РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Е.П. Истомин, С.Ю. Неклюдов, В.И. Романченко

ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ

учебник

Рекомендован Учебно-методическим объединением по образованию в области прикладной информатики в качестве учебника для студентов ВУЗов, обучающихся по специальности 080801 «Прикладная информатика (по областям)» и другим междисциплинарным специальностям

ООО «Андреевский издательский дом»
САНКТ-ПЕТЕРБУРГ, 2008

УДК 681.3.06.: 800.92
ББК 22.18
И 89

Истомин Е.П., Неклюдов С.Ю., Романченко В.И.
Информатика и программирование: Учебник. – СПб. ООО «Андреевский издательский дом», изд. 2-е, 2008 г. – 248 с.

ISBN 978-5-902894-19-3

Рецензенты: заведующий кафедрой вычислительных систем и информатики Санкт-Петербургского государственного университета водных коммуникаций, заслуженный деятель науки и техники РФ, доктор технических наук, профессор Гаскаров Д.В., профессор института экономики и права, доктор технических наук, профессор Григорьев Ю.Д.

Учебник «Информатика и программирование» предусматривает возможность подготовки специалистов широкого профиля. В книге представлены основы теории информации, технические средства передачи и обработки информации, а также подробно излагаются методы построения алгоритмов и программ на алгоритмическом языке высокого уровня ПАСКАЛЬ.

Материал изложен в соответствии с Государственным стандартом по специальности «Прикладная информатика (по областям)» в рамках дисциплины «Информатика и программирование».

Данная книга может быть использована в качестве учебника для студентов высших и средних учебных заведений по информационным дисциплинам как технических, так и гуманитарных специальностей, а также ею можно воспользоваться для практического освоения основ программирования на алгоритмическом языке ПАСКАЛЬ.

Истомин Е. П., Неклюдов С.Ю., Романченко В.И.
Информатика и программирование

Учебник

Редактор: Воронина О.Н.
Верстка: Истомин Д.Е.

Лицензия ЛП № 000216 от 14.07.1999 г.
ООО «Андреевский издательский дом»
197738, Санкт-Петербург, пос. Репино, Приморское шоссе, д. 394

E-mail: **biom@nm.ru**

Подписано в печать 20.08.2007 г.
Печ. листов 16,4. Тираж 2000 экз.

Отпечатано с готовых диапозитивов в ООО «Тетра»

ISBN 978-5-902894-19-3

ОГЛАВЛЕНИЕ

Глава 1. ОСНОВЫ ПЕРЕДАЧИ ИНФОРМАЦИИ И АППАРАТНЫЕ СРЕДСТВА

Глава 3. МИРОВЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

Глава 1. ОСНОВЫ ПЕРЕДАЧИ ИНФОРМАЦИИ И АППАРАТНЫЕ СРЕДСТВА

1. Теоретические основы передачи сообщения

1.1. Сообщения и сигналы

В общем случае под *информацией* понимают совокупность сведений о каких-либо событиях, явлениях или предметах. Для передачи или хранения информации используют различные знаки (символы), позволяющие выразить (представить) её в некоторой форме. Этими знаками могут быть слова и фразы в человеческой речи, жесты и рисунки, формы колебаний, математические знаки и т.п. Совокупность знаков, содержащих ту или иную информацию, называют *сообщением.* Так, при телеграфной передаче сообщением является текст телеграммы, представляющий собой последовательность отдельных знаков – букв и цифр. При разговоре по телефону сообщением является непрерывное изменение во времени звукового давления, отображающее не только содержание, но и интонацию, тембр, ритм и иные свойства речи. При передаче телевизионных изображений сообщение представляет собой изменение во времени яркости элементов этих изображений.

Средством передачи (переносчиком) сообщений на расстояние является *сигнал.* Под сигналом понимают физический процесс, отображающий (несущий) передаваемое сообщение. В информационных системах наиболее широко используются электрические сигналы в виде синусоидальных или импульсных токов и напряжений. Информационные сигналы формируются путем изменения одного из параметров электрического сигнала по закону передаваемого сообщения, т.е. путем модуляции.

По характеру изменения во времени различают непрерывные и дискретные сообщения.

Непрерывные по времени сообщения описываются непрерывной функцией времени. Например, речь при передаче телефонных разговоров, температура или давление воздуха при передаче телеметрических данных и т.п.

Дискретные по времени сообщения поступают только в определенные моменты времени и описываются дискретной функцией времени. Так как сообщения носят обычно случайный характер, то непрерывные сообщения описываются случайной функцией времени, а дискретные сообщения – как ряд случайных событий.

Сообщения могут быть также непрерывными и дискретными по состояниям.

Непрерывные по состояниям сообщения характеризуются тем, что функция, их описывающая, может принимать непрерывное множество значений в некотором интервале.

Дискретные по состояниям сообщения – это сообщения, которые могут быть описаны с помощью конечного набора чисел или дискретных значений некоторой функции.

Следует отметить, что в некоторых случаях сообщение не является функцией времени (например, текст телеграммы, неподвижное изображение и т.д.).

Сообщение с помощью датчиков обычно преобразуется в электрический сигнал, который в технике связи называют *первичным* сигналом. При передаче

речи такое преобразование выполняет микрофон, при передаче изображения – телевизионная камера. В большинстве случаев первичный сигнал является низкочастотным колебанием, которое отображает передаваемое сообщение.

Аналогичную сообщениям классификацию можно применить и для первичных сигналов, основные виды которых приведены на рис. 1.

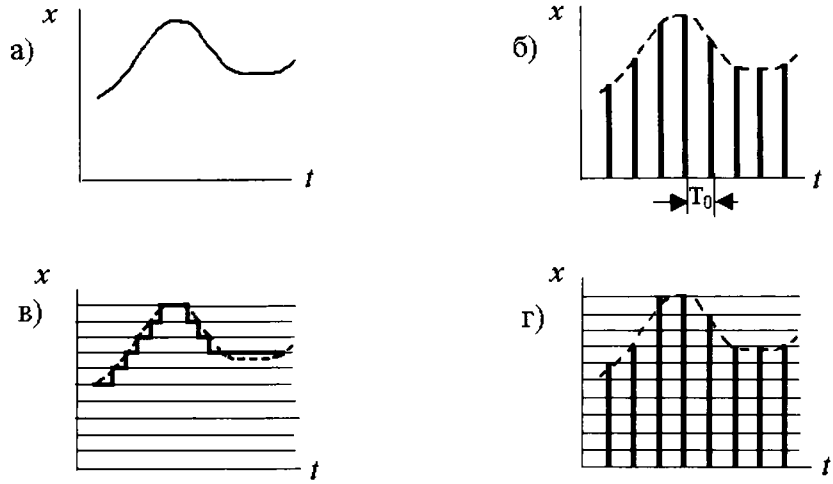


Рис. 1. Виды представления сигналов: а) – непрерывный; б) – дискретный; в) – квантованный; г) – дискретно-квантованный

В отличие от сообщений *сигнал* всегда *является функцией времени*, даже если сообщение таковым не является. Непрерывные сигналы описываются непрерывными функциями времени $x(t)$.

Дискретные сигналы позволяют перейти от аналоговой формы представления сигнала к цифровой, которая дает в ряде случаев значительные преимущества при передаче, хранении и обработке информации. Рассмотрим разновидности сигналов, которые описываются функцией $x(t)$.

Непрерывный сигнал. Описывается непрерывной функцией $x(t)$ непрерывного аргумента t (см. рис. 1.а). В этом случае сигнал (функция) принимает любые мгновенные значения на некотором отрезке времени и в диапазоне значений, ограничивающем его максимальную и минимальную величины, т.е.

$$x_{\min} \le x(t) \le x_{\max}.$$

Дискретный сигнал получается в результате дискретизации непрерывного сигнала $x(t)$ по времени. При этом непрерывный сигнал $x(t)$ заменяется последовательностью выборок (отсчетов), соответствующих мгновенным значениям сигнала $x(t)$, взятых в дискретные моменты времени $t = t_i$. Следовательно, он представляет собой непрерывную функцию дискретного аргумента $x(t_i)$, т.е. задан не на всей оси времени, а только в определенные моменты времени $t_i = i\Delta t$, где i = 0,1,2,…; Δt – интервал дискретизации. Величина $x(t_i)$ может принимать любое значение в диапазоне ($x_{\min}$, $x_{\max}$), т.е.

$$x_{\min} \le x(t_i) \le x_{\max.}.$$

Если Δt = const, то дискретизация называется *равномерной* с периодом $T_0 = \Delta t$ (см. рис. 1.б). В общем случае моменты отсчетов t_i могут выбираться неравномерно, например, “сгущаться” или “разряжаться” пропорционально скорости из-

менения сигнала (адаптивная дискретизация).

Квантованный сигнал получается в результате квантования непрерывного сигнала $x(t)$ по уровню. Квантование по уровню состоит в преобразовании непрерывного множества значений сигнала $x(t)$ в дискретное множество значений x_j. Таким образом, квантованный сигнал задается на всей оси времени, но определяется по величине только на дискретном множестве заранее установленных значений. В связи с этим он описывается дискретной функцией $x_j(t)$ непрерывного аргумента t (см. рис. 1.в), принимающей только определенные значения x_j при любом аргументе t. В частности, для случая *равномерного квантования:*

$$x_j(t) = j \cdot \Delta_x,$$

где j = 0, 1, 2,..., ($x_{max} - x_{min})/\Delta_x$; Δ_x – шаг квантования. При *неравномерном (адаптивном) квантовании* шаг квантования является переменным и изменяется в зависимости от текущих свойств сигнала.

Дискретно-квантованный сигнал является результатом совместного применения операций дискретизации и квантования. В этом случае непрерывный сигнал $x(t)$ преобразуется в дискретный сигнал по координатам x и t. Следовательно, он *описывается* дискретной функцией $x_j(t_i)$, заданной в дискретные моменты времени t_i и определяемой на дискретном множестве значений x_j, ближайших к мгновенным значениям $x(t)$ в моменты $t = t_i$. Таким образом, дискретно-квантованный сигнал представляет собой последовательность квантованных отсчетов $x_j(t_i)$, взятых в дискретные моменты времени $t_i = i\Delta t$, где i = 0, 1, 2,...; J = 0, 1, 2,... В частности, при $\Delta t = T_0$ и $\Delta_x = \text{const}$ $x_j(t_i) = j\Delta_x$, где j = 0, 1, 2,..., $(x_{max} - x_{min})/\Delta_x$ (см. рис. 1.г).

1.2. Количество информации в сообщениях

В настоящее время существуют три различных подхода к оценке количественных характеристик информации:

- При **структурном подходе** за единицу количества информации принимается некоторый квант данных, и количество информации оценивается простым их подсчетом. Например, в вычислительной технике за единицу количества информации принимается Байт (двоичная восьмиразрядная последовательность). При хранении или передаче данных их объем измеряется количеством Байт, содержащихся, например, в файле на магнитном диске или переданных по линии связи между машинами.
- В **семантической теории** учитывается, в основном, содержательная ценность информации, рассматривается ее изменение во времени (старение информации) и влияние этих изменений на эффективность управления в системе.
- **Статистическая теория** оценивает информацию с точки зрения меры неопределенности, снимаемой при получении информации. Она не затрагивает смыслового содержания информации, а основывается на вероятностных свойствах информационных элементов.

В этой книге будут рассматриваться, в основном, положения статистической теории информации. Связь между информативностью (количеством информации) некоторого события (сообщения) и вероятностью его возникновения можно проиллюстрировать следующими соображениями. Чем реже событие (сообщение), тем большую важность оно имеет, то есть содержит большее количество информации, нежели более частые события или сообщения.

Пусть имеется множество X_o, содержащее M событий (сообщений) X_{oj} (j = 1, 2, ..., M), которые возникают с вероятностями $p(X_{oj})$, при этом выполняется

соотношение

$$\sum_{j=1}^{M} p(X_{oj}) = 1 .$$

Количество информации I, содержащееся в событии X_{oj} (см. рис. 2), определяется по формуле:

$$I(X_{oi}) = - \log_2 p(X_{oi}). \qquad (1\text{-}1)$$

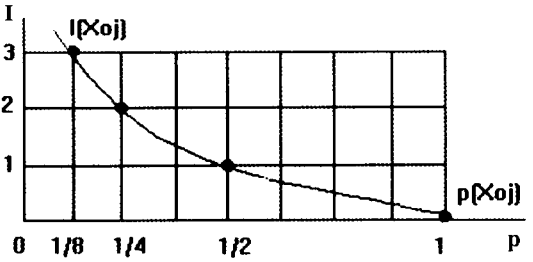


Рис. 2. Количество информации I, содержащееся в событии X_{oj}

При уменьшении вероятности возникновения сообщения количество информации в нем возрастает. Если вероятность события стремится к нулю, то его осуществление, граничащее с чудом, приносит количество информации, стремящееся к бесконечности. Если сообщение возникает с вероятностью равной 1, то есть заранее известно, что возникнет именно это сообщение, то количество информации в сообщении равно 0.

За единицу количества информации (1 бит) принимают количество информации, содержащееся в сообщении, имеющем вероятность возникновения равную 0,5 (см. рис. 3). Единицу измерения информации называют *битом.* Это слово предложил Тьюки.

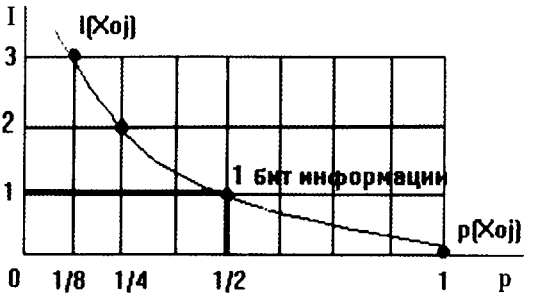


Рис. 3. Графическая интерпретация сообщения, имеющего вероятность возникновения равную p = 0,5

Пример. Имеется множество X_o из M = 4 сообщений с вероятностями появления $p(X_{oj})$, приведенными ниже во второй графе табл. 1.

Таблица 1.

Сообщение, X_{oj}	Вероятность возникновения, $p(X_{oj})$	Количество информации, $I(X_{oj})$
X_{o1}	1/2	
X_{o2}	1/4	
X_{o3}	1/8	
X_{o4}	1/8	

Требуется определить количество информации I, содержащееся в каждом из сообщений X_{oj}.

Выполним следующие расчёты:

1. $I(X_{01}) = - \log_2 p(X_{01}) = - \log_2 (1/2) = 1$ бит.
2. $I(X_{02}) = - \log_2 p(X_{02}) = - \log_2 (1/4) = 2$ бита.
3. $I(X_{03}) = - \log_2 p(X_{03}) = - \log_2 (1/8) = 3$ бита.
4. $I(X_{04}) = - \log_2 p(X_{04}) = - \log_2 (1/8) = 3$ бита.

Результаты расчётов количество информации $I(X_{oj})$ отражены в табл. 2.

Таблица 2.

Сообщение, X_{oj}	Вероятность возникновения, $p(X_{oj})$	Количество информации, $I(X_{oj})$ [бит]
X_{o1}	1/2	1
X_{o2}	1/4	2
X_{o3}	1/8	3
X_{o4}	1/8	3

Сообщения содержат разное количество информации. С учетом вероятностей возникновения сообщений количество информации $I_{ср}$, в среднем приходящееся на одно сообщение, будет равно:

$$I_{cp} = \sum p(X_{0j}) \times I(X_{0j}) \qquad (1\text{-}2)$$

Подставляя значения $p(X_{oj})$ и $I(X_{oj})$ из табл. 2 в (1-2) получается:

$$I_{ср} = 1/2 \times 1 + 1/4 \times 2 + 1/8 \times 3 + 1/8 \times 3 = 1.75 \text{ бит.}$$

Пример: Имеется множество X_0 содержащее M сообщений с одинаковыми вероятностями возникновения $p(X_{oj}) = 1/M$; j = 1, 2, …, M.

Следует отметить, что

$$\sum_{j=1}^{M} p(X_{oj}) = \sum_{j=1}^{M} \frac{1}{M} = \frac{M}{M} = \tilde{1}$$

Тогда в каждом сообщении будет содержаться одно и то же количество информации:

$$I = - \log_2 (1/M) = \log_2 M. \qquad (1\text{-}3)$$

Это известная формула предложена Хартли для определения количества информации M равновероятных сообщений. Графическая интерпретация формулы (1-3) приведена на рис. 4. Из (1-3) следует, что при увеличении числа равновероятных сообщений количество информации в сообщении увеличивается.

Множество M сообщений удобнее характеризовать средним количеством информации, приходящимся на одно сообщение, с учетом вероятностей возникновения этих сообщений. Это можно сделать, используя понятие энтропии.

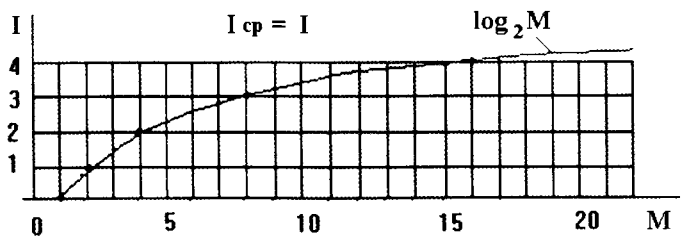


Рис. 4. Количество информации M равновероятных сообщений

1.3. Энтропия и ее свойства

Энтропия – это мера неопределенности состояния системы. Для определения энтропии используют формулу:

$$H = -\sum_{j=1}^{M} p(X_{oj}) \times \log_2 p(X_{oj}), \qquad (1\text{-}4)$$

где M – количество состояний системы; $p(X_j)$ – вероятность j-го состояния. Таким образом, энтропия является монотонной функцией от вероятностей состояния системы $p(X_j)$.

Не трудно заметить, что эта формула точно соответствует количеству информации, приходящемуся в среднем на одно сообщение из множества M сообщений. Если все сообщения из множества M равновероятны, то есть:

$$p(X_{oj}) = \frac{1}{M}; \; j = \overline{1, M};$$

что следует из

$$\sum_{j=1}^{M} p(X_j) = 1,$$

то

$$H = -\sum_{j=1}^{M} p(X_{oj}) \times \log_2 p(X_{oj}) = -\sum_{j=1}^{M} \frac{1}{M} \times \log_2 \frac{1}{M} = \log_2 M.$$

$$H = \log_2 M \qquad (1\text{-}5)$$

Получаем, что $H = \log_2 M$, но это есть формула Хартли, которая определяет количество информации, содержащееся в каждом из M равновероятных сообщений. Если же события неравновероятны, то в среднем в них содержится количество информации меньшее, чем показывает функциональная зависимость, представленная на рис. 5.

Из (1-5) следует утверждение: чем больше равновероятных событий, тем большее количество информации связано с каждым из этих событий.

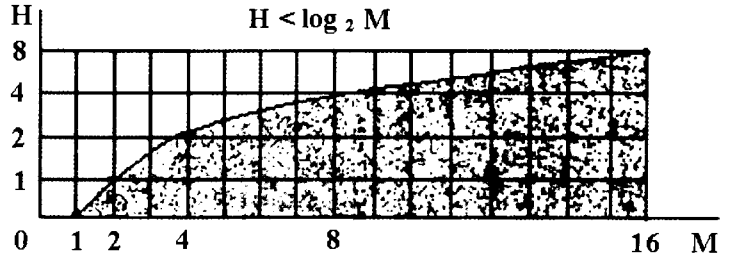


Рис. 5. Количество информации для M неравновероятных сообщений

Энтропия двух событий

При двух равновероятных событиях каждое из них содержит 1 бит информации (см. рис. 6).

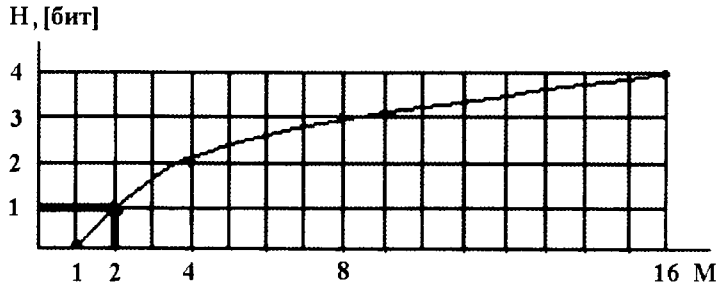


Рис. 6. Количество информации для двух равновероятных событий

Два события, имеющие вероятности P и 1 – P, содержат в среднем $H = - p \times \log_2 p - (1 - p) \times \log_2 (1 - p)$ двоичных единиц (бит) информации (см. рис. 7).

Рис. 7. Энтропия для двух равновероятных событий

Если события неравновероятны, то неопределённость состояния системы (энтропия, количество информации) уменьшается (см. рис. 8).

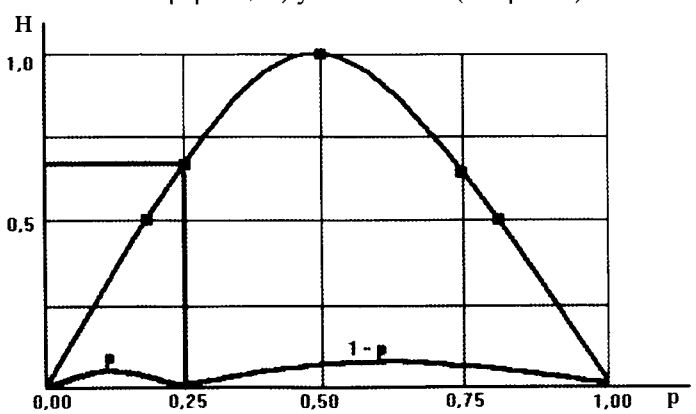


Рис. 8. Энтропия для двух неравновероятных событий

В среднем же каждое из них сдержит с учетом их вероятностей:

$$H = p \times I_1 + (1-p) \times I_2 = 0.25 \times 2 + 0.75 \times 0.415 = 0.811 \, бит .$$

Так как первое событие содержит $I_1 = -\log_2 p = -\log_2 0.25 = 2$ бита информации, а второе - $I_2 = -\log_2 (1-p) = -\log_2 0.75 = 0.415$ бит.

Если же p=0 (одно событие невозможно, а другое - достоверно), то неопределённость в системе отсутствует, и её информативность равна 0.

Энтропия инвариантна по отношению к перестановке значений вероятностей событий (см. рис. 9).

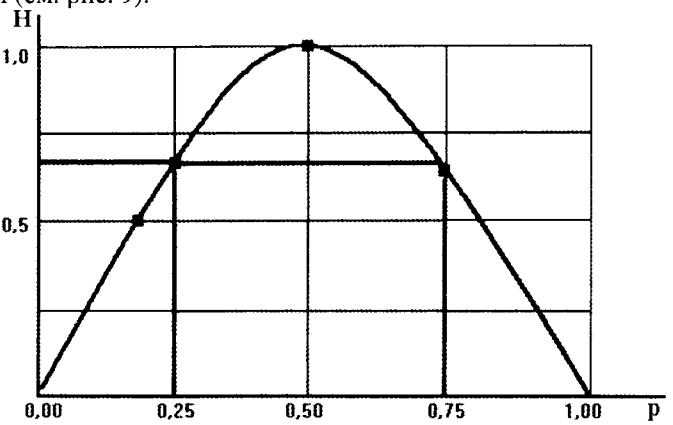


Рис. 9. Инвариантность энтропии

Энтропия для трех событий

Что представляет собой энтропия для ансамбля из трех событий?

Пусть дана система из трех событий (см. рис. 10) с вероятностями p1, p2, p3, где (p1 + p2 + p3 = 1), которая характеризуется энтропией:

$$H = -p1 \times \log_2 p1 - p2 \times \log_2 p2 - p3 \times \log_2 p3$$

с максимальным значением - $H_{max} = \log_2 3 = 1.58$ бит, при p1 = p2 = p3 = 1/3.

Плоскость abc определяет область существования энтропии (рис. 10.а) и, для рассматриваемого примера, является ограничением

$$p1 + p2 + p3 = 1.$$

Энтропия H(p1, p2, p3), как функция от трёх вероятностей p1, p2, p3, представляет собой выпуклую поверхность, схематично показанную на рис. 10.б.

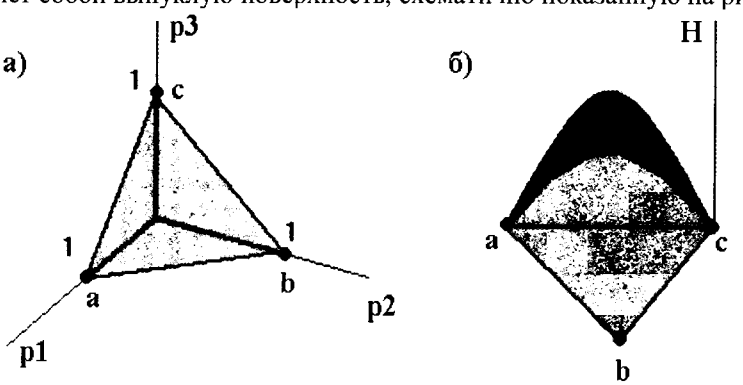


Рис. 10. Система для трёх событий: а) плоскость ограничений;
б) энтропия трёх событий abc

Точки a, b и c соответствуют ситуациям, когда одна из вероятностей равна 1, а две другие равны 0. В этих случаях неопределенность в системе отсутствует, и энтропия H равна 0. Если события равновероятны, то есть p1 = p2 = p3 = 1/3, то энтропия максимальна (см. рис. 11).

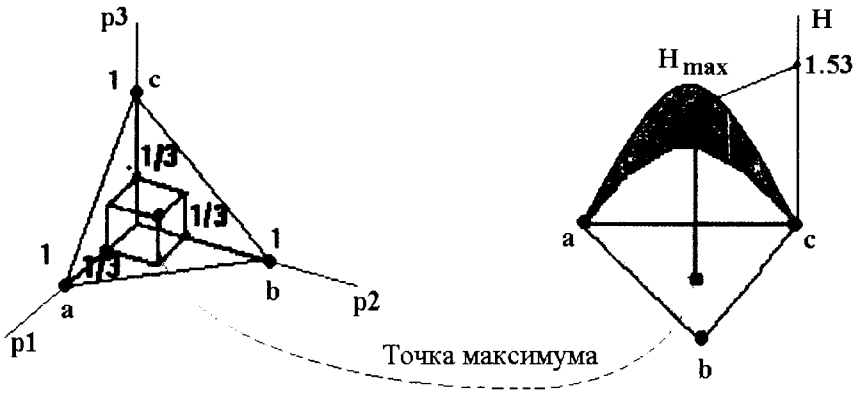


Рис. 11. Система для трёх равновероятных событий

Нетрудно найти значение энтропии для трёх равновероятных событий по формуле:

$$H_{\max} = H\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right) = -\sum_{i=1}^{3}\frac{1}{3}\times\log_2\frac{1}{3} = \log_2 3 \approx 1.53.$$

Третий вариант. Рёбра **ab**, **bc**, **ca** соответствуют ситуациям, когда одна из вероятностей равна 0. В этом случае имеется только два возможных события, при равенстве вероятностей которых получается максимальная энтропия равная 1 (см. рис. 12).

На рис. 12.а выделены точки посередине рёбер **ab**, **bc**, **ca.** Эти точки соответствуют двум равновероятным событиям при невозможном третьем событии. Энтропия в этих трёх случаях определяется по формуле:

$$H_i = \left(-\frac{1}{2}\times\log_2\frac{1}{2}\right)+\left(-\frac{1}{2}\times\log_2\frac{1}{2}\right)+\left(-0\times\log_2 0\right) = 1; i = 1,2,3.$$

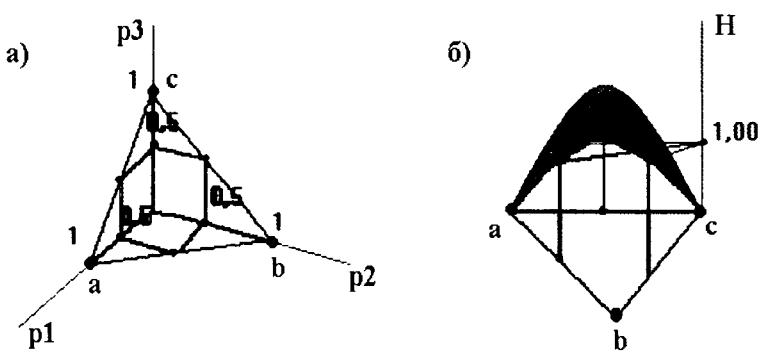


Рис. 12. Система для двух равновероятных событий и одного невозможного: а) область ограничений, б) энтропия

Энтропия и выбор события

Энтропия обладает свойством инвариантности к выбору события. На рис. 13 представлена система с тремя возможными событиями 1, 2, 3. Вероятности выбора событий p1, p2, p3 образуют полную группу, то есть $p1 + p2 + p3 = 1$.

На рис. 13.а выбор события производится за 1 шаг. Энтропия такой системы - H(p1, p2, p3). На рис. 13.б также система с тремя событиями 1, 2, 3. Выбор события осуществляется следующим образом. На первом шаге выбирается с вероятностью p1 первое событие. С вероятностью p0 = 1 - p0 требуется продолжить выбор на втором шаге. Энтропия первого шага H(p1, p0). Второй шаг происходит с вероятностью p0. Энтропию второго шага можно найти из формулы $H\left(\frac{p2}{p0}, \frac{p3}{p0}\right)$.

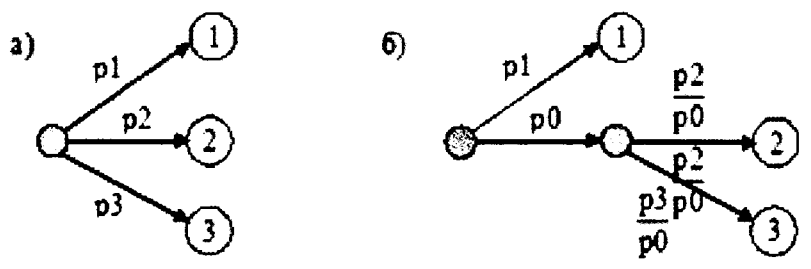


Рис. 13. Система с тремя событиями: а) выбор за 1 шаг;
б) выбор за 2 шага

Свойство выбора формулируется с помощью следующего тождества:

$$H(p1, p2, p3) \equiv H(p1, p0) + p0 \times H\left(\frac{p2}{p0}, \frac{p3}{p0}\right); \qquad (1\text{-}6)$$

при этом выполняются соотношения: $p1 + p2 + p3 = 1, p0 = p2 + p3$.

Тождество (1-6) нетрудно доказать, раскрывая правую часть с помощью формулы энтропии.

$$H(p1, p0) + p0 \times H\left(\frac{p2}{p0}, \frac{p3}{p0}\right) = -p1 \times \log_2 p1 - p0 \times \log_2 p0 +$$

$$p0 \times \left(-\frac{p2}{p0} \times \log_2 \frac{p2}{p0} - \frac{p3}{p0} \times \log_2 \frac{p3}{p0}\right) = -p1 \times \log_2 p1 - p0 \times \log_2 p0 - p2 \times \log_2 p2 +$$

$$p2 \times \log_2 p0 - p3 \times \log_2 p3 + p3 \times \log_2 p0 = -p1 \times \log_2 p1 - p2 \times \log_2 p2 -$$

$$p3 \times \log_2 p3 - p0 \times \log_2 p0 + (p2 + p3) \times \log_2 p0 = H(p1, p2, p3).$$

Что и требовалось доказать. Таким образом, справедливо утверждение: *энтропия системы зависит только от числа событий и их вероятностей и не зависит от порядка выбора.* Формула (1-6) позволяет понизить размерность энтропии, то есть выразить энтропию трёх событий через энтропии двух событий.

1.4. Системы связи

Совокупность технических средств, предназначенных для передачи сообщений от источника к потребителю, называется *системой связи.* Этими средствами являются передающее устройство, линия связи и приемное устройство. Иногда в понятие система связи включаются источник и потребитель сообщений.

Структурная схема системы связи

Простейшая система связи схематично изображена на рис. 14. Ниже рассматривается назначение отдельных элементов этой схемы.

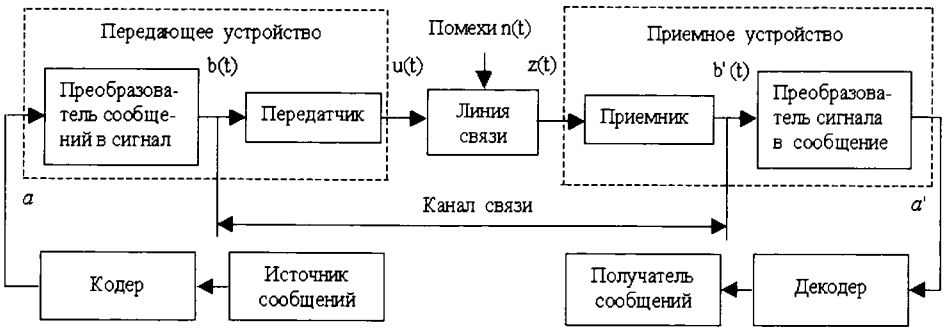


Рис. 14. Структурная схема системы связи

Источником и получателем сообщений в одних системах связи может быть человек, в других – технические средства, связанные с человеком.

Кодер, декодер – устройства, цель которых состоит в согласовании свойств дискретного источника сообщений со свойствами канала связи. Эти устройства решают две основные задачи:

- *Оптимальное кодирование.* Это существенное снижение среднего числа символов в коде на букву сообщения, что даёт выигрыш в объёме сообщения и времени его передачи, и, тем самым, повышает эффективность системы связи;
- *Помехоустойчивое кодирование.* При наличие помехи в канале связи, приводящей к искажению передаваемой информации, кодер вводит в код сообщения избыточность с учётом интенсивности и статистической закономерности помехи, а декодер обнаруживает ошибки и, при возможности, восстанавливает исходное сообщение.

Устройство, преобразующее сообщение в сигнал, называют *передающим устройством,* а устройство, преобразующее принятый сигнал в сообщение, - *приемным устройством.*

С помощью преобразователя в передающем устройстве сообщение a с любой физической природой (изображение, звуковое давление и т.п.) преобразуется в первичный электрический сигнал $b(t)$. В телефонии, например, эта операция сводится к преобразованию звукового давления в пропорционально изменяющийся ток микрофона. В телеграфии сначала производится кодирование, в результате которого последовательность элементов сообщения (букв) заменяется последовательностью кодовых символов (0, 1 или точка, тире), которая затем с помощью телеграфного аппарата преобразуется в последовательность электрических импульсов постоянного тока.

В *передатчике* первичный (низкочастотный) сигнал $u(t)$ преобразуется во вторичный (высокочастотный) сигнал $u(t)$, пригодный для передачи по используемому каналу. Это осуществляется посредством модуляции.

Под *линией связи* понимают физическую среду, по которой происходит распространение электромагнитной энергии, и технические устройства, обеспечивающие качественную передачу сигналов сообщения от передатчика к приемнику.

В зависимости от конкретных условий, в которых организуется связь, для передачи сигналов используют проводные - или радиолинии. К проводным линиям связи относятся воздушные и кабельные линии, волноводы, световоды и др. В радиолиниях сообщения передаются посредством радиоволн.

В процессе передачи сигналы подвергаются искажениям, которые обусловлены неидеальными характеристиками аппаратуры и воздействием различного рода помех $n(t)$.

Приемное устройство обрабатывает принятое колебание $z(t) = u(t) + n(t)$, представляющее собой сумму пришедшего искаженного сигнала u(t) и помехи $n(t)$, и восстанавливает по нему сообщение *a'*, которое с некоторой погрешностью отражает переданное сообщение *a*. Обратное преобразование принятого сигнала в сообщение осуществляется посредством демодуляции и, при необходимости, декодирования.

Системы связи могут быть одноканальными (см. рис. 14) и многоканальными. Система связи называется *многоканальной*, если она обеспечивает передачу нескольких сообщений по одной линии связи.

Энтропия сообщения

Под энтропией сообщения понимается среднее количество информации, содержащееся в одном из М сообщений отправителя:

$$H(S) = -\sum_{j=1}^{M} p(S_j) \times \log_2 p(S_j), \qquad (1\text{-}7)$$

где $p(S_j)$ – вероятность j-ого сообщения.

Перед отправкой или сохранением в запоминающем устройстве сообщение преобразуется кодирующим устройством в некоторый код с основанием К. Элементами этого кода являются К различных символов, например, символы 0, 1, 2, . . . , К – 1.

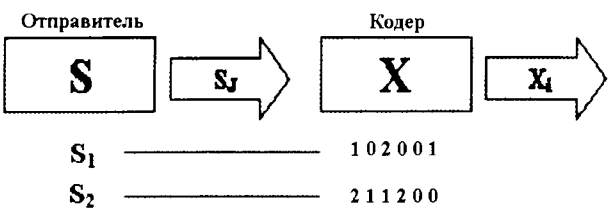


Рис. 15. Отправка сообщения

Кодирующее устройство можно рассматривать как источник различных кодовых символов (например, символов 0,1,2, … K-1). Среднее количество информации в одном сообщении определяется энтропией сообщения H(S).

Энтропия источника

Энтропия источника – среднее количество информации, переносимое одним символом кода:

$$H(X) = -\sum_{j=1}^{K} p(X_j) \times \log_2 p(X_j), \qquad (1\text{-}8)$$

где $p(X_j)$ – вероятность появления символа X_i.

Среднее количество информации в одном сообщении определяется энтропией сообщения H(S). Среднее число символов кода в одном сообщении можно

найти по формуле:

$$n = \frac{H(S)\left[\frac{бит}{сообщение}\right]}{H(X)\left[\frac{бит}{символ}\right]} = \frac{H(S)}{H(X)}\left[\frac{символ}{сообщение}\right], \quad (1\text{-}9)$$

где H(X) - энтропия источника.

Если все символы кода на выходе кодирующего устройства появляются с одинаковыми вероятностями $p(X_i) = \frac{1}{K}$, при i = 1,2, … K, то энтропия источника максимальна и равна $H_{max}(X) = \log_2 K$. При преобразовании получается минимальное среднее число символов на одно сообщение n_{min}:

$$n_{min} = \frac{H(S)}{\log_2 K} \quad (1\text{-}10)$$

где K – число различных символов, используемых для кодирования сообщений.

Величина R определяет долю «лишней» информации на выходе кодирующего устройства или, как говорят, *избыточность кода.*

$$R = \frac{n - n_{min}}{n} = 1 - \frac{n_{min}}{n} = 1 - \frac{H(X)}{\log_2 K}. \quad (1\text{-}11)$$

Отношение $\frac{H(X)}{\log_2 K}$ называют *относительной энтропией* источника.

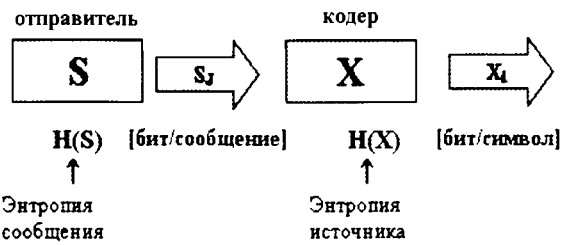


Рис. 16. Отправка сообщения

Снижая избыточность R, это отношение стремятся увеличить, выбирая наилучший метод кодирования.

1.5. Классификация каналов связи

Каналом связи называется совокупность технических средств, предназначенных для передачи сигналов сообщений любого вида от источника (отправителя) к получателю. Причем передача по одному каналу осуществляется независимо от других каналов. В состав канала связи входят: линия связи, передатчик и приемник (см. рис. 14).

Каналы связи могут быть классифицированы по различным признакам, основными из которых являются:

- область применения (назначение);
- способ распространения сигналов;
- физическая природа сигналов;
- диапазон используемых частот;

- условия распространения сигналов;
- вид входного и выходного сигналов.

По *назначению* каналы связи разделяются на телефонные, телеграфные, фототелеграфные, факсимильные, телевизионные, каналы передачи данных, звукового вещания, телеуправления и т.д.

В зависимости от *способа распространения* сигнала между пунктами связи (в свободном пространстве или по линии), выделяют каналы радио- и проводной связи.

По *физической природе сигналов*, используемых для передачи сообщений, различают акустические, оптические, электрические и радиоканалы. Акустические каналы применяются в таких областях техники, как гидролокация, дефектоскопия и др. Особое место занимают оптические каналы. Они могут быть организованы как по проводам (волоконным световодам), так и без проводов – со свободным распространением лазерного излучения в разных средах (атмосфере, воде, космосе). Наиболее распространенные каналы связи – электрические и радиоканалы.

Более существенна классификация каналов связи по *диапазону* используемых ими *частот*. По диапазону частот каналы и используемые ими линии связи располагаются в следующем порядке (см. табл. 3).

Таблица 3.

Тип канала связи	Тип линии связи (среда)	Диапазон частот
Акустический	Воздушная	до 20 кГц
	Магнитострикционные материалы	до 30 кГц
	Гидроакустическая	10 Гц – 10 МГц
	Пьезоэлектрические кристаллы	до 10^9 Гц
Электрический	Воздушная	до 150 кГц
	Симметричный кабель	до 1 МГц
	Коаксиальный кабель	до 300 МГц
Радиоканал	Радиосвязь	0,3 - 3·10^3 МГц
	Ионосферная	30 - 60 МГц
	Метеорная	30 - 300 МГц
	Тропосферная	300 - 5·10^3 МГц
	Радиорелейная	30 - 3·10^4 МГц
Оптический	Космическая	30 - 3·10^4 МГц
	Волоконно-оптическая	0,3 - 0,8·10^{15} Гц
	Воздушная	0,3 - 1·10^{15} Гц

Радиоканалы, в свою очередь, также можно классифицировать по диапазонам используемых ими волн (частот). Соответствующая классификация приведена в табл. 4.

В зависимости от *условий распространения* сигналов различают проводные, радио- и радиорелейные каналы, тропосферные, ионосферные, метеорные и космические (спутниковые) каналы связи.

Особенностью радиорелейных каналов является использование сигналов сантиметрового диапазона волн, которые распространяются только в пределах прямой видимости. Поэтому для организации связи на большие расстояния линии

связи этих каналов выполняют в виде цепочки *активных ретрансляторов*, где сигналы усиливаются и передаются от одного к другому.

Таблица 4.

Радиоканалы		Диапазон волн	Диапазон частот
Сверхдлинноволновые (СДВ)		10 – 100 км	3 – 30 кГц
Длинноволновые (ДВ)		1 – 10 км	30 – 300 кГц
Средневолновые (СВ)		100 – 1000 м	0,3 – 3 МГц
Коротковолновые (КВ)		10 – 100 м	3 – 30 МГц
Метровых волн	ультракоротковолновые (УКВ)	1 – 10 м	30 – 300 МГц
Дециметровых волн		0,1 – 1 м	300 – 3000 МГц
Сантиметровых волн		1 – 10 см	3 – 30 ГГц
Миллиметровых волн		1 – 10 мм	30 – 300 ГГц

В спутниковых системах связи, за счет размещения *активного ретранслятора на ИСЗ*, обеспечивается глобальная связь между любыми объектами, в том числе и с подвижными (судами, самолетами, космическими кораблями и др.).

В тропосферных, ионосферных и метеорных каналах связь на большие расстояния обеспечивается за счет отражения и рассеяния излученных сигналов в неоднородностях тропосферы, ионосферы и метеорных следов, выполняющих роли *пассивных ретрансляторов.*

Наибольший интерес для теории передачи сигналов представляет классификация каналов связи по *виду сигналов на входе и выходе канала.* Различают каналы:

а) непрерывные, если на входе и выходе канала сигналы непрерывны;

б) дискретные, если на входе и выходе канала сигналы дискретны;

в) дискретно–непрерывные, если на вход канала поступают дискретные сигналы, а с выхода снимаются непрерывные сигналы;

г) непрерывно–дискретные, если на вход канала поступают непрерывные сигналы, а с выхода снимаются дискретные сигналы.

Необходимо отметить, что дискретность и непрерывность канала не связана с характером передаваемых сообщений. Можно передать дискретные сообщения по непрерывному каналу и непрерывные сообщения по дискретному каналу.

1.6. Параметры и характеристики сигнала и канала связи

Сигнал является объектом транспортировки сообщения, поэтому целесообразно определить параметры сигнала, которые являются основными с точки зрения его передачи. Такими параметрами являются длительность сигнала T_C, его динамический диапазон D_C и ширина спектра F_C.

Всякий сигнал, рассматриваемый как временной процесс, имеет начало и конец. При этом *длительность сигнала* T_C является естественным его параметром, определяющем интервал времени, в пределах которого сигнал существует.

Динамический диапазон характеризует диапазон изменения мгновенной мощности сигналов связи, которая может принимать различные значения в весьма широких пределах. Динамический диапазон сигнала выражается, обычно, в децибелах и определяется по формуле

$$D_C = 10 \times \lg(P_{МАКС} / P_{МИН}), \text{ [дБ]} \qquad (1\text{-}12)$$

где $P_{МАКС}$ - максимальное, а $P_{МИН}$ - минимальное значения мгновенной мощности.

Динамический диапазон речи диктора, например, равен 25 - 30 дБ, небольшого вокального ансамбля – 45 - 65 дБ, симфонического оркестра – 70 - 95 дБ. Во избежание перегрузок канала в радиовещании динамический диапазон часто сокращают до 35 - 45 дБ.

Ширина спектра сигнала F_C - это диапазон частот, в пределах которого сосредоточена его основная энергия. Этот параметр дает представление о скорости изменения сигнала внутри интервала его существования. В технике связи спектр сигнала часто сознательно сокращают. Это связано с тем, что аппаратура и линия связи имеют ограниченную полосу пропускаемых частот. Сокращение спектра осуществляется исходя из допустимых искажений сигнала. Например, при телефонной связи требуется, чтобы речь корреспондентов была разборчива и узнаваема по голосу. Для выполнения этих условий достаточно передать речевой сигнал в полосе от 300 до 3400 Гц. Передача более широкого спектра речи в этом случае нецелесообразна, так как ведет к техническим усложнениям и увеличению затрат.

Спектр модулированного сигнала обычно шире спектра передаваемого сообщения (первичного сигнала) и зависит от вида модуляции. Более общей и наглядной информационной характеристикой сигнала является *объем сигнала,* определяемый из выражения:

$$V_C = T_C \; F_C \; D_C . \qquad (1\text{-}13)$$

Объем сигнала V_C дает общее представление о возможностях данного множества сигналов как переносчиков сообщений. Чем больше объем сигнала, тем больше информации можно "вложить" в этот объем и тем труднее передать такой сигнал по каналу связи.

Канал связи, как и сигнал, можно характеризовать тремя соответствующими *параметрами*: временем использования канала T_K, полосой частот, пропускаемых каналом F_K и динамическим диапазоном канала D_K. Под динамическим диапазоном канала понимают отношение (выраженное в децибелах) допустимой мощности передаваемого сигнала $P_{\text{с.max}}$ к мощности $P_{\text{п}}$ неизбежно присутствующей в канале помехи:

$$D_K = 10 \times \lg(P_{\text{с.max}}/P_{\text{п}}), \; [\text{дБ}] \qquad (1\text{-}14)$$

Основными характеристиками канала связи являются: емкость канала связи, скорость передачи информации и пропускная способность канала связи. *Емкость канала связи* V_K представляет собой произведение его параметров:

$$V_K = T_K \; F_K \; D_K . \qquad (1\text{-}15)$$

Неискаженная передача по каналу сигналов объема V_C возможна при условии $V_C \le V_K$, при котором объем сигнала полностью "вписывается" в емкость канала. Однако соотношение $V_C \le V_K$ выражает необходимое, но недостаточное условие согласования сигнала с каналом. Достаточным условием является согласование по всем параметрам

$$T_C \le T_K ; \quad F_C \le F_K ; \quad D_C \le D_K . \qquad (1\text{-}16)$$

Скорость передачи информации U определяет среднее количество инфор-

мации $I(X,Y)$ переносимое одним символом в единицу времени:

$$U = I(X,Y)/T_C = \{H(X) - H(X/Y)\}/T_C \quad \text{[ед. инф./с]}, \quad (1\text{-}17)$$

где: $H(X)$ - энтропия источника сигналов x, состоящих из символов алфавита $X(x_1, x_2, ..., x_m)$; $H(X/Y)$ - условная энтропия ансамбля сигналов x при известных выходных сигналах y (с алфавитом Y), определяемая помехой и зависящая от ее уровня и характера; T_C - среднее время передачи одного символа.

Наибольшая теоретически достижимая для данного канала скорость передачи информации называется *пропускной способностью канала*

$$C = U_{\max} = I_{\max}(X,Y)/T_C = \max\{H(X) - H(X/Y)\}/T_C. \quad (1\text{-}18)$$

Иначе говоря, пропускная способность определяет максимальное количество информации, которое можно передать в единицу времени. Если скорость передачи информации в общем случае зависит от статистических свойств сигнала, метода кодирования и свойств канала, то пропускная способность определяется только свойствами канала.

Дискретный канал без помех. В любом реальном канале всегда присутствуют помехи. Однако если их уровень настолько мал, что вероятность искажения практически равна нулю, можно условно считать, что все сигналы передаются неискаженными. В этом случае среднее количество информации, переносимое одним символом

$$I(X,Y) = I(X,X) = H(X). \quad (1\text{-}19)$$

Максимальное значение можно найти по формуле

$$\max\{I(X,Y\} = H_m(X), \quad (1\text{-}20)$$

где $H_m(X)$ - максимальная энтропия источника сигналов, имеющая место при равенстве вероятностей символов алфавита источника

$$p(x_1) = p(x_2) = ... = p(x_m) = 1/m.$$

С учетом известного соотношения для энтропии источника

$$H(X) = -\sum_{i=1}^{m} p(x_i)\log_a p(x_i).$$

Максимальная энтропия, выраженная в единицах информации на символ

$$H_m(X) = \log_a m.$$

Следовательно, *пропускная способность дискретного канала без помех* равна:

$$C = v_x \log_a m = F_T \log_a m, \quad (1\text{-}21)$$

где v_x - предельная скорость передачи по каналу элементарных сигналов x, численно равная тактовой частоте F_T следования символов.

Пропускная способность дискретного канала связи с помехами определяется следующим образом. Пусть p_0 - вероятность ошибочного приема символа. Тогда при передаче символов сигнала из алфавита объемом m, условная или остаточная энтропия

$$H(X/Y) = p_0 \log_a \frac{m-1}{p_0} + (1-p_0)\log_a \frac{1}{1-p_0}. \quad (1\text{-}22)$$

С учетом (1-22) выражение (1-18) принимает вид

$$C = \left[\log_a m + p_0 \log_a \frac{p_0}{m-1} + (1-p_0)\log_a (1-p_0) \right] / T_C, \quad (1\text{-}23)$$

где T_C - длительность символа.

Для симметричного двоичного канала ($m = 2$) пропускная способность

$$C = \left[1 + p_0 \log_2 p_0 + (1-p_0)\log_2 (1-p_0)\right] / T_C. \quad (1\text{-}24)$$

Пропускная способность непрерывного канала с аддитивным белым гауссовским шумом в полосе ΔF пропускания канала может быть найдена из формулы Шеннона:

$$C = \Delta F \times \log_2 (1 + P_C / P_{Ш}), \quad (1\text{-}25)$$

где: P_C - средняя мощность сигнала; $P_{Ш}$ - мощность шума в полосе ΔF. При этом предполагается, что сигнал $x(t)$ и шум в канале взаимно некоррелированы, т.е. статистически независимы.

Соотношение (1-25) определяет зависимость пропускной способности рассматриваемого канала от его технических характеристик, таких как ширина полосы пропускания и отношение сигнал – шум. Данная зависимость указывает на возможность обмена полосы пропускания на мощность сигнала, и наоборот. Учитывая что, величина C зависит от ширины полосы ΔF линейно, а от отношения $P_C / P_{Ш}$ - по логарифмическому закону, то компенсировать возможное сокращение полосы пропускания увеличением мощности сигнала, как правило, не выгодно. Более эффективным является обратный обмен мощности сигнала на полосу пропускания.

1.7. Дискретизация непрерывных сообщений по времени

Непрерывные сообщения часто приходится преобразовывать в форму, более удобную для передачи по тем или иным каналам связи. Наиболее распространенным преобразованием непрерывных сообщений является их *дискретизация по времени*. Это преобразование применяется при импульсных методах передачи сообщений, многоканальной передаче с временными методами уплотнения и разделения, а также как промежуточная операция при преобразовании непрерывных сообщений в цифровую форму.

Дискретизация непрерывного сообщения $x(t)$ по времени состоит в замене его последовательностью мгновенных значений (отсчетов) $x(t_k)$, взятых в дискретные моменты времени $t_k = k \cdot \Delta t$, где Δt – интервал дискретизации, а k – целые числа, как положительные, так и отрицательные (рис. 17).

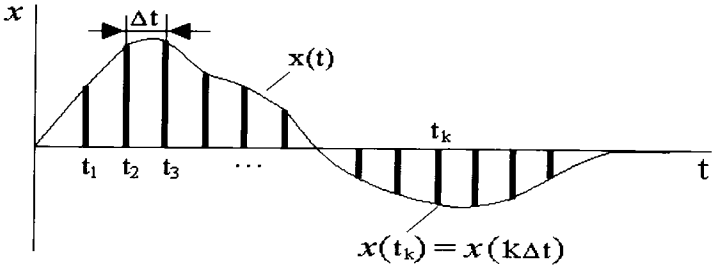


Рис. 17. Дискретизация непрерывного сообщения по времени

Математической моделью последовательности таких импульсов является сумма дельта – функций:

$$x(t_k) = \sum_{k=-\infty}^{\infty} x(k\Delta t)\delta(t - k\Delta t). \qquad (1\text{-}26)$$

При постоянном интервале Δt дискретизация будет равномерной с периодом $T_п=\Delta t$. В общем случае моменты отсчетов t_k могут выбираться неравномерно, т.е. пропорционально скорости изменения сообщения (адаптивная дискретизация). Равномерная дискретизация как более простая чаще используется на практике.

Технически дискретизация по времени реализуется с помощью ключевых устройств, управляемых периодической последовательностью коротких прямоугольных импульсов (см. рис. 18).

При дискретизации по времени возникают задачи, связанные с выбором интервала Δt и восстановлением с заданной точностью исходного сообщения по его отсчетам.

Как было ранее отмечено, для передачи сообщений без искажений необходим канал связи с бесконечной пропускной способностью. На практике передача сообщений всегда осуществляется с ограниченным (финитным) спектром частот и точностью, так как все каналы имеют ограниченную пропускную способность.

Возможность *восстановления по дискретным значениям* непрерывных сообщений с ограниченным спектром впервые была сформулирована В.А. Котельниковым в 1933 году в виде следующей *теоремы: "Если непрерывная функция времени x(t) не содержит составляющих с частотой выше F_m, то она полностью определяется мгновенными значениями, отсчитываемыми через интервалы времени $\Delta t = 1/2F_m$".*

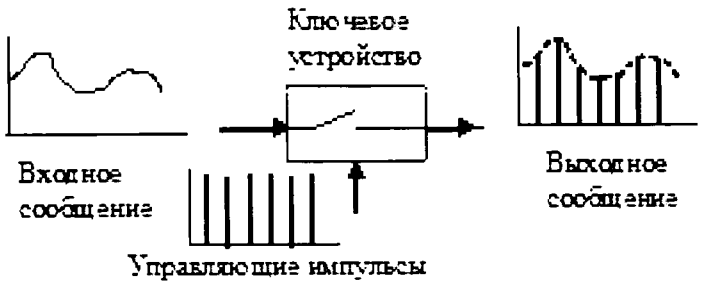


Рис. 18. Принцип технической реализации дискретизации по времени

Функцию с ограниченным спектром можно записать в виде тригонометрического интерполяционнго ряда:

$$x(t) = \sum_{k=-\infty}^{\infty} x(k\Delta t)\frac{\sin 2\pi F_m(t - k\Delta t)}{2\pi F_m(t - k\Delta t)}, \qquad (1\text{-}27)$$

где k – порядковый номер отсчета функции.

Как видно из выражения (1-27), непрерывная функция времени x(t) представляется суммой произведений, один из сомножителей которых есть выборка функции, а другой – так называемая функция отсчетов

$$\varphi_k(t) = \frac{\sin 2\pi F_m(t - k\Delta t)}{2\pi F_m(t - k\Delta t)} \quad => \quad , \quad \begin{cases} 1 \text{ при } t = k\Delta t \\ 0 \text{ при } t = i\Delta t \ (i \neq k) \end{cases}$$

График этой функции приведен на рис. 19.

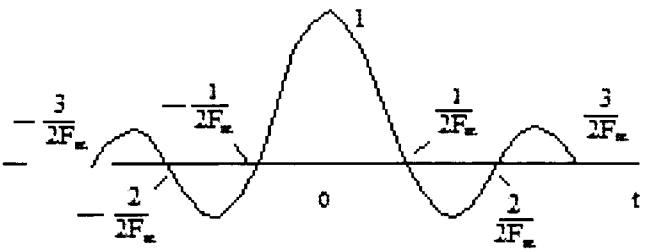


Рис. 19. Функция отсчетов

В этом случае значение ряда (1-27) в момент $t = k\Delta t$ определяется только k – м членом ряда, так как все другие члены в этот момент обращаются в нуль. Как видим свойства функции $\varphi_k(t)$ определяются свойствами функции

$$F(x) = \frac{\sin(x)}{x},$$

равной 1 при $x = 0$ и равной 0 при x кратных π.

Известно, что функция вида $(sin\ x)/x$ представляет собой реакцию идеального фильтра нижних частот (ФНЧ) с частотой среза $F_{ср} = F_m$ на δ - импульс, тождественный выборке функции в момент $k\Delta t$.

Таким образом, чтобы преобразовать идеально точно поток импульсов, описываемый функциями (1-26), в исходный непрерывный сигнал его нужно пропустить через фильтр нижних частот, суммируя при этом все выходные сигналы фильтра.

Теорема Котельникова устанавливает оптимальный метод интерполяции лишь для идеализированной модели непрерывной функции времени (с ограниченным спектром и бесконечной протяженности во времени). Все реальные сообщения имеют конечную протяженность во времени, поэтому такая модель для них, строго говоря, некорректна.

Кроме того, как видно из выражения (1-27), для точного восстановления исходной функции необходимо получить и просуммировать реакции фильтра на входные импульсы на всей оси времени $-\infty \le t \le +\infty$. Такая процедура и сам идеальный ФНЧ физически нереализуемы. Однако, для практических целей, идеально точное восстановление функций не требуется, необходимо лишь восстановление с заданной точностью.

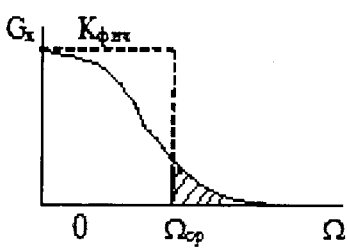


Рис. 20. Ограничение спектра реального сигнала

Для реальных процессов с помощью теоремы Котельникова можно получить только квазиоптимальное решение, если эти процессы до дискретизации пропускать через идеальный ФНЧ с частотой среза $F_{ср} = F_m$. В результате, исход-

ный процесс с неограниченной шириной спектра преобразуется в процесс со спектром ограниченной ширины $F_{ср}$, который может быть восстановлен без искажений с помощью идеального ФНЧ с той же частотой среза (рис. 20).

Однако, ограничение спектра сообщения частотой $F_{ср}=2\pi/\Omega_{ср}$ приводит к погрешности восстановления, относительная величина среднего квадрата которой равна:

$$\delta_u^2 = \frac{\int_{\Omega_{ср}}^{\infty} G_x(\Omega)d\Omega}{\int_0^{\infty} G_x(\Omega)d\Omega} = \frac{\Delta P_x}{P_x}. \qquad (1\text{-}28)$$

Эту величину можно трактовать как отношение средней мощности обрезанной части спектра ("хвоста") ΔP_x к средней мощности всего сообщения P_x. Отсюда, для заданной модели сообщения и заданной погрешности восстановления можно рассчитать частоту дискретизации (опроса) $F_0=1/\Delta t=2F_{ср}$, близкую к потенциально достижимой. Такой расчет служит ориентиром для оценки близости реальных методов интерполирования к квазиоптимальному. Таким образом, теорема Котельникова в теоретических и прикладных задачах играет важную роль как предельное соотношение. На практике частоту опроса часто определяют по формуле

$$F_0 = k_з F_m,$$

где $k_з$ – коэффициент запаса, задаваемый в пределах $1{,}5 \le k_з \le 6$.

1.8. Квантование непрерывных сообщений по уровню

Квантование по уровню как промежуточная операция при преобразовании аналоговых сообщений в цифровую форму широко используется в системах передачи информации, в измерительных системах, при высокоточном автоматическом управлении и контроле, обработке данных и т. д.

Сущность квантования (дискретизации) по уровню состоит в замене бесконечного множества значений, которые может принимать непрерывное сообщение, конечным множеством заранее установленных значений. Такая замена объясняется тем, что в результате действия помех и искажений близкие друг другу сообщения могут быть неразличимы при приеме. Появляется, как бы, некоторая зона неразличимости (неопределенности), в пределах которой нельзя установить истинное значение сообщения.

Этого можно избежать при конечном множестве заранее установленных значений сообщений, в котором интервал между соседними значениями сообщений (шаг квантования) определяется допустимой зоной неразличимости. Далее рассматриваются основные особенности данного преобразования.

При квантовании по уровню непрерывное множество значений функции $x(t)$ в диапазоне от x_{min} до x_{max}, называемое шкалой параметра, преобразуется в дискретное множество x_k – уровней квантования. В результате квантования образуется ступенчатая функция $x_{кв}(t)$. Квантование по уровню может быть равномерным и неравномерным.

При равномерном квантовании шкала параметра $L = (x_{max} - x_{min})$ разбивается на $(x_{max} - x_{min})/\Delta_x$ равных частей – интервалов квантования. Под шагом (интервалом) квантования понимается разность $\Delta_x=x_k - x_{k-1}$, где x_k, x_{k-1} – соседние уровни квантования. Число уровней квантования

$$M = (x_{max} - x_{min})/\Delta_x + 1.$$

При неравномерном квантовании интервал (шаг) квантования изменяется от уровня к уровню по некоторому правилу, учитывающих статистику квантуемых сообщений.

Квантование по уровню может осуществляться двумя способами. При первом способе мгновенное значение функции x(t) в интервале квантования округляется до значения нижнего (или верхнего) уровня квантования. При втором способе значение $x(t)$ округляется до значения ближайшего уровня квантования. Примеры первого (с округлением до нижнего уровня) и второго способов квантования приведены на рис. 21.а и рис. 21.б. соответственно.

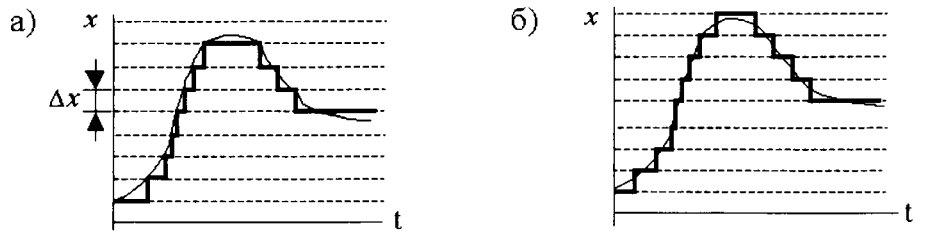


Рис. 21. Способы квантования сигналов

Устройство для квантования сигналов по уровню представляет собой нелинейный элемент, который может иметь два типа амплитудных характеристик в зависимости от используемых способов квантования. Амплитудные характеристики для первого и второго способов приведены соответственно на рис. 22.а и рис. 22.б.

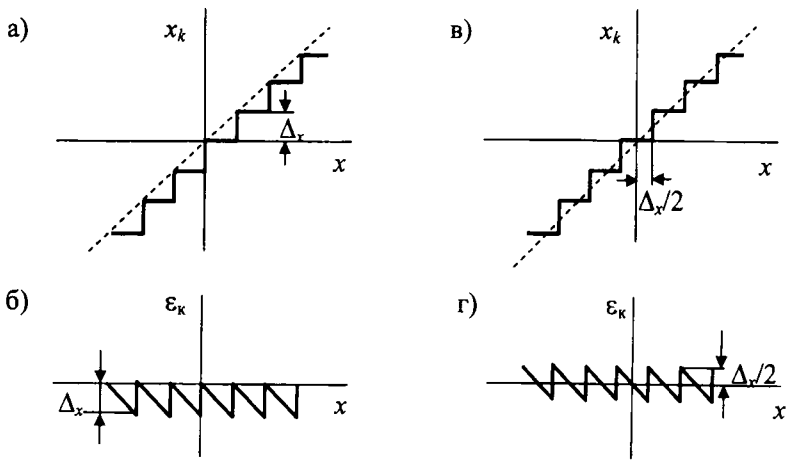


Рис. 22. Амплитудные характеристики и шумы квантования:
а) и б) – при первом способе квантования; в) и г) – при втором способе квантования

Квантование по уровню сопровождается шумами, или погрешностью квантования. Погрешность квантования $\varepsilon_k = x(t_i) - x_k$ связана с заменой истинного значения сигнала $x(t_i)$ его дискретным значением x_k. Максимальная погрешность зависит от способа квантования.

Для первого из рассмотренных способов (рис. 22.б) она равна:
$max\ \varepsilon_k = max\ |x(t_i) - x_k| = \Delta x.$

При втором способе квантования она по абсолютной величине не превосходит половину интервала квантования $0{,}5\Delta x$ (см. рис. 8.г).

Рассматривая значение $x(t_i)$ как случайную реализацию функции $x(t)$ с плотностью распределения вероятностей $w(x)$, можно найти характеристики случайной величины ε_k – погрешности квантования. Как видно из рис. 8, при равномерном квантовании погрешность Δ_x в каждом из интервалов квантования проявляет себя одинаково. Это позволяет найти ее числовые характеристики по любому из отдельно взятых интервалов квантования, например κ – му.

При достаточно малых интервалах квантования Δ_x по сравнению с диапазоном изменения сигнала можно пренебречь неравномерностью функции $w(x)$ в пределах k – го шага квантования (рис. 23).

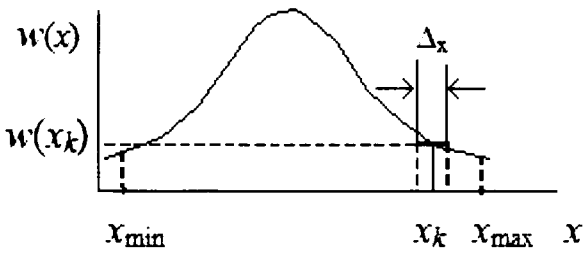


Рис. 23. Плотность вероятностей распределения мгновенных значений квантуемого сигнала

При этом плотность распределения $w(x)$ постоянна и равна $w(x_k)$, т.е. подчиняется равномерному закону распределения истинных (мгновенных) значений $x(t)$, а соответственно, и их отклонений ε_k от дискретного значения x_k в k – м интервале квантования. Поскольку все возможные значения погрешности ε_k лежат в рассматриваемом интервале $(x_k-\Delta x/2,\ x_k+\Delta x/2)$, то, как следует из теории вероятностей,

$$w(\varepsilon_k)d_x = w(x_k)\Delta_x = \int_{x_k-\Delta x/2}^{x_k+\Delta x/2} w(x)dx = 1. \qquad (1\text{-}29)$$

Следовательно, плотность вероятностей распределения погрешностей квантования (рис. 23) с достаточным приближением находится из равенства:

$$w(\varepsilon_k) = 1/\Delta x.$$

Далее необходимо определить математическое ожидание и дисперсию шума квантования для обоих способов равномерного квантования (см. рис. 22, эпюры а и б).

Для первого из них (с округлением значения x до нижней границы интервала квантования) плотность распределения $w(\varepsilon_k)$ описывается равномерным законом в интервале $-\Delta_x \le \varepsilon_k \le 0$ с математическим ожиданием $m_\varepsilon = -\Delta x/2$ (рис. 24.а).

Действительно, математическое ожидание погрешности квантования

$$m_\varepsilon = \int_{-\Delta x}^{0} \varepsilon_\kappa w(\varepsilon_\kappa)d\varepsilon_\kappa = \frac{1}{\Delta x}\int_{-d_x}^{0} \varepsilon_\kappa d\varepsilon_\kappa = -\frac{\Delta x}{2}. \qquad (1\text{-}30)$$

Тогда дисперсия погрешности квантования

$$\sigma_\varepsilon^{\,2} = \int_{-d_x}^{0} (\varepsilon_\kappa - m_\varepsilon)w(\varepsilon_\kappa)d\varepsilon_\kappa = \int_{-d_x}^{0} (\varepsilon_\kappa + \frac{\Delta x}{2})^2 \frac{1}{\Delta x} d\varepsilon_\kappa = \frac{\Delta x^2}{12}. \qquad (1\text{-}31)$$

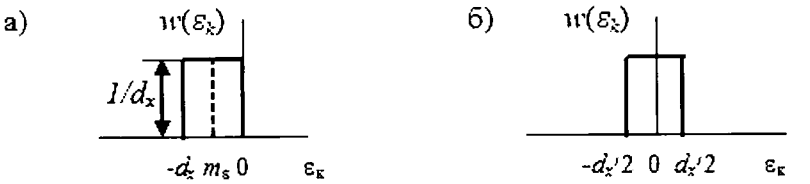


Рис. 24. Плотности вероятностей шума квантования: а) – для первого способа; б) – для второго способа

Для второго способа (с округлением значения x до ближайшей границы интервала квантования) плотность распределения $w(\varepsilon_к)$ описывается равномерным законом в интервале $-d_x/2 \le \varepsilon_к \le d_x/2$ с математическим ожиданием $m_\varepsilon = 0$ (рис. 24.б).

Дисперсия погрешности квантования

$$\sigma_\varepsilon^2 = \int_{-\Delta x/2}^{\Delta x/2} \varepsilon_к^2 w(\varepsilon_к) d\varepsilon_к = \int_{-\Delta x/2}^{\Delta x/2} \frac{1}{d} \varepsilon_к^2 d\varepsilon_к = \frac{\Delta x^2}{12}. \qquad (1\text{-}32)$$

Среднеквадратическое значение погрешности квантования

$$\sigma_\varepsilon = \frac{\Delta x}{2\sqrt{3}}. \qquad (1\text{-}33)$$

Значение среднеквадратической погрешности квантования $\gamma_к$, приведенной к шкале $L = \Delta x(M\text{-}1)$, равно

$$\gamma_k = \frac{\sigma_\varepsilon}{L} = \frac{1}{(M-1)2\sqrt{3}}, \text{ откуда } M = \frac{1}{\gamma_k \cdot 2\sqrt{3}} + 1.$$

Квантование лежит в основе цифровых методов передачи: передается не значение параметра, а соответствующий ему номер квантованного уровня. Если номер уровня кодировать двоичным кодом, то необходимое число n разрядов двоичного кода номера уровня квантования выбирается из условия

$$n \ge [log_2 M],$$

где $[\cdot]$ - символ округления до ближайшего целого числа (справа). Если задана погрешность квантования $\gamma_к$, то необходимое минимальное число разрядов

$$n = \left[\log_2 \left(\frac{1}{\gamma_k \cdot 2\sqrt{3}} + 1 \right) \right]. \qquad (1\text{-}34)$$

Следует отметить, что при первом способе квантования имеет место систематическая (методическая) погрешность, равная математическому ожиданию $m_\varepsilon = \pm\Delta x/2$. Она может быть устранена введением поправки $\varepsilon_п = \pm\Delta x/2$.

Для передачи номера квантованного уровня непрерывной величины часто используют коды, обеспечивающие наибольшую точность преобразования, например, двоичный код Грея (см. табл. 5). Особенность кода Грея состоит в том, что при аналого-кодовом преобразовании сигнала возникает погрешность квантования, и она уменьшается (по крайней мере, не увеличивается), если каждая последующая комбинация кода отличается от предыдущей только в одном разряде, что сводит ошибку к единице в младшем разряде кода.

Таблица 5.

Номер уровня	Код	Номер уровня	Код
0	0000	8	1100
1	0001	9	1101
2	0011	10	1111
3	0010	11	1110
4	0110	12	1010
5	0111	13	1011
6	0101	14	1001
7	0100	15	1000

2. Персональный компьютер. Основные понятия и определения

Компьютер представляет собой устройство, способное исполнять четко определенную последовательность операций, предписанную программой. Понятие «компьютер» является более широким, чем «электронно-вычислительная машина» (ЭВМ). В ЭВМ явный акцент делается на вычисления. Важнейшими характеристиками ЭВМ являются *быстродействие* и *производительность*. Эти характеристики тесно связаны, тем не менее, их не следует смешивать. *Быстродействие* характеризуется числом определенного типа команд, выполняемых ЭВМ в одну секунду. *Производительность* – это объем работ (например, число стандартных программ), выполняемый ЭВМ в единицу времени. Определение характеристик быстродействия и производительности представляет собой очень сложную инженерную и научную задачу, до настоящего времени не имеющую единых подходов и методов решения.

Персональный компьютер (ПК) характерен тем, что им может пользоваться один человек, не прибегая к помощи бригады обслуживающего персонала и не отводя под него специального зала с поддержанием климата, мощной системой электропитания и прочими атрибутами больших вычислительных машин. Этот компьютер обычно сильно ориентирован на интерактивное взаимодействие с одним пользователем (в играх иногда и с двумя), причем взаимодействие происходит через множество сред общения – от алфавитно-цифрового и графического диалога с помощью дисплея, клавиатуры и мыши до устройств виртуальной реальности, в которой пока не задействованы, наверное, только запахи. Когда используется аббревиатура PC (Personal Computer), подразумевается ПК, совместимый с самым массовым семейством персональных компьютеров фирмы IBM и их клонов. Конечно же, это не единственное в мире семейство — есть множество и других достойных компьютерных линий, но данная книга посвящена именно IBM PC-совместимым персональным компьютерам. ПК может быть использован и коллективно: возможности многих компьютеров этого семейства позволяют использовать их и в качестве серверов в сетях или локальных многотерминальных системах. Таким образом, можно объяснить словосочетание PC-сервер, которое неявно предполагает повышенную мощность (скорость вычислений, объем оперативной и внешней памяти) и особое конструктивное исполнение (просторный

корпус) компьютера. Словосочетание ПК-сервер уже звучит странновато, хотя в одноранговых сетях и этому словосочетанию можно найти объяснение — персональный компьютер может предоставлять свои ресурсы (например, дисковое пространство, принтеры или модемы) другим компьютерам, для которых он будет являться невыделенным сервером. Попутно следует отметить и термин рабочая станция (Workstation, WS), в который может быть вложено два значения. В компьютерной сети рабочей станцией называют компьютер пользователя (как противоположность серверу). Однако рабочая станция может быть и отдельно стоящим (Standalone Computer), но особенно мощным компьютером (его подключение к сети, конечно же, не исключается). В этом случае часто подразумевается архитектура, отличающаяся от IBM PC (например, компьютер на RISC-процессоре). Для мощного IBM PC - совместимого компьютера применяют англоязычный термин High End PC, которому короткого русского аналога пока нет.

Любой компьютер имеет три основные составные части: *процессор*, *память*, *периферийные устройства*. Они взаимодействуют между собой с помощью шин, стандартизация и унификация которых делает архитектуру компьютеров открытой.

Надежность – это способность компьютера при определенных условиях выполнять требуемые функции в течение заданного времени (стандарт ISO – Международная организация стандартов). Высокая надежность закладывается в процессе производства компьютера. Переход на новую элементную базу – сверхбольшие интегральные схемы (СБИС) – резко сокращает число используемых интегральных схем, а значит, и число их соединений друг с другом. Хорошо продуманная компоновка компьютера с обеспечением требуемых режимов работы (охлаждение, защита от пыли) также способствует повышению надежности. Модульный принцип построения позволяет легко проверять и контролировать работу всех устройств, проводить диагностику и устранять неисправности.

Точность – возможность различать почти равные значения величин (стандарт ISO). Точность получения результатов обработки, в основном, определяется разрядностью, которая в зависимости от класса компьютера может составлять 32, 64 и 128 двоичных разрядов.

Достоверность – свойство информации быть правильно воспринятой. Достоверность характеризуется вероятностью получения безошибочных результатов. Заданный уровень достоверности обеспечивается аппаратно-программными средствами контроля самого компьютера. Возможны методы контроля достоверности путем решения эталонных задач и повторных расчетов. В особо ответственных случаях проводятся контрольные решения на других компьютерах и сравнение полученных результатов.

2.1. Процессор

Основным принципом построения компьютеров является *программное управление*. В его основе лежит представление алгоритма решения любой задачи в виде программы вычислений. *Программа для ЭВМ* – упорядоченная последовательность команд, подлежащих выполнению. Каждая команда содержит указания на конкретную выполняемую операцию, местонахождение (адреса) операндов и ряд служебных признаков.

Операнды – переменные, значения которых участвуют в операциях преобразования данных. Список (массив) всех переменных (входных данных, промежуточных значений и результатов вычислений) является еще одним неотъем-

лемым элементом любой программы.

Для доступа к программам, командам и операндам используются их *адреса.* В качестве адресов выступают номера ячеек памяти, предназначенных для хранения данных. Информация (командная, числовая, текстовая и графическая) кодируется двоичными цифрами «0» и «1». Таким образом, дальнейшая обработка ее заключается в преобразовании последовательности бит.

Последовательность бит в формате, имеющая определенный смысл, представлена *полем.* Например, в каждой команде программы различают поле кода операций, поле адресов операндов. Применительно к числовой информации выделяют знаковые разряды, значащие разряды чисел, старшие и младшие разряды чисел.

Каждый тип информации имеет свои *форматы* – структурные единицы информации, которые кратны байту, т.е. состоят из целого числа байтов. Последовательность, состоящая из определенного, принятого для данной ЭВМ числа байтов, называется *словом.* Для компьютера размер слова составляет 2 или 4 байта. В качестве элементов информации различают также полуслово, двойное слово и др.

Процессор является основным «мозговым» узлом, в задачу которого входит выполнение программного кода, находящегося в памяти. В настоящее время под словом «процессор» подразумевают микропроцессор — микросхему, которая, кроме собственно процессора, может содержать и другие узлы, например, память. Процессор в определенной последовательности выбирает из памяти инструкции и исполняет их. Инструкции процессора предназначены для передачи, обработки и анализа данных, расположенных в пространствах памяти портов ввода/вывода, а также организации ветвлений и переходов в вычислительном процессе. В компьютере обязательно должен присутствовать центральный процессор (CPU — Central Processing Unit), который исполняет основную программу. В многопроцессорной системе функции центрального процессора распределяются между несколькими обычно идентичными, процессорами для повышения общей производительности системы. В помощь центральному процессору в компьютер часто вводят сопроцессоры, ориентированные на эффективное исполнение каких-либо специфических инструкций. Широко распространены математические сопроцессоры, эффективно обрабатывающие числовые данные в формате с плавающей точкой; графические сопроцессоры, выполняющие геометрические построения и обработку графических изображений; сопроцессоры ввода-вывода, разгружающие центральный процессор от несложных, но многочисленных операций взаимодействия с периферийными устройствами. Возможны и другие сопроцессоры, однако все они несамостоятельны. Выполнение основного вычислительного процесса осуществляется центральным процессором, который, в соответствии с программой, выдает задания сопроцессорам на исполнение отдельных инструкций.

2.2. Память компьютера

Другой важнейшей характеристикой компьютера является *емкость запоминающих устройств.* Она измеряется количеством структурных единиц информации, которые одновременно можно разместить в памяти машины. Этот показатель позволяет определить, какой набор программ и данных может быть одновременно размещен в памяти.

Наименьшей единицей информации является *бит* – одна двоичная цифра. Как правило, емкость памяти оценивается в более крупных единицах измерения –

байтах (байт равен восьми битам). Более крупными единицами измерения служат:

1 Кбайт = 2^{10} байта = 1024 байта;

1 Мбайт = 2^{10} Кбайт = 2^{20} байта = 1048576 байта;

1 Гбайт = 2^{10} Мбайт = 2^{20} Кбайт = 2^{30} байта = 1073741824 байта.

Обычно, отдельно характеризуют емкость оперативной памяти и емкость внешней памяти. Современные персональные компьютеры могут иметь емкость оперативной памяти равную 64 – 512 Мбайтам и даже больше. Этот показатель очень важен для определения, какие программные пакеты и их приложения могут одновременно обрабатываться в машине.

Память компьютера предназначена для кратковременного и долговременного хранения информации — кодов команд и данных. Информация в памяти хранится в двоичных кодах, каждый бит — элементарная ячейка памяти — может принимать значение «0» или «1». Каждая ячейка памяти имеет свой адрес, однозначно ее идентифицирующий в определенной системе координат. Минимальной адресуемой единицей хранения информации в памяти обычно является байт, состоящий, как правило, из 8 бит.

Существуют процессоры и компьютеры с разрядностью обрабатываемого слова не кратной 8 (например, 5, 7, 9...), и их байты не восьми битные, но в мире ПК таких мало. Также в некоторых системах (обычно коммуникационных), совокупность восьми соседних бит данных называют октетом. Название «октет» обычно подразумевает, что эти 8 бит, не имеют явного адреса, а характеризуются только своим местоположением в длинной цепочке бит.

Исторически сложилось деление памяти на внутреннюю и внешнюю. Под внутренней подразумевалась память, расположенная внутри системного блока (или плотно к нему примыкающая). Сюда входила и электронная, и магнитная память без подвижных носителей. Внешняя память представляла собой отдельные устройства с подвижными носителями — накопители на магнитных дисках и ленте. Со временем все устройства компьютера удалось разместить в один небольшой корпус, и прежнюю классификацию памяти применительно к ПК можно переформулировать так:

- внутренняя — электронная (полупроводниковая) память, устанавливаемая на системной плате или на платах расширения;
- внешняя — память, реализованная в виде устройств с различными принципами хранения информации и, обычно, с подвижными носителями. В настоящее время сюда входят устройства магнитной (дисковой и ленточной) памяти, оптической и магнитооптической памяти. Устройства внешней памяти могут размещаться как в системном блоке компьютера, так и в отдельных корпусах, достигающих иногда размеров небольшого шкафа.

Емкость внешней памяти зависит от типа носителя. Так, емкость одной дискеты составляет 1,2; 1,4; 2,88 Мбайта в зависимости от типа дисковода и характеристик дискет. Емкость жесткого диска и дисков DVD может достигать нескольких десятков Гбайт, емкость компакт-диска (CD-ROM) – сотни Мбайтов (640 Мбайт и выше). Емкость внешней памяти характеризует объем программного обеспечения и отдельных программных продуктов, которые могут устанавливаться в ЭВМ. Например, для установки операционной среды Windows 2000 требуется объем памяти жесткого диска более 600 Мбайт и не менее 64 Мбайт оперативной памяти ЭВМ.

Для процессора непосредственно доступной является внутренняя память,

доступ к которой осуществляется по адресу, заданному программой. Для внутренней памяти характерен одномерный (линейный) адрес, который представляет собой одно двоичное число определенной разрядности. Внутренняя память подразделяется на оперативную, информация в которой может изменяться процессором в любой момент времени, и постоянную, информацию которой процессор может только считывать. Обращение к ячейкам оперативной памяти может происходить в любом порядке, причем как по чтению, так и по записи, и оперативную память называют памятью с произвольным доступом — Random Access Memory (RAM), — в отличие от постоянной памяти (Read Only Memory, ROM). Внешняя память адресуется более сложным образом - каждая ее ячейка имеет свой адрес внутри некоторого блока, который, в свою очередь, имеет многомерный адрес. Во время физических операций обмена данными блок может быть считан или записан только целиком. В случае одиночного дискового накопителя адрес блока будет трехмерным: номер поверхности (головки), номер цилиндра и номер сектора. В современных накопителях этот трехмерный адрес часто заменяют линейным номером - логическим адресом блока, а его преобразованием в физический адрес занимается внутренний контроллер накопителя. Поскольку дисковых накопителей в компьютере может быть множество, в адресации дисковой памяти участвует номер накопителя, а также номер канала интерфейса. С такой сложной системой адресации процессор справляется только с помощью программного драйвера, в задачу которого в общем случае, входит копирование некоторого блока данных из оперативной памяти в дисковую и обратно. Дисковая память является внешней памятью с прямым доступом, что подразумевает возможность обращения к блокам (но не ячейкам) в произвольном порядке. Память на ленточных носителях имеет самый неудобный метод доступа — последовательный. В ней информация хранится также в виде блоков фиксированной или переменной длины, и в пределах одного носителя эти блоки имеют последовательные адреса. Для доступа к какому-либо блоку устройство должно найти некоторый маркер начала ленты (тома), после чего последовательным холостым чтением блока за блоком дойти до требуемого места и только тогда производить сами операции обмена данными. С такими неудобствами мирятся только потому, что ленточные носители являются самым дешевым хранилищем для больших объемов информации, к которой не требуется оперативного доступа.

Для подсистемы памяти важными параметрами являются следующие:

- объем хранимой информации. Нет необходимости объяснять, что чем он больше, тем лучше. Максимальный (в принципе неограниченный) объем хранят ленточные и дисковые устройства со сменными носителями, за ними идут дисковые накопители, и завершает этот ряд оперативная память;

- время доступа — усредненная задержка начала обмена полезной информацией относительно появления запроса на данные. Минимальное время доступа имеет оперативная память, за ней идет дисковая и после нее — ленточная;

- скорость обмена при передаче потока данных (после задержки на время доступа). Максимальную скорость обмена имеет оперативная память, за ней идет дисковая и после нее — ленточная;

- удельная стоимость хранения единицы данных — цена накопителя (с носителями), отнесенная к единице хранения (байту или мегабайту). Минимальную стоимость хранения имеют ленточные устройства со сменными носителями, их догоняют дисковые накопители, а самая дорогая — оперативная память.

Кроме этих параметров имеется и ряд других характеристик — энергоне-

зависимость (способность сохранения информации при отключении внешнего питания), устойчивость к внешним воздействиям, время хранения, конструктивные особенности (размер, вес) и т.п. У каждого типа памяти имеются различные реализации со своими достоинствами и недостатками.

Внутренняя и внешняя память используются различными способами. Внутренняя (оперативная и постоянная) память является хранилищем программного кода, который непосредственно может быть исполнен процессором. В ней же хранятся и данные, также непосредственно доступные процессору (а следовательно, и исполняемой программе). Внешняя память обычно используется для хранения файлов, содержимое которых может быть произвольным. Процессор (программа) имеет доступ к содержимому файлов только опосредованно через отображение их (полное или частичное) в некоторой области оперативной памяти. Исполнить программный код или обратиться к данным непосредственно на диске процессор не может в принципе. То же относится, естественно, и к ленточной памяти.

Однако на практике дисковая и оперативная память переплетаются сложным образом. Главный недостаток дисковой памяти - большое время доступа и низкая скорость обмена - устраняется при использовании виртуального диска, представляющего собой своеобразно используемую область оперативной памяти. В этой области хранятся файлы, и, с точки зрения операционной системы (и, тем более, прикладной программы), она выглядит как обычный, но очень быстрый диск. Конечно же, его объем ограничен, и этот объем вычитается из объема физически установленной памяти, доступной процессору в качестве обычной оперативной. Кроме того, виртуальный диск, в отличие от реального, не является энергонезависимым. Более того, информация на нем не переживет даже перезагрузки операционной системы. Но, несмотря на эти ограничения, виртуальный диск во многих случаях может повысить эффективность работы компьютера при интенсивном дисковом обмене. В операционной системе MS-DOS (и Windows 95) виртуальный диск реализуется загрузкой программного драйвера RAMDRIVE.SYS (в некоторых версиях - VDISK.SYS), похожие средства имеются и в других операционных системах. Другим способом решения проблем быстродействия дисковой памяти за счет оперативной является кэширование дисков. То есть хранение образов последних из использованных блоков дисковой памяти в оперативной в надежде на то, что к ним вскоре будет следующий запрос, который удастся удовлетворить из памяти. В MS-DOS кэшированием дисков занимается загружаемый драйвер SMARTDRV.EXE, в Windows 95 кэширование встроено в операционную систему (ОС). Даже и без этого драйвера «неглубокое» кэширование выполняет операционная система, и этим процессом можно управлять с помощью строки «BUFFERS = xxx» файла CONFIG.SYS. Если затребованный блок с диска уже находится в одном из буферов, ОС не будет «беспокоить» диск, а удовлетворит запрос из буфера. Чем больше параметр xxx, тем больше блоков может держать ОС в оперативной памяти, но область памяти для буферов, естественно, уменьшает объем памяти, доступной программам.

Конструктивно достижимый объем оперативной памяти во много раз меньше, чем у дисковой памяти (до сих пор это было справедливо на всех ступенях технического прогресса). Решить проблему увеличения объема оперативной памяти за счет дисковой позволяет виртуальная память, которую можно считать кэшированием оперативной памяти на диске.

Суть ее заключается в том, что программам предоставляется виртуальное

пространство оперативной памяти, по размерам превышающее объем физически установленной оперативной памяти. Это виртуальное пространство разбито на страницы фиксированного размера, а в физической оперативной памяти в каждый момент времени присутствует только часть из них. Остальные страницы хранятся на диске, откуда операционная система может их «подкачать» в физическую на место предварительно выгруженных на диск страниц. Для прикладной программы этот процесс прозрачен (если только она не критична ко времени обращения к памяти). Для пользователя этот процесс заметен по работе диска даже в тот момент, когда не требуется обращение к файлам. Расплатой за почти безмерное увеличение объема доступной оперативной памяти является снижение средней производительности памяти и некоторый расход дисковой памяти на, так называемый, файл подкачки (Swap File). Естественно, размер виртуальной памяти не может превышать размера диска (файл подкачки на нескольких дисках обычно не размещают). Виртуальная память реализуется операционными системами (и оболочками) защищенного режима (например, OS/2, MS Windows) на основе аппаратных средств процессоров класса не ниже 286, а наиболее эффективно — 32-разрядных процессоров 386 и старше.

Таково в общих чертах устройство памяти компьютера.

В общем случае в подсистему памяти обязательно входит оперативная память и энергонезависимая память, хранящая, по крайней мере, программу первоначальной загрузки компьютера. Дисковая память как таковая может и отсутствовать. Однако часто в понятие «диск» или «дисковое устройство» (Disk Device, Disk Drive) вкладывают значение «устройство внешней памяти прямого доступа». Так, например, виртуальный диск в ОЗУ и электронный диск на флэш-памяти отнюдь не имеют круглых, а тем более вращающихся деталей. Внешняя память с прямым доступом в том или ином виде — будь то действительно дисковые накопители, флэш-диск или сетевой диск, отображающий часть диска физически значительно удаленного компьютера-сервера, — является обязательным атрибутом персонального компьютера. Без внешней памяти компьютер вырождается в узкоспециализированное устройство с ограниченным набором функций (например, эмуляции терминала или интерпретатора языка Basic), «зашитых» в его постоянную память. Ленточная память является необязательной и используется обычно для хранения архивов.

2.3. Периферийные устройства

Периферийные устройства связывают компьютер с внешним миром, и без них он был бы «вещью в себе». Список устройств, делающих компьютер «вещью для нас», практически неограничен.

Сюда входят *устройства ввода* информации в ПК:

- *Клавиатура* – устройство для ввода числовой, текстовой и управляющей информации в ПК;
- *Манипуляторы* (устройства указания): *джойстик* – рычаг, *мышь*, *трекбол* – шар в оправе, *световое перо* и др. – для ввода графической информации на экран дисплея путем управления движением курсора по экрану с последующим кодированием координат курсора и вводом их в ПК;
- *Сенсорные экраны* – для ввода отдельных элементов изображения, программ или команд с полиэкрана дисплея в ПК;
- *Графические планшеты (диджитайзеры)* – для ручного ввода графической информации, изображений путем перемещения по планшету специального

пера; при перемещении пера автоматически выполняются считывание координат его местоположения и ввод этих координат в ПК;

- *Сканеры* (читающие автоматы) – для автоматического считывания с бумажных носителей и ввода в ПК машинописных текстов, графиков, рисунков, чертежей.

К *устройствам вывода информации* относятся:

- *Принтеры* – печатающие устройства для регистрации информации на бумажный носитель;
- *Графопостроители* (плоттеры) – для вывода графической информации (графиков, чертежей, рисунков) из ПК на бумажный носитель;
- *Другие устройства вывода* - алфавитно-цифровые и графические мониторы, акустические системы и прочие устройства в великом множестве их разновидностей.

Устройства *связи и телекоммуникации* используются для связи с приборами и другими средствами автоматизации:

- *Коммуникационные устройства* - модемы, адаптеры локальных и глобальных сетей. Сюда же часто относят дисковые и ленточные устройства хранения информации, но по выполняемым функциям их, все-таки, лучше включать в подсистему памяти.
- *Устройства связи с объектом.* К компьютеру можно подключать *датчики* и *исполнительные устройства* технологического оборудования, различные приборы. В общем, все, что, в конечном итоге, может вырабатывать электрические сигналы и (или) ими управляться.

Все периферийные устройства подключаются к компьютеру через внешние интерфейсы или с помощью специализированных адаптеров или контроллеров, встраиваемых в системную плату или размещаемых на платах (картах) расширения. Адаптер является средством сопряжения какого-либо устройства с какой-либо шиной компьютера. Контроллер служит тем же целям сопряжения, но при этом подразумевается его некоторая активность — способность к самостоятельным действиям после получения команд от обслуживающей его программы. Сложный контроллер может иметь в своем составе и собственный процессор. На эти тонкости терминологии не всегда обращают внимание, и понятия «адаптер» и «контроллер» считают почти синонимами. Все внешние интерфейсы компьютера, естественно, тоже имеют свои адаптеры или контроллеры. Для взаимодействия с программой (с помощью процессора или сопроцессоров) адаптеры и контроллеры периферийных устройств обычно имеют регистры ввода и вывода, которые могут располагаться либо в адресном пространстве памяти, либо в специальном пространстве портов ввода/вывода. Кроме того, используются механизмы аппаратных прерываний для сигнализации программе о событиях, происходящих в периферийных устройствах. Для обмена информацией с периферийными устройствами применяется и механизм прямого доступа к памяти DMA (Direct Memory Access). Контроллер DMA можно считать простейшим сопроцессором ввода/вывода, разгружающим центральный процессор от рутинных операций обмена.

2.4. Роль программного обеспечения

Таково, в общих чертах, устройство компьютера (естественно, подразумевается и наличие корпуса с блоком питания). Однако набор перечисленных выше устройств не может функционировать без программного обеспечения, которое в компьютере имеет многоуровневую организацию. Часть программного обеспече-

ния хранится в постоянной (энергонезависимой) памяти и обеспечивает тестирование и запуск при включении, загрузку операционной системы и связь операционной системы с аппаратными средствами компьютера. Эта часть называется базовой системой ввода/вывода BIOS (Basic Input-Output System). Следующий уровень — операционная система, основным назначением которой является загрузка прикладных программ и предоставление им некоторых сервисов. И, наконец, верхний уровень — прикладное программное обеспечение, ради исполнения которого и создавался компьютер. Именно возможность загрузки любой прикладной программы в совокупности с неограниченным ассортиментом периферийных устройств и позволяет считать персональный компьютер универсальным инструментом с неограниченными возможностями.

В любом компьютере имеются устройства ввода информации, с помощью которых пользователи вводят программы решаемых задач и данные к ним. Сначала введенная информация полностью или частично запоминается в оперативном запоминающем устройстве, а затем переносится во внешнее запоминающее устройство, где преобразуется в специальный объект – файл. *Файл* – это имеющий имя информационный массив (программа, данные, текст и т.п.), размещаемый во внешней памяти и рассматриваемый как неделимый информационный объект при пересылке и обработке.

Организация обмена информационными потоками между запоминающими устройствами различных уровней позволяет рассматривать иерархию памяти как *абстрактную виртуальную память*. Децентрализация управления структуры ЭВМ позволила перейти к более сложным *многопрограммным (мультипрограммным) режимам*. При этом в компьютере одновременно может обрабатываться несколько программ пользователей. В компьютере, имеющем один процессор, многопрограммная обработка является кажущейся. Она предполагает параллельную работу отдельных устройств, задействованных в вычислениях по различным задачам пользователей. Например, компьютер может производить распечатку каких-либо документов и принимать сообщения, поступающие по каналам связи.

В компьютерах или вычислительных системах, имеющих несколько процессоров обработки, многопрограммная работа может быть более глубокой. Автоматическое управление вычислениями предполагает усложнение структуры за счет включения в ее состав систем и блоков, исключающих возможность возникновения взаимных помех и ошибок (системы прерываний и приоритетов, защиты памяти).

2.5. Основные этапы развития ЭВМ

На пути развития электронной вычислительной техники можно выделить ряд поколений ЭВМ, отличающихся элементной базой, функционально-логической организацией, конструктивно-технологическим исполнением, программным обеспечением, техническими и эксплуатационными характеристиками, степенью доступа пользователей к ЭВМ.

Первое поколение, 50-е годы: ЭВМ на электронных вакуумных лампах. В качестве устройства ввода-вывода сначала использовалась стандартная телеграфная аппаратура (телетайпы, ленточные перфораторы, трансмиттеры и др.), а затем электромеханические запоминающие устройства на магнитных лентах, барабанах и дисках. Машины первого поколения имели внушительные размеры, потребляли большую мощность, имели сравнительно малое быстродействие, малую емкость оперативной памяти, невысокую надежность работы и недостаточно развитое программное обеспечение. Однако в ЭВМ этого поколения были заложены осно-

вы логического построения машин и продемонстрированы возможности цифровой вычислительной техники.

Второе поколение, 60-е годы: ЭВМ на дискретных полупроводниковых приборах (транзисторах). В отличие от ламповых ЭВМ, транзисторные машины обладали большими быстродействием, емкостью оперативной памяти и надежностью. Существенно уменьшились размеры, масса и потребляемая мощность. Значительным достижением явилось применение печатного монтажа. Повысилась надежность электромеханических устройств ввода-вывода, удельный вес которых возрос. Особенность машин второго поколения – их дифференциация по применению. Появились машины для решения научно-технических и экономических задач, для управления производственными процессами и различными объектами (управляющие машины).

Третье поколение, 70-е годы: ЭВМ на полупроводниковых интегральных схемах с малой и средней степенью интеграции (сотни тысяч транзисторов в одном корпусе). В машинах третьего поколения существенно расширены возможности непосредственного доступа к ним абонентов, находящихся на различных, в том числе и значительных (десятки и сотни километров), расстояниях. Развитые операционные системы обеспечивают управление работой ЭВМ в различных режимах: пакетной обработки, разделения времени, запрос-ответ и др. Удобство общения абонента с машиной достигается за счет развитой сети абонентских пунктов, снабженных периферийными устройствами ввода-вывода и связанных с ЭВМ информационными каналами связи.

Четвертое поколение, 80-е годы: ЭВМ на больших и сверхбольших интегральных схемах – микропроцессорах (десятки тысяч - миллионы транзисторов в одном кристалле). Высокая степень интеграции способствует увеличению плотности компоновки электронной аппаратуры, повышению ее надежности и быстродействия, снижению стоимости.

Пятое поколение, 90-е годы: ЭВМ с многими десятками параллельно работающих микропроцессоров, позволяющих строить эффективные системы обработки знаний; ЭВМ на сверхсложных микропроцессорах с параллельно-векторной структурой, одновременно выполняющих десятки последовательных команд программы.

ЭВМ четвёртого и пятого поколения широко представлены персональными компьютерами, которые, как говорят, совместимы с IBM PC. Эти компьютеры образуют самостоятельное родословное дерево и, в свою очередь, делятся на модели (классы) со следующими характерными особенностями:

– IBM PC первой модели имел процессор Intel 8088, адресуемую память 1 Мбайт, шину расширения ISA (8 бит), накопители на гибких дисках (НГМД) до 360 Кбайт.

– IBM PC/XT (eXtended Technology — расширенная технология) — появились винчестеры — накопители на жестких дисках (НЖМД) - и возможность установки математического сопроцессора Intel 8087.

– IBM PC/AT (Advanced Technology — прогрессивная или «продвинутая» технология): процессор Intel 80286/80287, адресуемая память 16 Мбайт, шина ISA 16 бит, НГМД 1,2 и 1,44 Мбайт, НЖМД.

В настоящее время класс машин AT развивается в нескольких направлениях:

- 16-разрядный процессор заменен на 32-разрядный (386 и старше),
- память адресуется в пространстве до 4 и даже 32 Гбайт,

- применяются более эффективные шины расширения (EISA, VLB, PCI) с сохранением ISA 16 бит как дешевой шины для обеспечения совместимости со старыми адаптерами,
- расширяется состав устройств, имеющих системную поддержку на уровне BIOS.

Более совершенными, по сравнению с предыдущими поколениями процессоров фирмы Intel, являются процессоры Intel Pentium. Данные процессоры изготавливаются по более совершенной и перспективной технологии, чем у 486. Серия процессоров Pentium состоит из изделий с частотами 100, 120, 133, 150, 166, 200 МГц. Правда, следует отметить, что процессоры с частотами 60-133 МГц фирмой Intel уже не выпускаются. С 1997 года рекламируются, выпускаются и активно применяются еще более мощные и современные процессоры - Pentium Pro, Pentium MMX, Pentium II, Celeron, Pentium II Xeon, Pentium III, Pentium III Xeon, которые получили название семейства **P6**. Серия Pentium Pro состоит из изделий с частотами 150, 166, 180, 200 МГц, Pentium MMX - 166, 200, 233 МГц, Pentium II -233, 266, 300, 333, 350, 400 МГц, Celeron – 500 МГц, Pentium III - 800 МГц. Процессор Pentium Pro предназначен, в основном, для использования в серверах. Pentium MMX - в программах с мультимедийными приложениями. Наибольшая эффективность данных процессоров - в графических программах с использованием расширенного набора команд. Процессорами семейства P6 в настоящее время комплектуется 85% выпускаемых в мире ПК.

Шестое и последующие поколения: оптоэлектронные ЭВМ с массовым параллелизмом и нейронной структурой – с распределенной сетью большого числа (десятки тысяч) несложных микропроцессоров, моделирующих архитектуру нейронных биологических систем.

2.6. Основные определения

Дадим определение некоторым терминам, относящимся к аппаратным средствам современных компьютеров. Поскольку персональные компьютеры, увы, имеют иностранное происхождение (опала, в которую попала кибернетика в нашей стране, не позволила удержать приоритеты в этой области), приходится мириться с рядом иностранных слов, вошедших в технический русский язык в виде, даже не всегда правильных транслитераций. Однако, во многих случаях для профессионалов они понятнее, поскольку много информации по данной теме черпается исключительно из зарубежных, чаще всего, англоязычных источников. Далее рассматриваются варианты названий основных элементов компьютера.

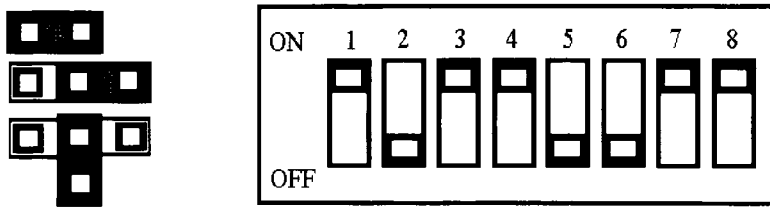
Системной платой (System Board), или *материнской платой* (Mother Board), называют основную печатную плату, на которой устанавливается процессор, оперативная память, ROM BIOS и некоторые другие системные компоненты.

Платой расширения, или *картой расширения* (Expansion Card), называют печатную плату с краевым разъемом, устанавливаемую в слот расширения. Карты расширения, привносящие в PC какой-либо дополнительный интерфейс, называют *интерфейсными картами* (Interface Card). Поскольку интерфейсная карта представляет собой «приспособление» для подключения какого-либо устройства, к ней применимо и название *адаптер* (Adapter). К примеру, дисплейный адаптер (Display adapter) служит для подключения дисплея-монитора. Адаптер и интерфейсная карта, практически, синонимы, и, например, NIC (Network Interface Card - карта сетевого интерфейса) часто переводится как адаптер ЛВС (локальной вычислительной сети).

Слот (Slot) представляет собой щелевой разъем, в который устанавливается какая-либо печатная плата. *Слот расширения* (Expansion Slot) в ПК представляет собой разъем системной шины в совокупности с прорезью в задней стенке корпуса компьютера - то есть посадочное место для установки карты расширения. Слоты расширения имеют разъемы шин ISA/EISA, PC1, MCA, VLB или PC Card (PCMCIA). Внутренние слоты используются и для установки модулей оперативной памяти (DIMM), кэш-памяти (COAST), процессоров Pentium II, а также процессорных модулей и модулей памяти в некоторых моделях ПК.

Сокет (Socket) представляет собой гнездо, в которое устанавливаются микросхемы. Его контакты рассчитаны на микросхемы со штырьковыми выводами в корпусах DIP, PGA во всех модификациях или же микросхемы в корпусах SOJ и PLCC с выводами в форме буквы «J». Гнездо *Zip-Socket* (Zero Insertion Force - с нулевым усилием вставки) предназначено для легкой установки при высокой надежности контактов. Эти гнезда имеют замок, открыв который можно установить или изъять микросхему без приложения усилия к ее выводам. Для работы после установки замок закрывают, при этом контакты сокета плотно обхватывают выводы микросхемы.

Джампер (Jumper) представляет собой съемную перемычку, устанавливаемую на торчащие из печатной платы штырьковые контакты (см. рис. 25.а).



а) б)

Рис. 25. Аппаратные средства конфигурирования

Джамперы используются для конфигурирования различных компонентов, как выключатели или переключатели, для которых не требуется оперативного управления. Джамперы переставляют с помощью пинцета, что рекомендуется делать только при выключенном питании, поскольку есть опасность их уронить в неподходящее место или закоротить пинцетом близко расположенные контакты.

DIP-переключатели (DIP Switches) представляют собой малогабаритные выключатели в корпусе DIP (см. рис. 25.б), применяемые для тех же целей, что и джамперы. Их преимущество в более легком переключении, которое удобно производить шариковой ручкой. Недостатком переключателей является большее, по сравнению с джамперами, занимаемое на плате место и более высокая цена. Кроме того, несмотря на название, они обычно являются только выключателями, что делает их применение менее гибким, чем применение джамперов.

В современных компонентах стремятся сокращать количество переключателей или джамперов, стараясь переложить все конфигурационные функции на программно-управляемые электронные компоненты. Платы (карты), в которых удается изжить джамперы полностью (но которые требуют конфигурирования), называют Jumperless Cards - карты, свободные от джамперов. Компоненты, которые после установки конфигурируются автоматически, относят к классу P&P (Plug and Play - вставляй и играй).

Чип (Chip) — это полупроводниковая микросхема, причем обычно неяв-

но подразумевается ее функциональная сложность. Чипсет (Chip Set) — это набор интегральных схем, при подключении которых друг к другу формируется функциональный блок вычислительной системы. Чипсеты широко применяются в системных платах, графических контроллерах и других сложных узлах, функции которых в одну микросхему заложить не удается.

2.7. Структурная схема и основные компоненты ПК

На рис. 26, с помощью структурной схемы, рассматривается состав, связь и назначение основных блоков персональной ЭВМ.

Микропроцессор (МП) – центральный блок ПК, предназначенный для управления работой всех блоков машины и для выполнения арифметических и логических операций над информацией.

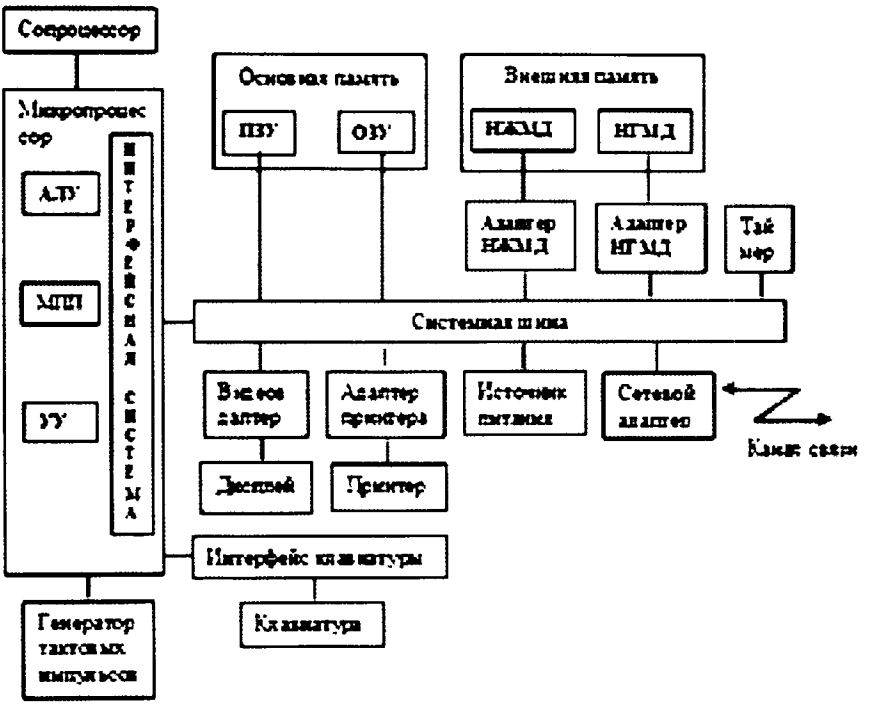


Рис. 26. Структурная схема персонального компьютера

В состав микропроцессора входят:

- *Устройство управления* (*УУ*), которое формирует и подает во все блоки машины сигналы управления (управляющие импульсы); формирует адреса ячеек памяти, используемых выполняемой операцией и передает эти адреса в соответствующие блоки ЭВМ;
- *Арифметико-логическое устройство* (*АЛУ*), которое предназначено для выполнения всех арифметических и логических операций над числовой и символьной информацией. В некоторых моделях ПК, для ускорения выполнения операций, к АЛУ подключается дополнительный *математический сопроцессор*;
- *Микропроцессорная память* (*МПП*), которая служит для кратковременного хранения, записи и выдачи информации, непосредственно используемой в

вычислениях в ближайшие такты работы машины. МПП строится на регистрах и используется для обеспечения высокого быстродействия машины;

- *Интерфейсная система микропроцессора*, которая обеспечивает сопряжение и связь с другими устройствами ПЭВМ; включает в себя внутренний интерфейс МП, буферные запоминающие регистры и схемы управления портами ввода-вывода (ПВВ) и системной шиной. *Интерфейс* (interface) – совокупность средств сопряжения и связи устройств компьютера, обеспечивающая их эффективное взаимодействие. Порт ввода-вывода (Input/Output port) – аппаратура сопряжения, позволяющая подключить к микропроцессору другое устройство;
- *Генератор тактовых импульсов* генерирует последовательность электрических импульсов, частота которых определяет тактовую частоту машины. Интервал времени между соседними импульсами равен одному такту работы машины. Частота тактовых импульсов является одной из основных характеристик персонального компьютера и во многом определяет скорость его работы, так как каждая операция в нем выполняется за конечное число тактов.

Системная шина. Это основная интерфейсная система компьютера, обеспечивающая сопряжение и связь всех устройств между собой. Системная шина включает в себя:

- *Кодовую шину данных*, содержащую провода и схемы сопряжения для параллельной передачи всех разрядов числового кода (машинного слова) операнда;
- *Кодовую шину адреса*, включающую провода и схемы сопряжения для параллельной передачи всех разрядов кода адреса ячейки основной памяти или порта ввода-вывода внешнего устройства;
- *Кодовую шину управления*, содержащую провода и схемы сопряжения для передачи управляющих сигналов (импульсов) во все блоки ПК;
- *Шину питания*, имеющую провода и схемы сопряжения для подключения блоков ПК к системе энергопитания.

Системная шина обеспечивает три направления передачи информации:

1) Между микропроцессором и основной памятью;
2) Между микропроцессором и портами ввода-вывода внешних устройств;
3) Между основной памятью и портами ввода-вывода внешних устройств (в режиме прямого доступа к памяти).

Все блоки, а точнее их порты ввода-вывода, через соответствующие разъемы (стыки) подключаются к шине единообразно: непосредственно или через *контроллеры* (*адаптеры*). Управление системной шиной осуществляется микропроцессором непосредственно, либо, что чаще, через дополнительную микросхему – *контроллер шины*, формирующей основные сигналы управления. Обмен информацией между внешними устройствами и системной шиной выполняется с использованием ASCII-кодов.

Основная память (**ОП**) предназначена для хранения и оперативного обмена информацией с прочими блоками машины. ОП содержит два вида запоминающих устройств: постоянное запоминающее устройство (ПЗУ) и оперативное запоминающее устройство (ОЗУ).

ПЗУ служит для хранения неизменяемой (постоянной) программной и справочной информации, позволяет оперативно только считывать хранящуюся в нем информацию (изменять информацию в ПЗУ нельзя).

ОЗУ предназначено для оперативной записи, хранения и считывания ин-

формации (программ и данных) в текущий период времени. Главным достоинствами оперативной памяти являются ее высокое быстродействие и возможность прямого адресного доступа к ней. Недостатком ОЗУ является невозможность сохранения информации в ней после выключения питания машины.

Внешняя память персонального компьютера используется для долговременного хранения любой информации, которая может потребоваться когда-либо для решения задач. В частности, во внешней памяти хранится все программное обеспечение компьютера. Наиболее распространенными устройствами внешней памяти служат накопители на жестких (НЖМД) и гибких (НГМД) магнитных дисках. В качестве устройств внешней памяти используются также запоминающие устройства на кассетной магнитной ленте (стримеры), накопители на оптических дисках (CD-ROM – compact Disk Read Memory – компакт-диск с памятью, только читаемой) и др.

Источник питания служит для автономного и сетевого энергопитания ПК.

2.8. Материнская плата

Материнская плата (см. пример на рис. 27) является основной составной частью каждого ПК. На ней расположены основные элементы: процессор, оперативная память, BIOS, Chipset, вспомогательные микросхемы и т.д. Материнская плата не только сердце компьютера, но и самостоятельный элемент, который управляет внутренними связями и взаимодействует через прерывания с другими внутренними устройствами. В этом отношении материнская плата является элементом внутри ПК, влияющим на общую производительность компьютера. Супербыстрый винчестер или гиперпроизводительная видеокарта не смогут увеличить его производительность, если тормозится поток данных от (к) материнской плате.

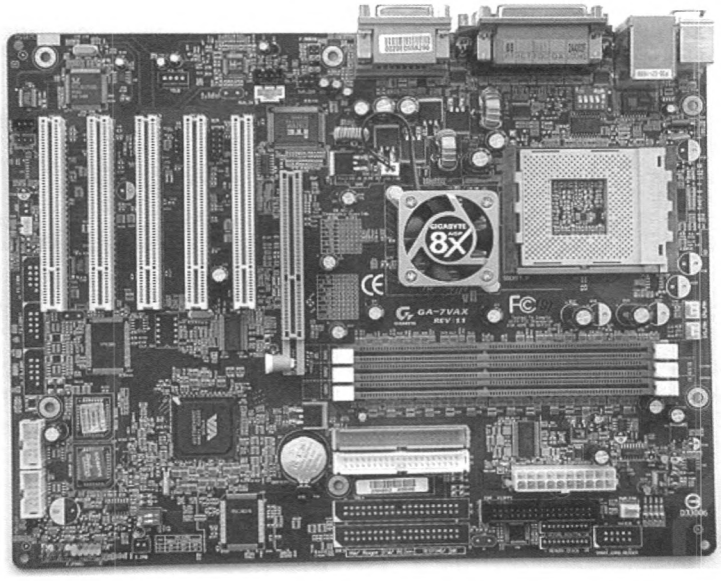


Рис. 27. Материнская плата Gigabyte 7VAX

Компоненты материнской платы. Возможности компьютера определяются системной, или, как чаще говорят, материнской (от англ. motherboard)

платой. На ней обычно размещаются: базовый микропроцессор, оперативная память, сверхоперативное ЗУ, называемое также кэш-памятью, ПЗУ с системной BIOS (базовой системой ввода/вывода, см. рис. 28), а также:

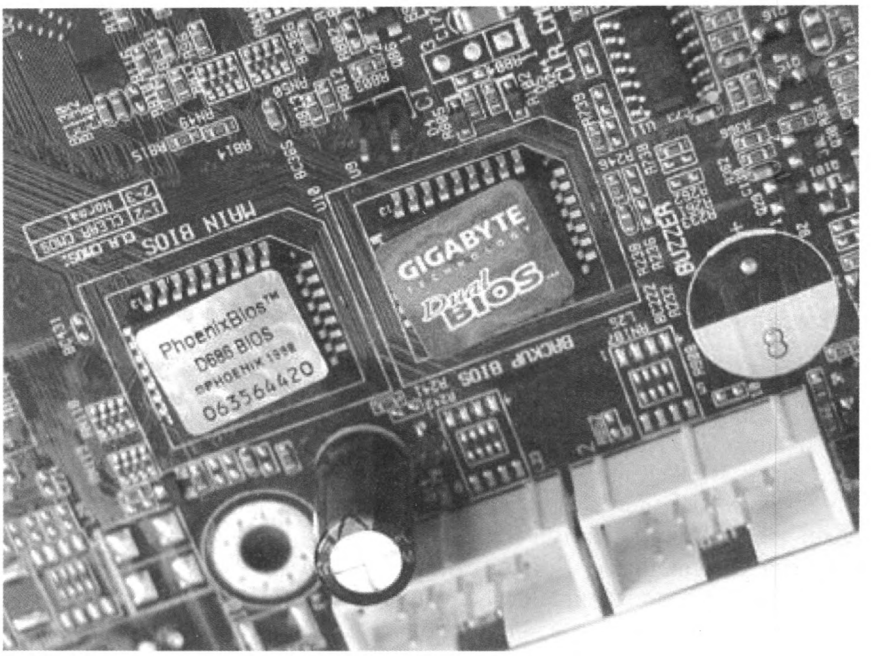


Рис. 28. Расположение Bios на материнской плате

- набор управляющих микросхем, или *чипсетов* (chipset), *вспомогательных микросхем* и *контроллеров ввода/вывода* (см. рис. 29);

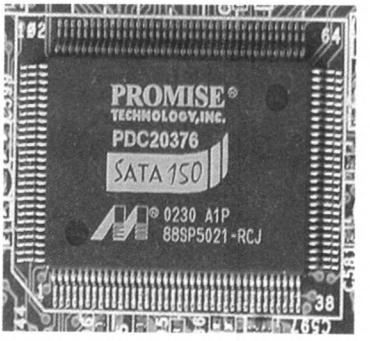


Рис. 29. Управляющие микросхемы (чипсеты)

- CMOS-память с данными об аппаратных настройках и аккумулятором для ее питания;
- разъемы расширения, или слоты (slot) (см. рис. 30);

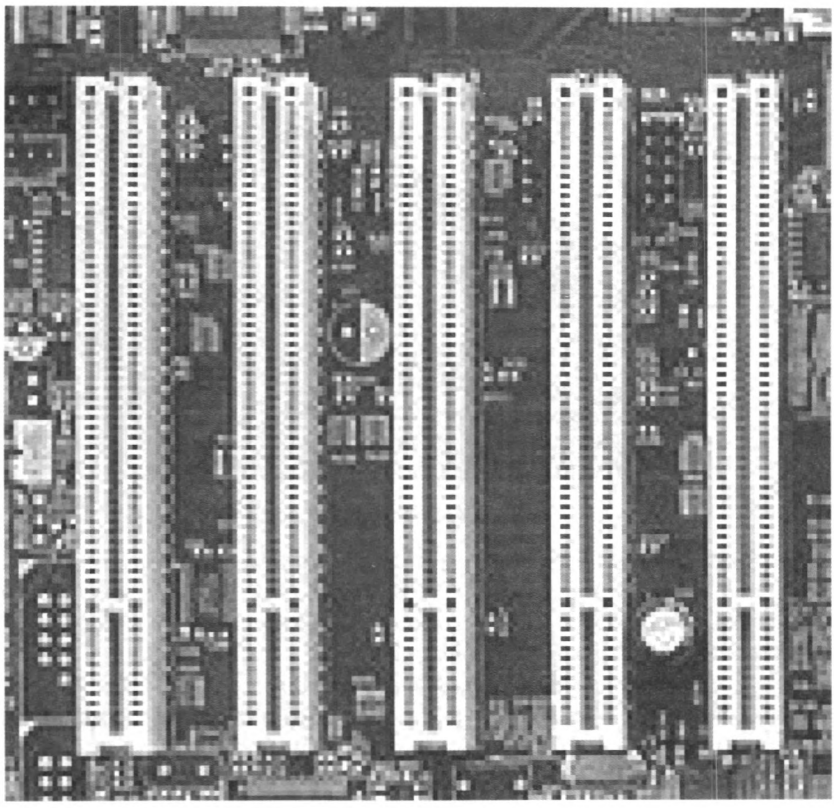


Рис. 30. Слоты расширения

- *разъемы для подключения интерфейсных кабелей* жестких дисков, дисководов, последовательного и параллельного портов, инфракрасного порта, а также универсальной последовательной шины USB (см. рис. 31);

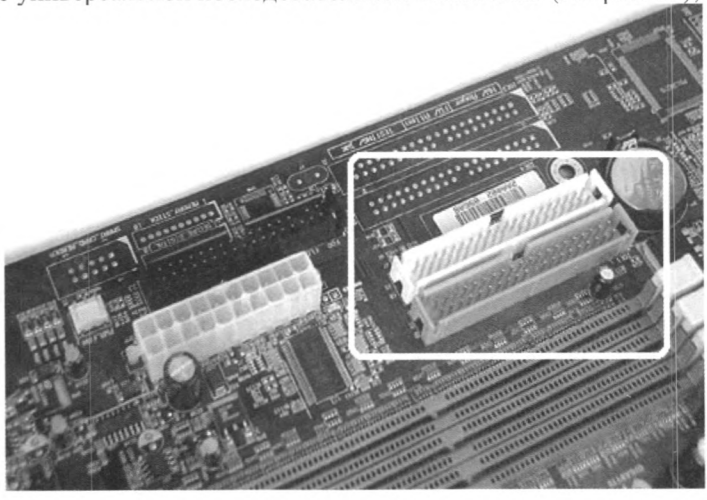


Рис. 31. Разъемы для подключения интерфейсных кабелей жестких дисков

- *разъемы питания* (см. рис.32);

Рис. 32. Разъемы питания

- *преобразователь напряжения с 5 вольт на более низкое для питания процессора (например, процессоры i486DX4, Intel Pentium, Intel Pentium Pro потребляют 3,3 В, а современные Intel Pentium III и 4, равно как AMD Athlon и Duron потребляют менее 2 В);*
- *разъем для подключения клавиатуры и ряд других компонентов.*

На платах формата ATX и NLX также находятся разъемы мыши и клавиатуры в стандарте PS/2, разъемы параллельного и последовательного портов. На материнской плате также могут находиться микросхемы видеоадаптера и звуковой платы (т. н. платы All-In-One), а также контроллера SCSI.

Для подключения индикаторов, кнопок и динамика, расположенных на корпусе системного блока, на материнской плате имеются специальные миниатюрные разъемы-вилки. Подобные же разъемы служат как *контакты для перемычек* (jumpers, см. рис.33) при задании аппаратной конфигурации системы. Если на системной плате сосредоточены все элементы, необходимые для его работы, то она называется All-In-One. У большинства персональных компьютеров системные платы содержат лишь основные функциональные узлы, а остальные элементы расположены на отдельных печатных платах (платах расширения), которые устанавливаются в разъемы расширения. Например, устройство формирования изображения на экране монитора – видеоадаптер – пока, чаще всего, располагается на отдельной плате расширения - видеокарте.

Все компоненты материнской платы связаны друг с другом системой проводников (линий), по которым происходит обмен информацией. Эту совокупность линий называют *шиной* (Bus). В отличие от других систем соединения, линии шины делятся на три группы в зависимости от типа передаваемой информации: линии данных, линии адреса и линии управления. Шины в компьютере различаются и по своему функциональному назначению.

В настоящее время единого стандарта на компоновку материнской платы у производителей не существует. Однако, компоновка материнской платы всегда указывается в ее описании. Поэтому все вопросы по ее структуре решаются с использованием руководства к системной плате.

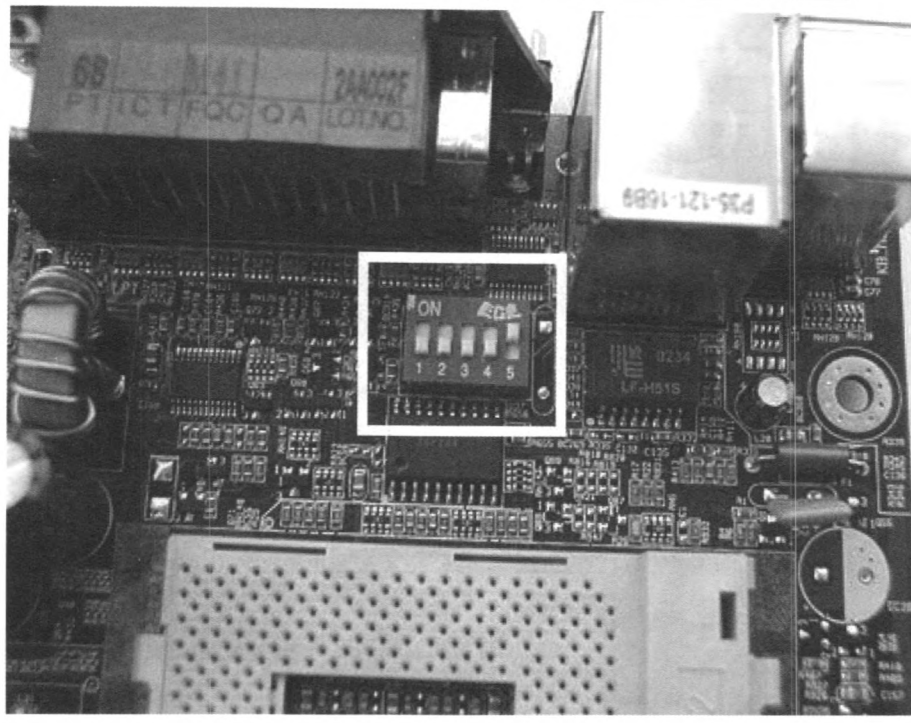


Рис. 33. DIP-переключатели (jumpers)

На рис. 34 рассматривается стандартная компоновка системной платы 6A815E1 на основе ее руководства, взятого с сайта фирмы-производителя материнской платы.

Компоненты рассматриваемой материнской платы помечены цифрами:

1. Панель стандартных разъемов (разъемы, спецификации и шины которых поддерживаются чипсетом, такие как: USB, клавиатура, мышь, порт COM1, стандартные звуковые разъемы и т.д.);
2. Разъем для получения аудио-потока с CD-ROM;
3. Разъем для подключения порта COM2;
4. Разъем для подключения передней панели индикаторов и кнопок;
5. Разъемы для подключения вентиляторов;
6. Разъем для подключения интерфейсного кабеля дисковода;
7. Разъемы для подключения интерфейсных кабелей жестких дисков;
8. Разъем для подключения инфракрасного порта;
9. Разъем для подключения дополнительного USB-порта;
10. Разъем питания;
11. Перемычки блокировки flash BIOS;
12. Индикатор состояния Suspend to RAM;
13. Перемычка установки параметра "Включать компьютер при нажатии на клавиатуру";
14. Перемычка настройки сетевой карты (встроенной в материнскую плату);
15. Перемычки очистки данных CMOS.

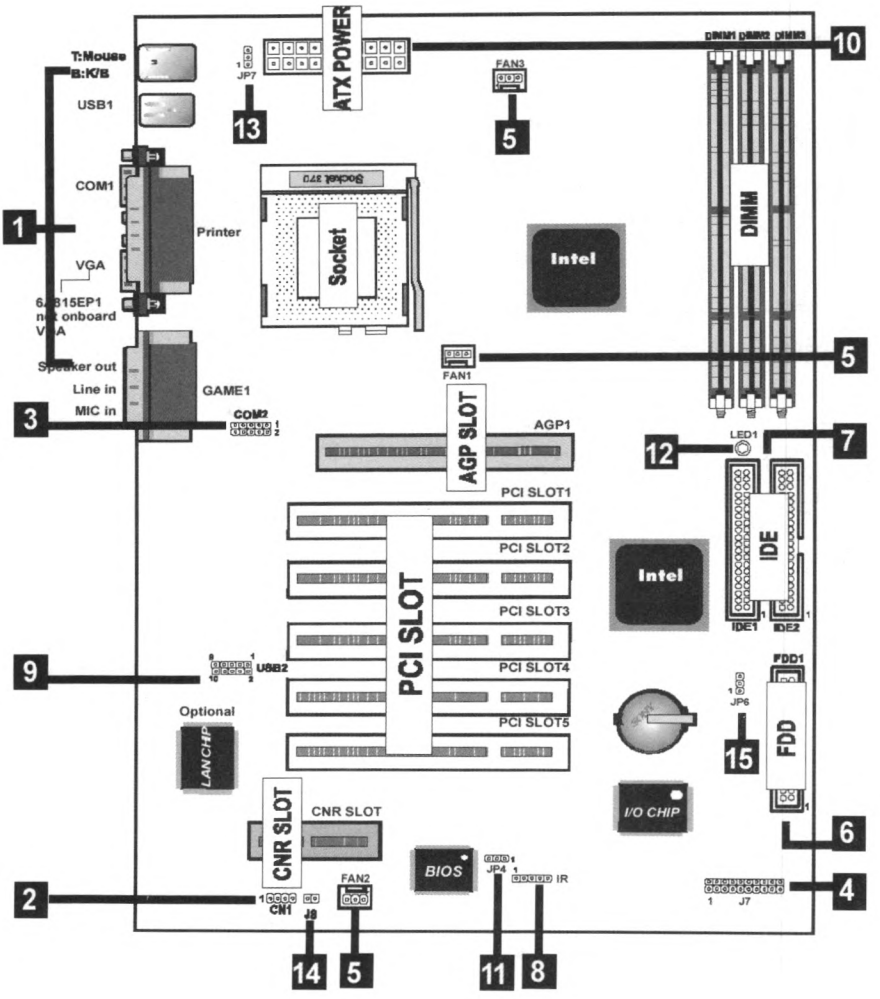


Рис. 34. Компоновка материнской платы 6A815E1

Кроме компоновки материнской платы существует еще один неотъемлемый аспект – компоновка разъемов на задней панели компьютера. Они также указываются в технической документации на материнскую плату, так как разъемы непосредственно интегрированы в нее (см. рис. 35) и представляют собой:

1. Разъем для клавиатуры PS/2;
2. Порт COM1;
3. Порт COM2;
4. Линейный выход;
5. Линейный вход;
6. Микрофон;
7. Разъем PS/2 для мыши;
8. Разъемы USB-шины;
9. Параллельный порт (LPT);
10. MIDI / Игровой порт.

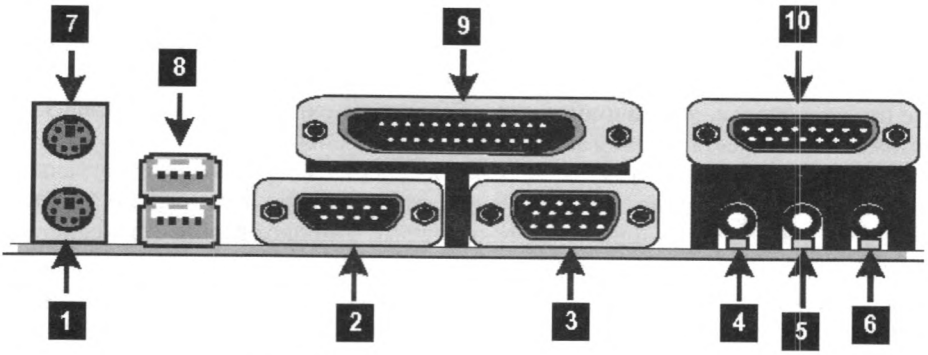


Рис. 35. Компоновка разъемов материнской платы

Типоразмеры (форм-факторы) материнских плат. На сегодняшний день существует четыре преобладающих типоразмера материнских плат – AT, ATX, LPX и NLX. Кроме того, есть уменьшенные варианты формата AT (Baby-AT), ATX (Mini-ATX, microATX) и NLX (microNLX). Более того, недавно выпущено расширение к спецификации microATX, добавляющее к этому списку новый форм-фактор – FlexATX. Все эти спецификации определяют форму и размеры материнских плат, а также расположение компонентов на них и особенности корпусов.

2.9. Поддержка процессора

Процессор должен устанавливаться на материнскую плату. Существуют различные конструктивы процессоров и, соответственно, разъемов под них, некоторые из которых рассмотрены ниже.

Процессорный разъем Socket 7. Существует также, так называемый, стандарт Super 7, предназначенный для работы с FSB 66-100 МГц, но разъемы Super 7 и Socket 7 не отличаются абсолютно ничем, что бы там ни говорили особо компетентные продавцы. На верхней части разъема может быть надпись Socket 7 или Super 7. Имеется один "ключ" - скошенный угол, благодаря которому ориентация процессора в разъеме не может быть неверной, так как ровно в том же месте у процессора отсутствует ножка. Размер - 37×37 мм, 5 рядов выводов, расположенных в шахматном порядке. Крепление кулера осуществляется с помощью клипсы-коромысла, причем все процессоры под Socket 7, кроме K6-2+ и K6-III, имеют сравнительно небольшое тепловыделение, так что покупка дорогих кулеров не всегда имеет смысл. Процессоры, выпущенные под этот разъем: Intel Pentium MMX, AMD K6, K6-2, K6-2+, K6-III.

Чипсет (от английского "chip set" - набор микросхем) - набор системной логики, обеспечивающий функционирование как самого процессора, так и периферийных устройств, "сердце" всей вычислительной системы. Именно он определяет основные функциональные возможности платы: типы поддерживаемых процессоров, максимальный объем и тип оперативной памяти и т. п. Кроме того, от чипсета также зависят такие характеристики, как максимальная штатная частота системной шины и шины памяти, синхронный или асинхронный режим их работы, поддерживаемые делители для шин AGP и PCI, а также максимально возможные режимы работы AGP и UDMA, количество и состав внешних портов.

То есть чипсет является тем звеном, которое связывает воедино все остальные компоненты системы и обеспечивает их функционирование, им же описываются правила этого функционирования. Но что такое чипсет сам по себе? Просто набор микросхем с описанием поддерживаемых интерфейсов. В таком виде он, конечно же, функционировать не может. Именно физическая реализация возможностей чипсета и возлагается на материнскую плату.

Чипсет обычно состоит из нескольких специализированных интегральных микросхем (ASIC — application-specific integration circuits), как правило, от одной до четырех, выпущенных одним производителем. Широкое распространение чипсеты получили в эпоху 386-х и 486-х процессоров, а до этого чаще применялись наборы микросхем, состоявшие из множества контроллеров средней степени интеграции. Переход к чипсетам позволил уменьшить стоимость материнских плат и повысить взаимную совместимость компонентов, что облегчило задачу проектирования материнских плат.

В настоящее время существует большое количество компаний, выпускающих чипсеты для процессоров различных поколений. К ним относятся: Intel, VIA Technology, SiS, OPTi, UMC, ALi, Micron Technologies, VLSI, ETEQ, Cyrix, AMD. При обозначении чипсета обычно приводят название производителя и номер серии: например, SiS 496 обозначает микросхему SiS 85C496. Часто производитель использует имена собственные для обозначения чипсета: Samurai, Triton, Alladin, Viper, Saturn. Некоторые производители наносят на чипсет свою торговую марку (PC Chips, ExpertChip).

Чипсеты конструктивно привязаны к типу используемого процессора, причем за время жизненного цикла процессора успевает смениться несколько поколений чипсетов для него, и если первые чипсеты позволяют использовать преимущества нового процессора лишь отчасти, то последние позволяют выжать из процессора максимальную производительность и использовать широкий спектр процессоров.

Обычно чипсет представляет собой две микросхемы (см. структуру на рис. 36), одна из которых - северный мост, вторая - южный мост.

Северный мост - отвечает за обмен данными между процессором, оперативной памятью, AGP-портом, а также южным мостом.

Южный мост - как правило, содержит контроллеры USB, ATA, PCI, ISA, AMR, CNR, ACR, AC'97, контроллеры портов ввода/вывода, а также, возможно, Ethernet 10/100Mbit и Home PNA.

На рис. 36 обозначены: 1 – параллельный и последовательные порты, дисковод, клавиатура и мышь; 2 – звуковые и модемные кодеки.

В зависимости от структуры чипсета определяется возможность или невозможность поддержки материнской платой нескольких процессоров.

На рис 37 приводится пример двухпроцессорной реализации (поддержки) чипсета, полученной с Веб-сайта www.km.ru.

Микропроцессор, чипсет, память, контроллеры, порты ввода/вывода и разъемы различных шин еще не исчерпывают конструкцию материнской платы. Для создания полной системы необходимы также вспомогательные микросхемы, такие как преобразователь напряжения, тактовый генератор, таймер, различные контроллеры, буферы адреса и данных и т. п. Функции многих из них интегрированы в чипсете, однако некоторые компоненты, в любом случае, остаются снаружи.

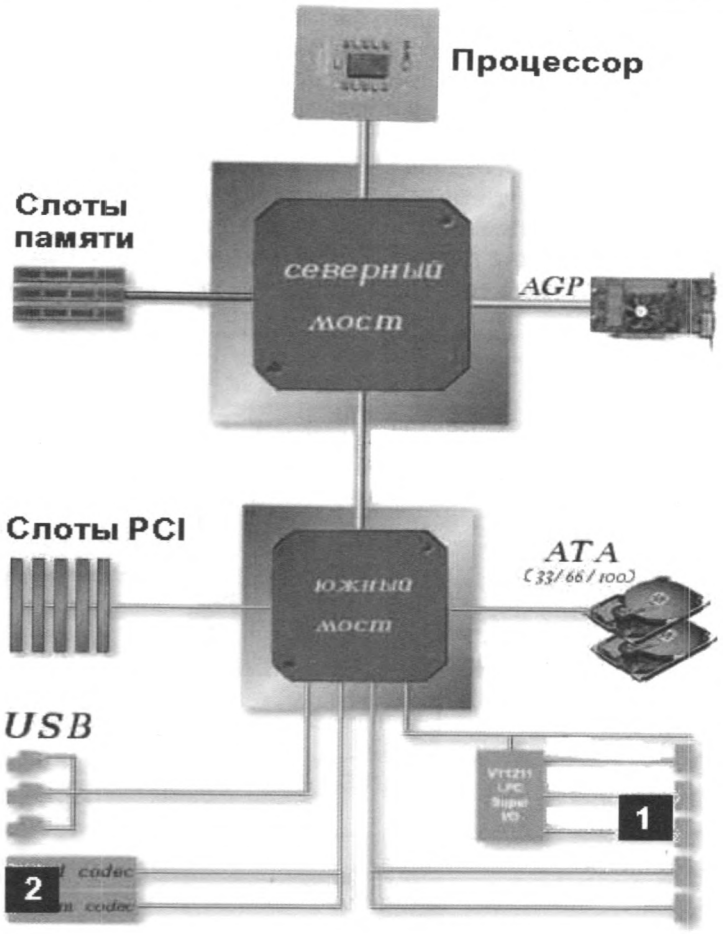


Рис. 36. Структура чипсета

Во-первых, это преобразователь напряжения. Дело в том, что блок питания на материнских платах формата AT выдает лишь 5 и 12 В различной полярности, ATX - 5, 12 и 3,3 В. Для питания же процессоров с двойным напряжением, которым может требоваться от 2,0 до 3,2 В, в старых Pentium-платах предусматривался VRM (Voltage Regulator Module) - модуль регулятора напряжения, который выглядел как специальный двухрядный разъем с пластмассовым обрамлением, расположенный рядом с процессором. Позднее регулятор напряжения был реализован на самой материнской плате. С его помощью на современных платах, как правило, можно при помощи переключателей задавать различные напряжения с шагом 0,1 В.

Существует два типа регуляторов: линейный и импульсный. Применявшийся в более старых платах линейный регулятор напряжения представлял собой микросхему, понижающую напряжение за счет рассеяния его избытка в виде тепла. С уменьшением требуемого напряжения росла тепловая мощность, рассеиваемая такими регуляторами, поэтому они снабжались массивными радиаторами, по которым их легко было найти на материнской плате. При установке в материн-

скую плату процессора, потребляющего большую мощность, регулятор (а с ним и материнская плата) мог выйти из строя из-за перегрева. Поэтому в современных материнских платах применяется импульсный регулятор, содержащий сглаживающий фильтр низких частот, на который подается последовательность коротких импульсов полного напряжения. За счет инерционности фильтра импульсы сглаживаются в требуемое постоянное напряжение. КПД такого преобразователя весьма высок, поэтому паразитного нагрева почти не происходит. Узнать импульсный регулятор напряжения на плате можно по катушкам индуктивности. Часто применяют смешанные варианты: импульсный регулятор понижает напряжение с 5 до 3 В, а линейный - с 3,3 до 2,8 В, так как нагрев при этом небольшой.

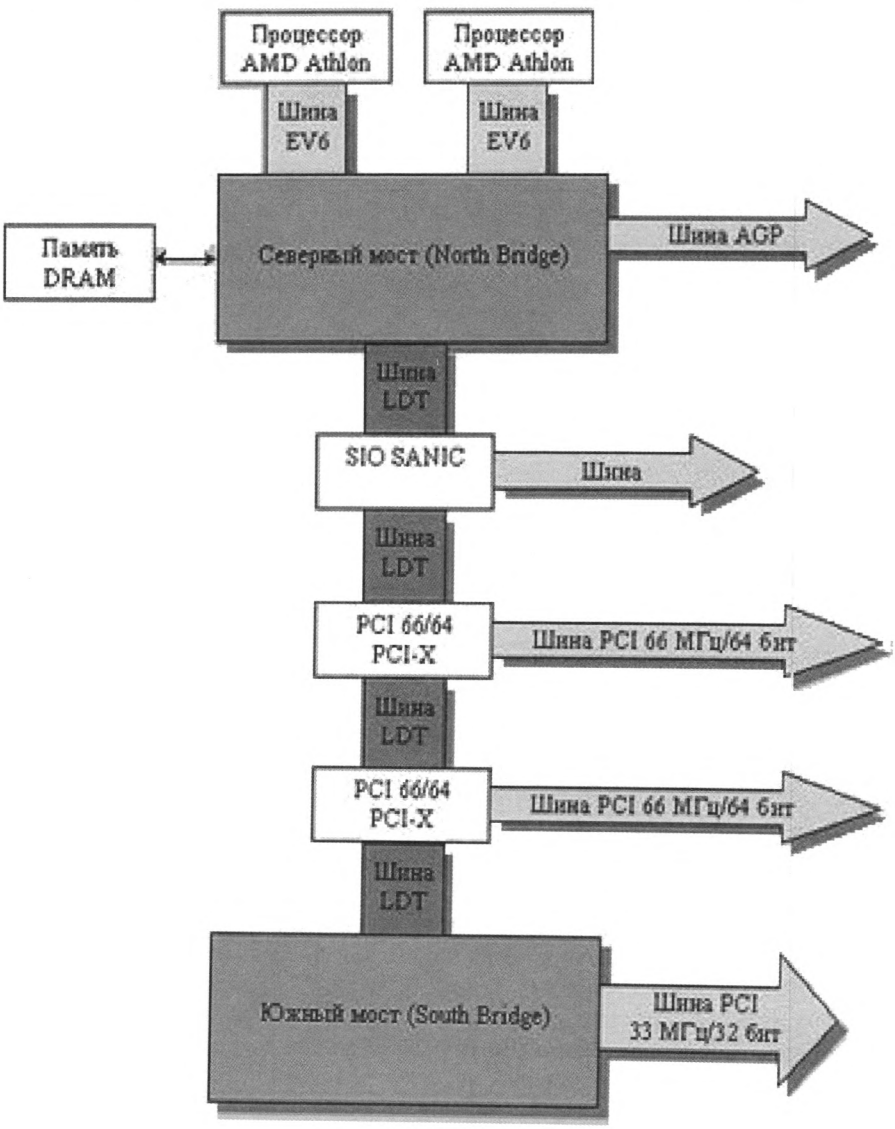


Рис. 37. Поддержка чипсетом двух процессоров

Во-вторых, обязательным устройством любой материнской платы является тактовый генератор. Так как большинство логических элементов компьютера должно работать синхронно, именно генератор тактовой частоты вырабатывает специальные импульсы, служащие тактовыми сигналами для всех электронных устройств на системной плате.

В-третьих, обязательным устройством (в настоящее время входящим в чипсет) является контроллер прерываний. Что такое прерывание? Понятно, что процессор в одно и то же время может обслуживать только одно событие. При этом непонятно, как компьютер может выполнять несколько задач параллельно. Например, компьютер выполняет расчеты в Excel, а пользователь в это время переместил мышь. При этом возникает прерывание (Interrupt), то есть процессор на время откладывает (прерывает) расчеты, запускает короткую программу (так называемый обработчик прерывания), которая считывает координаты и перемещает указатель мыши на экране в новое положение, после чего процессор вновь возвращается к прерванным расчетам. Все это происходит настолько быстро, что у пользователя создается иллюзия параллельности работы.

Прерывания используются для управления работой множества устройств: видеокарт, дисковых и других накопителей, звуковых карт, контроллеров SCSI, сетевых карт и др. При этом используется контроллер прерываний, который устанавливает для каждого из своих входов определенный уровень важности - приоритет. У всех современных компьютеров имеется 16 линий запроса прерывания (Interrupt ReQuest, IRQ). Приоритет убывает в порядке возрастания номера линии - наивысший приоритет имеет линия запроса прерывания IRQ0. Линии прерывания IRQ8 - IRQ15 являются расширением линии IRQ2, то есть имеют приоритет ниже, чем IRQ1, но выше IRQ3. Из этих 16 линий прерывания для дополнительных периферийных устройств, таких как звуковые и сетевые карты, а также различных дополнительных контроллеров, остаются свободными всего 2-3 линии. Из-за этого возникает, так называемый, конфликт прерываний (или конфликт ресурсов), когда два устройства пытаются использовать одно и то же прерывание. Для преодоления таких конфликтов была разработана технология автоконфигурирования Plug-and-Play. Другим способом ухода от конфликта прерываний является использование устройств USB, FireWire или SCSI, которые, будучи расположенными на одной шине, используют всего одно прерывание на шину.

Для быстрого обмена данными с периферийными устройствами обычно используются каналы прямого доступа в память (ПДП или DMA, Direct Memory Access). Всего таких каналов семь, и, как минимум, один (2-й) всегда задействован. При распределении ресурсов между периферийными устройствами возможен также конфликт и по DMA.

В-четвертых, каждая материнская плата имеет контроллер клавиатуры. Он может быть либо выполнен в виде отдельной микросхемы (самой длинной на плате), либо интегрирован в чипсете платы.

В-пятых, современная материнская плата имеет двухканальный EIDE-контроллер и контроллер ввода-вывода, обслуживающий дисководы гибких дисков и порты ввода/вывода. Кроме того, на ней могут быть контроллеры USB и FireWire, дополнительные интегрированные контроллеры (видео, сетевой, SCSI, звук и т. п.). Все эти устройства описаны в соответствующих разделах.

К дополнительным компонентам материнской платы можно отнести температурные датчики LM78, собирающие информацию о температуре процессора, материнской платы, скорости вращения вентилятора и др.

2.10. Кэш-память

Кэшированием данных называется размещение наиболее важных данных в области памяти с более быстрым доступом. В качестве житейской аналогии можно привести библиотеку школьника, у которого нужные каждый день учебники лежат на рабочем столе, изредка читаемые классики стоят на книжной полке, а старые ненужные тетради валяются где-нибудь на балконе. В случае необходимости, время доступа к этим источникам будет разным, однако, и вероятность того, что потребуется учебник или старая тетрадь, тоже разная.

В мире компьютерной памяти этот принцип применим потому, что более быстрая память обычно стоит существенно дороже более медленной, однако применение малого объема быстрой (но дорогой) памяти, называемой кэш-памятью (cache memory), в комплексе с большим объемом медленной (но дешевой) памяти позволяет создать приемлемое по цене и скорости решение. Применение кэширования особенно эффективно, когда доступ к данным осуществляется преимущественно в последовательном порядке. Тогда после первого запроса на чтение данных, расположенных в медленной (кэшируемой) памяти, можно заранее выполнить чтение следующих блоков данных в кэш-память для того, чтобы при следующем запросе на чтение данных почти мгновенно выдать их из кэш-памяти. Такой прием называется упреждающим чтением.

Упреждающее чтение применяется во всех современных жестких дисках, имеющих от 64 до 1024 Кбайт кэш-памяти, выполненной на основе динамической RAM. Считываемые с диска данные, с некоторым запасом, помещаются в кэш-память диска и определенное время там хранятся. При повторном обращении к тем же данным они считываются уже из кэш-памяти, что происходит в 10-1000 раз быстрее.

Кэширование данных применяется также в процессорах. Внутри кристалла процессора находится малый объем (от 1 до 1024 Кбайт) очень быстрой статической памяти, работающей на частоте процессора. Эта память используется для кэширования существенно более медленной оперативной памяти, выполненной на основе динамической RAM.

Таким образом, в различных ситуациях одна и та же память может быть как кэшем, так и кэшируемой памятью.

Кэш-память также может быть организована в виде иерархической структуры. В случае процессоров x86 характерно использование кэша первого уровня (Level 1 или L1-кэша), расположенного непосредственно на кристалле процессора, и более медленного кэша второго уровня (Level 2 или L2-кэша), расположенного в другой микросхеме или вообще на другой плате. При этом кэш первого уровня кэширует L2-кэш, а тот, в свою очередь, кэширует еще более медленную оперативную память. В RISC-процессорах зачастую используется L3-кэш и кэш более высоких порядков.

Существуют различные алгоритмы работы кэш-памяти, которые очень сильно влияют на эффективность процедуры кэширования. Помимо кэширования операций чтения данных можно выполнять кэширование записи данных (это называется отложенной записью, или lazy writes, для жестких дисков и обратной записью, или write back, для процессоров). Применение отложенной записи еще больше увеличивает скорость работы диска, но повышает риск потери данных, которые не успели записаться из кэш-памяти в кэшируемую память, в случае внезапного краха системы.

3. Основы телекоммуникаций

3.1. Модем. Основные понятия

Как известно, данные в компьютере представлены в цифровой форме - закодированы в виде нулей и единиц, которым физически соответствует низкий или высокий уровень напряжения. Для передачи данных по аналоговому каналу необходимо управлять одним из параметров сигнала несущей частоты, например, амплитудой, частотой или фазой. Типичным аналоговым каналом является телефонный канал. В нем при передаче речи воспринимаемые микрофоном звуки преобразуются в электрические сигналы, которые модулируют сигнал несущей частоты. При передаче цифровой информации управление параметрами несущего колебания производят информационные биты – потенциальный импульсный код. При этом модулируемый аналоговый сигнал называется несущим (carrier) и, обычно, представляет собой сигнал постоянной частоты и амплитуды (несущая частота).

Таким образом, если для передачи цифровой информации по каналу связи необходимо поток битов преобразовать в аналоговые сигналы, то при приеме информации из канала связи в ЭВМ нужно выполнить обратное действие – преобразовать аналоговые сигналы в поток битов, которые может обрабатывать ЭВМ. Такие преобразования выполняет специальное устройство (см. рис. 27), включаемое между компьютером и телефонной линией, называемое модемом (modem). Слово "модем" происходит от сочетания "МОдулятор/ДЕМодулятор" и используется для обозначения широкого спектра устройств передачи цифровой информации при помощи аналоговых сигналов путем их модуляции - изменения во времени одной или нескольких характеристик аналогового сигнала: частоты, амплитуды и (или) фазы.

Рис. 27. Модем

Типовая схема организации связи при помощи двух модемов показана на рис. 28.

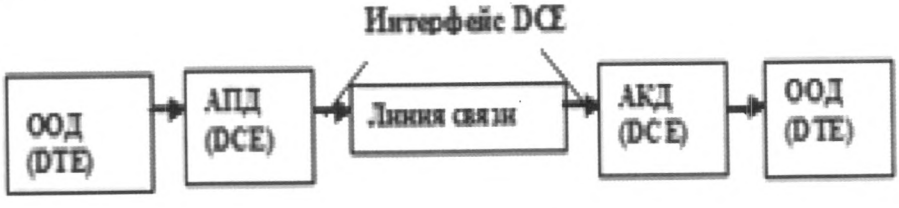


Рис. 28. Типовая схема передачи данных

Оконечное оборудование данных (ООД) – в терминологии систем связи

обозначает оконечные цифровые устройства, генерирующие или получающие данные. Соответствующий ООД международный термин имеет аббревиатуру DTE (Data Terminal Equipment). В качестве DTE может выступать персональный компьютер, большая ЭВМ, терминал, устройство сбора данных, кассовый аппарат, приемник сигналов глобальной навигационной системы или любое оборудование, способное передавать или принимать данные.

Оконечное оборудование данных передает и (или) принимает данные посредством аппаратуры передачи данных (АПД) и канала связи. Соответствующий АПД международный термин – DCE (Data Communication Equipment – оборудование передачи данных). Этим термином обозначаются модемы.

Линия связи между DCE – аналоговая, между DCE и DTE – цифровая.

Важную роль во взаимодействии DTE и DCE играет интерфейс, который состоит из входящих/исходящих цепей, разъемов и соединительных кабелей. Интерфейс определяет логику работы DTE и DCE.

Если для связи DTE и DCE используется унифицированный цифровой интерфейс, это зачастую дает возможность связать два расположенных рядом DTE прямой цифровой линией - так называемым нуль-модемным кабелем. В случае разнесения DTE на большое расстояние в разрыв вместо нуль-модемного кабеля включается пара модемов и аналоговая линия связи, обеспечивая прозрачное соединение и передачу данных.

3.2. Каналы связи

Под *каналом связи* понимают совокупность среды распространения и технических средств передачи между двумя канальными интерфейсами DCE. В зависимости от типа передаваемых сигналов различают два больших класса каналов связи: цифровые и аналоговые.

Цифровой канал является битовым трактом с цифровым (импульсным) сигналом на входе и выходе канала. На вход *аналогового канала* поступает непрерывный сигнал, и с его выхода также снимается непрерывный сигнал (рис. 29).

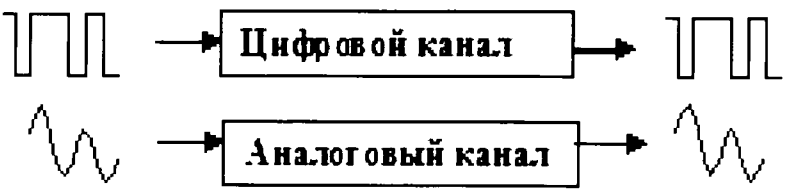


Рис. 29. Сигналы цифровых и аналоговых каналов передачи

Цифровыми каналами являются каналы систем ИКМ (Импульсно-Кодовой Модуляции), ISDN (Integrated Services Digital Network), каналы типа T1/E1 и многие другие. Вновь создаваемые системы передачи данных (СПД) строят, в основном, на основе цифровых каналов, обладающих рядом преимуществ перед аналоговыми каналами.

Аналоговые каналы наиболее распространены по причине длительной истории развития и относительной простоты их реализации. Типичным примером аналогового канала является канал тональной частоты (ТЧ), а также групповые тракты на 12, 60 и более каналов тональной частоты. Телефонный канал общего пользования (ТфОП), как правило, включает многочисленные коммутаторы, устройства разделения, групповые модуляторы и демодуляторы. Для ТфОП парамет-

ры канала и его физический маршрут будут меняться при каждом очередном вызове.

Для передачи данных на входе аналогового канала должно находиться устройство, которое преобразовывало бы цифровые данные, приходящие от DTE, в аналоговые сигналы, посылаемые в канал. Приемник должен содержать устройство, которое преобразовывало бы обратно принятые непрерывные сигналы в цифровые данные. Этими устройствами являются модемы. Аналогично при передаче по цифровым каналам данные от DTE приходится приводить к виду, принятому для данного конкретного канала. Этим преобразованием занимаются цифровые модемы, очень часто называемые адаптерами ISDN, адаптерами каналов E1/T1, линейными драйверами и т.д. (в зависимости от конкретного типа канала или среды передачи).

Телефонные каналы

Модемы могут работать с двумя типами телефонных каналов, это:

• *коммутируемые* (к которым пользователь подключает обычный телефонный аппарат). По коммутируемому каналу можно соединиться с любым абонентом сети.

• *арендованные* (непосредственно соединяющие пользователя и телефонный канал, арендованный у телефонной компании) или (очень редко) прокладываются заинтересованной организацией..

Работа по коммутируемому каналу. Модемы вызывают друг друга посредством набора номера телефона, к которому они подключены. Такие каналы принципиально содержат в своем составе коммутационное оборудование телефонных станций (АТС). Обычные телефонные аппараты используют коммутируемые каналы ТфОП. Кроме того, коммутируемые каналы предоставляет цифровая сеть с интеграцией служб ISDN.

Работа по арендованному каналу. В этом режиме 2 модема соединены по специально выделенному телефонному каналу и находятся в постоянном взаимодействии. Качество этих каналов выше, чем у коммутируемых каналов, так как на них не влияет коммутационная аппаратура АТС. Этот режим используется при передаче очень больших объемов информации или при необходимости срочной доставки, а также если абоненты предъявляют специальные требования к защите информации от несанкционированного доступа.

Работа при 2-х и 4-х проводном окончании. Соединение пользователя с телефонным каналом может осуществляться с помощью 4-х и 2-х проводного окончания. При 4-х проводном соединении передача и прием происходит независимо друг от друга. В таких каналах практически полностью отсутствует влияние сигналов, передаваемых во встречном направлении. А при работе с 2-х проводным соединением передача информации идет в дуплексном режиме (передача информации одновременно в двух направлениях). Такие каналы позволяют экономить на стоимости кабелей, но требуют усложнения каналообразующей аппаратуры и аппаратуры пользователя. В данном режиме работы передатчик порождает помехи на входе своего приемника (т.к. невозможно идеально настроить системы, отвечающие за разделение приема и передачи информации). Двухпроводные каналы требуют разделения принимаемого и передаваемого сигналов, что реализуется при помощи дифференциальных систем, обеспечивающих необходимое затухание по встречным направлениям передачи. Наличие дифференциальных систем приводит к искажениям амплитудно-частотных характеристик канала и к специ-

фической помехе в виде эхо-сигнала. Передача данных по телефонным каналам с 2-х проводным окончанием организуется с помощью следующих методов:

• Полудуплексный режим - режим передачи данных по 2-х проводному каналу, при котором в каждый момент времени передача ведется в одном направлении.

• Частотное разделение направлений передачи (при дуплексном режиме). Может быть симметричной и асимметричной в зависимости от равенства или неравенства скоростей передачи в разных направлениях.

• Одновременной передачи в обоих направлениях с подавлением на приеме отраженного сигнала собственного передатчика (дуплексный режим с эхо компенсацией). Передача также может вестись асинхронным и синхронным методом.

При асинхронной передаче обмен информацией между модемами ведется Старт-стопными знаками. Каждый старт-стопный сигнал состоит из стартового бита,7-ми или 8-ми битов информации, бита паритета и 1-го или 2-х стоповых битов. Бит проверки (parity bit) служит для контроля честности (even) или нечетности (odd), ему могут передаваться фиксированные значения ("1" - mark или "0" - space), или он может вообще отсутствовать (none). В последнем случае количество битов информации равно 8. Полезная скорость передачи информации при асинхронной передаче ниже, чем скорость передачи элементов сигнала, что обусловлено необходимостью передачи стартового, стопового сигналов и бита проверки. Так как в настоящее время в модемах преимущественно используются асинхронные методы модуляции, в них вводятся устройства, выполняющие асинхронно-синхронное и обратное преобразование информации.

Полоса пропускания

Полоса пропускания является основополагающим понятием в области связи. Единицей измерения является Герц (Гц, Hz). Полоса пропускания, или ширина полосы пропускания, определяется как диапазон между нижней и верхней частотами, используемыми в конкретной СПД. Для фильтров, аттенюаторов, усилителей, линий передачи и др. электромагнитного оборудования границы полосы пропускания определяются как точки, в которых средняя мощность сигнала P_c составляет половину от его максимальной мощности $P_{c.макс}$ (см. рис. 30).

Рис. 30. Определение полосы пропускания

Вносимое ими затухание уменьшает мощность сигнала на границах полосы пропускания в два раза, что соответствует трем децибелам в относительных логарифмических единицах

$$\alpha_з = 10 \times \lg(P_{с.макс}/P_0) = 10 \times \lg 2 \cong 3 \text{ [дБ]}.$$

Применительно к ТфОП полоса телефонного канала тональной частоты определена как диапазон частот от 300 до 3400 Гц, т.е. ширина канала равна 3100 Гц.

Скорость передачи данных

В компьютере данные принято измерять байтами. Байт – это взаимосвязанная последовательность из восьми битов. В устройствах, которые осуществляют параллельную передачу данных, скорость передачи данных измеряют в байтах в секунду. К таким устройствам относятся, например, все устройства, подключаемые к параллельному порту. В них одновременно по группе проводников передаются сразу восемь битов, составляющих байт. Однако телефонный модем к ним не относится. Через него данные передаются последовательно бит за битом. Поэтому измерять пропускную способность (производительность модема) в байтах в секунду не принято – только в битах в секунду. Итак, передающему модему от компьютера передается поток бит. В зависимости от физического протокола передачи данных, по которому работает модем, при модуляции, модем ставит в соответствие каждому биту или последовательности бит цифровой информации некий аналоговый сигнал (тон). Единицей скорости изменения сигнала (т.е. скорости передачи в канале) является бод (от англ. baud).

Некоторая путаница в понятиях бод и бит/с связана с самыми ранними моделями факс-модемов, где эти величины действительно совпадали. Действительно, если в одном периоде сигнала передается один бит информации, то скорость передачи в канале, измеряемая в бодах, соответствует скорости передачи данных в битах за секунду. Современные модемы кодируют несколько битов информации в одном изменении состояния сигнала и, очевидно, что скорость передачи цифровых данных и скорость работы канала в этом случае не совпадают. Поэтому не следует смешивать понятия бод и бит/с. Например, если при скорости 2400 бод в одном периоде сигнала закодировано два бита, то производительность модема – 4800 бит/с, если три бита, то производительность – 7200 бит/с и т. д.

В зависимости от типа протокола, поддерживаемого модемами, соединение можно устанавливать на следующих скоростях:

- **V.32bis** - максимальная скорость составляет 14400 бит/с.
- **V32** - 9600 бит/с.
- **V22/V22bis** - 2400 бит/с.

Иногда в старых технических описаниях или книгах встречается скорость передачи, измеряемая в символах в секунду (cps, от англ. - character per second). В среднем, при передаче данных через модем, каждым десяти переданным битам соответствует 1-байт или символ машинописного текста. Поэтому передаче данных на скорости 14400 бит/с будет соответствовать приблизительно 1440 cps (для асинхронного метода передачи).

Некоторые коммуникационные программы тоже измеряют скорость передачи в символах в секунду, но в этом случае речь идет о самых обыкновенных информационных символах, длина которых равна восьми битам. Поэтому путать символьную скорость модема с программной скоростью приема данных не следует. Сейчас в состав задач модема входит: защита от ошибок; исправление ошибок при передаче; сжатие данных. Это позволяет радикально увеличить достоверность и скорость передачи информации.

3.3. Стандарты для протоколов и кодов передачи данных.

Теоретическую основу передачи данных в современных информационных сетях определяет базовая эталонная модель взаимодействия открытых систем (OSI – Open system Interconnection). Она описана стандартом ISO 7498 Международной организации стандартов (ISO – International Standards Organisation). Эта эталонная модель является международным стандартом для передачи данных. Согласно эталонной модели взаимодействия OSI выделяются семь уровней, образующих область взаимодействия открытых систем. Основные функции семиуровневой модели OSI приведены в табл. 6.

Таблица 6.

№ уровня	Название уровня	Функция
7	Прикладной	Поддержка прикладного программного обеспечения пользователей
6	Представительный	Представление и интерпретация (шифрование, сжатие и кодирование) данных
5	Сеансовый	Установление и поддержка сеанса связи между абонентами; обмен данными между ними
4	Транспортный	Обеспечение сквозного обмена данными между сетями
3	Сетевой	Маршрутизация, сегментирование и объединение блоков данных; обнаружение ошибок и сообщение о них
2	Канальный	Управление каналом передачи данных; формирование кадров; управление доступом к среде передачи; передача данных по каналу; обнаружение ошибок в канале и их коррекция
1	Физический	Физический интерфейс с каналом передачи данных; битовые протоколы модуляции и линейного кодирования

Основная идея этой модели заключается в том, что каждому уровню отводится конкретная роль. Благодаря этому общая задача передачи данных расщепляется на отдельные конкретные задачи. Функции уровня, в зависимости от его номера, могут выполняться программными, аппаратными либо программно-аппаратными средствами. Как правило, функции высших уровней носят программный характер. Функции канального и сетевого уровней могут быть исполнены как программными, так и аппаратными средствами. Физический уровень обычно выполняется в аппаратном виде.

Каждый уровень определяется группой стандартов, включающих спецификации на протокол и обеспечиваемый для вышестоящего уровня сервис. Под протоколом понимается набор правил и форматов, определяющих взаимодействие объектов одного уровня модели OSI. Иерархически организованный набор протоколов, достаточный для организации взаимодействия узлов системы или сети передачи данных, называется *стеком или профилем* протоколов. В настоящее время используется большое число стеков, из которых наиболее популярны TCP/IP, IPX/SPX, NetBIOS/SMB, DECnet, SNA и OSI. Все стеки, кроме SNA, на нижних уровнях (канальном и физическом) используют одни и те же хорошо стандарти-

зированные протоколы для работы в локальных и глобальных сетях – Ethernet, Token Ring, ATM, PPP, V.24, V.25, V.35 другие. Из протоколов физической связи в нашей стране наибольшее распространение получили протоколы, представленные в табл. 7.

Таблица 7.

Название	Описание протокола
V.22 (ITU-T)	Дуплексный, симметричный, использует относительную фазовую модуляцию ОФМ (Differential Phase Shift Keying - DPSK), передающую информацию путем сдвига фазы несущего сигнала. Несущие частоты - 1200 и 2400 Гц, скорость модуляции - 600 Бод. Протокол имеет два режима, в одном из которых одной модуляцией передается один бит, а в другом - два бита (дибит). Соответственно, в первом случае имеется две, а во втором - четыре позиции модуляции с относительным сдвигом фазы на 180 и 90 градусов, а скорость передачи равна 600 и 1200 бит/с. Реализация протокола предусматривает наличие эквалайзера, корректирующего частотные и фазовые характеристики сигнала.
V.22 bis (ITU-T)	Развитие V.22 путем исключения однопозиционной и введения шестнадцати позиционной квадратурно-амплитудной модуляции с передачей четырех битов (квадбита) за одну модуляцию сигнала. Соответственно, максимальная скорость передачи увеличена до 2400 бит/с.
V.32 (ITU-T)	Использует шестнадцати позиционную КАМ и Trellis-кодирование, скорости передачи - 4800 и 9600 бит/с.
V.32 bis (ITU-T)	Расширение V.32 со скоростью передачи до 14400 бит/с, введены промежуточные скорости 7200 и 12000 бит/с. В протокол включена поддержка процедур автоматического изменения скорости во время сеанса при изменении качества линии, однако в ряде модемов реализованы лишь процедуры ее снижения без возврата к исходной скорости.
HST	Оригинальный помехоустойчивый несимметричный протокол с передачей в одну сторону со скоростью до 16800 бит/с, в обратном канале скорость фиксирована - 300 или 450 бит/с. Протокол автоматически ориентируется в сторону наиболее плотного потока данных; при потоках сравнимой плотности происходит периодический "разворот" протокола.
V.32 terbo (AT&T)	Расширение V.32bis со скоростью передачи до 19200 бит/с, промежуточная скорость - 16800 бит/с.
V.32 terbo/ASL (USR)	Расширение V.32bis со скоростью до 21600 бит/с. ASL - Adaptive Speed Leveling, адаптивная коррекция скорости в зависимости от качества передачи. Управление осуществляется по протоколу V.42. Поддерживаются быстрые пересоединения (retrain) без полной настройки систем эхоподавления. Начальное соединение для надежности выполняется на скорости 7200.

Продолжение табл. 7

Продолжение табл. 7

V.34 (ITU-T)	Протокол последнего поколения со скоростью передачи до 28800 бит/с, промежуточные скорости – 2400...26400 бит/с с дискретностью 2400. Принятию стандарта ITU предшествовали протоколы ряда производителей под названиями V.Fast и V.FC. Модуляция - 256-позиционная КАМ с дополнительным временным кодированием, при котором решение на приемном конце принимается по двум смежным состояниям сигнала. В связи с увеличением размера передаваемого за одну модуляцию элемента данных вместо понятия "бод" используется "символ в секунду"; в данном случае размер символа равен 8 битам, или одному байту. Соответственно, введено понятие "символьная скорость" - 2400, 2743, 2800, 3000, 3200, 3429 симв./с. Две последние скорости формально не укладываются в стандартную полосу пропускания телефонного тракта, однако ряд телефонных линий реально обладает нужной пропускной способностью.
ZYX (ZyXEL)	Оригинальный протокол со скоростью передачи от 7200 до 16800 бит/с в обычных моделях, и до 19200 бит/с - в моделях Plus. Дискретность изменения скорости - 2400 бит/с.
ZyCELL	Оригинальный помехоустойчивый протокол, ориентированный на работу по сотовым (cellular) линиям связи.
V.34bis (ITU-T)	Расширение V.34 до скорости 33600 бит/с с промежуточной скоростью 31200 бит/с.
V.90 (ITU-T)	Несимметричный, "полуцифровой" скоростной протокол, позволяющий поднять скорость передачи в одну сторону до 56 кбит/с. Стандарту предшествовали протоколы x2 (USR/3COM) и k56flex (Rockwell/Lucent). Данная группа протоколов известна также под названиями V.PCM и 56k. Протоколы 56k реализуются только на несимметричных линиях, когда с одной стороны устанавливается блок прямого сопряжения ("цифровой модем") с подключением к цифровому каналу T1/E1, ISDN и др., а с другой - аналоговый модем с поддержкой V.90. При таком соединении сигнал со стороны цифрового канала большую часть расстояния передается в неизменной цифровой форме, и только от абонентского комплекта до обычного модема - в аналоговой. Поскольку преобразование из цифровой формы в аналоговую сопряжено с меньшими потерями информации, чем обратно, предельная пропускная способность цифрового канала (64 кбит/с) понижается только до 56 кбит/с (реально обычно до 45-53 кбит/с). В обратную сторону предельной является скорость 33.6 кбит/с.

Протоколы 56k ориентированы в первую очередь, на централизованные системы связи - провайдеры Internet (ISP - Internet Service Provider), банковские и информационные сети и т.п., где преобладает передача информации от центра к абоненту (download), а передача от абонента к центру (upload) встречается гораздо реже.

При передаче информации между различными вычислительными системами должно применяться одинаковое кодовое представление используемых машинных слов и алфавитно-цифровых знаков. Например, прикладные программы взаимодействующих пользователей должны работать с одинаковыми кодовыми таблицами. Наибольшее распространение получили двоичный код BCD (Binary Code Digit), международный телеграфный код (МТК-5), а также ASCII (American Standard Code for Information Interchange) и EBCDIC (Extended Binary Coded Decimal Interchange Code) коды. Их основные виды и характеристики приведены в табл. 8.

Таблица 8.

Код	Область применения	Число бит	Число знаков кода
BCD	Цифровая информация	4	16
(МТК-5)	Телеграфия	5	32
ASCII	МиниЭВМ и микроЭВМ	7	128
EBCDI	Большие и миниЭВМ	8	256

Часто используются всевозможные национальные расширения перечисленных кодов, например, основная и альтернативная кодировки кириллицы для кода ASCII. В этом случае основание кода увеличивается до восьми бит. Задача согласования форматов данных, поступающих к пользователю, возлагается на уровень представления модели ISO.

Функции современных модемов относятся к наиболее "далеким" от пользователя уровням – физическому и канальному.

Передача данных организуется на основе набора протоколов, каждый из которых устанавливает правила взаимодействия связывающихся устройств. Протоколы, используемые в модемах, делятся на четыре основные группы:

- протоколы модуляции и передачи данных;
- протоколы коррекции ошибок;
- протоколы сжатия передаваемых данных;
- протоколы связи DTE и DCE.

Первые три группы относятся только к связи DCE-DCE, последняя - только к связи DCE-DTE.

Первая группа протоколов устанавливает правила вхождения модемов в связь, ее поддержания и разрыва, параметры аналоговых сигналов, правила кодирования и модуляции. Эти протоколы непосредственно относятся к сигналам, передаваемым по межмодемной аналоговой линии связи. Соединение двух модемов возможно только в случае поддержки ими каких-либо общих или совместимых протоколов этой группы. В семиуровневой иерархии протоколов связи OSI эта группа протоколов имеет уровень 1 (физический) и формирует канал цифровой связи в реальном времени, однако не защищенный от ошибок передачи.

Протоколы физической связи могут быть симплексными (simplex) - реализующими в каждый момент времени передачу только в одну сторону, и дуплексными (duplex) - с одновременной двунаправленной передачей. Чаще всего применяются дуплексные протоколы, которые могут быть симметричными, когда скорости передачи в обоих направлениях равны, и несимметричными, когда скорости различаются. Несимметричный дуплекс применяется для повышения скорости передачи в одну сторону за счет ее снижения в обратную сторону, когда поток передаваемых данных имеет выраженную асимметрию.

Для определения направления передачи в физическом канале использу-

ются понятия вызывающего (инициирующего соединение) и отвечающего модемов; направление передачи определяется со стороны вызывающего модема.

Вторая группа устанавливает правила обнаружения и коррекции ошибок, возникающих на этапе передачи с помощью протоколов первой группы. Эти протоколы имеют дело только с цифровой информацией. Для проверки целостности информации, она разделяется на блоки (пакеты), снабжаемые контрольными избыточными кодами (CRC - Cyclic Redundancy Check). При несовпадении контрольного кода на приемном конце переданный пакет считается ошибочным, и запрашивается его повторная передача. Эта группа протоколов формирует из ненадежного физического канала надежный (защищенный от ошибок) канал более высокого уровня, однако это приводит к потере связи в реальном времени и дается ценой определенных накладных расходов. В модели OSI эта группа соответствует уровню 2 (канальный).

Третья группа устанавливает правила сжатия передаваемых данных путем уменьшения их избыточности. При этом на передающем конце происходит их анализ и упаковка, а на приемном - распаковка в исходный вид. Сжатие позволяет повысить скорость передачи сверх физической пропускной способности канала за счет уменьшения объема реально передаваемых данных. Реализация сжатия также требует некоторых накладных расходов на анализ информации и формирование пакетов. В случае неэффективного сжатия скорость передачи может оказаться ниже скорости физического канала.

Последняя группа протоколов задает правила взаимодействия DCE и DTE. Они подразделяются на физические, касающиеся кабелей, разъемов и сигналов взаимодействия, и информационные, относящиеся к формату и смыслу передаваемых сообщений. Посредством этих протоколов реализуется общение DTE и DCE во время подготовки к вхождению в связь, организации вызова и ответа, а также в процессе самого обмена данными.

3.4. Классификация модемов

Строгой классификации модемов не существует и, вероятно, не может существовать по причине большого разнообразия, как самих модемов, так и сфер применения и режимов их работы. Тем не менее, можно выделить ряд признаков, по которым и возможно провести условную классификацию. К таким признакам можно отнести следующие:

- По области применения: для передачи данных; факсимильные модемы; комбинированные модемы.
- По функциональному назначению.
- По реализации протоколов модуляции, исправления ошибок и сжатия данных.
- По технической реализация: программные модемы; аппаратные модемы.
- По типу используемого канала: модемы для коммутируемых каналов; модемы для арендованных каналов; комбинируемые модемы.
- По скорости передачи информации: малоскоростные модемы; среднескоростные; высокоскоростные.
- По исполнению: внутренние (изготовленные в виде плат); внешние (имеющие разъемы для подключения телефонной линии, телефонного аппарата, управляющего терминала).

Область применения. Современные модемы можно разделить на несколько групп:

- для коммутируемых телефонных каналов;
- для выделенных (арендуемых) телефонных каналов;
- для физических соединительных/абонентских линий (xDSL) модемы;
- для цифровых систем передачи (CSU/DSU);
- для сотовых систем связи;
- для пакетных радиосетей;
- для спутниковых каналов связи;
- для локальных радиосетей;
- для телевизионных кабельных сетей.

Подавляющее большинство выпускаемых модемов предназначено для использования на коммутируемых телефонных каналах. Такие модемы должны уметь работать с АТС, различать их сигналы и передавать свои сигналы набора номера.

Основное отличие модемов для физических линий от других типов модема состоит в том, что полоса пропускания физической линии зависит, в основном, от типа физической среды (витая пара, коаксиальный кабель и др.) и ее длины. С точки зрения используемых для передачи сигналов, модемы для физических линий могут быть разделены на модемы низкого уровня (линейные драйверы), использующие цифровые сигналы, и модемы "основной полосы" (baseband), использующие аналоговые сигналы (как в телефонных каналах).

В модемах низкого уровня обычно используются цифровые методы биимпульсной передачи, позволяющие формировать импульсные сигналы без постоянной составляющей и часто занимающие более узкую полосу частот, чем исходная цифровая последовательность.

В модемах "основной полосы" часто используются различные виды квадратурной амплитудной модуляции, позволяющие радикально сократить требуемую для передачи полосу частот. В результате, на одинаковых физических линиях такими модемами может достигаться скорость передачи до 100 кбит/с, в то время как модемы низкого уровня обеспечивают только 19,2 кбит/с.

Модемы для цифровых систем передачи напоминают модемы низкого уровня. Однако в отличие от них обеспечивают подключение к стандартным цифровым каналам, таким как E1/T1 или ISDN, и поддерживают функции соответствующих канальных интерфейсов.

Модемы для сотовых систем связи отличаются компактностью исполнения и поддержкой специальных протоколов модуляции исправления ошибок, позволяющих эффективно передавать данные в условиях помех и постоянно изменяющимися параметрами. Среди таких протоколов выделяются ZyCELL, ETC и MNP10.

Пакетные радиомодемы предназначены для передачи данных по радиоканалу между мобильными пользователями. При этом несколько модемов используют один и тот же радиоканал в режиме множественного доступа, например, множественного доступа с контролем несущей, в соответствии с AX.25. Радиоканал по своим характеристикам близок к телефонному и организуется с использованием типовых радиостанций, настроенных на одну и ту же частоту в УКВ либо КВ диапазоне.

Локальные радиосети являются быстроразвивающейся перспективной сетевой технологией, дополняющей обычные локальные сети. Ключевым их элементом являются специализированные радиомодемы (адаптеры локальных радиосетей). В отличие от упомянутых пакетных радиомодемов, такие модемы обеспечи-

вают передачу данных на небольшие расстояния (до 300 м) с высокой скоростью (2-10 Мбит/с), сопоставимой со скоростью передачи в проводных локальных вычислительных сетях (ЛВС). Кроме того, радиомодемы локальных радиосетей работают в определенном диапазоне частот с применением сигналов сложной формы, таких как сигналы с псевдослучайной перестройкой рабочей частоты.

Модемы для телевизионных кабельных сетей используют свободные телевизионные каналы с полосой пропускания в 6 МГц в диапазоне от 60 до 450 МГц для передачи своих сигналов. Большие полосы пропускания ТВ каналов обуславливают и высокие скорости передачи, которые достигают порядка 36 Мбит/с.

3.5. Устройство модемов

Практически все современные модемы имеют похожие функциональные схемы (рис. 31), состоящие из адаптеров портов канального DCE и DTE–DCE интерфейсов, универсального (PU) и цифрового сигнального (DSP) процессоров, постоянного запоминающего устройства (ПЗУ, ROM), оперативного запоминающего устройства (ОЗУ, RAM), перепрограммируемого запоминающего устройства (Non – Volatile RAM, NVRAM – не разрушающаяся память с прямым доступом), собственно модулятора/демодулятора, схемы индикаторов состояния модема и динамика.

Порт интерфейса DTE–DCE обеспечивает взаимодействие с DTE. Если модем внутренний, вместо интерфейсов DTE-DCE может применяться интерфейс внутренней шины компьютера ISA или PSI. Порт канального интерфейса обеспечивает согласование электрических параметров с используемым каналом связи.

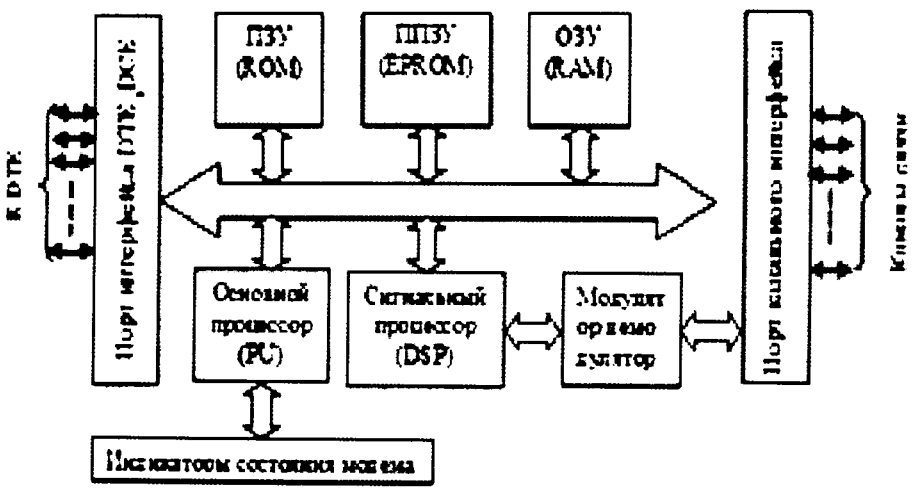


Рис. 31. Обобщенная структурная схема модема

Основной процессор фактически является встроенным микрокомпьютером, отвечающим за прием и выполнение команд, буферизацию и обработку данных – кодирование, декодирование, сжатие/распаковку и т.п., а также за управление сигнальным процессором. В большинстве модемов используются специализированные процессоры на основе типовых наборов микросхем, а в некоторых (US Robotics, ZyXEL) – процессоры общего назначения (Intel, Zilog, Motorola).

На сигнальный процессор (DSP, Digital signal Processor – цифровой сиг-

нальный процессор), как правило, возлагаются задачи по реализации основных функций протоколов модуляции (кодирование сверточным кодом, относительное кодирование, скремблирование и т.д.), за исключением собственно операций модуляции/демодуляции. Последние операции обычно выполняются специализированным модемным процессором – модулятором/демодулятором. В качестве таких процессоров используются либо специализированные, ориентированные на конкретный набор способов и протоколов модуляции (AT&T, Rockwell, Exar), либо универсальные - со сменной микропрограммой (например, TMS), позволяющие дорабатывать и изменять алгоритмы работы.

В зависимости от типа и сложности модема, основная интеллектуальная нагрузка смещается в сторону DSP или модулятора/демодулятора. В низкоскоростных (300...2400 бит/с) модемах основную работу выполняет модулятор/демодулятор, в скоростных (4800 бит/с и выше) – DSP.

В ПЗУ хранятся программы для основного и сигнального процессоров (firmware). ПЗУ может быть однократно программируемым (PROM), перепрограммируемым со стиранием ультрафиолетом (EPROM) или перепрограммируемым электрически (EEPROM, flash ROM). Последний тип ПЗУ позволяет оперативно менять прошивки по мере исправления ошибок или появления новых возможностей.

ОЗУ используется в качестве временной памяти при работе основного и сигнального процессоров; оно может быть как раздельным, так и общим. В ОЗУ хранится также текущий набор параметров модема (active profile).

В NVRAM хранятся сохраненные наборы параметров модема (stored profiles), один из которых загружается в текущий набор при каждом включении или сбросе. Обычно имеется два сохраненных набора – основной (profile 0) и дополнительный (profile 1). По умолчанию для инициализации используется основной набор, но есть возможность переключиться на дополнительный. Ряд модемов имеет более двух сохраненных наборов.

Схемы согласования с линией включают разделительный трансформатор для передачи сигнала, оптопару для опознания сигнала звонка (Ring), реле подключения к линии ("поднятия трубки", off-hook) и набора номера, а также элементы создания нагрузки в линии и защиты от перенапряжений. Вместо реле могут применяться бесшумные электронные ключи. В некоторых модемах применяются дополнительные оптопары для контроля напряжения в линии. Подключение к линии и набор номера могут выполняться как одним, так и раздельными ключами. На динамик (speaker) выводится усиленный сигнал с линии для слухового контроля ее состояния. Динамик может быть включен на время набора номера и соединения, во время всего соединения, а также отключен совсем.

Внешние модемы дополнительно содержат схему формирования питающих напряжений (обычно +5, +12 и −12В) из одного переменного (реже – постоянного) напряжения источника питания. Кроме этого, внешние модемы содержат интерфейсные цепи для связи с DTE.

3.6. Характеристики процесса передачи данных

Любая коммуникационная сеть должна включать следующие основные компоненты: сообщение, средства передачи, приемник.

Сообщение – цифровые данные определенного формата, предназначенные для передачи. Сообщением может быть файл базы данных, таблица, ответ на запрос, текст или изображение.

Приемник – устройство, принимающее данные.

Средства передачи – физическая передающая среда и специальная аппаратура, обеспечивающая передачу сообщений.

Для передачи сообщений в вычислительных сетях используются различные типы каналов связи. Наиболее распространены выделенные телефонные каналы и специальные каналы для передачи цифровой информации. Применяются также радиоканалы и каналы спутниковой связи.

Особняком в этом отношении стоят локальные вычислительные сети, где в качестве передающей среды используются витая пара проводов, коаксиальный кабель и оптоволоконный кабель.

Для характеристики процесса обмена сообщениями в вычислительной сети используются следующие понятия: режим передачи, код передачи, тип синхронизации.

Режим передачи. Существуют три режима передачи: симплексный, полудуплексный и дуплексный.

Симплексный режим – передача данных только в одном направлении (рис. 32.).

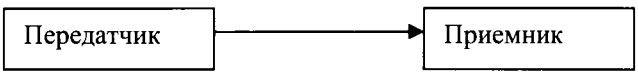


Рис. 32. Симплексный режим передачи

Примером симплексного режима передачи является система, в которой информация, собираемая с помощью датчиков, передается для обработки на ЭВМ. В вычислительных сетях симплексная передача, практически. не используется.

Полудуплексный режим – попеременная информация, когда источник и приемник последовательно меняются местами (рис. 33.).

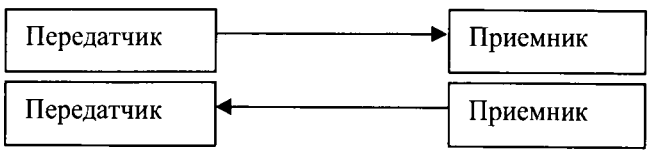


Рис. 33. Полудуплексный режим передачи

Яркий пример работы в полудуплексном режиме – разведчик, передающий в Центр информацию, а затем принимающий инструкции из центра.

Дуплексный режим – одновременная передача и прием сообщений. Пример дуплексного режима – телефонный разговор.

Дуплексный режим (рис. 34) является наиболее скоростным режимом работы и позволяет эффективно использовать вычислительные возможности быстродействующих ЭВМ в сочетании с высокой скоростью передачи данных по каналам связи.

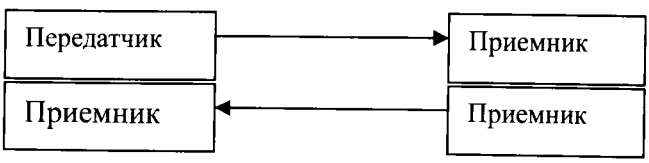


Рис. 34. Дуплексный режим передачи

Коды передачи данных

В компьютерной сети ЭВМ могут быть объединены в комплекс с помощью интерфейсного кабеля или с помощью двухпроводной линии связи. Интерфейсный кабель – это набор проводов, число которых равно числу бит в передаваемой кодовой комбинации. Для каждого бита отводится отдельный провод.

Если ЭВМ связаны между собой интерфейсным кабелем, то в этом случае, имеет место передача данных *параллельным кодом*. При этом каждая кодовая комбинация передается за время, равное одному такту (периоду тактовых импульсов). Предпочтение такой передаче отдается при организации локальных вычислительных сетей при небольших расстояниях между абонентами сети. Передача параллельным кодом обеспечивает высокое быстродействие, но требует повышенных затрат на создание физической передающей среды и обладает плохой помехозащищенностью.

При передаче кодовой комбинации по двухпроводной линии группа битов передается по одному проводу бит за битом. Это передача информации *последовательным кодом*. Она осуществляется с меньшей скоростью, так как только на передачу одной кодовой комбинации требуется несколько тактов, число которых определяется количеством переданных бит. Несмотря на это, передача последовательным кодом экономически более выгодна и широко используется в распределенных сетях на большом расстоянии абонентов друг от друга.

Синхронизация данных

Процессы передачи и приема информации в вычислительных сетях могут быть привязаны к определенным временным отметкам, определяющим начало и конец посылок с передаваемыми данными. Такие процессы называются синхронными.

Синхронизация данных – это установление временного соответствия между процессами передачи и приема данных в вычислительных сетях.

В системах передачи данных используются два способа передачи данных: синхронный и асинхронный.

При *синхронной передаче* (рис. 35) данные передают побитно, группируя их в отдельные блоки (пакеты) различной длины, снабженные заголовком и контрольной суммой.

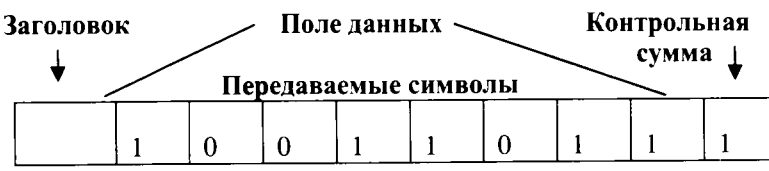


Рис. 35. Синхронная передача данных

В состав блока, кроме информационных символов, включаются также и служебные символы, обеспечивающие контроль состояния физической передающей среды. При синхронной передаче в конце блока данных в канал связи выдается контрольная сумма, в виде символа, дополняющего число единиц в информационной посылке до четного числа. Данный символ позволяет, путем проверок на четность, обнаруживать на приемной стороне одиночные ошибки в передаваемых сообщениях и разделять (нарезать) принимаемый поток данных на исходные

блоки (пакеты) данных.

Синхронная передача – высокоскоростная и почти безошибочная. Синхронная передача требует дорогостоящего оборудования.

При асинхронной передаче (рис. 36) данные передаются побайтно. При этом каждый байт ограничивается стартовым и стоповым битами. В линиях связи с низкой надежностью используют несколько таких битов.

Рис. 36. Асинхронная передача данных

Стартовые и стоповые биты между байтами обеспечивают правильное опознание начала и конца каждого байта. Таким образом, минимальной единицей передачи данных является байт. Дополнительные стартовые и стоповые биты несколько снижают эффективную скорость передачи данных и, соответственно, пропускную способность канала связи. В то же время, асинхронная передача не требует дорогостоящего оборудования и отвечает требованиям организации диалога в вычислительной сети при взаимодействии персональных ЭВМ.

Защита от ошибок

При передаче по телефонным каналам сигнал может быть искажен помехами. Для защиты от ошибок данные, передаваемые модемом, разбиваются на группы, называемыми пакетами или кадрами. В каждый передаваемый пакет, кроме информации, вводится специальная последовательность двоичных символов, называемая проверочной комбинацией. Проверочная комбинация является результатом сложной математической операции, проводимой над данными. При приеме данных удаленным модемом над пакетом проводится аналогичная операция, и полученная проверочная комбинация сравнивается с принятой от передающего модема проверочной комбинацией. Если имеет место несовпадение сравниваемых проверочных комбинаций, следовательно, при передаче пакета произошла ошибка, и модем запрашивает повторную передачу такого пакета до тех пор, пока он не будет принят без ошибок. Благодаря этому, данные между модемами могут быть переданы, практически без ошибок.

Радикальным методом помехоустойчивости приема стало применение сочетания модуляции с «решетчатым» кодированием. «Решетчатое» кодирование-метод исправления ошибок, применяемый в модемах на скоростях передачи выше 4800 бит/сек. Для создания возможности кодирования, пространство сигналов расширяется путем добавления к каждой группе информации битов, на которые делится поток при модуляционном кодировании, дополнительного бита. Этот бит образуется путем выполнения операций сверхточного кодирования над частью указанных групп битов информации сигнала.

Расширенная таким образом группа битов представляется затем в виде вектора многопозиционного пространства модулируемых сигналов и передается в канал связи. На приеме, с помощью декодера, выполняется сверхточное декодирование сигналов, позволяющее на основе анализа корреляционных связей между группами битов осуществить исправление значительной части ошибок.

Очень редко данные, передаваемые модемом, носят чисто случайный характер, и очень часто они содержат много повторяющихся знаков, например знаков пробела и др. Для устранения такой избыточности в модемах выполняется дополнительное кодирование данных таким образом, что более часто встречающиеся знаки передаются с помощью более коротких последовательностей, а более редко встречающиеся с помощью длинных последовательностей. При этом используется определенный справочник сокращенных комбинаций, фиксированных или адаптивно изменяемых.

В результате, для передачи одного и того же объема информации требуется передача меньшего количества двоичных символов. Это позволяет сократить время на передачу информации, что уменьшает риск искажения и утери данных. В ряде типов модемов применяются другие методы повышения эффективности передачи данных. Эти методы базируются на оценке качества приема последовательности (sequence estimation). Результаты оценки используются для соответствующего изменения структуры передаваемых данных (скорости передачи, длины пакета и т.д.).

3.7. Реализация передачи данных

Цифровые данные по проводной линии передаются путем смены текущего напряжения: нет напряжения – "0", есть напряжение – "1". Существуют два способа передачи информации по физической передающей среде: цифровой и аналоговый.

При *цифровом* или *узкополосном способе* передачи (рис. 37) данные передаются в их естественном виде на единой частоте.

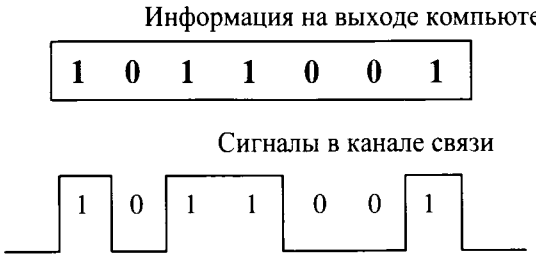


Рис. 37. Цифровой способ передачи

Узкополосный способ обеспечивает возможность одновременного использования передающей среды только двумя пользователями и допускает нормальную работу на ограниченном расстоянии (длина линии связи не более 1000 м).

В то же время, узкополосный способ передачи обеспечивает высокую скорость обмена данными – до 10 Мбит/с и позволяет создавать легко конфигурируемые вычислительные сети. Подавляющее число локальных вычислительных сетей используют узкополосную передачу.

Аналоговый способ передачи цифровых данных (рис. 38) обеспечивает широкополосную передачу за счет использования в одном канале сигналов различных несущих частот.

При аналоговом способе передачи происходит управление параметрами сигнала несущей частоты в соответствии с передаваемым сигналом.

Сигнал несущей частоты представляет собой гармоническое колебание, описываемое уравнением

$$u(t) = U_0 cos(\omega t + \varphi_0),$$

где: U_0, ω, φ_0 – соответственно, амплитуда, частота и начальная фаза колебаний; t – текущее время.

Передать цифровые данные по аналоговому каналу можно, управляя одним из параметров сигнала несущей частоты: амплитудой, частотой или фазой. Так как необходимо передавать данные в двоичном коде (последовательность единиц и нулей), то можно предложить следующие способы управления (модуляции): амплитудный, частотный, фазовый.

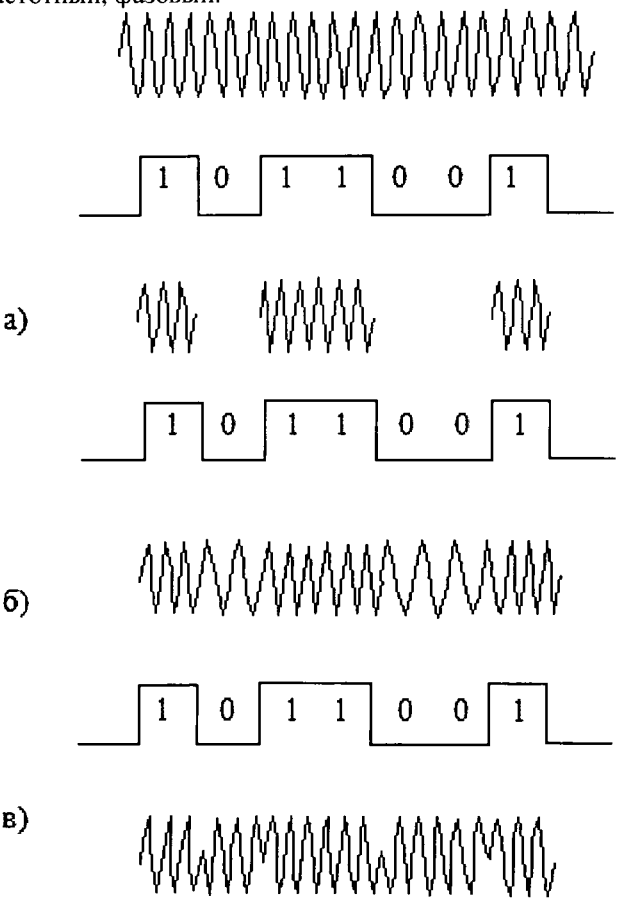


Рис. 38. Способы передачи цифровых данных по аналоговому каналу
а) – амплитудная, б) – частотная, в) – фазовая модуляции

Наиболее просто осуществляется *амплитудная модуляция*: "0" – отсутствие сигнала, т.е. колебаний несущей частоты; "1" – наличие сигнала, т.е. наличие колебаний несущей частоты.

Частотная модуляция предусматривает передачу сигналов 0 и 1 на разной частоте. При переходе от 0 к 1 и обратно происходит изменение частоты сигнала несущей частоты (см. рис. 38.б).

Наиболее сложной для понимания является *фазовая модуляция*. Суть ее в том, что в момент смены символа (перехода от 0 к 1 и от 1 к 0) имеет место пере-

скок фазы несущих колебаний. На рис. 38.в этот перескок составляет $\Delta\varphi = \pi$, что соответствует 180°. В этом случае направление колебаний изменяется на противоположное.

3.8. Факсимильная связь

Факсимильная связь предназначена для передачи на расстояние информации в виде текстов, чертежей, рисунков, схем, фотоснимков и т.п. По существу, факсимильный способ передачи информации заключается в дистанционном копировании документов. В основу факсимильной связи положен метод передачи временной последовательности электрических сигналов, отображающих яркость отдельных элементов передаваемого документа. Разложение передаваемого изображения на элементы называется *разверткой*, а просмотр и считывание этих элементов называется *сканированием*. Скорость развертки при передаче элементов изображения составляет 60, 90, 120, 180, 240 строк в минуту. Сигнал яркости, пропорциональный коэффициенту отражения элементов изображения, преобразуется в цифровой вид (код) и передается по каналу связи с использованием того или иного способа модуляции.

Для организации факсимильной связи используют чаще всего, телефонные каналы, реже – телеграфные каналы и радиоканалы связи. Структурная схема факсимильной связи приведена на рис. 39.

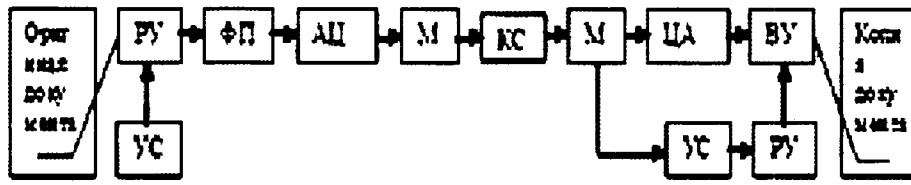


Рис. 39. Структурная схема факсимильной связи

Изображение (оригинал), подлежащее передаче, подвергается сканированию световым пучком требуемых размеров. Пучок формируется светооптической системой, содержащей источник света и фокусирующее устройство. Перемещение пучка по поверхности оригинала осуществляется развертывающим устройством (РУ). Часть светового потока, падающего на элементарную площадку оригинал, отражается и поступает на фотоэлектрический преобразователь (ФП), в котором происходит его преобразование в электрический видеосигнал. Амплитуда видеосигнала пропорциональна величине отраженного светового потока. Далее видеосигнал поступает на вход аналого-цифрового преобразователя (АЦП), где преобразуется в цифровой код. Этот код поступает на вход модулятора (модема), где посредством использования одного из протоколов модуляции спектр цифрового видеосигнала переносится в область частот используемого канала связи.

На приемной стороне приходящий из канала связи модулированный сигнал последовательно обрабатывается в демодуляторе модема и цифро-аналоговом преобразователе (ЦАП). Далее видеосигнал поступает в воспроизводящее устройство (ВУ), где с помощью развертывающего устройства (РУ) на бланке воспроизводится копия переданного изображения. Процесс получения конечной факсимильной копии, обратный процессу сканирования, носит название репликации. Для обеспечения синхронности и синфазности разверток на передающей и приемной сторонах используются устройства синхронизации.

Таким образом, аппарат факсимильной связи (факс) очень напоминает ксерокс, в котором оригинал и копию разделяют многие километры.

Современные факс-модемы имеют в своем составе все составные части факсимильных аппаратов за исключением сканирующего и воспроизводящего устройств. Они «умеют» связываться с обыкновенными факсами, при этом принимаемая информация о передаваемом изображении передается на компьютер, где программой передачи факсимильных сообщений преобразуется в один из распространенных графических форматов. В дальнейшем, полученный таким образом документ можно отредактировать, вывести на принтер или передать другому корреспонденту, имеющему факс или компьютер с факс-модемом.

3.9. Факсимильные аппараты

Выпускаемые в настоящее время факсимильные аппараты отличаются способом воспроизведения изображения, видом развертки и разрешающей способностью.

По способу воспроизведения изображения факсимильные аппараты классифицируются следующим образом: фотографические, электрохимические, электромеханические, электрографические, термографические, струйные и лазерные.

Фотографические факсимильные аппараты лучше других передают полутона и имеют высокую разрешающую способность (до 10 точек/мм), но используют дорогую фотографическую бумагу.

Разрешающая способность *электрохимических* и *электромеханических* аппаратов примерно одинакова – 4-6 точек/мм, но электромеханические аппараты не передают полутонов (их часто называют штриховыми аппаратами). Достоинствами электромеханических аппаратов являются использование обычной бумаги и простота конструкции.

Современные аппараты, чаще всего *термографического* типа: они недорогие и имеют достаточно хорошие характеристики: 7-10 точек/мм, 20 уровней серого цвета. Примерно такого же класса электрографические и струйные факсимильные аппараты, их важная особенность – использование обычной бумаги. Лучшие характеристики имеют *лазерные* факсимильные аппараты: до 15 точек/мм, 64 уровней серого, но они сравнительно дорогие.

По виду развертки факсимильные аппараты делятся на *плоскостные* (Xerox 7024, Panafax UF-60V, “Березка”) и *барабанные* (“Нева”,Xerox 7245, Panasonic KX-F700 BX).

В плоскостных аппаратах передаваемые документы ограничиваются размером только по ширине (может передаваться рулонный документ), а в барабанных – и по ширине и по длине.

Согласно классификации МККТТ факсимильные аппараты делятся на четыре группы (табл. 9).

Таблица 9.

Группа	Тип передачи сообщений	Время передачи А4, с	Разрешающая способность
1	Аналоговые ЧМ сигналы	до 360	от 4 точ./мм
2	Аналоговые АМ сигналы	до 180	от 4 точ./мм
3	Кодирование со сжатием	до 60	7-9 точ./мм
4	Высокоскоростная передача	5-10	до 16 точ./мм

Скорость передачи факсимильной информации по телефонным каналам связи лежит в пределах 4800 – 14440 бод (стандарт МККТТ V.34).

4. Безопасность установки и эксплуатации компьютерного оборудования

Общеизвестно, что компьютерная техника является источником излучений и электромагнитных полей, потенциально опасных для здоровья человека, особенно при неправильном ее использовании. Исследованиям проблем электромагнитной безопасности компьютеров давно занимаются специалисты в нашей стране и за рубежом. Однако, специальных публикаций, доступных для широкого круга читателей, о специфике проблемы и способах обеспечения безопасности людей, работающих долгое время за компьютерами, и, особенно, в местах их скопления (банках, компьютерных классах учебных заведений, предприятиях и учреждениях), пока нет.

Цель данной главы - помочь пользователям персональных вычислительных машин (ПК): преподавателям информатики в учебных заведениях, специалистам по технике безопасности на предприятиях и учреждениях - понять суть проблемы, познакомить их с методами правильного использования ПК, обучить их достаточно простым практическим способам обеспечения электромагнитной безопасности компьютерной техники.

4.1. Основные понятия

В докомпьютерную эпоху человеческого развития основными техническими средствами интенсификации умственного труда инженеров и финансовых работников были логарифмическая линейка, счеты и предел мечтаний - арифмометр. Эти нехитрые механические устройства не оказывали, практически, никакого влияния на среду их применения и, соответственно, на пользователя, а потому и не привлекали к себе внимание экологов, гигиенистов и службы охраны труда.

Появление в практике настольных электронно-счетных машин и калькуляторов не внесло ничего нового в экологию рабочих мест, т.к. по своим пространственным и частотным характеристикам электромагнитных полей указанные устройства мало отличались от широко применяемых настольных ламп и вентиляторов.

Стремительное развитие компьютерной техники привело к двум (с точки зрения гигиены труда) важным явлениям:

- Во-первых - на рабочих местах пользователей этого нового достижения техники появились сложные электронные устройства, обладающие не только пространственными свойствами традиционных потребителей электроэнергии промышленной частоты 50 Гц, но и генерирующие внутри себя целый спектр электрических сигналов различной частоты и интенсивности;
- Во-вторых - стал резко расширяться круг пользователей современной техники - от узких специалистов до многочисленных менеджеров и руководителей не только предприятий и фирм, но и государства; что особенно важно, новая техника вошла в быт, стала доступной, в том числе и детям, - как дома, так и в школьных и дошкольных учреждениях.

Как известно, непременной составляющей персонального компьютера является дисплей (синоним - "монитор" или обобщающий термин - "видеодисплейный терминал" - ВДТ), обеспечивающий связь машины с оператором. Известно также, что дисплей является порождением телевизионной техники, и это последнее обстоятельство привело к возникновению проблемы. Дело в том, что вокруг работающего телевизора (или дисплея) из-за наличия высокого напряжения и ши-

рокого спектра электрических сигналов образуются статические и переменные электрические и магнитные поля (далее – электромагнитные поля), отрицательные результаты воздействия которых на человека хорошо изучены.

В области телевидения проблема обеспечения санитарно-гигиенических требований по исключению воздействия на зрителя электромагнитных полей была решена элементарно просто, исходя из того физического факта, что интенсивность этих низкочастотных электрических и магнитных полей резко падает при удалении от их источника. Поэтому достаточно было в инструкциях по использованию телевизоров внести запись о предпочтительном просмотре телепрограмм с расстояния не менее 2 - 3-х метров и проблемы не стало. При этом не требовалось ни разработки каких-либо норм и нормативных документов, ни проведения доработки указанных технических средств, ни применения дополнительных средств защиты при их использовании.

В компьютерной технике проблема состоит в том, что электрические и магнитные поля от дисплеев столь же интенсивны, как и от телевизоров, а усадить пользователя компьютера на расстояние 2 - 3 метра от дисплея невозможно. Таким образом, пользователь компьютера волей-неволей должен быть близок к дисплею, подвергая себя воздействию этих полей. Именно это обстоятельство привело к появлению многочисленных сведений об отрицательных последствиях такой "близости".

4.2. Источники и характеристики электромагнитных полей на рабочем месте с ПК

Современный персональный компьютер является энергонасыщенным аппаратом с потреблением до 200-250 Вт, содержащим несколько электро- и радиоэлектронных устройств с различными физическими принципами действия. Поэтому он создает вокруг себя поля с широким частотным спектром и пространственным распределением, такие как:

- электростатическое поле;
- переменные низкочастотные электрические поля;
- переменные низкочастотные магнитные поля.

Потенциально возможными вредными факторами могут быть также:

- рентгеновское и ультрафиолетовое излучения электронно-лучевой трубки (ЭЛТ) дисплея ПК;
- электромагнитное излучение радиочастотного диапазона;
- электромагнитный фон (электромагнитные поля, создаваемые сторонними источниками на рабочем месте с компьютерной техникой).

Следует отметить, что рентгеновское и ультрафиолетовое излучения экранов ВДТ можно назвать лишь потенциально существующими вредными факторами. Дело в том, что экраны современных дисплеев делают из стекла, не прозрачного для рентгеновского излучения, возникающего в трубке, а ультрафиолетовое излучение при испытаниях не обнаруживается даже в самых старых моделях дисплеев. Излучения радиочастотного диапазона от электронных узлов компьютерной техники также существенно ниже предельно допустимых уровней, регламентируемых санитарными нормами.

Электростатическое поле возникает за счет наличия электростатического потенциала (ускоряющего напряжения) на экране ЭЛТ. При этом появляется разность потенциалов между экраном дисплея и пользователем ПК. Наличие электростатического поля в пространстве вокруг ПК приводит, в том числе, к тому,

что пыль из воздуха оседает на клавиатуре ПК и затем проникает в поры на пальцах, вызывая заболевания кожи рук.

Электростатическое поле вокруг пользователя ПК зависит не только от полей, создаваемых дисплеем, но также от разности потенциалов между пользователем и окружающими предметами. Эта разность потенциалов возникает, когда заряженные частицы накапливаются на теле человека в результате ходьбы по полу с ковровым покрытием, при трении материалов одежды друг о друга и т.п.

В современных моделях дисплеев приняты кардинальные меры для снижения электростатического потенциала экрана. Но нужно помнить, что разработчиками дисплеев применяются различные технические способы для борьбы с данным фактором, в том числе и, так называемый, компенсационный способ, особенность которого заключается в том, что снижение потенциала экрана до требуемых норм обеспечивается лишь в установившемся режиме работы дисплея. Соответственно, подобный дисплей имеет повышенный (в десятки раз более установившегося значения) уровень электростатического потенциала экрана в течение 20 - 30-ти секунд после своего включения и до нескольких минут после выключения; что достаточно для электризации пыли и близлежащих предметов.

Источниками переменных электрических и магнитных полей в ПК являются узлы, в которых присутствует высокое переменное напряжение, и узлы, работающие с большими токами. Типичные пространственные распределения переменного магнитного поля и переменного электрического поля вокруг дисплея ПК показаны на рис. 40 и рис. 41, соответственно.

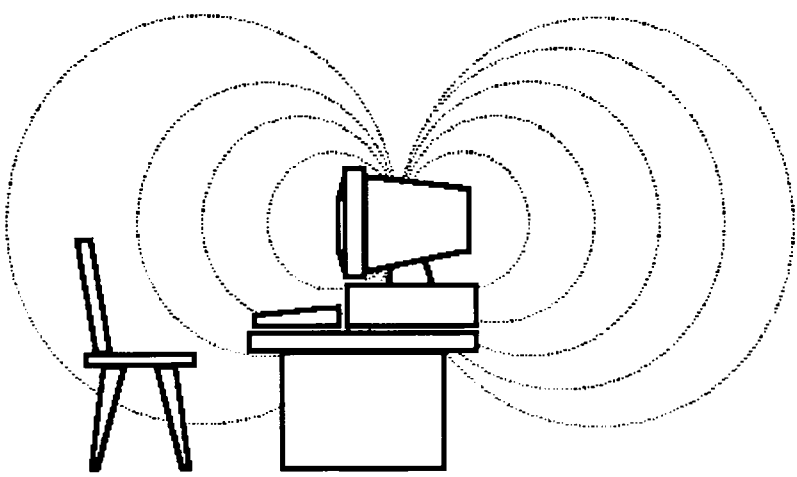


Рис. 40. Силовые линии магнитного поля вокруг дисплея

По частотному спектру эти электромагнитные поля разделяются на две группы:

- поля, создаваемые блоком сетевого питания и блоком кадровой развертки дисплея (основной энергетический спектр этих полей сосредоточен в диапазоне частот до 1 кГц);
- поля, создаваемые блоком строчной развертки и блоком сетевого питания ПК (в случае, если он импульсный); основной энергетический спектр этих полей сосредоточен в диапазоне частот от 15 до 100 кГц.

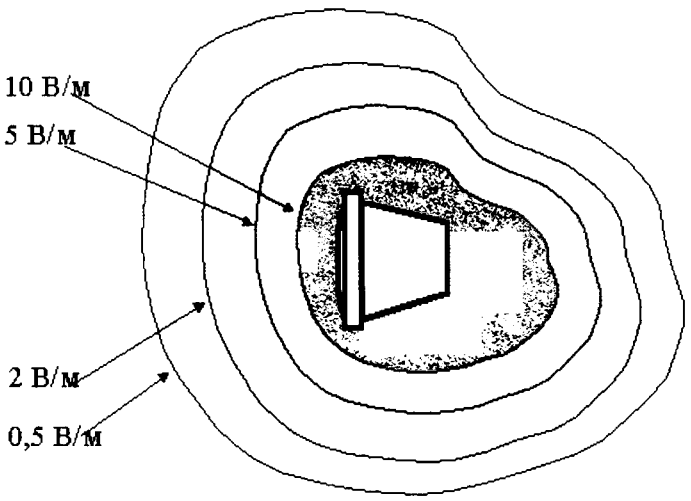


Рис. 41. Пространственная диаграмма распределения в горизонтальной плоскости интенсивности электрического поля вокруг монитора ПК

Электромагнитные поля, порожденные посторонними (не входящими в состав ПК) источниками, называют иногда фоновыми полями. Характер этих полей, их пространственное распределение и уровни определяются физическими особенностями источников, положением их по отношению к рабочему месту. Часто фоновые поля имеют общий источник - сеть электропитания, дающую существенный вклад в общий энергетический спектр полей на частоте 50 Гц и ее гармониках. Это вклад во многом зависит от организации электросети и контура заземления, удаленности и расположения рабочего места относительно розеток питания и других элементов сети. Источниками фоновых низкочастотных полей являются также другие технические средства, в том числе бытовые (кондиционеры, вентиляторы, пылесосы, кухонная техника), а также массивные не заземленные металлические предметы (решетки, стеллажи и т.п.).

Особого внимания требуют случаи появления экстремальных электрических и магнитных полей посторонних источников, которые могут не только многократно превышать гигиенические требования, но и нарушать нормальную работу ПК и другой, связанной с ними техники. Так, например, магнитное поле промышленной частоты 50 Гц с напряженностью более тысячи нанотесла (1 мкТл) вызывает заметную для глаз пространственную и временную нестабильность (дрожание и мерцание) изображения на экране дисплея ПК с частотой, равной разности между частотой кадровой развертки дисплея и частотой 50 Гц.

В таких случаях возникают эффекты опосредованного влияния на оператора ПК магнитного поля промчастоты 50 Гц. (см. диаграмму на рис. 42):

1. Непосредственное влияние магнитного поля на оператора ПК;
2. Воздействие магнитного поля на отклоняющую систему дисплея, вызывающее нестабильность изображения на его экране;
3. Дискомфорт, повышенная утомляемость при восприятии нестабильного изображения оператором ПК.

Наличие механизмов неблагоприятного опосредованного влияния магнитных полей на человека является еще одной отличительной особенностью при

использовании ПК в сфере жизнедеятельности человека по сравнению с использованием им других технических средств.

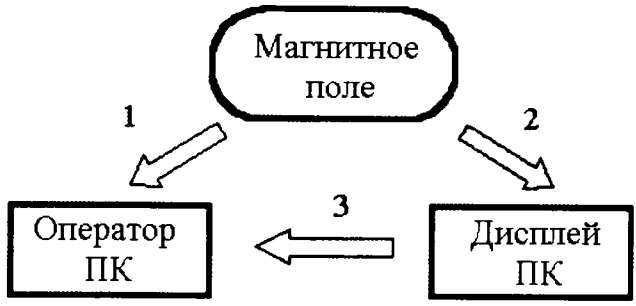


Рис. 42. Схема опосредованного влияния магнитного поля промчастоты 50 Гц на оператора ПК

Выявление, анализ и устранение повышенных и экстремальных магнитных полей промчастоты 50 Гц представляет серьезную самостоятельную задачу. Исследование причин их появления, путей снижения и устранения требует привлечения специализированных организаций, имеющих опыт решения таких задач и необходимую для этого аппаратуру.

Наиболее активное и результативное участие в решении проблемы достижения экологически приемлемых условий, связанных с эксплуатацией устройств визуального отображения информации, приняли шведские научные и общественные организации, так или иначе связанные с безопасностью производственных процессов и здравоохранением.

Основную роль в деле изучения, концентрации и осмысления, так называемых, излучательных и эргономических характеристик, по поручению Правительства Швеции, сыграло Национальное Управление по измерениям и тестированию - MPR (впоследствии - "Управление SWEDAC"). Начиная с 1987 года, при участии широкого круга экспертов из самых разнообразных областей науки и техники, были выполнены комплексные испытания различных вариантов устройств визуального отображения и собраны сведения о влиянии электромагнитных полей и излучений присущего дисплеям частотного диапазона на состояние и здоровье людей. Особое внимание при этом было уделено установлению предельно допустимых уровней вредных физических факторов и разработке системы добровольного тестирования устройств визуального отображения на предмет выполнения установленных норм.

В 1990 году результаты исследований с учетом накопленного опыта были оформлены Управлением SWEDAC в виде двух документов: справочника для пользователя по оценке устройств визуального отображения (MPR 1990:8) и методов проверки устройств визуального отображения (MPR 1990:10), которые получили широкую известность под названием "Шведские стандарты".

Эти стандарты легли в основу созданных во многих странах национальных систем тестирования и сертификации - как различных дисплеев, так и ПЭВМ в целом. Ценность этих документов - в комплексности решения проблемы. В этих документах не только установлены предельно допустимые уровни полей и даны методы и средства контроля, но и приведены подробные технические требования к техническим средствам для калибровки и поверки средств измерений и контро-

ля.

Представляя нормативы излучательных характеристик дисплеев, авторы указанных выше шведских нормативных документов оговариваются, что эти нормы "не являются предельными значениями с точки зрения санитарии", а имеют своим назначением оказание помощи пользователям в выборе подходящих для них технических средств.

В то же время необходимо знать, что исходной предпосылкой при создании норм на излучательные характеристики было то, что "устройство визуального отображения не должно увеличивать уровни излучений, имеющихся в нормальном офисе". То есть, здесь видится третий вариант подхода к нормированию, как наиболее гуманный. Более того, при обосновании выбранных норм, авторы главное внимание уделяют физическим факторам, оказывающим влияние на здоровье пользователей.

Вероятно, эти обстоятельства привели к тому, что в большинстве стран "Шведский стандарт" был воспринят как санитарно-гигиенический, и на его основе создавались национальные нормативные акты. Директивой Совета ЕЭС от 29 мая 1990 г. № 90/270/EEC данный документ введен с июня 1992 года в качестве общеевропейского стандарта.

В России два основополагающих стандарта (гармонизированные с MPR 1990:8 и MPR 1990:10) введены в действие в 1997 году. Это ГОСТ Р 50948-96. "Средства отображения информации индивидуального пользования. Общие эргономические требования и требования безопасности" и ГОСТ Р 50949-96 "Средства отображения информации индивидуального пользования. Методы измерений и оценки эргономических параметров и параметров безопасности".

С учетом данных стандартов Госсанэпиднадзор России разработал и с первого января 1997 года ввел в действие обязательные санитарные правила и нормы - СанПиН 2.2.2.542-96 "Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работы".

В последнем документе электромагнитные поля ВДТ представлены как "неионизирующие излучения". Рентгеновское же излучение, принципиальное присутствие которого возможно ввиду наличия высокого (более 22 кВ) напряжения на электронно-лучевой трубке дисплея, законно представлено как "ионизирующее".

Кроме характеристик, присущих только дисплеям, СанПиН содержат санитарно-гигиенические требования к ПК вообще, требования к помещениям, где эксплуатируются ПК, к микроклимату, акустическим шумам и вибрациям, освещению, организации и оборудованию рабочих мест с ВДТ и ПК, как для взрослых пользователей, так и учащихся и детей дошкольного возраста.

Введение допустимых значений параметров неионизирующих излучений построено в СанПиН 2.2.2.542-96 так, как будто произошла с 1 января 1997 года замена устаревших ПДУ на новые, более жесткие. При этом, естественно, возникает желание сравнить " старые" и "новые" нормы. Но дело в том, что, как уже отмечалось выше, "старые" нормы разрабатывались совсем для других случаев, в то время, когда проблемы с "излучениями" дисплеев не существовало.

Новое же нормирование исходит из возможности одновременного воздействия на пользователя дисплея всех рассматриваемых физических факторов. Это означает, что гигиеническое сравнение "старых" и "новых" ПДУ представляется некорректным, особенно если учесть, что работа с ПК является одним из

наиболее сложных видов интеллектуальной деятельности человека. Если же подойти чисто формально, то "новые" ПДУ примерно в 20 раз жестче "старых".

Нормы по электрическим и магнитным полям, действующим с 1-го января 1997 г. приведены в табл. 10.

Таблица 10.

1	Напряженность переменного электрического поля на расстоянии 50 см вокруг дисплея:	
	В диапазоне частот 5 Гц - 2 кГц	Не более 25 В/м
	В диапазоне частот 2 кГц - 400 кГц	Не более 2,5 В/м
2	Плотность магнитного потока (магнитная индукция):	
	В диапазоне частот 5 Гц - 2 кГц	Не более 250 нТл
	В диапазоне частот 2 кГц - 400 кГц	Не более 25 нТл
3	Поверхностный электростатический потенциал экрана дисплея	Не более 500 В

При установлении электромагнитной безопасности рабочего места с компьютерной техникой должно быть подтверждено его соответствие трем нормативным документам:

- СанПиН 2.2.2.542-96 по требованиям к электрическим и магнитным полям дисплеев и ПК;
- СанПиН 5802-91 по требованиям к электрическим полям промчастоты 50 Гц;
- СанПиН 2.2.4.723-98 по требованиям к магнитным полям промчастоты 50 Гц.

4.3. Требования к помещениям для размещения компьютерной техники

Общие гигиенические требования к помещениям для эксплуатации ВДТ и ПК изложены в разделе 4 СанПиН 2.2.2.452-96. Из них основными, с точки зрения обеспечения электромагнитной безопасности, являются следующие требования:

- Площадь, приходящаяся на одно рабочее место с ПК, должна составлять не менее 6 кв.м., что позволяет расположить технические средства на безопасном для пользователя расстоянии при любых известных в настоящее время излучательных характеристиках данных технических средств.
- Рекомендуемый объем, приходящийся на одно рабочее место с ПК, должен составлять не менее 20 куб.м. (24 куб.м. - во всех учебных и дошкольных учреждениях), что позволяет кроме обеспечения общей гигиены, снижать концентрацию пылевидных частиц и аэроионов.
- С целью предотвращения накопления статических зарядов рекомендуется увлажнять воздух в помещениях с ВДТ и ПК, например, с помощью увлажнителей, заправляемых дистиллированной или прокипяченной водой.
- Для снижения восприимчивости пользователей к воздействию вредных факторов, помещения с ВДТ и ПК должны быть расположены и оборудованы так, чтобы можно было обеспечить там температуру, относительную влажность и скорость движения воздуха, соответствующие действующим санитарным нормам микроклимата производственных помещений. При этом в помещениях, где работа с ВДТ и ПК является основной (диспетчерские, операторские, расчетные, кабины и посты управления, залы вычислительной техники и др.), должны обеспечиваться оптимальные параметры микроклимата. Требования к микроклимату в дошкольных, средних специальных и высших учебных заведениях оговорены особо в

СанПиН 2.2.2452-96.

Однако как показывает опыт, на практике данных требований оказывается недостаточно для обеспечения как нормальной электромагнитной обстановки в помещении, так и условий для нормального функционирования ПК. При неверной общей планировке помещения, неоптимальной разводке питающей сети и неоптимальном устройстве контура заземления (хотя и удовлетворяющем всем регламентируемым требованиям электробезопасности) собственный электромагнитный фон помещения может оказаться настолько сильным, что обеспечить на рабочих местах пользователей ПК требования СанПиН по уровням электромагнитных полей не представляется возможным ни при каких ухищрениях в организации самого рабочего места и ни при каких (даже "суперсовременных" и экологически безопасных) компьютерах. Более того - сами компьютеры, будучи помещенными в сильные электромагнитные поля, становятся неустойчивыми в работе, появляется эффект дрожания изображения на экранах дисплеев, существенно ухудшающий их эргономические характеристики.

Можно сформулировать следующие дополнительные требования, которыми необходимо руководствоваться при выборе помещений для обеспечения в них нормальной электромагнитной обстановки, а также обеспечения условий устойчивой работы ПК в условиях электромагнитного фона:

1. Помещение должно быть удалено от посторонних источников электромагнитных полей, создаваемых мощными трансформаторами и электроустройствами, электрическими распределительными щитами, кабелями электропитания с мощными энергопотребителями, радиопередающими устройствами и пр. Если данная возможность в выборе помещения отсутствует, настоятельно рекомендуется предварительно (до установки компьютерной техники) провести обследование помещения по уровню низкочастотных электромагнитных полей. Затраты на последующее обеспечение устойчивой работы ПК в не оптимально выбранном по данному критерию помещении несравнимо больше, чем стоимость подобных обследований.
2. Если на окнах помещения имеются металлические решетки, то они должны быть заземлены. Как показывает опыт, несоблюдение данного правила может привести к резкому локальному повышению уровня полей в какой-либо точке (точках) помещения и сбоям в работе компьютера, случайно установленного в данной точке.
3. Групповые рабочие места (характеризующиеся значительной скученностью компьютерной и другой оргтехники) желательно размещать на нижних этажах зданий. При подобном размещении рабочих мест минимально их влияние на общую электромагнитную обстановку в здании (энергонагруженные кабели питания не идут по всему зданию). Существенно снижается также общий электромагнитный фон на рабочих местах с компьютерной техникой (вследствие минимального значения сопротивления заземления именно на нижних этажах зданий).

В самих помещениях при организации и планировке расположения рабочих мест необходимо руководствоваться следующими правилами:

- Должно быть обеспечено заземление (или трехпроводная сеть с третьим, соединенным с землей проводом), подводимое непосредственно к каждому рабочему месту. При организации заземления следует предостеречь от одной распространенной ошибки. Речь идет об использовании при разводке питания фильтров типа "Пилот" и других типов удлинителей с евророзетками, снабженными заземляющими контактами. Практика многочисленных обследований показывает, что

нередки случаи ненадежного (а иногда и полностью отсутствующего) электрического контакта между заземляющими узлами такой розетки электропитания и вилки сетевого шнура компьютера. В связи с этим организацию заземления посредством использования заземляющего контакта евророзеток можно рекомендовать только в тех случаях, когда надежность этого контакта подтверждена замерами сопротивления заземления, а стыковочный узел надежно защищен от произвольных пространственных перемещений и в процессе эксплуатации не подвергается многочисленным операциям стыковки и расстыковки.

- Крайне нежелательными является вариант одной линии питания, обходящей помещение по всему периметру, и наличие замкнутого по периметру контура заземления. При подобных схемах питания и организации контура заземления может резко возрасти магнитная составляющая поля в диапазоне частот измерения 5 Гц - 2 кГц.
- Провода питания желательно проводить в экранирующих металлических оболочках или трубах.
- Места группового подключения ПК целесообразно оборудовать экранированными щитками, обеспеченными достаточным количеством розеток и размещенными с учетом наибольшей равно удаленности их от рабочих мест пользователей ПК и других сотрудников, постоянно работающих в помещении.
- Целесообразно к каждому групповому месту подключать не более 2 - 3-х пользователей ПК.
- Желательно, чтобы установленные сетевые розетки позволяли изменять полярность включения вилки питания дисплея и системного блока ПК в сетевую розетку. В дальнейшем (при обследовании рабочего места) это позволит выбрать ту ориентацию вилки в сетевой розетке, при которой поля на рабочем месте минимальны.

Выполнение перечисленных выше требований может обеспечить снижение в десятки и сотни раз общего электромагнитного фона в помещении. Задача обеспечения нормальной электромагнитной обстановки на рабочих местах пользователей ПК при этом будет сведена к задаче правильной организации самих рабочих мест.

4.4. Размещение компьютерной техники на рабочем месте

Раздел 8 СанПиН 2.2.2.542-96, в котором основное внимание уделяется эргономическим требованиям к оборудованию рабочих мест с ПК и ВДТ, дает лишь несколько общих рекомендаций по организации рабочих мест, полезных с точки зрения электромагнитной безопасности:

- рабочее место должно быть автономным;
- экран дисплея ПК должен находиться от глаз пользователя на оптимальном расстоянии 60 - 70 см, но не ближе 50 см.;
- в помещениях с ПК и ВДТ ежедневно должна проводиться влажная уборка.

Однако, этих требований (установленных в СанПиН 2.2.2.542-96) недостаточно для обеспечения электромагнитной безопасности рабочих мест.

При неправильной организации электропитания рабочего места источниками электрических и магнитных полей могут быть не только дисплей ПК, импульсный источник питания системного блока ПК и сетевые кабели (провода) электропитания, но и периферийные устройства ПК (клавиатура, принтер, модем и т.п.).

Полную гарантию безопасности рабочего места может дать лишь его де-

тальное обследование по уровням полей и аттестация рабочего места уполномоченными на это организациями и специалистами. Вместе с тем, можно сформулировать ряд конкретных практических рекомендаций по организации рабочего места и размещению на нем компьютерной техники, выполнение которых заведомо улучшит электромагнитную обстановку и с намного большей вероятностью обеспечит аттестацию рабочего места без принятия для этого каких-либо дополнительных специализированных мер.

1. Основные источники импульсных электрических и магнитных полей, а также электростатических полей - дисплей и системный блок ПК (в том числе, совмещенный с клавиатурой в учебных ПК) - должны быть в пределах рабочего места максимально удалены от пользователя.

2. Должно быть обеспечено надежное заземление (с периодическим контролем) системного блока и источника питания ПК. Если имеется техническая возможность, целесообразно заземлить системный блок не только через заземляющий контакт трехконтактной вилки питания (естественно, при наличии соответствующей и правильно подключенной розетки), но и путем соединения отдельным проводником корпуса системного блока с контуром заземления в помещении.

3. Должно быть обеспечено наибольшее удаление пользователя от сетевых розеток и проводов электропитания. Не рекомендуется использование различных удлинителей (переносок) и сетевых фильтров, выполненных в виде переносок. Использование рекламируемых в торговле сетевых фильтров в виде переносок можно признать целесообразным только в том случае, если достоверно установлено наличие сбоев в работе ПК из-за помех из сети питания. Крайне не рекомендуется использование двухпроводных удлинителей, переносок и сетевых фильтров, а также подобных устройств с трехконтактными розетками и вилками питания, но с незадействованным на шину заземления заземляющим контактом. Использование таких устройств можно допустить только в том случае, если имеется отдельно выполненное заземление системного блока ПК.

4. Должно быть обеспечено надежное заземление (с периодическим контролем) защитного экранного фильтра дисплея ПК. Наиболее правильным способом является заземление фильтра на корпус системного блока ПК (например, под винт крепления источника питания). Не рекомендуется заземление защитного экранного фильтра в другие точки схемы электропитания (на "нулевой" провод в розетке питания, заземляющую шину в помещении и т.п.). Хотя эти точки и связанны гальванически между собой и с корпусом системного блока, но, как показывает практический опыт, реальные защитные свойства установленного на экран дисплея фильтра при этом снижаются.

5. При организации электропитания рабочего места целесообразно предусмотреть возможность изменения полярности включения в розетку сетевой вилки питания системного блока и дисплея ПК и предусмотреть при этом маркировку фазного и нулевого проводов. Это позволит при обследовании рабочего места специальной аппаратурой для контроля электромагнитных полей, оперативно выбрать и зафиксировать ту ориентацию подключения вилки питания, при которой поля на рабочем месте минимальны.

Особо необходимо остановиться на организации рабочего места с большим количеством периферийных устройств - когда пользователь в силу обстоятельств окружен различной оргтехникой. С высокой степенью достоверности можно сказать - при надежном заземлении каждого из периферийных устройств,

при исправности заземляющей шины информационных цепей, связывающих периферийные устройства, последние не вносят существенного вклада в общий уровень электромагнитных полей. Основное внимание при этом необходимо уделить максимальному удалению от пользователя дисплея и системного блока.

На рис. 43 - 45 показаны различные варианты компоновки рабочего места (рекомендуемые и не рекомендуемые с точки зрения электромагнитной безопасности).

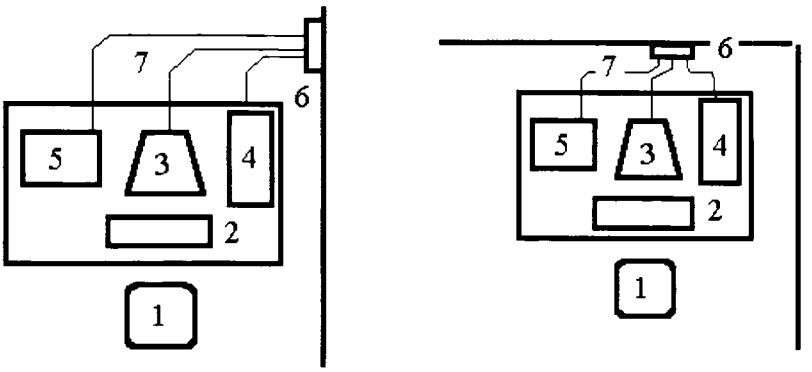


Рис. 43. Рекомендуемые компоновки рабочего места оператора ПК

Наиболее оптимальной следует признать планировку, когда полностью разделены зона местонахождения пользователя ПК и зона, где расположены кабели электропитания технических средств рабочего места, включая розетки сетевого электропитания.

Цифрами на рис. 43 - 45 помечены:

1. Рабочее место оператора;
2. Клавиатура ПК;
3. Дисплей ПК;
4. Системный блок ПК;
5. Принтер;
6. Розетки питания;
7. Сетевые кабели питания блоков ПК.

Менее оптимальной является планировка, представленная на рис. 44, когда рядом с пользователем расположены сетевые кабели электропитания рабочего места. Данную планировку нежелательно использовать, если на рабочем месте установлено большое количество технических средств со значительным энергопотреблением.

В этом случае по сетевым кабелям электропитания текут значительные токи, и пользователь ПК находится в зоне воздействия магнитных полей промчастоты 50 Гц.

Недопустимой является планировка рис. 45. Из-за реально ненулевого значения сопротивления цепи заземления пользователь ПК может находиться не только в зоне воздействия магнитных, но и электрических полей промчастоты 50 Гц. При отсутствии возможности иной организации рабочего места можно рекомендовать способ снижения уровня полей за счет расположения кабелей электропитания в металлической (стальной) заземленной трубе. Однако следует особо подчеркнуть, что данную планировку рабочего места можно использовать только при наличии документального подтверждения соответствия уровней полей требо-

ваниям действующих СанПиН при контроле специальной аппаратурой. В случае отсутствия объективных замеров уровней полей на рабочем месте подобную планировку использовать нельзя.

В любом случае рекомендуется модернизировать рабочее место на рис. 45, так как изображено на рис. 44.

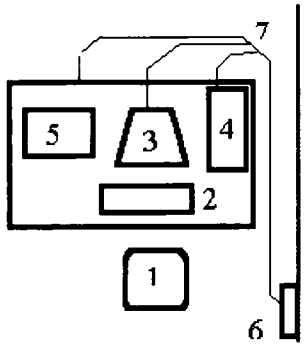


Рис. 44. Нежелательная компоновка рабочего места

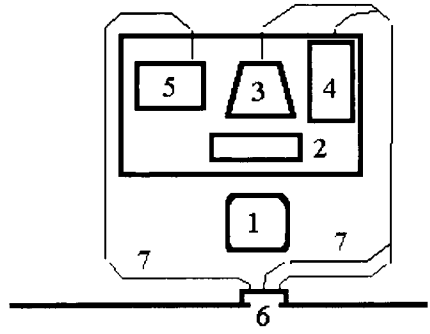


Рис. 45. Недопустимая компоновка рабочего места

В приведенные обобщенные схемы могут вноситься уточнения и изменения, обусловленные специфическими особенностями конструкции ПК и дисплея ПК, особенностями пространственных диаграмм электрических и магнитных полей и особенностями пространственной конфигурации помещения. В заключение следует отметить еще два немаловажных обстоятельства.

Первое. При реализации какого-либо варианта в организации рабочего места необходимо учитывать возможное влияние его электромагнитных полей на постоянно работающих рядом людей и осуществлять корректировку их расположения относительно рабочего места с ПК.

Второе. Реализуемые варианты компоновки рабочего места (безопасные по электромагнитным полям) ни в коей мере не должны противоречить установленным в санитарных правилах и нормах требованиям по эргономическим характеристикам. Должны выполняться требования по освещенности экрана дисплея и освещенности документа, требования по расположению рабочего места с учетом направления освещения для исключения бликов на экране дисплея и т.п. Последствия для здоровья при пренебрежении эргономическими требованиями могут быть не меньшими, чем от повышенного уровня электромагнитных полей.

4.5. Размещение и организация в помещении нескольких рабочих мест с ПК

При определении допустимого количества рабочих мест в помещении следует учитывать требование СанПиН о необходимости обеспечения не менее 6-ти кв. м на одно рабочее место с ВДТ или ПК. Схемы размещения РМ должны учитывать допустимые расстояния между рабочими столами с ВДТ (которые должны быть не менее 2 м по фронту и обеспечивать расстояние между боковыми поверхностями мониторов не менее 1,2 м). Выполнение указанных требований с учетом требований к помещениям и рекомендаций по компоновке каждого из рабочих мест, дает весьма высокую гарантию обеспечения нормальной электромагнитной обстановки на рабочих местах.

Основные принципы размещения в помещении значительного количества

рабочих мест с ПК должны быть следующие:

1. Автономное размещение отдельных рабочих мест, их автономное электропитание.
2. Выбор наиболее безопасных схем размещения рабочих мест, обеспечивающих:
   2.1. максимально возможную удаленность от каждого пользователя сетевых элементов и аппаратуры соседних рабочих мест;
   2.2. расположение аппаратуры и пользователя с учетом размещения их на соседних рабочих местах;
   2.3. учет диаграмм излучений от аппаратуры соседних рабочих мест и принятие мер к уменьшению их воздействия соответствующим размещением, ориентацией или экранировкой.
3. Периодический контроль электромагнитной обстановки.

Автономное размещение отдельных рабочих мест и их автономное электропитание резко повышает мобильность рабочего места при выполнении применительно к нему конкретных мероприятий по снижению уровней электромагнитных полей в зоне расположения пользователя. Это позволяет оперативно отключать не эксплуатируемые в конкретный момент рабочие места с целью уменьшения общего электромагнитного фона в помещении, позволяет оптимально выполнить схему электропитания. Последнее важно не только для снижения уровней напряженности полей промчастоты на рабочем месте оператора, но и для снижения напряженности магнитной составляющей промчастоты в зоне расположения монитора ПК. Повышенный уровень магнитного поля промчастоты оказывает негативное влияние на качество изображения в связи с появлением эффектов его дрожания и мерцания.

Выбор наиболее безопасных схем размещения рабочих мест зависит, во многом, от конкретных характеристик помещения. Это связано с тем обстоятельством, что при обеспечении электромагнитной безопасности в конкретном производственном или ином помещении необходимо учитывать необходимость обеспечения одновременно и других требований по охране труда - требований электробезопасности, пожарной безопасности, требований по освещенности на рабочем месте, микроклимату и т.п.

Ниже приводятся наиболее оптимальные варианты по размещению компьютерной техники (которые можно реализовать на практике, исходя из конкретных условий) с оценкой их достоинств и недостатков, а также варианты, не рекомендуемые для размещения (однако, зачастую встречающиеся на практике).

Для примера рассматриваются помещения наиболее часто встречающейся прямоугольной геометрии.

В варианте размещения, показанном на рис. 46, рабочие места расположены друг за другом в несколько рядов.

Для упрощения, на схеме указаны лишь мониторы и клавиатура на столах и места расположения работающих. Жирными линиями обозначена подводка электропитания от рабочих мест к розеткам в стенах помещения.

Достоинство такого расположения - возможность визуального общения руководителя с подопечными и боковое освещение экранов дисплеев естественным светом (что рекомендовано в СанПиН 2.2.2.542-96).

Существенный недостаток - опасность облучения большинства пользователей с тыльной стороны дисплея, установленного на соседнем рабочем месте, а также близость к пользователю элементов сетевого питания соседнего рабочего

места. Данную планировку можно рекомендовать только в том случае, если в помещении имеется возможность разнесения рабочих столов на требуемое санитарными правилами и нормами расстояние между ними в два и более метра или при использовании компьютеров с крайне низким уровнем собственных полей. Сетевые провода электропитания каждого рабочего места должны быть максимально локализованы с тыльной его стороны, а пользователь соседнего рабочего места должен находиться в максимально возможном удалении (в пределах существующего пространства каждого рабочего места) от данных проводов. При невыполнении указанных выше условий, планировка рабочих мест, приведенная на рис. 46, является недопустимой.

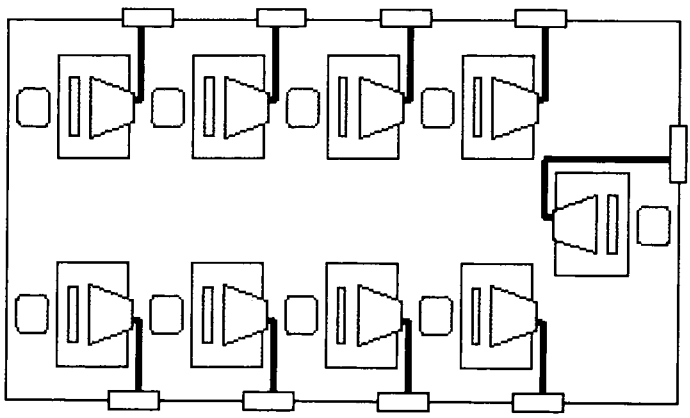


Рис. 46. Наименее приемлемый (с точки зрения обеспечения электромагнитной безопасности) вариант взаимного расположения рабочих мест

Улучшить электромагнитную обстановку на указанных рабочих местах без существенной переделки системы подводки электропитания (к сожалению, именно фактор возможных вариантов реализации электропитания является зачастую решающим при выборе той или иной планировки в условиях реальных помещений) можно при встречно-противоположном и многорядном расположении рабочих мест (см. рис. 47).

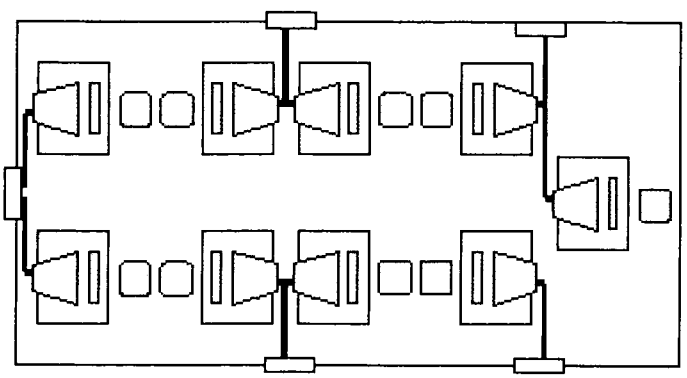


Рис. 47. Рекомендуемая перепланировка рабочих мест, изображенных на рис. 46

Недостаток такой планировки - частичная потеря зрительного контакта руководителя с подопечными. На рис. 48 показана планировка, с которой пришлось столкнуться при обследовании компьютерных классов. Можно понять преподавателей, которые стремились с помощью такой планировки не допустить недозволенных контактов во время занятий между учениками соседних рабочих мест. Однако худшего расположения (по электромагнитной безопасности) придумать трудно. При использовании старых типов дисплеев, зачастую характеризующихся гипертрофированными магнитными полями с максимальной интенсивностью в боковых направлениях, такое расположение рабочих мест опаснее, чем последовательное их расположение друг за другом, кА показано на рис. 46.

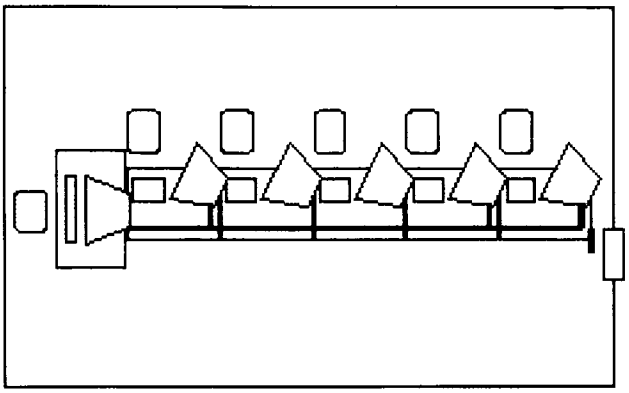


Рис. 48. Наиболее опасное расположение мониторов ПК

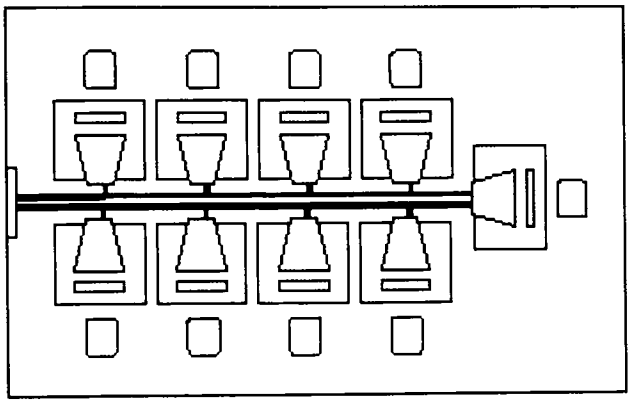


Рис. 49. Рекомендуемая перепланировка рабочих мест, изображенных на рис.48

При расположении рабочих мест по схеме, показанной на рис. 49, удается удалить пользователей на безопасное расстояние от источников полей (как своих, так и соседних рабочих мест), правда, при не всегда высокой эффективности использования площади помещения.

Особенность данного расположения в том, что питание ПК осуществляет-

ся общей сетевой линией, проходящей между двумя рядами столов. По ней протекает увеличенный ток, и при низком уровне полей ПК создается опасность повышенного уровня полей промчастоты. Это требует применения дополнительных средств обеспечения электромагнитной безопасности (например, экранировки проводов), что повышает затраты на оборудование помещения.

Другой недостаток такого расположения - возможная засветка экранов ВДТ одного из рядов рабочих мест естественным светом, падающим из светопроемов.

При малом боковом расстоянии между рабочими местами при планировке, изображенной на рис. 48, возможно увеличение уровня полей на каждом рабочем месте из-за влияния мониторов соседних рабочих мест. В таком случае можно рекомендовать некоторую модификацию размещения рабочих мест, когда электромагнитные поля в точке местоположения пользователей будут существенно снижены за счет взаимной экранировки полей аппаратуры каждого рабочего места (рис. 50).

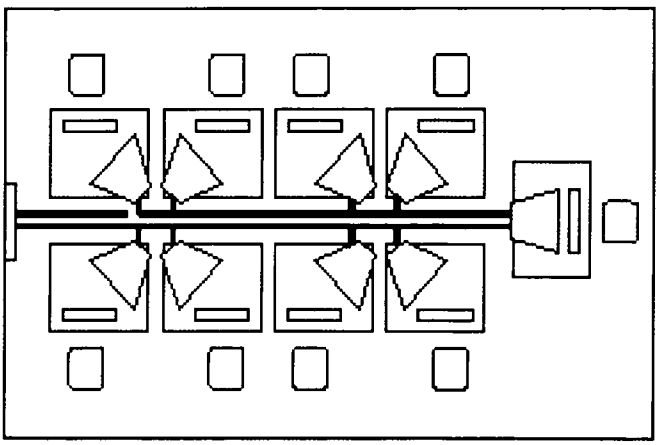


Рис. 50. Расположение мониторов на рабочих местах
с взаимной экранировкой их полей

В основе планировки, показанной на рис. 51, лежит расположение рабочих мест вдоль боковых и торцевых стен помещения с ориентацией тыльной стороны каждого рабочего места к стене.

При таком расположении удается:

- разделить сетевое питание вычислительной техники в помещении на ряд параллельных линий, уменьшив этим токи в отдельных силовых линиях и, соответственно, создаваемые ими поля;
- кардинально решить вопросы максимальной отдаленности пользователей от сетевых элементов;
- добиться размещения в помещении заметно большего количества рабочих мест без ухудшения электромагнитной обстановки и обеспечения требований санитарных норм по допустимому минимальному расстоянию между рабочими местами (так как регламентируемые санитарными нормами боковые расстояния между столами меньше, чем регламентируемые фронтальные).

По этим причинам схема на рис. 51 может быть предпочтительнее предыдущих при прочих равных условиях и обстоятельствах.

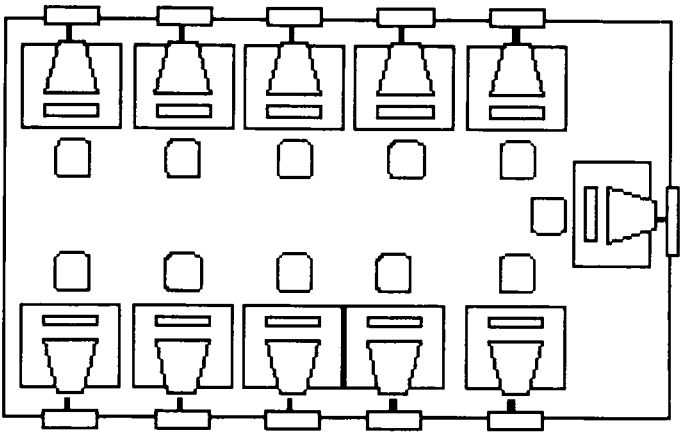


Рис. 51. Наиболее оптимальная планировка большого количества рабочих мест

Следует особо остановиться на еще одном, часто практикуемом варианте компоновки рабочих мест. Речь идет об их секционном расположении - в отдельных (находящихся рядом друг с другом) кабинах, разделенных перегородками. Подобная компоновка практикуется в компьютерных классах школ, на рабочих местах операционистов в банках и в кассовых залах.

Здесь особенно важно правильно подойти к компоновке каждого из рабочих мест, так как перегородки между кабинами (в особенности, если они выполнены из обычных пластиковых материалов или сухого дерева) практически не влияют на электромагнитную обстановку.

На рис. 52. показано не рекомендуемое в подобных случаях (но часто используемое) расположение рабочих мест.

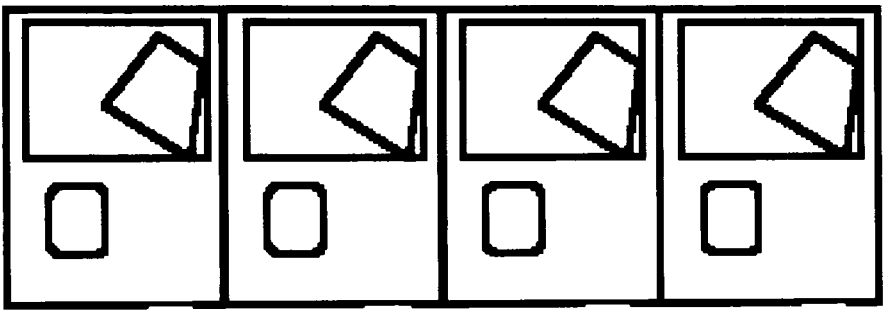


Рис. 52. Не рекомендуемая планировка рабочих мест при их секционном расположении

Здесь каждый из пользователей находится в зоне электромагнитных полей мониторов и электропроводки соседнего рабочего места.

Планировка рабочих мест, представленная на рис. 53, лишена отмеченных выше недостатков. Здесь каждый из пользователей ПК расположен в максимально возможном удалении от зон максимума электромагнитных полей. При прочих равных условиях именно подобную планировку необходимо рекомендовать при секционном расположении рабочих мест.

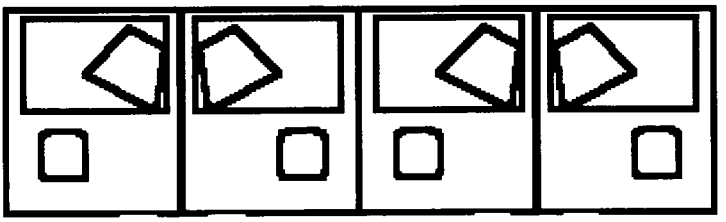


Рис. 53. Рекомендуемая планировка рабочих мест при их секционном расположении

При планировке рабочих мест с компьютерной техникой нельзя забывать о других, постоянно находящихся в помещении сотрудниках. Их рабочие места не должны располагаться в зонах концентрации электромагнитных полей, в особенности - с тыльной и боковых сторон мониторов ПК.

Нужно также иметь в виду, что зачастую стены между помещениями (в особенности, если они выполнены из дерева или пластиковых материалов) не являются сколь либо значительным препятствием для низкочастотных электромагнитных полей. Об этом нужно помнить при расположении рабочих мест с ПК в помещении вдоль стен смежных с ним помещений.

4.6. Выбор компьютерной техники и средств защиты

В последние годы появились портативные ПК, в которых системный блок, дисплей и клавиатура размещены в одном корпусе, так называемые "Note Book", работающие от автономного источника (аккумулятора) или от электрической сети через преобразователь. По своим функциональным возможностям они сравнимы с обычными ПК и отражают последние достижения в развитии полупроводниковой электроники. Дисплеи таких ПК представляют собой матрицу (обычно жидкокристаллических элементов) черно-белого или цветного изображения.

Главной частью ПК является системный блок, определяющий технические параметры компьютера и его возможности в целом. Практически всегда в системный блок входят устройства ввода-вывода информации в виде накопителей на жестком и гибком магнитных дисках, а в последнее время - устройства считывания записи с компакт-дисков.

В настоящее время отечественные компьютеры (производимые в России) представляют собой продукт сборочного производства из заграничных элементов (причем, не всегда лучшего качества).

Сказанное относится к системным блокам ПК. Периферийные устройства не собираются, а покупаются целиком, в основном в странах юго-восточной Азии: - Тайвань, Китай, Малайзия, Южная Корея. Отечественные производители компьютеров предлагают продукцию, которая по своим техническим возможностям сравнима с продукцией заграничных фирм, но несколько уступает им по надежности. Несмотря на широкий рынок предложений со стороны иностранных компьютерных фирм, они находят в стране сбыт из-за своей дешевизны.

Следует отметить, что если еще совсем недавно персональными электронно-вычислительными машинами были оснащены только предприятия и научные учреждения (где они были прерогативой инженеров и научных работников), то сейчас ПК стали предметом массового пользования. Это во многом объяснимо структурными изменениями в обществе, его открытостью по отношению к внешнему миру, возможностью подсоединяться к мировым компьютерным сис-

темам. В учебных заведениях уже не редкость компьютерные классы, где учащиеся изучают основы программирования и приобретают навыки работы на самых современных ПК.

Внедрение компьютеров во все сферы деятельности человека объяснимо несомненной их полезностью и разносторонними возможностями. Однако при определенных обстоятельствах они могут доставлять пользователям и вред. Бывает, что пользователи жалуются на утомление глаз, нервные расстройства, иногда и более серьезные болезни. Особенную тревогу для врачей представляют дети, так как они не умеют рационально планировать свое рабочее время.

Для любого пользователя ПК очень большое значение имеет дисплей, при помощи которого он ведет "разговор" с компьютером, к экрану дисплея приковано все его внимание; и независимо от характера труда, именно дисплей, практически всегда определяет собой рабочее место.

Сразу же можно дать одну общую рекомендацию - если предполагается проводить значительное время за компьютером и работать при этом не только с текстовыми материалами - нельзя не экономить на дисплее. Необходимо приобрести дисплей, который не только удовлетворяет всем установленным в России требованиям гигиенической сертификации, но и устраивает пользователя по чисто субъективным ощущениям.

Основной элементом современного дисплея - электронно-лучевая трубка, на экране которой электронный луч формирует изображение. К электродам трубки подводится высокое напряжение (десятки киловольт), а в катушках отклоняющей системы протекает импульсный ток. Это является причиной появления в пространстве перед дисплеем электростатического, а вокруг дисплея - электромагнитного поля, спектральные составляющие которого сосредоточены в диапазоне частот от 5 Гц до 400 кГц.

Дисплеи первых ПЭВМ 70-х годов создавали на расстоянии 50 см от своего экрана значительные поля - напряженностью более 100 В/м на частотах 5 Гц - 400 кГц, и электростатический потенциал на расстоянии 10 см от экрана более 10 кВ. По требованию врачей и сориентированной ими общественности началась разработка нормативных документов, которые определили границу безопасных норм и контролируемый частотный диапазон излучений. За рубежом имеется директива Европейского экологического сообщества N90/270/EEC, которая гласит: "Оператор работающий с дисплеем, должен быть информирован о мерах безопасности и сохранении здоровья, а также о мерах, предпринимаемых с целью уменьшения или устранения любого риска".

Впервые наиболее жесткие нормы были созданы в Швеции - стандарты MPR I, MPR II, TCO 95. Нормативы Швеции включены в официальные документы Европейского экономического сообщества и являются базовыми для создания единого стандарта стран ЕЭС. На требования значений TCO 91 и TCO 95 ориентированы разрабатываемые стандарты США. Предлагаемые в них нормы отражают современное понятие степени биологической безопасности с одной стороны и технические возможности электронной промышленности с другой стороны.

В России разработаны аналогичные государственные стандарты: - ГОСТ Р 50948-96, ГОСТ Р 50949-96, по которым в испытательных лабораториях проводят сертификационные испытания дисплеев. Также введены в действие санитарные правила и нормы на рабочие места с использованием дисплеев (СанПиН 2.2.2.542-96).

Материалы сертификационных испытаний показывают, что для совре-

менных моделей дисплеев контролируемые параметры, в основном, соответствуют норме. В настоящее время не вызывают претензий дисплеи ведущих фирм Sony, Samsung, изготовленные в Японии или Ю. Корее. Встречаются дисплеи производства Тайвань, не обеспечивающие требований по уровню напряженности электрического поля, но в большинстве случаев эти превышения незначительны. Однако последнее относится лишь к дисплеям выпуска позднее 1995-96 годов.

В целом по результатам сертификационных испытаний можно сделать несколько выводов:

- В характеристиках компьютерной техники, продаваемой на рынках России, в последние годы произошло качественное изменение по уровням электромагнитных полей. Серьезная конкуренция в данной области техники, принятие в России санитарных норм, гармонизированных с международными нормами, привели к повышению качества дисплеев ПК (являющихся одним из основных источников электромагнитных полей на рабочих местах) по параметрам электромагнитной безопасности. Но многие дисплеи все еще не соответствуют требованиям СанПиН, особенно дисплеи выпуска до 1995 года, не взирая на обозначение на них "Low Radiation".

- Производимые различными фирмами дисплеи характеризуются весьма разнообразными пространственными диаграммами низкочастотных электромагнитных излучений. Соответственно, при планировке групповых рабочих мест в обязательном порядке необходимо проведение детальных обследований с использованием специализированной аппаратуры для установления наиболее оптимальных вариантов компоновки каждого из рабочих мест с учетом используемых в нем типов технических средств.

- В последние годы резко повысилось качество дисплеев по уровню электростатического потенциала экрана. Данный параметр улучшен фирмами- изготовителями дисплеев в сотни раз. Вместе с тем, исследования показали, что во многих типах дисплеев используется, так называемый, компенсационный метод снижения электростатического потенциала монитора, когда схема компенсации начинает эффективно работать только через некоторое время после включения дисплея, а до этого периода на экране дисплея присутствует значительный электростатический потенциал. Следовательно, для подобных типов дисплеев требуется применение защитных экранных фильтров. Этот факт необходимо учитывать при комплектовании рабочих мест средствами защиты (исходя из конкретных типов используемой техники).

- В последнее время некоторые фирмы (в том числе и отечественные) начали комплектовать производимые ими ПК специальными экранированными сетевыми шнурами. Применение таких сетевых шнуров (вместо обычных трехпроводных) в десятки раз снижает на рабочем месте уровень переменного электрического поля в диапазоне частот 5 Гц - 2 кГц. При покупке дисплея или ПК необходимо приобрести их в комплектации именно с таким сетевым кабелем питания. Стоимость приобретенного оборудования возрастет незначительно, а положительный эффект для здоровья пользователя будет существенным.

- Не подтвердилось широко бытующее и, к сожалению, ошибочное мнение о полной безопасности портативных персональных компьютеров типа "Note Book". Такие персональные компьютеры совершенно безопасны по уровню электростатического потенциала (по причине отсутствия в них высоких напряжений постоянного тока). Однако, из-за наличия в своем составе высокочастотных пре-

образователей источников питания некоторые типы данных персональных компьютеров имеют значительное превышение норм по уровням электромагнитных полей с четко выраженной диаграммой направленности (как в горизонтальной, так и в вертикальной плоскости).

Особо следует отметить тот факт, что требования международного стандарта MPR II не распространяются на персональные компьютеры типа Note Book. Следовательно, соответствие данного типа ПК международно признанным нормам по уровням электромагнитных излучений находится полностью на совести производителя. Но справедливости ради, нужно сказать, что фирмы-производители принимают здесь конкретные меры. Так, например, в некоторых типах обследованных Note Book не было обнаружено сколь - либо значительного превышения уровня полей в направлении пользователя.

К сожалению, мы не считаем возможным назвать здесь эти фирмы, так как у других типов компьютеров Note Book тех же фирм ситуация была совершенно противоположной. Обобщающие выводы здесь делать рано. Чтобы обезопасить себя, при покупке компьютеров типа Note Book советуем требовать в обязательном порядке у продавца сертификат соответствия продаваемой техники нормам безопасности ГОСТ Р 50948-96. Дело в том, что введенный в действие на территории России с 1-го июля 1997 года стандарт безопасности для компьютерной техники распространяется также (в отличие от стандарта MPR II) и на портативные персональные компьютеры типа Note Book.

Наблюдения показывают, что для индивидуальных пользователей часто решающим значением при выборе является цена. На производстве или учреждении ситуация такова, что покупают аппаратуру одни люди, а работают с ней другие. Часто оператору приходится работать не с тем компьютером, который ему нравится, а с тем, который предложат. В любом случае, при выборе компьютера следует ориентироваться не только техническими характеристиками, ценой, моделью, но и посмотреть, в какой стране он изготовлен. Можно попросить продавца показать сертификат соответствия, но практика показывает, что сертификатами снабжена продукция практических всех торговых фирм, и в наше время он не всегда отражает реальность. Поэтому предпочтительно ориентироваться на фирмы-изготовители, продукция которых отличается стабильным качеством.

Труднее обстоит дело при ориентации своего выбора на отечественную продукцию. В России прекращены разработки собственной компьютерной техники для широкого применения, а собираемая из зарубежных элементов (блоков) продукция часто не отличается стабильностью качества и привлекает покупателя своей дешевизной. Очень часто, если спросить у производителя или продавца отечественных ПК о соответствии их (этих ПК) действующим в настоящее время в России нормам электромагнитной безопасности, он представит необходимые документы только на дисплей (который является импортным) и попытается убедить, что этого достаточно. Продавец или производитель в этом случае не правы. ПК отечественного производства или их системные блоки (если дисплеи ПК импортные) должны иметь свой гигиенический сертификат. В соответствии с требованиями СанПиН 2.2.2.542-96 *запрещены постановка на производство и продажа отечественных ПК без гигиенической оценки их безопасности для здоровья человека.*

Как уже отмечалось, при выборе компьютерной техники необходимо придавать большее значение выбору дисплея, особенно цветного изображения. Если на компьютере выполняется простая и монотонная работа, то лучше остано-

вить свой выбор на дисплее с черно-белым изображением. Такие дисплеи меньше утомляют зрение. Хорошим качеством изображения и параметрами безопасности обладают модели мониторов фирм Sony, Samsung производства Японии или Ю. Кореи. При испытаниях они показывают результаты, превышающие в лучшую сторону требования нормативных документов. Низкочастотные электромагнитные поля в них сведены до минимума, почти отсутствуют электростатический потенциал экрана, а в последних моделях на экранах имеется антибликовое покрытие, что дает возможность не применять защитные экранные фильтры.

Компьютер (точнее, его дисплей) чувствителен к мощным магнитным полям, которые могут создаваться сильноточными линиями (кабелями) промышленной частоты или мощными электрическими установками. Внешние магнитные поля с величиной индукции более 1000 нТл вызывают дрожание изображения на экране дисплея, что крайне вредно отражается на психике пользователя ПК. Требования по устойчивости дисплеев к данным внешним воздействиям в настоящее время не регламентированы. Поэтому, на экране приобретенного компьютера с самым современным дисплеем может обнаружится рябь изображения, его подергивание или медленное покачивание. Причина тому - повышенные переменные магнитные поля промчастоты на рабочем месте. Какие-либо защитные экраны в таких случаях малоэффективны вследствие низкой частоты внешнего воздействующего поля. Требуется перепланировка рабочих мест или перемонтаж разводки электропитания - вплоть до перемонтажа разводки электропитания всего здания. Если по каким-либо причинам предлагаемое выше неосуществимо, необходимо приобрести для рабочих мест с повышенным уровнем магнитных полей компьютеры (дисплеи) с жидкокристаллическими экранами, которые во много раз более устойчивы к внешним магнитным полям, либо дисплей на ЭЛТ с возможностями подстройки в широких пределах частоты кадровой развертки. Регулируя частоту кадровой развертки такого дисплея, можно опытным путем существенно уменьшить эффект зрительного восприятия дрожания изображения на его экране.

4.7. Рекомендации по обеспечению электромагнитной безопасности от средств вычислительной техники при проектировании, строительстве и реконструкции производственных объектов

Уменьшение напряженности низкочастотного электромагнитного поля до безопасного уровня в новых производственных объектах может быть обеспечено следующими мерами:

1. Целесообразно построенной и технически правильно реализованной системой электропитания.

2. Эффективной системой заземления.

3. Правильным проектированием и выбором помещений, предназначенных для работы с компьютерной техникой.

Ниже приводится более детальная расшифровка каждого из указанных направлений.

1. Система электропитания средств вычислительной техники (СВТ) должна быть спроектирована и построена с соблюдением следующих требований:

1.1. Электрическая и пространственная независимость от осветительной сети.

1.2. Отсутствие кондуктивной и индуктивной связи с информационными линиями.

1.3. Прокладка силовых трехфазных транзитных и магистральных линий внутри

здания 5-ти проводным кабелем (или 4-х проводным с металлическим экраном) с раздельными нулевым и земляным проводниками, соединяемыми только на входном центральном щите.

1.4. Трехпроводность сетевых линий (наличие в линиях третьего, соединенного с землей провода), или, в крайнем случае, подвод "земли" в виде отдельных клемм ко всем щиткам или местам группового размещения розеток.

1.5. По возможности равномерное распределение нагрузки на сетевые отводы от магистрали. Предпочтительно использовать два или больше сетевых вводов в помещение со многими потребителями, нежели одну линию, обходящую помещение по всему периметру.

1.6. Прокладка линий сети в заземленных жестком (для магистрали и внутрикомнатных линий) и гибком (отводы к розеткам потребителей) экранах. Коэффициент заполнения сечения жесткого экрана не должен превышать 0,65.

1.7. Внутрикомнатные линии целесообразно прокладывать вблизи пола.

1.8. Розетки потребителей должны располагаться, по возможности, ниже.

1.9. Необходимое число розеток целесообразно объединять в группы и оборудовать экранированными щитками. Щитки желательно располагать на возможно большем расстоянии друг от друга из расчета экологически безопасного размещения одного-двух пользователей. Щитки должны быть снабжены выключателем подачи напряжения на их розетки и световыми сигнализаторами включения питания.

2. Система заземления должна отвечать следующим условиям:

2.1. Объект должен иметь контур заземления, оборудованный в соответствии с требованиями техники электробезопасности.

2.2. Сопротивление заземления в любой точке системы не должно превышать 4 Ом. Для вычислительных центров специально рекомендуется сопротивление заземления не выше 2 Ом.

2.3. На контур заземления посажены третий провод трехпроводной линии электропитания и экранирующие оболочки сетевых линий.

2.4. На щитках питания необходимо выводить "землю" отдельной клеммой.

3. При проектировании или выборе рабочих помещений для пользователей ПК необходимо учитывать следующие требования и соображения:

3.1. Согласно требованиям СанПиН 2.2.2.542-96, раздел 4, помещения для эксплуатации дисплеев и ПК должны иметь оконные проемы, ориентированные, преимущественно, на север и северо-восток, обеспечивающие коэффициент естественной освещенности в 3-м световом климатическом поясе не ниже 1,2% (в зонах с устойчивым снежным покровом) или 1,5% - на остальной территории. Не допускается расположение рабочих мест с дисплеями и ПК в подвальных помещениях, а также в цокольных помещениях (для учебных заведений и дошкольных учреждений).

Помещения должны оборудоваться системами отопления, кондиционирования воздуха или рассчитанной эффективной приточно-вытяжной вентиляцией. Они не должны граничить с помещениями, в которых уровни шума и вибраций превышают нормируемые значения (СанПиН, приложения 7, 8, 9).

3.2. Необходимая площадь помещения оценивается исходя из требования выделения не менее 6 кв. м. на каждое рабочее место (не считая размеров проходов между соседними рабочими местами согласно требованиям противопожарной безопасности) при объеме не менее 20 или 24 куб. м. (в учебных и дошкольных учреждениях).

3.3. При строительстве новых и реконструкции действующих средних, средних

специальных и высших учебных заведений помещения для ВДТ и ПК следует проектировать высотой (от пола до потолка) не менее 4,0 м.

3.4. Выбираемое помещение должно быть удалено от потенциальных источников сильных внешних электромагнитных полей и токов - подстанций, мощных трансформаторов, электрических щитов, антенн передающих ТВ -, радио-, метеостанций, РЛС и т.п. Для уменьшения воздействия внешнего электромагнитного поля может потребоваться дополнительное экранирование, например с помощью заземленных оконных решеток.

3.5. При планировании размещения в одном помещении многих пользователей желательно располагать их рабочие места в ряд вдоль боковых стен. Многорядовое размещение с использованием перегородок, также как и неупорядоченное, может усложнить и ухудшить электромагнитную обстановку.

3.6. При размещении дисплеев и ПК непосредственно вдоль перегородок и стен следует учитывать возможное воздействие их полей на находящихся за ними людей и принимать меры к уменьшению такого воздействия.

4.8. Нормативные документы по безопасности компьютерной техники

1. Директива ЕЭС № 90/270/ЕЕС.

"Оператор, работающий с дисплеем, должен быть информирован о мерах безопасности и сохранения здоровья и о мерах, предпринимаемых с целью уменьшения или устранения любого риска".

2. "Шведский стандарт" MPR 1990:10 1990-12-31 комплекса стандартов MPR II. Справочное руководство пользователя для оценки качества дисплеев. (Введен в качестве общеевропейского стандарта с июня 1992 г. директивой Совета ЕЭС от 29 мая 1990 г. № 90/270/ЕЕС)

3. "Шведский стандарт" MPR 1990:8 1990-12-01 комплекса стандартов MPR II. Методика проведения испытаний дисплеев. Визуальные эргономические характеристики. Характеристики излучений. (Введен в качестве общеевропейского стандарта с июня 1992 г. директивой Совета ЕЭС от 29 мая 1990 г. № 90/270/ЕЕС).

4. Шведский стандарт SS 436 1490 "Компьютерная техника - Методы измерения создаваемых ими электрического и магнитного поля", 1995 г.

5. СанПиН 2.2.2.542-96. Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организация работы. (Утверждены и ведены в действие на территории РФ с момента утверждения Постановлением Госсанэпиднадзора России от 14 июля 1996 г. №14; при этом дата введения международных норм на параметры электромагнитных полей - с 1-го января 1997 г.).

6. ГОСТ Р 50948-96. Средства отображения информации индивидуального пользования. Общие эргономические требования и требования безопасности. (Утвержден и введен в действие на территории РФ с 1-го июля 1997 г. Постановлением Госстандарта России от 11 сентября 1996 г. № 576).

7. ГОСТ Р 50949-96. Средства отображения информации индивидуального пользования. Методы измерений и оценки эргономических параметров и параметров безопасности. (Утвержден и введен в действие на территории РФ с 1-го июля 1997 г. Постановлением Госстандарта России от 11 сентября 1996 г. № 577).

8. ГОСТ Р 50923-96. Рабочее место оператора. Общие эргономические требования и требования к производственной среде. Методы измерения. (Утвержден и введен в действие на территории РФ с 1-го июля 1997 г. Постановлением Госстандарта России от 10 июля 1996 г. № 451).

ГЛАВА 2. ПРОГРАММИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ НА АЛГОРИТМИЧЕСКОМ ЯЗЫКЕ PASCAL

1. Основные понятия

1.1. Алфавит и словарь языка

Программа формируется из предложений, состоящих из лексем и разделителей, которые, в свою очередь, формируются из конечного набора литер, образующих алфавит языка Borland Pascal. Этот язык состоит из букв латинского алфавита: прописных - A, B, C, D, E, F, ... X, Y, Z и строчечных - a, b, c, d, e, f, ... x, y, z; арабских цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; и специальных символов.

Использование строчных букв эквивалентно построению соответствующих конструкций из прописных букв и используется для стилистического оформления программы.

Разделителями являются: пробел, конец строки, ';' - точка с запятой (конец предложения) и комментарий, представляющий собой {текстовый литерал}, ограниченный слева и справа фигурными скобками.

Лексемы включают: зарезервированные слова, идентификаторы (стандартные и пользовательские), специальные символы (простые и составные), метки.

Зарезервированные слова представляют собой составную часть языка, имеют фиксированное начертание и определенный смысл. Перечень и перевод на русский язык наиболее часто используемых зарезервированных слов приведен в табл. 1.

Таблица 1

Обозначение	Перевод	Обозначение	Перевод
AND	и	MOD	модуль
ARRAY	массив	NIL	ничто
BEGIN	начало	NOT	не
CASE	в случае	OF	из
CONST	константа	OR	или
DIV	делить	PROCEDURE	процедура
DO	выполнять	PROGRAM	программа
DOWNTO	назад	RECORD	запись
ELSE	иначе	REPEAT	повторить
END	конец	SET	множество
EXTERNAL	внешний	THEN	тогда
FILE	файл	TYPE	тип
FOR	для	TO	к, до
FUNCTION	функция	UNIT	модуль
GOTO	перейти к	UNTIL	пока не
IF	если	VAR	переменная
IN	в	WHILE	пока
LABEL	метка	WITH	с

Стандартные идентификаторы служат для определения заранее зарезервированных идентификаторов предопределенных типов данных, констант,

функций и процедур. Перечень стандартных идентификаторов приведен в табл. 2.

Идентификаторы пользователя применяются для обозначения констант, переменных, процедур и функций. Пользователь должен выбирать имя идентификатора отличное от зарезервированных слов и стандартных идентификаторов.

Правила составления идентификаторов:

1. Идентификатор начинается с буквы или знака подчеркивания.
2. Содержит только буквы, цифры или знак подчеркивания.
3. Между двумя идентификаторами должен стоять разделитель.
4. Максимальная длина 127 символов. Все символы значимы.
5. Идентификатор не может повторять зарезервированное слово.
6. Если идентификатор пользователя повторяет стандартный, то действие стандартного идентификатора отменяется.

Таблица 2

CONST (константы)	FALSE, TRUE, MAXINT, MAXLONGINT, NIL;
TYPE (стандартные типы данных)	ARRAY, BOOLEAN, BYTE, CHAR, FILE, INTEGER, REAL, RECORD, SET, STRING, TEXT, WORD;
FUNCTION (стандартные функции)	ABS, ARCTAN, CHR, COS, EOF, EOLN, EXP, INT, FRAC, LN, ODD, PI, ORD, PRED, ROUND, RANDOM, SIN, SQR, SQRT, SUCC;
PROCEDURE (встроенные процедуры)	ASSIGN, BREAK, CLOSE, CONTINUE DISPOSE, DEC, INC, NEW, READ, READLN, RESET, REWRITE, SEEK, WRITE, WRITELN.

Примеры пользовательских идентификаторов: s, ALPHA_2, X15, _beta.

Специальные символы приведены в табл. 3, табл. 4.

Простые символы Таблица 3

+	плюс	<	меньше	{ }	фигурные скобки
–	минус	_	подчеркивание	[ ]	квадратные скобки
*	звездочка	.	точка	( )	круглые скобки
/	дробная черта	,	запятая	^	тильда
=	равно	:	двоеточие	'	апостроф
>	больше	;	точка с запятой	$	знак денежной единицы

Составные символы Таблица 4

:=	присвоение	<=	меньше, или равно
<>	не равно	>=	больше, или равно
..	диапазон значения		

Метки используются для идентификации операторов в программе при переходе по оператору GOTO. Правила написания меток отличаются от правил составления идентификаторов, следующим – на первом месте может стоять цифра.

Примеры меток: Blok_12, 25, M61, QUIT, 15GX.

1.2. Скалярные, стандартные типы данных

Данные скалярного типа имеют в качестве своего значения одну единственную величину. В этом разделе речь пойдет о четырех важнейших, стандартных типах данных - INTEGER, REAL, BOOLEAN и CHAR.

Константы и переменные

При решении любой задачи требуются данные, над которыми выполняются действия для получения результата. Любое данное является либо константой, либо переменной.

Константы – это данные, значения которых известны заранее и в процессе выполнения программы не изменяются.

Переменные – это данные, которые меняют свое значение по ходу выполнения программы.

Тип INTEGER (целый)

Этот тип представляет множество целых чисел не симметричное относительно нуля диапазона от –32768 до 32767. В памяти ЭВМ под целое число отводится два байта (16 бит). Наибольшему значению целого числа 32767 соответствует стандартный идентификатор MAXINT, а наименьшему – выражение NOT (MAXINT) = – (MAXINT+1), или число –32768.

Операции, определенные над целыми числами: + сложение, – вычитание, * умножение, DIV – целочисленное деление, MOD – остаток от целочисленного деления, AND – арифметическое 'И', OR – арифметическое 'ИЛИ', NOT – арифметическое отрицание, XOR – исключающая дизъюнкция. Примеры использования этих операций приведены в табл. p1 приложения 1.

Любая из этих операций выполнима над двумя целыми числами, если абсолютная величина результата не превышает MAXINT (для умножения). В противном случае возникает прерывание программы, связанное с переполнением.

Например: требуется вычислить выражение 1000 * 4000 div 2000. Поскольку операции умножения и деления имеют один приоритет и выполняются слева направо в порядке записи арифметического выражения, то при умножении произойдет прерывание, связанное с переполнением. Выход из этой ситуации возможен при изменении порядка выполнения операций умножения и деления для чего используются круглые скобки ==> 1000 * (4000 div 2000).

Предусмотрено представление целых чисел в шестнадцатеричной системе счисления. Форма записи таких чисел $X, где X – целая константа, а символ $ – признак.

Примеры: $57, $1FF. Следует помнить, что в шестнадцатеричной системе счисления цифры 10, 11, 12, 13, 14 и 15 заменяются латинскими буквами A, B, C, D, E и F, соответственно.

Кроме типа INTEGER в языке Borland Pascal предусмотрены и другие целые типы данных BYTE, SHORTINT, WORD и LONGINT (см. табл. 5). Все эти типы определены на множестве целых чисел, характеризуются одним набором арифметических операций и отличаются диапазоном значений и объемом занимаемой памяти.

Целые типы данных Таблица 5

Название типа	Длина, байты	Диапазон значений числа
BYTE	1	0 .. 255
SHORTINT	1	–128 .. 127
WORD	2	0 .. 65535
INTEGER	2	–32768 .. 32767
LONGINT	4	–2147483648 .. 2147483647

Тип REAL (вещественный)

Число типа REAL занимает три слова (шесть байтов). При работе с вещественными числами нужно помнить, что на любом отрезке вещественной оси существует бесчисленное множество чисел. Поскольку для кодирования вещественного числа отведено всего шесть байтов памяти, то расчеты выполняются всегда с конечной точностью, которая зависит от формата числа.

Вещественное число записывается и хранится в памяти компьютера в виде $X = m \cdot B^n$, где m – мантисса, B – основание представления числа с плавающей точкой, n – порядок (целое число). Имеют место ограничения –M1 < m < +M2; –E1< n < +E2. В этих выражениях B, E и M – константы, характеризующие представление числа. В табл. 6 приведены значения этих констант для вещественных типов данных, используемых в Borland Pascal.

Вещественные типы данных Таблица 6

Название типа	Длина, байты	Мантисса, количество значащих цифр	Диапазон десятичного порядка
SINGL	4	7 .. 8	-45 .. 38
REAL	6	11 .. 12	-39 .. 38
DOUBLE	8	15 .. 16	-324 .. 308
EXTENDED	10	19 .. 20	-4951 .. 4932

Так для типа REAL основание B равно 10. Размер мантиссы 11 - 12 десятичных чисел. Диапазон десятичного порядка равен [-39, +38]. Таким образом, на отрезке оси вещественных чисел в заданном диапазоне можно закодировать только конечное число значений, и поскольку на оси таких чисел бессчетное множество, то выбирается интервал, дискрет, на который этот диапазон (отрезок) делится. Число таких интервалов конечно. Каждый дискрет ставится в соответствие значению вещественного числа. Конечное множество определенных таким образом представителей вещественных чисел называется *континуумом*. Результаты вычислений округляются до чисел этого множества, поэтому необходимо говорить о точности вычислений. Округление результата происходит до ближайшего вещественного числа большего данного по модулю. Следует также отметить, что эти интервалы не являются равными. В соответствии с полулогарифмическим способом своего представления интервалы "растягиваются" с увеличением порядка. Наибольшая точность расчетов достигается в центральной части диапазона изменения вещественного числа X (например, в районе 1.0E+00 погрешность вычислений 0,00000000001), и наименьшая - на его краях (например, в окрестностях числа 1.0E+38 погрешность вычисления равна 10000000000000000000000000000).

Существует две формы отображения вещественных чисел (см. табл. 7): полулогарифмическая (с плавающей точкой) и естественная (с фиксированной точкой).

Правильное представление вещественных чисел Таблица 7

С плавающей точкой	С фиксированной точкой
0.000000E +00 (нормальная форма)	0.0
1.340000E +02	134.0
-7611.E -02	-76.11
-98.3569E -05	-0.000983569

Ниже, в табл. 8 приведены примеры ошибочного представления вещественных чисел.

Таблица 8

E +05	нет мантиссы	8.	отсутствует дробная часть вещественного числа
-5.1 E +02	пробел между мантиссой и основанием	-.500	отсутствует целая часть вещественного числа
155.27.54	две точки в числе		

Над вещественными числами определены операции сложения (+), вычитания (-), умножения (*) и деления (/). Операция возведения в степень не предусмотрена. Примеры использования арифметических операций над вещественными числами приведены в табл. 49 приложения 1.

Использование типа REAL у начинающего программиста часто вызывает ряд ошибок, приводящих к искажению результата по следующим причинам:

- Ошибки *ввода* - недостаточная точность исходных данных при сборе, подготовке и их вводе в ЭВМ;
- Ошибки *представления* обуславливаются ограниченной точностью внутреннего представления данных в конкретной ЭВМ, используемой для расчетов;
- Ошибки *вычислений* возникают за счет несовершенства математических методов, выбранных для решения задачи. Необходимо оценивать погрешность и держать ее в заданных пределах.

Тип BOOLEAN (булевский, логический)

Логический тип в языке Паскаль задается как перечисляемый тип, содержащий всего два значения, имеющих идентификаторы FALSE (ложь) и TRUE (истина). Элементам этого типа поставлены в соответствие номера 0 значению FALSE и 1 - TRUE, поэтому FALSE < TRUE.

В памяти ЭВМ переменные этого типа занимают один байт. Над данными этого типа определены операции: дизъюнкция ($\vee$) OR, конъюнкция ($\wedge$) AND, исключающее ИЛИ ($\oplus$), отрицание ($\neg$) NOT, а также отношения <, >, <=, >=, <>, =. Результаты выполнения логических операций над булевыми переменными P и Q приведены в табл. 9.

Таблица 9

P	Q	$P \wedge Q$	$P \vee Q$	$P \oplus Q$	$\neg P$	$\neg Q$
False	False	False	False	False	True	True
True	False	False	True	True	False	True
False	True	False	True	True	True	False
True	True	True	True	False	False	False

Следует отметить, что операции сравнения данных любых типов имеют результат типа BOOLEAN. Например, если даны переменные с именами P, Q типа BOOLEAN и X, Y, Z типа REAL, причем X = 5.8, Y = 8, а Z = 10.3, то справедливы утверждения:

Q := (X < Y) ∧ (Y <= Z) ====> TRUE
P := X = Y ====> FALSE

Наиболее часто булевский тип данных используется для управления порядком выполнения операторов в программе.

В языке имеется функция ODD(X), где X целое число. Если X четно, то ODD(X) принимает значение FALSE, если X нечетно, то ODD(X) - TRUE.

БУЛЕВЫ СООТНОШЕНИЯ

1.	P ∨ Q = Q ∨ P, P ∧ Q = Q ∧ P;	Коммутативность
2.	(P ∨ Q) ∨ R = P ∨ (Q ∨ P), (P ∧ Q) ∧ R = P ∧ (Q ∧ R);	Ассоциативность
3.	(P ∧ Q) ∨ R = (P ∨ R) ∧ (Q ∨ R), P ∨ Q) ∧ R = (P ∧ R) ∨ (Q ∧ R);	Дистрибутивность
4.	¬ (P ∨ Q) = ¬ P ∧ ¬ Q, ¬ (P ∧ Q) = ¬ P ∨ ¬ Q.	Законы Де Моргана

При решении практических задач могут возникнуть самые разнообразные условия, которые следует записать в виде логических выражений, использующих логические операции и отношения. В этом случае бывает полезно помнить о следующих булевых соотношениях и эквивалентных преобразованиях.

ЭКВИВАЛЕНТНЫЕ ПРЕОБРАЗОВАНИЯ

X ≠ Y	<------->	¬ (X = Y);
X ≤ Y	<------->	¬ (X > Y) ∨ (X = Y);
X ≥ Y	<------->	¬ (X < Y);
X > Y	<------->	¬ (X < Y) ∧ ¬ (X = Y).

Тип CHAR (литерный, символьный)

Этот тип задает конечное упорядоченное множество символов (литер), допускаемое в конкретной реализации языка. В IBM PC для внутреннего представления символов, хранения символьной информации на внешних носителях и кодирования знакогенераторов дисплеев и печатающих устройств используется восьмиразрядный код ASCII (American Standard Code for Information Interchange - стандартный американский код, используемый для обмена информацией). Емкость кода 256 единиц.

Общая часть кодовой таблицы для всех персональных компьютеров содержит символы, имеющие коды от 32 до 127 сведенные в табл. 10.

Первые позиции 0 - 31 заняты под коды управления устройствами (монитор, принтер и др.) и могут иметь разное воздействие на разные устройства и в табл. 10 не включены. Например, код 7 вызывает звуковой сигнал при выводе информации на дисплей — WRITELN('Проверьте принтер!', CHR(7));.

Таблица кодов (общая часть) Таблица 10

32 - пр.	48 - 0	64 - @	80 - P	96 - `	112 - p
33 - !	49 - 1	65 - A	81 - Q	97 - a	113 - q
34 - "	50 - 2	66 - B	82 - R	98 - b	114 - r
35 - #	51 - 3	67 - C	83 - S	99 - c	115 - s
36 - $	52 - 4	68 - D	84 - T	100 - d	116 - t
37 - %	53 - 5	69 - E	85 - U	101 - e	117 - u
38 - &	54 - 6	70 - F	86 - V	102 - f	118 - v
39 - '	55 - 7	71 - G	87 - W	103 - g	119 - w
40 - (	56 - 8	72 - H	88 - X	104 - h	120 - x
41 -)	57 - 9	73 - I	89 - Y	105 - i	121 - y
42 - *	58 - :	74 - J	90 - Z	106 - j	122 - z
43 - +	59 - ;	75 - K	91 - [	107 - k	123 - {
44 - ,	60 - <	76 - L	92 - \	108 - l	124 - ¦
45 - -	61 - =	77 - M	93 -]	109 - m	125 - }
46 - .	62 - >	78 - N	94 - ^	110 - n	126 - ~
47 - /	63 - ?	79 - O	95 - _	111 - o	127 -

Коды 13 и 10 для дисплея или принтера осуществляют перевод курсора в начало текущей строки и переход на следующую строку. Эти коды можно использовать для вывода информационного сообщения, составляющего несколько строк, с помощью одного оператора вывода:

```
WRITELN('Внимание!' + CHR(13) + CHR(10) + 'Следите за экраном.');.
```

Таблица кодов (переменная часть) Таблица 11

128 - А	144 - Р	160 - а	176 - ▒	192 - └	208 - ╨	224 - р	240 - Ё
129 - Б	145 - С	161 - б	177 - ▒	193 - ┴	209 - ╤	225 - с	241 - ё
130 - В	146 - Т	162 - в	178 - ▓	194 - ┬	210 - ╥	226 - т	242 - Є
131 - Г	147 - У	163 - г	179 - │	195 - ├	211 - ╙	227 - у	243 - є
132 - Д	148 - Ф	164 - д	180 - ┤	196 - ─	212 - ╘	228 - ф	244 - Ї
133 - Е	149 - Х	165 - е	181 - ╡	197 - ┼	213 - ╒	229 - х	245 - ї
134 - Ж	150 - Ц	166 - ж	182 - ╢	198 - ╞	214 - ╓	230 - ц	246 - Ў
135 - З	151 - Ч	167 - з	183 - ╖	199 - ╟	215 - ╫	231 - ч	247 - ў
136 - И	152 - Ш	168 - и	184 - ╕	200 - ╚	216 - ╪	232 - ш	248 - °
137 - Й	153 - Щ	169 - й	185 - ╣	201 - ╔	217 - ┘	233 - щ	249 - ●
138 - К	154 - Ъ	170 - к	186 - ║	202 - ╩	218 - ┌	234 - ъ	250 - •
139 - Л	155 - Ы	171 - л	187 - ╗	203 - ╦	219 - ▒	235 - ы	251 - √
140 - М	156 - Ь	172 - м	188 - ╝	204 - ╠	220 - ▒	236 - ь	252 - №
141 - Н	157 - Э	173 - н	189 - ╜	205 - ═	221 - ▒	237 - э	253 - ¤
142 - О	158 - Ю	174 - о	190 - ╛	206 - ╬	222 - ▒	238 - ю	254 - ■
143 - П	159 - Я	175 - п	191 - ┐	207 - ╧	223 - ▒	239 - я	255 -зб.

Примечание: в табл. 10 и 11 сокращения (пр.) и (зб.) означают пробел и забой, соответственно.

Переменная часть кодовой таблицы содержит национальный алфавит, символы псевдографики и специальные нестандартные символы. Коды 128 - 255, приведенные в табл. 11, отражают модифицированную кодировку ГОСТа для подключения кириллицы.

Значения констант и переменных типа CHAR есть один символ из допустимого набора, например: 'Z', 'j', '2', '*', 'Ц', 'д', 'г'. Второй способ записи символа в программе состоит в использовании префикса # перед номером литеры. Примеры символов: #90, #106, #50, #42, #150, #164.

Описываются переменные этого типа как VAR CH1, CH2:CHAR;

Использование переменных типа CHAR в арифметических выражениях запрещено. К данным этого типа могут применяться только операции сравнения, при этом результат зависит от порядковых номеров литер в кодовой таблице символов.

Например: 'B' > 'A' ==> FALSE, '1' <= '9' ==> TRUE.

Множество цифр и букв не только упорядочено в соответствии с кодом литер от 32 до 255, но и связно - код последующей литеры больше кода предшествующей на 1. Таким образом, '0' < '1' < '2' < ... < '9'; 'A' < 'B' < 'C' < 'D' < ... < 'Z'; 'а' < 'б' < 'в' < 'г' < ... < 'я'.

Для работы с литерами часто используются функции CHR, ORD, PRED, SUCC, описание которых приведено в табл. 15.

Пример 1. Вывести на экран монитора литеры, коды которых начинаются с 32 и заканчиваются - 255.

```
PROGRAM PR1;
VAR   I: INTEGER;
BEGIN
      FOR I:=32 TO 255 DO WRITELN('код =', I:-3,' ==> ', CHR(I))
END.
```

1.3. Встроенные функции

Наиболее часто встречающиеся операции над скалярными типами данных реализованы в языке Паскаль с помощью встроенных (иногда говорят - стандартных) функций и процедур. Наиболее известные функции над переменными целого, вещественного, логического и литерного типов приведены в табл. 12, 13, 15.

Встроенные арифметические функции Таблица 12

Функция	Содержание
ABS(X)	Модуль (абсолютная) величина X, $\lvert X \rvert$
ARCTAN(X)	Главное значение арктангенса X, ArctgX
COS(X)	Косинус от X, заданного в радианах, CosX
EXP(X)	Показательная функция от X, e^X
FRAC(X)	Дробная часть от X, {X}
INT(X)	Целая часть числа X в вещественной форме,]X[
LN(X)	Натуральный логарифм от X, LnX
SIN(X)	Синус от X, заданного в радианах, SinX
SQR(X)	Квадрат (вторая степень) числа X, т.е. X^2

Продолжение табл. 12

SQRT(X)	Корень квадратный из X, $\sqrt{X}$
RANDOM(N)	Равномерно распределенное псевдослучайное целое число от 0 до N
RANDOM	Равномерно распределенное псевдослучайное вещественное число от 0 до 1
ROUND(X)	Возвращает значение X округленное до ближайшего целого числа
TRUNC(X)	Возвращает ближайшее целое число, меньшее или равное X, если X > 0, и большее, или равное X, если X < 0

В табл. 13 приведены примеры вычислений по функциям INT, ROUND, TRUNC для пояснения особенностей их использования.

Таблица 13

Функция:	X	INT(X)	ROUND(X)	TRUNC(X)
Тип:	REAL	REAL	INTEGER	INTEGER
Номер примера:				
1	123.44	123.0	123	123
2	34.50	34.0	35	34
3	1.70	1.0	2	1
4	-25.14	-25.0	-25	-25
5	-10.70	-10.0	-11	-10
6	-0.41	0.0	0	0
7	-0.50	0.0	-1	0

Встроенные логические (булевские) функции Таблица 14

Функция	Содержание
ODD(N)	N - целочисленная переменная, результат TRUE, если N - нечетное число и FALSE, если N - четное число
EOF(F)	Возвращает значение TRUE, если достигнут конец файла F, в противном случае FALSE
EOLN(F)	Принимает значение TRUE, если при чтении текстового файла F достигнут конец текущей строки, FALSE - в противном случае

Встроенные функции над перечисляемыми типами данных Таблица 15

SUCC(s)	SUCC('O')	'П'	Возвращает значение следующего за S данного перечисляемого типа	совпадает с типом аргумента S
	SUCC(-90)	-89		
	SUCC('a')	'b'		
PRED(s)	PRED('П')	'O'	Возвращает значение предшествующего S в порядке возрастания в списке	совпадает с типом аргумента S
	PRED(-90)	-91		
	PRED('b')	'a'		
ORD(s)	ORD('Щ')	153	Возвращает порядковый номер идентификатора S в списке	INTEGER
	ORD('4')	52		
	ORD(FALSE)	0		
CHR(s)	CHR(87)	'W'	Возвращает литеру с кодом S, если она существует.	CHAR
	CHR(201)	'r'		
	CHR(53)	'5'		

1.4. Структура программы

Программист вводит текст программы, произвольно располагая строки на экране. Отступ слева выбирает сам программист, чтобы программа была более «читабельной». В одной строке допускается писать несколько операторов. Длинные операторы можно переносить на следующую строку. Перенос допускается в любом месте, где можно сделать пробел. Максимальная длина строки не должна превышать 127 символов. Из соображений наглядности, удобства просмотра и отладки программы рекомендуется длину строки ограничивать 80 символами. Программы имеют жесткую структуру, описанную в табл. 16.

Структура программы Таблица 16

Фрагмент программы	Содержание	Примечание
Заголовок	Program <имя программы>	Необязательный
Раздел 0	USES, описание модулей, библиотек	Разделы описаний программного блока
Раздел 1	LABEL, описание меток	
Раздел 2	CONST, описание констант	
Раздел 3	TYPE, описание типов данных	
Раздел 4	VAR, описание переменных	
Раздел 5	PROCEDURE, FUNCTION - описание процедур и функций, используемых в программе	
Раздел 6	BEGIN … END. – тело программы	Раздел исполняемых операторов

Синтаксические правила построения предложений языка можно описывать следующими способами:

- Схема (формат предложения, или раздела). В учебнике выбран именно этот способ, поскольку он наиболее понятен начинающему программисту.
- Синтаксическая диаграмма. Этот способ детально формализует синтаксис предложения и используется разработчиками трансляторов с языка Паскаль.
- Порождающими правилами РАСШИРЕННЫХ БЭКУСА - НАУРА ФОРМ (РБНФ). Это весьма компактный и, в то же самое время, наглядный способ записи языковых конструкций. Этот способ используется в статьях и научных разработках. В данном учебнике используются только четыре элемента РБНФ, описанные в табл. 17.

Таблица 17

Соглашение	Толкование
Угловые скобки < >	Угловые скобки заключают в себе элемент синтаксиса, который Вы должны задать. Текст внутри угловых скобок характеризует элемент, однако, не описывает синтаксис этого элемента.
Квадратные скобки [ ]	Квадратные скобки в синтаксических конструкциях заключают в себе один или несколько необязательных элементов.
Вертикальная черта |	Разделяет два альтернативных элемента синтаксической конструкции, один из которых нужно выбрать.
Фигурные скобки { }	Фигурные скобки заключают в себе несколько элементов, разделенных 'I'. Из них нужно выбрать один.
Многоточие ...	Многоточие показывает, что можно повторить некоторый элемент один и более раз.

Раздел описания модулей USES

Раздел имеет структуру: USES Модуль 1, Модуль 2, ... Модуль N, где за ключевым словом USES указывается список, в котором перечисляются все имена библиотек (модулей) стандартных и пользовательских, к процедурам и функциям которых есть обращение в программе. Если таких обращений нет, то раздел USES не нужен.

Пример: USES CRT, GRAPH, HELP, MYLIB;

В этом примере две стандартные библиотеки - CRT, GRAPH, и две пользовательские библиотеки - HELP, MYLIB.

Раздел описания меток LABEL

Раздел имеет структуру: LABEL Метка 1, Метка 2, ..., Метка N, где за ключевым словом LABEL указывается список, в котором перечисляются все имена меток, встречающихся в программе.

Пример: LABEL M1, 12_BL, 9999;

Метки позволяют менять естественный ход выполнения программы. Ссылка на метку осуществляется оператором GOTO <метка>. Если в программе меток нет, то раздел LABEL отсутствует. В теле программы (в разделе операторов) метка ставится перед требуемым оператором и отделяется от него двоеточием.

Пример: M27: X := A * B – C/2;

Областью действия метки является блок, где она описана. Ниже приведена схема использования меток в тексте программы.

```
LABEL метка 1, метка 2;
      . . .
BEGIN
  метка 1: <Оператор I>;
      . . .
  метка 2: <Оператор J>;
      . . .
END.
```

Раздел описания констант CONST

Раздел существует, если в алгоритме используется, по крайней мере, одна константа, то есть величина не изменяющая свое значение в процессе выполнения программы. Попытка изменить значение константы в теле программы будет обнаружена на этапе трансляции.

В стандарте на Паскаль константы определяются следующим способом:

```
CONST    <Идентификатор 1> = <Значение 1>;
                <Идентификатор 2> = <Значение 2>;
                                    . . .
                <Идентификатор N> = <Значение N>;
```

Примеры констант:

```
CONST  A = 15.7; BXZ = 'Серия N123/5'; MIN_IND = $15D;
C_10 = –0.57E–6;  L125 = 695; FLAG = TRUE;
```

Константа может иметь только предопределенный (стандартный) тип

данных. Тип присваивается константе по внешнему виду значения и в соответствии с этим типом отводится память для хранения значения константы.

В нарушении всех принципов стандартного Паскаля в Borland Pascal в разделе констант разрешается присваивать начальные значения элементам массива. Например:

```
CONST  N = 5;
       X: ARRAY[1..N] OF REAL = (1.5,-8,0.56,89,-100.56);
```

Далее по тексту программы с массивом X можно работать так же, как и с любым массивом заданным в неявной форме в разделе VAR.

В качестве расширения стандартного Паскаля разрешено использовать выражения, составленные из ранее определенных констант и некоторых стандартных функций (Abs, Chr, Hi, Length, Lo, Odd, Ord, Pred, Prt, Round, SizeOf, Succ, Swap, Trunc). Примеры использования константных выражений:

```
CONST Min = 0; Max = 250; Centr = (Max-Min) div 2;
      Beta = Chr(225); NumChars = Ord('2') - Ord('A')+1;
      Message = 'не хватает памяти';
      ErrStr  = 'Ошибка: ' + Message + '.'
      Ln10  = 2.302585092994045884; Ln10R = 1/Ln10;
```

Константные выражения вычисляются компилятором без выполнения программы на этапе ее создания.

Раздел описания типов TYPE

Стандартные типы данных (REAL, INTEGER, BOOLEAN, CHAR) не требуют описаний в этом разделе. Описания требуют только типы, образованные пользователем.

Концепция типов - одно из основных понятий в языке. С каждым данным связывается один, и только один определенный тип.

Тип - это множество значений + множество операций, которые можно выполнять над этими значениями, то есть правила манипулирования данными. Использование типов позволяет выявлять многочисленные ошибки, связанные с некорректным использованием значений или операций, еще на этапе трансляции без выполнения программ.

О Паскале говорят, что он строго типизирован, то есть программист должен описать все объекты, указывая их типы, и использовать в соответствии с объявленными типами. Программы становятся более надежными и качественными. При компиляции информация используется для уточнения вида операции. Так знаком + для данных типа REAL и INTEGER обозначается операция сложения, а для множеств (тип SET) - объединение. Структура раздела описания типов имеет вид:

```
TYPE    <имя типа 1> = <значение типа 1>;
        <имя типа 2> = <значение типа 2>;
                  . . .
        <имя типа L> = <значение типа L>;
```

Имя типа представляет собой идентификатор, который может употребляться в других типах, описанных вслед за данным типом. Раздел TYPE не является обязательным, так как тип можно описать и в разделе переменных VAR.

Примеры описания пользовательских типов:

```
TYPE   DAY = 1..31;  Year = 1900 .. 2000;  {Интервальный тип}
       LatBukv = ('A', 'C', 'D', 'G', 'H');{Перечисляемый тип}
       Matr = array[-1..12, 'A' .. 'F] of real; {Регулярный тип}
```

Раздел описания переменных VAR

Это обязательный раздел. Любая встречающаяся в программе переменная должна быть описана. В языке нет переменных, объявляемых по умолчанию. Основная цель этого раздела - определить количество переменных в программе, какие у них имена (идентификаторы) и данные каких типов хранятся в этих переменных. Таким образом, переменная - это черный ящик, а тип показывает, что в него можно положить. Структура раздела имеет вид:

```
VAR        <список 1 идентификаторов переменных>:<тип 1>;
           <список 2 идентификаторов переменных>:<тип 2>;
                                  . . .
           <список N идентификаторов переменных>:<тип N>;
```

Тип переменных представляет собой имя (идентификатор), описанный в разделе TYPE, при явном описании типа, или собственно описание типа в случае его неявного задания. Примеры описания переменных:

```
TYPE       DAY = 1..31; Matr = ARRAY[1..5, 1..8] OF INTEGER;
VAR        A, B: DAY; X, Y: Matr;     { явное описание типов }
           YEAR: 1900 .. 2000; LES: (LPT, PRN); { неявное описание типов }
           A, B, CD, FER51: REAL;     { описание переменных стан-}
           EQUAL: BOOLEAN; SH: CHAR; { дартных типов производится }
           I, J, K: INTEGER;          { только в разделе VAR }
```

Раздел описания процедур и функций

Стандартные процедуры и функции, имена которых включены в список зарезервированных слов, в этом разделе не описываются. Описанию подлежат только процедуры и функции, определяемые пользователем.

```
PROCEDURE <имя процедуры> (<параметры>);  { заголовок процедуры }
        < разделы описаний >   { тело процедуры }
BEGIN
        < раздел операторов >
END;
FUNCTION <имя функции>(<параметры>): <тип результата>; { заголовок }
        < разделы описаний >                    { тело функции }
BEGIN
        < раздел операторов >
END;
```

Структура процедур и функций та же самая, что и у основной программы. Отличие описаний состоит в том, что идентификаторы констант, переменных, процедур и функций, описанных в соответствующих разделах описаний пользовательских процедур и функций, распространяются только на блоки, где они описаны и на блоки внутренние по отношению к ним. На внешние блоки, в том числе на тело основной программы, они не распространяются.

Раздел операторов

Это основной раздел, именно в нем в соответствии с предварительным описанием переменных, констант, функций и процедур выполняются действия, позволяющие получать результаты, ради которых программа и писалась.

Синтаксис раздела операторов основной программы:

```
BEGIN
        < Оператор 1; >          { Операторы выполняются }
        < Оператор 2; >          { строго последовательно   }
              . . .                        { друг за другом. }
        < Оператор N >
END.
```

Комментарий

Это пояснительный текст, который можно записать в любом месте программы, где разрешен пробел. Текст комментария ограничен: слева – '{', справа – '}', и может содержать любые символы. Комментарий игнорируется транслятором и на программу влияния не оказывает. Пример использования комментария:

```
PROGRAM PR;
        < Разделы описаний >
BEGIN
          < Оператор 1; >
          < Оператор 2; >
        {  < Оператор 3; >
                . . .
           < Оператор N >   }
END.
```

Средства комментария часто используются для отладки. Так, в приведенном выше примере, операторы - 3, 4, 5 ... N, заключенные в фигурные скобки, временно не выполняются.

Правила пунктуации

Основным средством пунктуации является символ точка с запятой - ";".

1. Точка с запятой не ставится после слов LABEL, TYPE, CONST, VAR, а ставится после каждого описания этих разделов.
2. Точка с запятой не ставится после BEGIN и перед END, так как эти слова - операторные скобки.
3. Точка с запятой разделяет операторы, и ее отсутствие вызовет:

```
A := 333                    { ошибка - нет ';' }
B := A/10;;;;;    { четыре пустых оператора }
```

4. Возможна ситуация:

```
        END;          следует писать          END
    END;          ================>      END
END;                                          END;
```

5. Допускается запись метки на пустом операторе - <Метка> : ;
6. Точка с запятой не ставится после операторов WHILE, REPEAT, DO и перед UNTIL.
7. В условных операторах ";" не ставится после THEN и перед ELSE.

2. Программирование вычислительных процессов

Решение задачи на ЭВМ - это сложный процесс, в ходе которого пользователю приходится выполнять целый ряд действий, прежде чем он получит интересующий его результат.

Идеальная модель процесса решения задач на компьютере показана с помощью табл. 18.

Таблица 18

Этапы решения задачи	Исходные данные, результаты
Начало решения.	Формулировка цели. Концептуальная, содержательная постановка задачи.
Постановка задачи. Построение модели.	Формальная математическая постановка задачи для ЭВМ.
Алгоритмизация.	Алгоритм.
Запись алгоритма на языке программирования.	Исходный текст программы на алгоритмическом языке. Программа в кодах ЭВМ.
Выполнение программы.	Результаты выполнения программы - числа, диаграммы, графики.
Анализ и использование результатов работы программы.	Выводы по результатам решения исходной задачи.
Конец решения.	

Детальное рассмотрение этапов решения задачи на ЭВМ выходит за рамки данного учебника. Лишь приводятся некоторые существенные определения и понятия, которые будут использоваться далее по тексту.

Алгоритмизация - это процесс проектирования алгоритма, то есть выделение совокупности действий, используемых в математическом методе, и сведения их к совокупности действий, которые будет выполнять ЭВМ.

Алгоритм - совокупность точно описанных действий, приводящих от исходных данных к желаемому результату и удовлетворяющих следующим свойствам:

- *Определенность* - алгоритм должен однозначно, точно и понятно задавать выполняемые действия (для одних и тех же исходных данных должен получаться один и тот же результат);
- *Дискретность* - алгоритм должен представлять действие в виде последовательности составных частей;
- *Результативность* - алгоритм должен приводить к желаемому точному результату за конечное время;
- *Массовость* - алгоритм должен быть приемлем для решения всех задач определенного класса.

Структурное проектирование - проектирование сверху вниз, это подход к разработке и реализации, который состоит в преобразовании алгоритма в такую

последовательность все более конкретных алгоритмов, окончательный вариант которой представляет собой программу для вычислительной машины.

Существуют различные способы записи алгоритмов (словесный, формульно-словесный, графический, аналитический). Каждый из способов имеет свои плюсы и минусы. В данном учебнике, в основном, используется два способа графического представления алгоритма: блок-схема и структурограмма.

Блок-схема - это графическое представление алгоритма, дополняемое словесными записями. Каждый блок алгоритма отображается геометрической фигурой (блоком). Правила выполнения схем алгоритмов регламентируют ГОСТ 19.002-80, ГОСТ 19.003-80. Графические фигуры соединяются линиями, потоками информации. Эти линии задают порядок переходов от блока к блоку. Блок-схемы наглядны для представления простых вычислительных алгоритмов и ориентированы для программирования задач на языках АССЕМБЛЕР, АЛГОЛ, БЕЙСИК, ФОРТРАН, ПЛ. В этих языках программировании требуется четко отслеживать последовательность команд, и широко используется оператор GOTO.

В ПАСКАЛЕ и ему подобных языках структурного программирования блок-схемы использовать в классическом виде стало неудобно. Это связано, во-первых, с громоздкостью алгоритма, реализованного в виде блок-схем, во-вторых, с нарушением основного принципа структурного проектирования программ сверху вниз. В соответствии с этим принципом рисуются укрупненные блоки алгоритма, которые потом уточняются. Этот процесс продолжается до тех пор, пока каждый алгоритмический блок не станет однозначно отражать одну или несколько языковых конструкций выбранного разработчиком языка программирования. Такие операторы как For ... To, While ... Do, Repeat ... Until, Case ... of отразить с помощью блок-схем, не описывая принципов их работы, невозможно. Поэтому для пояснения работы этих и некоторых других операторов Паскаля в тексте используются блок-схемы, как алгоритмы нижнего уровня. А для объяснения вычислительных процессов решения сложных задач, в том числе численных методов, были использованы изображения алгоритма с помощью структурограммы (схемы Насси-Шнайдермана). Эти схемы позволяют заменить линии передачи управления между блоками на правила вложения структур. Для Паскаля использование структурограмм существенно упрощает запись алгоритма, так как явно описываются такие операторы как: IF, FOR, WHILE ... DO, REPEAT ... UNTIL, CASE ... OF, что делает алгоритм компактным, наглядным и легко программируемым.

2.1. Линейные процессы вычислений

Простейший алгоритм представляет собой цепочку блоков (операторов): от начального блока до конечного. Каждый блок должен быть выполнен один единственный раз. Это линейный алгоритм. Он отражает линейный вычислительный процесс. Основой линейных процессов является последовательность операторов, обеспечивающих ввод исходных данных, вычисление выражений, вывод результатов расчетов на экран или печать.

Операторы ввода (чтения)

Ввод числовых данных, символов, строк и т.д. с клавиатуры обеспечивается операторами вызова стандартных процедур:

READ(X1, X2, X3,...), или READLN(X1, X2, X3, ...).

Где X1, X2, X3, ... - идентификаторы скалярных переменных. Данные, со-

ответствующие переменным X1, X2, X3, ... вводятся с клавиатуры и разделяются либо пробелом, либо Entr. После ввода последнего данного всегда нажимается клавиша Entr.

Отличие оператора READLN от READ заключается в том, что после считывания последнего в списке X1, X2, X3, ... значения, данные для следующего оператора READLN будут считываться с начала новой строки. То есть, если список ввода X1, X2, X3,... оператора READLN меньше, чем число набранных в одну строку через пробел чисел, то оставшиеся в строке числа будут проигнорированы. Оператор READ сохранит оставшиеся числа для дальнейшего ввода. Вводимые данные должны строго соответствовать типам переменных, описанных в разделе Var, в противном случае будут возникать сообщения об ошибках ввода.

Оператор READLN без параметров вызывает приостановку программы до момента нажатия клавиши Entr.

Операторы вывода (записи)

Вывод числовых данных, символов, строк и булевских значений на экран дисплея осуществляется с помощью операторов вызова стандартных процедур: WRITE(X1, X2, X3, ...), или WRITELN(X1, X2, X3, ...).

Отличие этих операторов состоит в том, что WRITELN после распечатки списка выражений X1, X2, X3, ... автоматически переводит курсор в начало следующей строки, а после выполнения оператора WRITE курсор остается за последним символом, выведенным на экран.

Оператор WRITELN с пустым списком выводимых данных выводит строку пробелов. Управление форматом выводимых данных осуществляется непосредственно в операторе вывода. Для этого используется запись элемента из списка $\{ X_i \}$ в виде - X [: B [: C]], где X - выражение, идентификатор переменной или константа, B - ширина поля для вывода данного X, C - точность (только для типа REAL). Точность представления определяет количество цифр после точки (дробная часть числа). Если указанная ширина поля оказывается "слишком большой", то значение данного выводится со стоящими впереди пробелами. Если указанная ширина поля "мала", то в строке вывода для значения этого данного добавляются (автоматически) необходимые позиции. Параметры формата (ширина поля B и точность C) могут быть константой, переменной или выражением целого типа.

Пример 2. Описать формат вывода арифметического выражения X, численное значение которого $|X| < 1000$, с точностью до пяти знаков после десятичной точки.

Решением этой задачи является оператор:

```
WRITELN( X : 10 : 5 )
```

На рис. 1 показана схема формата X:10:5. Цифра 10 определяет ширину поля, то есть общее количество литер, отведенное для отображения вещественного числа вместе со знаком и десятичной точкой. Цифра 5 – точность - указывает количество цифр мантиссы. В этом примере результат вычисления X выводится на экран дисплея в форме вещественного числа с фиксированной точкой.

Рис. 1. Схема отображения вещественного числа с фиксированной точкой

Если задать формат в виде X:8, то вещественное число будет представлено в формате с плавающей точкой и будет включать для своего представления восемь литер. Ограничение ширины поля скажется на разрядности мантиссы. Для вывода целой части X можно форматировать - X:5:0. В этом случае "точность" равна 0, и десятичная точка не отображается на экране. Дробная часть вещественного числа округляется с указанной точностью, а не отбрасывается.

Примеры форматов вещественных чисел Таблица 19

X	Тип	Формат	Результат	Примечание
12.336	REAL	X:5:2	12.34	Формат вещественного числа с фиксированной точкой.
12.334	REAL	X:5:2	12.33	
-12.339	REAL	X:6:2	-12.34	
-0.0123	REAL	X:6:3	-0.012	
12.334	REAL	X:5:0	12	Округление вещественного числа до целого числа с помощью формата.
12.534	REAL	X:5:0	13	
-12.534	REAL	X:5:0	-13	
-123.456	REAL	X:10	-1.235E+02	Формат вещественного числа с плавающей точкой.
-12.3456	REAL	X: 8	-1.2E+01	
12.3456	REAL	X: 8	1.2E+01	
0.012	REAL	X:11	1.2000E-02	
0.0156	REAL	X	1.5600000000E+02	Бесформатный вывод вещественного числа.
-12.3456	REAL	X:5:0	-1.2345600000E+01	

Оператор присваивания

Вычисления, в большинстве случаев, реализуются с помощью оператора присваивания, который имеет формат:

<Идентификатор> := <Выражение>;

Оператор присваивания заменяет значение переменной, идентификатор которой стоит в левой части, на значение, задаваемое выражением в правой части. Выражение строится из операндов (переменных и констант), функций, операций и круглых скобок.

Арифметические выражения используют арифметические операции: *, /, DIV, MOD, AND, OR, +, - . Операнды имеют тип REAL или INTEGER.

Пример арифметического выражения:

X := (1-B)*EXP(-0.5*A)/(1-A).

Список встроенных арифметических функций, наиболее часто используемых в программах на Паскале, приведен в табл. 12. Последовательность вы-

полнения операций в выражении определяется их приоритетом. В первую очередь делаются операции умножения и деления (*, /), а в последнюю - сложения и вычитания (+, -) слева направо. Приоритетность арифметических операций указана в приложении 1, табл. 50.

В языке существуют ограничения на преобразование типов данных путем присваивания. Переменной A типа REAL можно присвоить значение переменной B типа INTEGER ==> A := B. Однако обратное присвоение B := A вызовет прерывание по причине несоответствия типов. Для этого случая предусмотрены функции преобразования типов TRUNC(A) или ROUND(A), то есть используется присвоение вида B := TRUNC(A), или B := ROUND(A).

Примеры линейных программ

Пример 3. Рассчитать площадь шара в кв. см. Радиус шара ввести с клавиатуры в миллиметрах.

```
PROGRAM PR3;     { Программа вычисляет площадь поверхности шара }
VAR  PL: REAL;   { PL - площадь шара }
     R: INTEGER; { R  - радиус }
BEGIN
     WRITELN('введите радиус шара, мм'); READLN(R);
     PL := 4*PI*SQR(R)/100;
     WRITELN('Площадь шара =', PL:8:1, 'кв. см')
END.
```

Пример 4. Осуществить расчеты по формуле (2-1):

$$Y = arctg\frac{0.5\sqrt{a^2+b^2}\cdot\sin(\alpha+\varphi+15^\circ)+\sqrt{c^2+d^2}\cdot\sin(\beta+\psi+75^\circ)}{\sqrt{a^2+b^2}\cdot\cos(\alpha+\varphi+15^\circ)+\sqrt{c^2+d^2}\cdot\cos(\beta+\psi+75^\circ)} \qquad (2\text{-}1)$$

где: $\varphi = arctg(b/a)$, $\psi = arctg(d/c)$, $c = n \cdot a$, $d = m \cdot b$.

Поскольку набор символов, используемых в идентификаторах переменных в программе (латиница), не включает традиционные для тригонометрии символы греческого алфавита α, β, φ, ψ, необходимо составить таблицу имен, которая установит соответствие между идентификаторами переменных и этими символами. В таблице имен фиксируются промежуточные (рабочие) переменные, упрощающие программирование исходной формулы.

Таблица имен

Математическая величина, выражение	Идентификатор переменной в программе
α	ALPHA
β	BETA
φ	FI
ψ	PSI
$\alpha+\varphi+15^\circ$	AF
$\beta+\psi+75^\circ$	BP
$\sqrt{a^2+b^2}$	SQAB
$\sqrt{c^2+d^2}$	SQCD

Программирование линейных вычислительных процессов очень похоже на вычисления по формулам, которые математик осуществляет на бумаге. Алгоритм таких вычислений, как правило, не составляется в виде блок-схем. Наиболее удобной формой представления такого алгоритма является формульно-словесный способ, при котором действия пронумерованы пунктами 1, 2, 3 и т.д. Каждое действие поясняется словами, формулами и расчетами.

Алгоритм решения этой задачи описан формульно-словесным способом:

1. Ввод с клавиатуры параметров A, B, M, N, ALPHA, BETA.
2. Вычисление аргументов тригонометрических функций (2-1) по формулам:

$$AF = \alpha + \varphi + \frac{15^\circ \cdot \pi}{180^\circ},\ BP = \beta + \psi + \frac{75^\circ \cdot \pi}{180^\circ},$$

где AF и BP промежуточные рабочие переменные, которые в исходной формуле встречаются по два раза - в числителе и знаменателе. Следует отметить, что аргументы встроенных функций Sin и Cos в Паскале должны задаваться в радианах. В исходной формуле подразумевается, что углы α, β, φ, ψ измеряются в радианах. Поэтому углы 15° и 75° подлежат пересчету в радианы, что и сделано в приведенных выше формулах для расчета AF и BP.

3. Последовательное вычисление величин C, D, FI, PSI по формулам:

$$C = n \cdot a,\ D = m \cdot b,\ FI = arctg(b/a),\ PSI = arctg(d/c).$$

4. Нахождение значений промежуточных переменных SQAB и SQCD по формулам:

$$SQAB = \sqrt{a^2 + b^2} \text{ и } SQCD = \sqrt{c^2 + d^2}.$$

5. Вычисление Y по упрощенной формуле за счет уже выполненных в предыдущих пунктах алгоритма расчетов.

$$Y = arctg\frac{0.5 \cdot SQAB \cdot \sin(AF) + SQCD \cdot \sin(BP)}{SQAB \cdot \cos(AF) + SQCD \cdot \cos(BP)}$$

6. Последним пунктом этого алгоритма является вывод найденного значения Y на экран монитора.

```
PROGRAM PR4;
VAR ALPHA, BETA, FI, PSI, SQAB, SQCD, AF, BP, A, B, C, D, N, M, Y: REAL;
BEGIN WRITELN('введите значения A, B, M, N'); READLN(A, B, M, N);
WRITELN('введите значения АЛЬФА, БЕТТА'); READLN(ALPHA, BETA);
   AF = ALPHA + FI + 15*PI/180; BP = BETA + PSI + 75*PI/180;
   C := N*A; D:=M*B; FI := ARCTAN(B/A); PSI :=ARCTAN(D/C);
   SQAB := SQRT(A*A + B*B); SQCD := SQRT(C*C + D*D);
   Y := ARCTAN((0.5*SQAB*SIN(AF)+SQCD*SIN(BP)) /
                (SQAB*COS(AF) + SQCD*COS(BP)));
  WRITELN('Y = ', Y:7:3)
END.
```

Следует выделить следующие типичные действия программиста при разработке программ такого класса (формализация линейного вычислительного процесса):

1. Формирование таблицы имен. На этом этапе подбираются латинские обозначения (идентификаторы) для отображения в программе математических величин, используемых в формулах. Для некоторых выражений, встречающихся два и

более раза, в формулах можно ввести свои идентификаторы (временные переменные). Эти величины рассчитываются один раз перед основной формулой (формулами), что упрощает исходные формулы и ускоряет расчеты.

2. Учитывая последовательный принцип выполнения операторов в программе - друг за другом по мере их написания, необходимо установить порядок расчета формул. Основное требование состоит в том, чтобы при расчете формулы все переменные и параметры были ранее вычислены или введены с клавиатуры. Если формулы можно упростить путем алгебраических преобразований, то это нужно сделать до начала программирования.

3. Все математические величины нужно разбить на две группы: константы и переменные. Константы следует определить в разделе Const программы, а переменные - в разделе Var.

4. Проанализировав возможные значения переменных и требуемую точность расчетов, следует определить тип каждой переменной.

5. Требуется проанализировать все переменные из раздела VAR и определить значения, какие из них вводятся с клавиатуры, а какие вычисляются по ходу программы.

6. Если в тригонометрических функциях в качестве аргументов используются величины в градусах, то необходимо в программе сделать преобразование этих величин в радианы.

7. При выводе результатов расчетов на экран нужно выбрать формат, способ представления результатов (с плавающей или с фиксированной точкой) и задать точность (число значащих чисел).

Пример 5. Осуществить расчеты по формуле (2-2):

$$Y = \sqrt[n]{\left| X^{n+1} + Log_n \left| X + 1 \right| \right|} . \qquad (2\text{-}2)$$

Для решения этой задачи следует использовать известные математические преобразования, которые приведут исходную формулу (2-2) к виду, удобному для программирования. Эти преобразования описанные в табл. 21.

Таблица 21

Исходная формула	Формула для программирования	Текст программы
$\sqrt[n]{Z}$, $Z > 0$	$e^{LnZ/n}$	EXP(LN(Z)/N)
Z^{n+1}, $Z > 0$	$e^{(n+1)\cdot LnZ}$	EXP((N+1)*LN(Z))
$Log_N \left| X + 1 \right|$	$\frac{Ln\left| X + 1 \right|}{Ln\,N}$	LN(ABS(X+1))/LN(N)

```
PROGRAM PR5;
VAR          X, Y: REAL; N: INTEGER;
BEGIN WRITELN('введите значения X, N'); READLN(X, N);
      Y = EXP(LN(ABS(EXP((N+1)*LN(X)) + LN(ABS(X+1))/LN(N)))/N);
      WRITELN('Y = ', Y:8:4)
END.
```

Пример 6. Осуществить расчеты по формуле (2-3):

$$Y = 0.014 \cdot X^4 + 0.5 \cdot X^3 + 1.64 \cdot X^2 + 10 \cdot X + Ln\sqrt[3]{15 \cdot X + 12.65} . \qquad (2\text{-}3)$$

При вычислении подобных формул желательно сделать алгебраические преобразования, сводя операции возведения в степень к операциям умножения и деления. Такие преобразования, во-первых, существенно уменьшают время вы-

полнения программы, а во-вторых, уменьшают погрешность расчетов.

После тривиальных преобразований получится формула (2-4):

$$Y = X \cdot (X \cdot (X \cdot (0.014 \cdot X + 0.5) + 1.64) + 10) + \frac{Ln(15 \cdot X + 12.65)}{3}. \qquad (2\text{-}4)$$

```
PROGRAM PR6;
VAR          X, Y: REAL;
BEGIN
WRITELN('введите значения X'); READLN(X);
Y = X*(X*(X*(0.014*X + 0.5) + 1.64) + 10) + LN(15*X + 12.65)/3;
WRITELN('Y = ', Y:8:4)
END.
```

2.2. Разветвляющийся вычислительный процесс

Если вычислительный процесс зависит от определенных условий и реализуется по одному из нескольких заранее предусмотренных направлений, он называется разветвляющимся вычислительным процессом, а каждое из этих направлений - ветвью вычислений. Для выбора ветви вычислений в Паскале используются операторы IF и CASE. Однако прежде, чем перейти к рассмотрению этих операторов, необходимо познакомиться с понятиями составного оператора, логическими операциями и выражениями.

Составной оператор

Составной оператор предписывает выполнение составляющих его операторов в порядке их написания. Зарезервированные слова BEGIN и END являются операторными скобками. Формат оператора:

```
BEGIN                { Начало составного оператора}
        <Оператор 1;>
        <Оператор 2;>
          . . . .
        <Оператор n >
END;                          { Конец составного оператора }
```

Составной оператор используется в тех конструкциях, где по синтаксису языка должен быть только один оператор, а для решения задачи требуется более одного. В составном операторе все операторы 1, 2, ... , n выполняются последовательно друг за другом.

Логические выражения

Одним из нечисловых видов данных является тип BOOLEAN. Булевы (логические) переменные имеют только два значения FALSE (ложь), TRUE (истина). Существует несколько форм конструирования логического выражения:

- константа, описанная в разделе CONST;
- переменная, которой можно присвоить булевы значения, например, (FLAG := TRUE);
- отношение между переменными скалярных и некоторых структурированных типов.

В Паскале допускаются отношения, перечисленные в табл. 22.

Операции отношений Таблица 22

Отношение	Содержание	Отношение	Содержание
=	равно	<>	не равно
>	больше, чем	>=	больше, чем ..., или равно
<	меньше, чем	<=	меньше, чем ..., или равно

Пример 7. Пусть заданы вещественные переменные A, B и логическая переменная FLAG. Требуется построить примеры простых логических выражений, содержащих отношения между A и B.

Если: VAR FLAG, FLAG1, FLAG2: BOOLEAN; A, B: REAL;

тогда допустимы выражения вида:

FLAG := A <= B; Значение TRUE "истина" присваивается переменной FLAG, если A меньше или равно B.

FLAG1 := A <> B; Значение TRUE "истина" присваивается переменной FLAG1, если A не равно B.

FLAG2 := A = B; Значение TRUE "истина" присваивается переменной FLAG2, если A равно B.

Помимо указанных выше отношений (см. табл. 22), логические выражения конструируются с помощью булевых операций, описанных в табл. 23.

Логические операции Таблица 23

Математическое обозначение	НАЗВАНИЕ	Обозначение в Паскале
¬	Логическое отрицание, НЕ	NOT
∧	Логическая конъюнкция, И	AND
∨	Логическая дизъюнкция, ИЛИ	OR
⊕	Исключающее ИЛИ	XOR

Пример 8. Сформулировать логическое условие попадания точки с координатами (x, y) в область S (см. рис. 2).

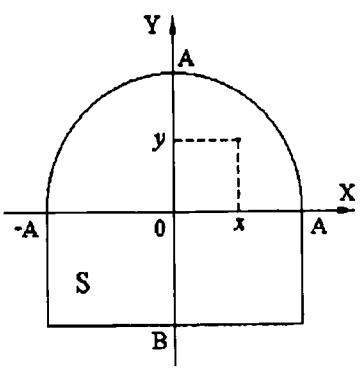


Рис. 2. Область S

Пусть: VAR FLAG: BOOLEAN;

Уравнение окружности, которая ограничивает область S в первом и втором квадранте системы координат XOY имеет вид:

$$Y = \sqrt{A^2 - X^2}\ .$$

Тогда величину FLAG, которая принимает значение TRUE, в том случае, когда точка с координатами (x, y) принадлежит области S, можно найти по формуле:

```
FLAG := (X>=-A) AND (X<=A) AND (((Y<=SQRT(A*A-X*X)) AND (Y>=0)) OR
((Y<0) AND (Y>=B)));
```

В языке Паскаль логическое выражение просчитывается до тех пор, пока результат не становится очевидным. После чего вычисления прекращаются. Так в данном случае используется конъюнкция трех условий: X больше A, X меньше A, и ограничение на значение Y. Достаточно любой логической величине принять значение FALSE, и остальные величины, стоящие правее в логическом выражении, уже не просчитываются, так как переменная FLAG независимо от значений оставшихся отношений будет равна FALSE. В настоящем случае это удобно! Потому, что уравнение окружности определено для значений X, удовлетворяющих условию $-A \le X \le A$. Именно это условие и проверяется в двух левых отношениях, поэтому используемое логическое выражение Y<=SQRT(A*A-X*X) для расчета переменной FLAG корректно для любых значений X.

```
PROGRAM PR8;
VAR    A, B, X, Y: REAL; FLAG: BOOLEAN;
BEGIN
WRITELN('Введите параметры A и B'); READLN(A, B);
WRITELN('Введите координаты X и Y'); READLN(X, Y);
FLAG := (X>=-A) AND (X<=A) AND (((Y<=SQRT(A*A - X*X)) AND (Y>=0)) OR
((Y<0) AND (Y>=B)));
IF FLAG
THEN  WRITELN('Точка в области S')
ELSE  WRITELN('Точка вне области S')
END.
```

В стандартном Паскале предусмотрен порядок старшинства операций в булевых выражениях: высший - (скобки); NOT; AND; (OR, XOR); (>, =, >=, <=, <>) - низший. Однако в различных версиях языка эти требования могут и не соблюдаться, поэтому надежнее использовать скобки для уточнения последовательности вычислений.

Не следует путать логические операции AND, OR и XOR с одноименными арифметическими операциями, которые приведены в приложении 1, табл. 49.

Существуют встроенные булевы функции, наиболее известные из которых - ODD(X), EOF(F), EOLN(F), описание которых приведено в табл. 14.

Логическое выражение может быть достаточно сложным и включать в себя арифметические и логические функции, например:

```
FLAG := ODD( I*3+K ) AND (( SQR(C) > SIN(D/2) ) OR ( A = 5 ));
```

Переменная FLAG принимает значение TRUE, если целочисленное выражение I*3 + K принимает нечетное значение, и квадрат C больше, чем синус D, деленной пополам, или A равно 5. В противном случае FLAG принимает значение FALSE.

В приведенных примерах в правой части оператора присваивания расположено логическое выражение, а в левой части - логическая переменная.

Организация условного перехода. Оператор IF

Для программирования разветвляющихся процессов, содержащих две ветви, используется оператор IF условного перехода (ветвления), имеющий две конструкции. Сокращенная форма:

```
IF              < Условие >    {Алгоритм приведен:}
```

THEN < Оператор >; {блок-схема на рис. 3.б.}
{структурограмма на рис. 4.б.}

В качестве условия можно использовать переменную или выражение логического типа. В этой конструкции Оператор выполняется в том случае, когда логическое выражение принимает значение TRUE. Вторая ветвь процесса, содержащая пустой оператор и в явном виде не описываемая, соответствует значению условия FALSE.

Другая, полная форма этого оператора имеет синтаксическую структуру:

IF < Условие > {Алгоритм приведен:}
THEN < Оператор 1> {блок-схема на рисунке 3.а.}
ELSE < Оператор 2>; {структурограмма на рис. 4.б.}

Здесь обе ветви процесса значимы. При значении логического выражения (переменной) TRUE выполняется Оператор 1, который всегда располагается за ключевым словом THEN, при FALSE - выполняется Оператор 2 (ветвь ELSE). Если в одну из ветвей следует включить более одного оператора, то их следует объединить составным оператором BEGIN ... END.

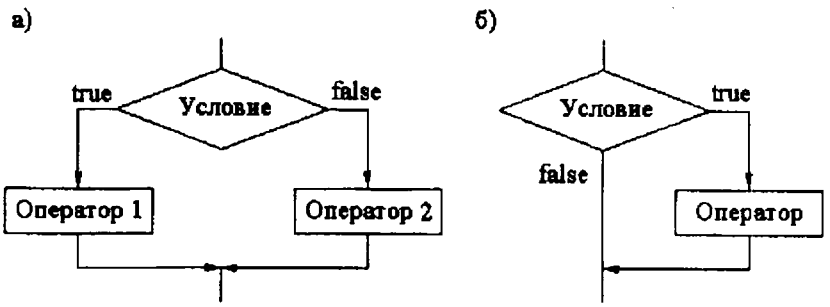


Рис. 3. Блок-схема оператора IF: а) полная форма, б) сокращенная форма

Вложенная конструкция операторов условного перехода используется в случае, если существует более двух ветвей в вычислительном процессе. В случае вложенной конструкции Оператор 1, или Оператор 2, или оба вместе представляют собой также операторы условного перехода. В этих случаях легко запутаться при отладке программы. Но следует помнить, что ELSE всегда относится к ближайшему слева THEN.

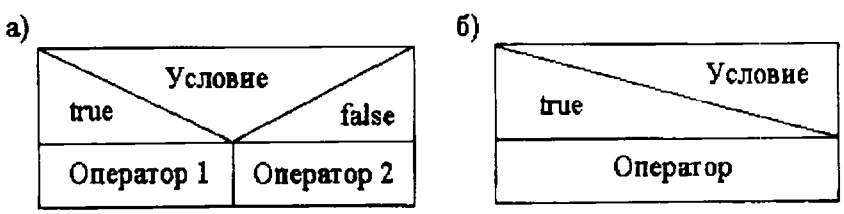


Рис. 4. Структурограмма оператора IF: а) полная форма, б) сокращенная форма

Пример 9. Для заданного с клавиатуры значения X вычислить Y по формуле:

$$Y=\begin{cases}\frac{1}{2}\cdot\sqrt[3]{X}, & X\ge 1;\\ \frac{1}{3}\cdot X^2, & 0<X<1;\\ \frac{1}{4}\cdot\sqrt[4]{|X|}, & X\le 0.\end{cases} \qquad (2\text{-}5)$$

Эту задачу можно решить двумя способами. Первый способ предусматривает использование трех операторов IF сокращенной формы. Алгоритм решения достаточно прост (см. структурограмму на рис. 5.а.). Последовательно проверяется три взаимно исключающих друг друга условия, образующих полную группу событий. Для любого X только одно условие примет значение true, остальные два условия равны FALSE. Таким образом, оператор присвоения выполнится только один раз, и этот оператор будет соответствовать условию, имеющему значение TRUE. Программная реализация - PR9_1.

Второй способ предусматривает использование двух вложенных операторов IF полной формы. Алгоритм этого способа реализован в виде блок-схемы на рис. 5.б. Программная реализация - PR9_2.

Сравнительный анализ этих двух способов показывает, что первый - проще в понимании и легче в отладке. Второй способ более сложный, однако, более компактный и более быстрый.

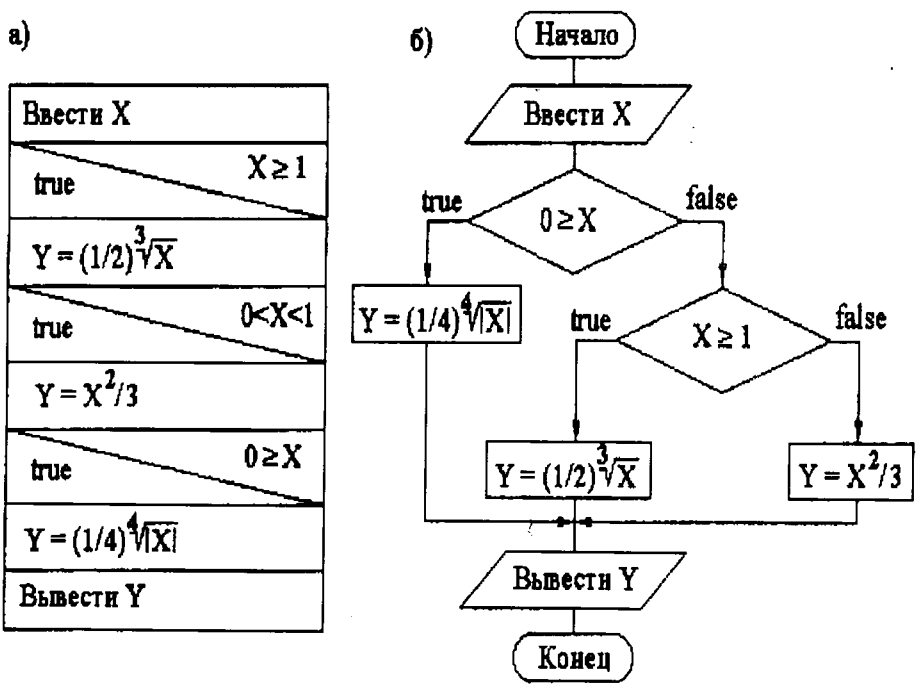


Рис. 5. Алгоритмы решения формулы (2-5):
а) структурограмма программы PR9_1, б) блок-схема программы PR9_2

```
PROGRAM PR9_1;
VAR  X, Y: REAL;
BEGIN
WRITELN('ВВЕДИТЕ X');
READLN(X);
IF X >= 1
THEN Y:= EXP(LN(X)/3)/2;
IF (X > 0) AND (X<1)
THEN Y:= X*X/3;
IF X <= 0
THEN Y:= EXP(LN(ABS(X))/4)/4;
WRITELN('Y=', Y:10:6)
END.
```

```
PROGRAM PR9_2;
VAR  X, Y: REAL;
BEGIN
WRITELN('ВВЕДИТЕ X');
READLN(X);
IF X <= 0
THEN Y:= EXP(LN(ABS(X))/4)/4
ELSE IF X >= 1
THENY:= EXP(LN(X)/3)/2
ELSE Y:=X*X/3;
WRITELN('Y=', Y)
END.
```

Конечно, при одной реализации вычислительного процесса это преимущество не столь существенно, но если таких реализаций десятки тысяч, то несколько секунд можно сэкономить.

Пример 10. Составить программу, которая по введенному значению X вычисляет и выводит значение Y = F(X), где F(X) задана графически на рис. 6.

Эта задача отличается от задачи, описанной в предыдущем примере, только тем, что нужно самому составить формулу для функции F(X). В предыдущей задаче эта формула (2-5) была задана в качестве исходных данных. Таким образом, требуется сделать формальную, математическую постановку задачи, которая в данном случае сводится к составлению по графику формулы аналогичной (2-5).

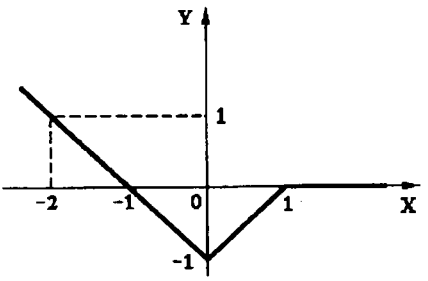


Рис. 6. График функции F(X)

Глядя на график (см. рис. 6) нетрудно увидеть, что на нем изображена кусочно-линейная функция, содержащая три прямых линии. Первая прямая имеет уравнение Y = –X–1 и определена для $X \le 0$. Вторая линия определена на отрезке $0 \le X \le 1$ и имеет уравнение Y = X – 1. Третья прямая линия имеет уравнение Y = 0 и определена для $X \ge 1$. С учетом выше сказанного, искомая формула будет иметь вид:

$$Y = \begin{cases} 0, & X \ge 1; \\ X - 1, & 0 < X < 1; \\ -X - 1, & X \le 0. \end{cases} \qquad (2\text{-}6)$$

Текст программы составляется по аналогии с программой PR9_2, которая подробно описана в предыдущем примере.

```
PROGRAM PR10;
VAR  X, Y: REAL;
BEGIN
WRITELN('ВВЕДИТЕ X'); READLN(X);
IF X <= 0
THEN Y:= –X–1
ELSE IF X >= 1
        THEN Y:= 0
        ELSE Y := X – 1;
WRITELN('Y=', Y)
END.
```

Пример 11. Найти расстояние от внешней, произвольной точки M с координатами (X, Y) до контура геометрической фигуры, изображенной на рис. 7.

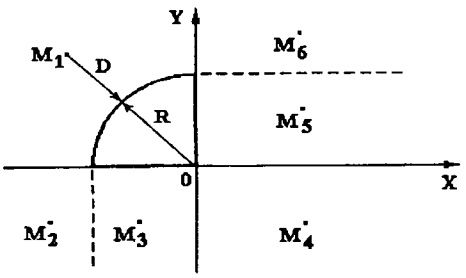


Рис. 7. Чертеж геометрической фигуры

В этом задании нужно рассмотреть чертеж геометрической фигуры. Нетрудно заметить, что во втором квадранте декартовой системы координат изображен сектор круга, центр которого расположен в центре системы координат. Радиус сектора равен R. Для решения поставленной задачи требуется выделить области и для каждой из них написать уравнение, с помощью которого можно вычислить расстояние D до контура фигуры. На чертеже выделено шесть областей, в каждой из которых поставлена одна произвольная точка M_K. Номер точки K соответствует номеру области. Любая точка в области M_1 имеет кратчайшее расстояние до дуги сектора, вычисляемое по формуле (2-7):

$$D = \sqrt{X^2 + Y^2} - R,\ X \le 0,\ Y \ge 0, \sqrt{X^2 + Y^2} \ge R\,. \qquad (2\text{-}7)$$

Вторая область, помеченная точкой M_2, имеет кратчайшее расстояние до точки с координатами (– R, 0). Это расстояние можно найти по формуле (2-8):

$$D = \sqrt{(X + R)^2 + Y^2},\quad X < -R,\ Y < 0\,. \qquad (2\text{-}8)$$

Третья область, помеченная точкой M_3, имеет кратчайшее расстояние равное расстоянию до оси абсцисс. Это расстояние легко найти по формуле (2-9):

$$D = |Y|,\quad 0 \ge X \ge -R,\ Y \le 0\,. \qquad (2\text{-}9)$$

Четвертая область, которой соответствует точка M_4, имеет кратчайшее расстояние, равное расстоянию до начала координат. Это расстояние находится по формуле (2-10):

$$D = \sqrt{X^2 + Y^2},\quad 0 \le X,\ Y \le 0\,. \qquad (2\text{-}10)$$

Пятая область, помеченная точкой M_5, имеет кратчайшее расстояние, равное расстоянию до оси ординат.

Это расстояние легко найти по формуле (2-11):

$$D = X, \quad 0 \le X,\ R \ge Y \ge 0. \tag{2-11}$$

Последняя, шестая область, помеченная точкой M_6, имеет кратчайшее расстояние до верхней точки сектора с координатами (0, R). Это расстояние можно найти по формуле (2-12):

$$D = \sqrt{X^2 + (Y-R)^2}, \quad X \ge 0,\ Y \ge R. \tag{2-12}$$

На основании вышесказанного, математическая постановка задачи будет такой: если точка находится вне сектора, изображенного на рис. 7, то будет истинно условие:

$$\overline{X < 0 \wedge\ Y > 0 \wedge \sqrt{X^2 + Y^2} < R}. \tag{2-13}$$

Учитывая (2-7, 2-8, 2-9, 2-10, 2-11, 2-12), можно найти кратчайшее расстояние до контура фигуры по формуле:

$$D = \begin{cases} \sqrt{X^2 + Y^2} - R, & X \le 0,\ Y \ge 0, \\ \sqrt{(X+R)^2 + Y^2}, & X < -R,\ Y < 0, \\ |Y|, & 0 \ge X \ge -R,\ Y \le 0, \\ \sqrt{X^2 + Y^2}, & 0 \le X,\ Y \le 0, \\ X, & 0 \le X,\ R \ge Y \ge 0, \\ \sqrt{X^2 + (Y-R)^2}, & X \ge 0,\ Y \ge R. \end{cases} \tag{2-14}$$

Теперь, когда формальная постановка задачи сделана, и найдены формулы (2-13, 2-14), можно перейти к этапу алгоритмизации.

Структурограмма

Ввести значение X с клавиатуры	
$X < 0 \wedge\ Y > 0 \wedge \sqrt{X^2 + Y^2} < R$	
True	False
Вывод на монитор сообщения: "Точка M внутри контура геометрической фигуры"	$X \le 0 \wedge\ Y \ge 0$
	True
	$D = \sqrt{X^2 + Y^2} - R$ {Область M_1}
	$X < -R \wedge\ Y < 0$
	True
	$D = \sqrt{(X+R)^2 + Y^2}$ {Область M_2}
	$0 \ge X \ge -R \wedge\ Y \le 0$
	True
	$D = \lvert Y \rvert$ {Область M_3}
	$0 \le X \wedge\ Y \le 0$
	True
	$D = \sqrt{X^2 + Y^2}$ {Область M_4}
	$0 \le X \wedge\ R \ge Y \ge 0$
	True
	$D = X$ {Область M_5}
	$X \ge 0 \wedge Y \ge R$
	True
	$D = \sqrt{X^2 + (Y-R)^2}$ {Область M_6}
	Вывод монитор значения D

```
PROGRAM PR11;
VAR X, Y, R, D: REAL;
BEGIN
WRITELN('ВВЕДИТЕ РАДИУС СЕКТОРА R'); READLN(R);
WRITELN('ВВЕДИТЕ КООРДИНАТЫ ТОЧКИ X, Y'); READLN(X, Y);
IF (X<0) AND (Y>0) AND (SQRT(X*X + Y*Y)<R)
THEN WRITELN('ТОЧКА РАСПОЛОЖЕНА ВНУТРИ СЕКТОРА')
ELSE BEGIN
   IF  (X<=0) AND (Y>=0)
   THEN D:= SQRT(X*X + Y*Y)-R;
   IF  (X<=-R) AND (Y<0)
   THEN D:= SQRT(SQR(X + R)+Y*Y);
   IF  (X<=0) AND (X>=-R) AND (Y<=0)
   THEN D:= ABS(Y);
   IF  (X>=0) AND (Y<=0)
   THEN D:= SQRT(X*X + Y*Y);
   IF  (X>=0) AND (Y<=R) AND (Y>=0)
   THEN D:= X;
   IF  (X>=0) AND (Y>=R)
   THEN D:= SQRT(X*X + SQR(Y-R))
   END;
WRITELN('РАССТОЯНИЕ ДО КОНТУРА СЕКТОРА D =', D:5:2)
END.
```

Оператор варианта CASE

Иногда его называют также оператором выбора. Это оператор CASE, который является обобщением оператора IF, и позволяет сделать выбор из произвольного числа имеющихся вариантов. Формат оператора варианта:

```
CASE        <Выражение-селектор> OF
            < Список 1>: <Оператор 1  >;
            <Список 2>: <Оператор 2  >;
                        . . .
            <Список N>: <Оператор N  >
            [ ELSE <Оператор N+1> ]
END;
```

В этой конструкции "Выражение селектор" может быть любого перечисляемого типа: стандартного (INTEGER, BOOLEAN, CHAR и т.д.) или пользовательского. "Список i" представляет собой подмножество значений, разделенных через запятую, выражения-селектора, при которых следует выполнить оператор i. Оператор ELSE может отсутствовать. Если выражение-селектор принимает значение, которое не входит ни в один из списков 1, 2, ..., N, то в этом случае выполняется оператор N+1, стоящий за ELSE. Когда оператор ELSE отсутствует, вместо оператора N+1 выполняется пустой оператор. Ниже приведен алгоритм оператора CASE ... OF в форме структурограммы.

Структурограмма

Анализ значения выражения-селектора				
Список 1	Список 2	...	Список N	В противном случае
Оператор 1	Оператор 2		Оператор N	Оператор N + 1

Вариант, помеченный словами "В противном случае", соответствует ветке ELSE оператора CASE. Если ELSE отсутствует, то в структурограмме нет этой колонки. Оператор CASE ... OF работает следующим образом:

- Вычисляется значение выражения селектора G.
- Значение G сравнивается с множеством значений, представленных в списке 1. Если в списке есть такое значение, то выполняется оператор 1.
- Если в списке 1 нет значения G, то проверяется список 2. Если найдено в списке 2 значение G, то выполняется оператор 2 и т.д.
- Если значение G не найдено ни в одном из списков с номерами 1, 2, 3, ... , N, то выполняется оператор N + 1.
- Если в операторе CASE нет ветки ELSE и значение G не найдено ни в одном из списков 1, 2, 3, ... , N, то выполняется пустой оператор.

Пример 12. Составить программу для вычисления площади одной из пяти геометрических фигур (прямоугольника, треугольника, трапеции, круга и сектора).

```
PROGRAM PR12;
VAR    S, A, B, H, R, FI: REAL; N: INTEGER;
BEGIN WRITELN('1 - прямоугольник');
                  WRITELN('2 - треугольник');
                  WRITELN('3 - трапеция');
                  WRITELN('4 - круг');
                  WRITELN('5 - сектор');
                  READLN(N); S :=0;
                  CASE N OF
   1: BEGIN       WRITELN('Введите стороны - A, B');
                  READ(A, B); S:= A*B; END;
   2: BEGIN       WRITELN('Введите основание и высоту - A, H');
                  READ(A, H); S:=A*H/2; END;
   3: BEGIN       WRITELN('Введите основания и высоту - A, B, H');
                  READ(A, B, H); S:= (A + B)*H/2; END;
   4: BEGIN       WRITELN('Введите радиус - R');
                  READ(R); S:= PI*R*R; END;
   5: BEGIN       WRITELN('Введите радиус и угол, град - R, FI');
                  READ(R, FI); S:=PI*R*R*FI/360; END
     END; { CASE }
        WRITELN('Площадь фигуры = ', S:8:2)
END.
```

В этой программе оператор CASE используется для организации простейшего меню для выбора фигуры по предлагаемому номеру из списка номеров. Оператор указывает номер фигуры, и управление передается соответствующей ветви вычислительного процесса.

Пример 13. Для заданного целого положительного K и значения вещественного числа X вычислить Y = F(X) по формуле:

$$Y=\begin{cases} X^K+X+1, & K=2 \text{ или } K=3,\\ \dfrac{1}{|X+1|}, & K=4 \text{ или } K=5 \text{ или } K=6,\\ \sqrt{|X+K|}+\sqrt{|X-K|}, & K>7 \text{ или } K<2. \end{cases} \qquad (2\text{-}15)$$

Алгоритм решения этой задачи описан с помощью структурограммы:

Ввод с клавиатуры значений X и K.		
	Анализ значения K	
K = 2, 3	K = 4 .. 6	В противном случае
$Y = X^K + X + 1$	$Y = \dfrac{1}{\lvert X+1\rvert}$	$Y = \sqrt{\lvert X+K\rvert} + \sqrt{\lvert X-K\rvert}$
Вывести значение Y.		

```
PROGRAM PR13;
VAR             X, Y : REAL; K: INTEGER;
BEGIN WRITELN('Укажите X и K'); READLN(K);
            CASE K OF
        2, 3 :  Y := EXP(K *LN(X)) + X + 1;      { Список 1 }
        4 .. 6 : Y := 1/ABS(X+1);                { Список 2 }
        ELSE  Y:= SQRT(ABS(X +K))+SQRT(ABS(X–K))  { В противном случае }
        END; { CASE }
        WRITELN('Y = ', Y:5:1)
END.
```

В примерах 12 и 13 не следует путать метки - значения выражения селектора CASE с метками, используемыми оператором GOTO. Попытка войти в метку оператора CASE оператором безусловного перехода приведет к ошибке.

2.3. Программирование арифметических циклов

Цикл - это последовательность операторов, которая может выполняться более одного раза. В языке Паскаль разработано три механизма для конструирования циклов, использующие операторы FOR, WHILE, REPEAT.

Оператор безусловного перехода

Иногда его называют просто оператором перехода. Он имеет вид GOTO <Метка>, где метка - это идентификатор, описанный в разделе LABEL. Метка ставится перед оператором, на выполнение которого нужно передать управление и отделяется от него двоеточием. Использование оператора GOTO в Паскале сильно ограничено и не рекомендуется. Так, нельзя входить извне в такие сложные операторы как IF, CASE, FOR, WHILE, REPEAT, BEGIN ... END, процедуры, функции и другие, хотя транслятор может и не обнаружить ошибки. Переходы рекомендуется осуществлять внутри одного уровня или при выходе на более высокий уровень.

Областью действия метки является тот блок, где она описана, и переходы по этой метке возможны только в этом блоке.

С помощью операторов IF и GOTO можно организовать циклический процесс. Поскольку операторы в теле программы выполняются последовательно друг за другом, то необходимо один из операторов пометить меткой, а затем, с помощью оператора GOTO, повторно вернуться на выполнение помеченного оператора. Чтобы этот возврат был управляемым, то есть для избежания бесконечных циклов, необходим анализ определенных условий. Именно для этого и нужен условный оператор IF.

Пример 14. Вычислить $Y = \sum_{i=1}^{\infty} \frac{1}{(2 \cdot i - 1) \cdot (2 \cdot i + 1)}$. Вычисления закончить при выполнении условия $|Y - 0,5| < EPS$.

Алгоритм программы, в которой предполагается использование операторов безусловного перехода GOTO с помощью структурограмм, представить невозможно. Для графического отображения таких программ используют только блок-схемы. В блок-схеме оператору GOTO <метка> соответствует графический объект – стрелка. Эта стрелка передает управление блоку, который помечен в верхнем левом углу меткой. В нашем случае идентификатором метки является цифра 1.

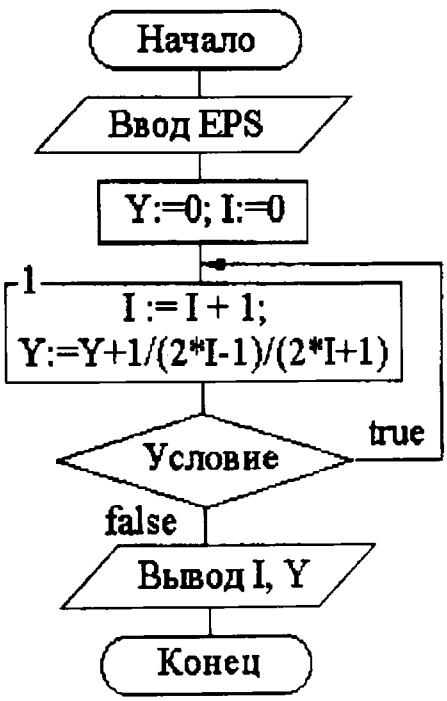


Рис. 8. Блок-схема алгоритма вычислений PR14

Циклический алгоритм программы приведен на рис. 8. На рис. 8 видно, что блок 1 повторяется до тех пор, пока условие имеет значение true. Как только условие станет равным false, искомая величина Y и номер текущей итерации I будут выведены на экран монитора.

```
PROGRAM PR14;
LABEL 1;
VAR  Y, EPS: REAL; I: INTEGER;
BEGIN
   WRITELN('ВВЕДИТЕ EPS');
   READLN(EPS);
   Y := 0; I := 0;
 1: I := I + 1;
   Y := Y + 1/(2*I - 1)/(2*I + 1);
   IF ABS(Y - 0.5) >= EPS
   THEN GOTO 1;
   WRITELN('I = ', I, ', Y = ', Y:6:4)
END.
```

Следует отметить, что организация циклов с помощью операторов GOTO не рекомендуется. Причина кроется в том, что именно использование этого оператора приводит к основной доле ошибок, которые допускает программист. Но эти ошибки, к тому же, самые трудные в диагностике и отладке программы. Именно для борьбы с этими ошибками использовали такую трудоемкую процедуру, как трассировка программы. Для борьбы с использованием оператора GOTO идеологами структурного программирования были предложены, так называемые, структурированные операторы циклов FOR, WHILE и REPEAT.

Циклы с параметром. Оператор FOR

Эти циклы организуются в программах, где заранее известно число повторений. При этом повторное выполнение сопровождается изменением управляющего параметра (переменной цикла).

FOR < Идентификатор параметра цикла > := < Выражение 1 >
{TO | DOWNTO} <Выражение 2> DO <Оператор>;

"Выражение 1" определяет первое значение параметра, а "Выражение 2" задает последнее значение параметра цикла. Параметр цикла, первое и последнее значения могут быть только одного любого перечисляемого типа (вещественный тип к перечисляемым типам не относится). "Выражение 1" и "Выражение 2" вычисляются один раз при вхождении в цикл и не могут изменяться внутри цикла. Служебное слово TO означает выбор следующего значения параметра. Для множества целых чисел выбор следующего значения тождественно увеличению параметра на единицу (см. функцию SUCC(s) в табл. 15). Служебное слово DOWNTO означает выбор предыдущего значения параметра, что для целых чисел адекватно сложению с -1 (см. функцию PRED(s) в табл. 15). Служебное слово DO распространяется только на один оператор (тело цикла), следующий за ним, поэтому в качестве этого оператора часто используется составной оператор. Для пояснения алгоритма работы оператора FOR (см. рис. 9), должны быть введеныследующие обозначения: I - идентификатор параметра цикла; M - значение выражения 1; N - значение выражения 2.

Оба варианта оператора цикла могут быть представлена с помощью одной и той же структурограммы, представленной на рис. 10. Циклы с параметром часто называют арифметическими циклами.

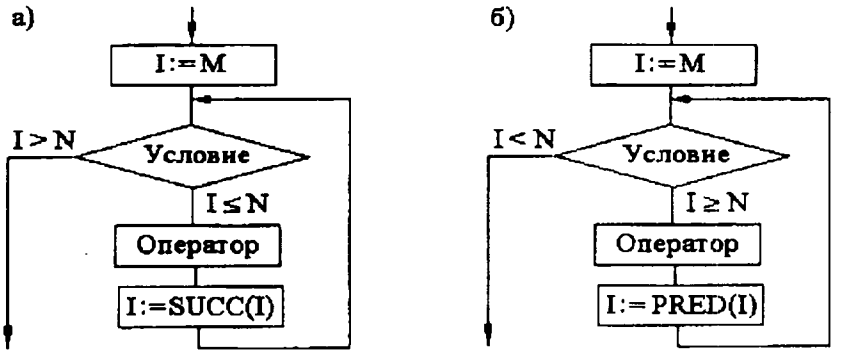


Рис. 9. Оператор цикла FOR: а) слово TO, б) слово DOWNTO

Для I от M до N с шагом ± 1 делать:
Оператор

Рис. 10. Блок цикла с параметром

Вычисление конечных сумм, произведений

Пример 15. Вычислить сумму $S = A + B \cdot \sum_{i=m}^{n} \frac{i^2}{n+i}$. Значения величин A, B, m и n ввести с клавиатуры.

Алгоритм решения этой задачи очевиден и наглядно описывается с помощью приведенной ниже структурограммы.

Ввести с клавиатуры A, B, M и N; S = 0;
Для I от M до N с шагом + 1 делать:
$S = S + \frac{I^2}{N+I}$
Вывести на экран значение A+B×S

```
PROGRAM PR15;
VAR  A, X, S: REAL; I, M, N: INTEGER;
BEGIN
WRITELN('ВВЕДИТЕ A, M И N'); READLN(A, M, N);
S := 0;
FOR  I := M TO N
DO S := S + I*I /(N + I);
WRITELN('S = ', A+B*S:6:4)
END.
```

Замечание. Большое значение имеет стиль написания программы. Для облегчения просмотра текста рекомендуется, по возможности, придерживаться следующих правил при составлении текста программы:

- Оператор END начинать с той же позиции, что и соответствующий ему опе-

ратор BEGIN. Это удобно в тех случаях, когда в программе используются вложенные составные операторы.

- Оператор DO писать под соответствующим оператором FOR.
- Операторы THEN и ELSE писать под соответствующим оператором IF.

Пример 16. Найти сумму S всех целых чисел, кратных 3 на отрезке [M, N].

Эта задача похоже на предыдущую. Отличие состоит в том, что, просматривая все числа из интервала [M, N], нужно проверить, делится ли число I без остатка на 3 прежде, чем его суммировать к S. Для проверки деления используется операция mod деление с остатком целых чисел (см. табл. 49 приложения 1). Таким образом, условие деления числа I на 3 без остатка будет иметь вид: I mod 3 = 0.

Алгоритм решения этой задачи отличен от предыдущего наличием условия в теле арифметического цикла и описан с помощью приведенной ниже структурограммы.

Ввести с клавиатуры M и N; S = 0;	
Для I от M до N с шагом + 1 делать:	
	I mod 3 = 0
	True
	S = S + I
Вывести на экран значение S	

```
PROGRAM PR16;
VAR  X, S: REAL; I, M, N: INTEGER;
BEGIN
WRITELN('ВВЕДИТЕ M И N'); READLN(M, N);
S := 0;
FOR  I := M TO N
DO       IF  I MOD 3 = 0
         THEN  S := S + I;
WRITELN('S = ', S:6:4)
END.
```

Пример 17. Для заданного значения N найти $Y = (-1)^N \cdot N!!$.

Для вычисления Y необходимо сделать следующие преобразование исходной формулы:

$$Y = \begin{cases} 2 \cdot 4 \cdot 6 \cdot \ldots \cdot (N-2) \cdot N, & \text{если N четное,} \\ -1 \cdot 3 \cdot 5 \cdot \ldots \cdot (N-2) \cdot N, & \text{если N нечетное.} \end{cases} \qquad (2\text{-}16)$$

```
PROGRAM  PR17;
VAR     I, N:  INTEGER; Y: REAL;
BEGIN
WRITELN('ВВЕДИТЕ N'); READLN(N);
Y := 1;
FOR     I:= 0 TO (N-1) DIV 2     { ВЫЧИСЛЕНИЕ N!! }
DO      IF ODD(N)
        THEN Y:= Y*(2*I+1)
        ELSE Y:= Y*2*(I+1);
IF  ODD(N)                       { УМНОЖЕНИЕ Y НА -1 В СТЕПЕНИ N }
```

```
THEN Y := -Y;
WRITELN ('Y=', Y: 10: 0)
END.
```

Табулирование функции

Задача табулирования функции предполагает получение таблицы значений функции при изменении аргумента с фиксированным шагом. В качестве исходной информации должны быть заданы: X_0, X_n - начало и конец промежутка табулирования, при этом ($X_0 < X_n$); n - число шагов разбиения промежутка [X_0, X_n]; F(X) - описание табулируемой функции.

При составлении алгоритма предполагается, что X - текущее значение аргумента; h - шаг изменения аргумента (иногда его называют шагом табуляции функции); i - текущий номер точки, в которой вычисляются функция (i = 0 .. n).

Количество интервалов n, шаг табуляции h и величины X_0, Xn связаны между собой формулой:

$$h = \frac{X_n - X_0}{n}. \qquad (2\text{-}17)$$

Интерпретация переменных (т. е. их обозначение в математической постановке задачи, смысл и тип, обозначения в структурограмме и программе) приведены в таблице имен.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
n	N	Число интервалов разбиения [X_0, Xn]	integer
X_0	X0	Начало промежутка	real
Xn	XN	Конец промежутка	real
X	X	Текущее значение аргумента	real
F(X)	Y	Текущее значение функции	real
h	H	Шаг табулирования	real

Пример 18. Табулировать функцию F(X) в **N** равноотстоящих точках, заданную на промежутке [X_0, **Xn**], где $F(X) = \sin(X+1)\cdot e^{-(x^2-2)}$.

```
PROGRAM PR18;
VAR   I, N: INTEGER; X, Y: REAL;
      H, X0, XN: REAL;
BEGIN
WRITELN('ВВЕДИТЕ X0, XN, N');
READLN(X0, XN, N);
H := (XN - X0)/N;
FOR I := 0 TO N
DO BEGIN
    X := X0 + I*H;
    Y:= SIN(X+1)*EXP(2-X*X);
```

Структурограмма

Ввод X0, XN, N
H := (XN-X0)/N
Для i от 0 до N с шагом 1 делать
X := X0 + i * h
$Y = \sin(x+1)\cdot e^{-(x^2-2)}$
Вывод X, Y

```
    WRITELN (X:4:1, ' ', Y:9:6)
    END
END.
```

В следующем примере задача табуляции функции выглядит весьма завуалированной. Поэтому в процессе решения этой задачи предстоит перед алгоритмизацией сделать формальную постановку.

Пример 19. Вычислить значения функции, $F(X) = X^2 \cdot \sqrt{|Sin(X)|}$ для всех X, где X = – 3.0, – 2.7, – 2.4, … , 3.0.

В этой задаче аргумент X функции F(X) меняется с постоянным шагом, h = 0.3. Известны начальное $X_0 = -3.0$, и конечное $X_N = 3.0$ значения диапазона изменения аргумента функции. Величину N можно найти из известной формулы:

$$N = \left\rceil \frac{X_n - X_0}{h} \right\lceil . \qquad (2\text{-}18)$$

Значения аргумента X вычисляется по формуле: $X = X_0 + i \cdot h$, i = 0, 1, 2, ..., N.

Легко заметить, что получены все необходимые формулы, и задача полностью повторяет предыдущую. Отличие только в табулируемой функции, а также в том, что значения H = 0.3; X0 = –3.0; XN = 3.0 нет необходимости вводить с клавиатуры, поскольку они уже известны из условия задачи.

```
PROGRAM PR19;
CONST H = 0.3; X0 = –3.0; XN = 3.0;
VAR          I, N: INTEGER; X, Y: REAL;
BEGIN
N:= TRUNC((XN – X0)/H);      {N - должно быть целым числом}
FOR I:=0 TO N
DO BEGIN
    X:= X0 + I*H;
    Y:= X * X * SQRT(ABS(SIN(X)));
    WRITELN (X:4:1, ' ', Y:9:6)
    END
END.
```

Арифметический цикл с рекуррентной зависимостью

Многие циклические вычислительные процессы используют рекуррентные зависимости при решении различных математических задач. В общем виде формулу для рекуррентных вычислений можно представить так:

$$Y_i = F(Y_{i-1}, Y_{i-2}, \ldots, Y_{i-k}). \qquad (2\text{-}19)$$

В этой рекуррентной формуле для вычисления i -го члена последовательности Y_i, где $i \geq k$, используются k предыдущих членов последовательности $Y_{i-1}, Y_{i-2}, \ldots, Y_{i-k}$. Для вычислений по формуле (2-19) нужно задать k первых членов последовательности - $Y_0, Y_1, \ldots, Y_{k-1}$.

Использование рекуррентных формул, как правило, сокращает текст программы и время выполнения на компьютере. Однако, в большинстве случаев рекуррентную формулу нужно написать программисту, что в ряде случаев вызывает определенные трудности.

Пример 20. Вычислить значение $tgX = \cfrac{X}{1-\cfrac{X^2}{3-\cfrac{X^2}{5-\cfrac{X^2}{7-\cfrac{X^2}{9-X^2}}}}}$.

Знаменатель формулы для вычисления tgX представляет собой цепную дробь. Для вычисления такого знаменателя в цикле удобно использовать рекуррентную зависимость с памятью в один член последовательности $A_i = F(A_{i-1})$. Для вывода рекуррентной формулы следует использовать табл. 24.

Таблица 24

Номер I	Член последовательности	Величина
0	A_0	1
1	A_1	$9-\frac{X^2}{A_0}$
2	A_2	$7-\frac{X^2}{A_1}$
3	A_3	$5-\frac{X^2}{A_2}$
4	A_4	$3-\frac{X^2}{A_3}$
5	A_5	$1-\frac{X^2}{A_4}$

Из табл. 24 легко заметить, что рекуррентная формула принимает вид:

$$A_i = 11 - 2 \cdot i - \frac{X^2}{A_{i-1}},\ A_0 = 1,\ i = 1, 2, 3, 4, 5. \qquad (2\text{-}20)$$

Структурограмма

Ввод с клавиатуры X; A:= 1;
Для I от 1 до 5 с шагом 1 делать
A := 11 2*I+1 - X *X/A;
Вывод на экран монитора значения tg(x) = X/A

```
PROGRAM PR20;
VAR    X, A: REAL; I: INTEGER;
BEGIN
       WRITELN('ВВЕДИТЕ X'); READLN(X);
       A := 1;
       FOR I := 1 TO 5 DO A :=11 - 2 * I - X * X/A;
       WRITELN('tg X = ', X/A:8:5)
END.
```

Пример 21. Пользуясь рекуррентной формулой, для заданного N вычислить $S_N = \sum_{i=0}^{N} Y_i$, если известны Y_0, Y_1, Y_2, а Y_i ($i \geq 3$) вычисляется по формуле

$$Y_i = Ln\left| Y_{i-1}^2 + Y_{i-3} + 1 \right|. \qquad (2\text{-}21)$$

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
N	N	Номер последнего члена последовательности S_N.	integer
Y_0, Y_{i-3}	Y3	Член последовательности с номером i-3.	real
Y_1, Y_{i-2}	Y2	Член последовательности с номером i-2.	real
Y_2, Y_{i-1}	Y1	Член последовательности с номером i-1.	real
Y_i	Y	Член последовательности с номером i.	real
S_N	S	Искомая сумма	real

Первым шагом в работе алгоритма является ввод данных Y_0, Y_1, Y_2, N. При вводе трех первых значений последовательности нужно использовать рабочие ячейки Y3, Y2 и Y1, соответственно. На втором шаге требуется проанализировать значение N. Если N < 3, то рекуррентная формула для подсчета S суммы первых N членов не потребуется. Для определения S при условии N < 3 в алгоритме предусмотрен переключатель (оператор CASE), имеющий три ветви:

N = 0, N = 1 и в противном случае, куда попадает и случай N = 2.

Для каждой ветви подсчитывается соответствующая сумма S. Третий шаг выполняется только в том случае, если N > 2. На этом шаге для I от 3 до N по рекуррентной формуле вычисляются Y_I и подсчитывается их сумма S. Найденное значение S на последнем, четвертом шаге выводится на экран.

Структурограмма

Ввод Y_0 в Y3, Y_1 в Y2, Y_2 в Y1, N;		
Анализ N		
N = 0	N = 1	В противном случае
S = Y3	S = Y3 + Y2	S = Y3 + Y2 + Y1
N > 2		
True		
Для I от 3 до N с шагом 1 делать:		
$Y = Ln\left\| Y_{i-1}^2 + Y_{i-3} + 1 \right\|$; S = S + Y;		
Y3 = Y2; Y2 = Y1; Y1 = Y;		
Вывод S.		

```
PROGRAM PR21;
VAR    Y3, Y2, Y1, Y, S: REAL; I, N: INTEGER;
BEGIN
WRITELN('ВВЕДИТЕ Y0, Y1, Y2, N'); READLN(Y3, Y2, Y1, N);
```

```
CASE N OF
   0:  S := Y3;
   1:  S := Y3 + Y2
   ELSE  S := Y3 + Y2 + Y1
   END;          { CASE }
   IF N > 2
   THEN  FOR I := 3 TO N
    DO  BEGIN
          Y := LN(ABS(Y1*Y1 + Y3 + 1)); S := S + Y;
          Y3 := Y2; Y2 := Y1; Y1 := Y
          END;
   WRITELN('S=', S:10:8)
END.
```

Рекуррентных формул может быть несколько. Ниже приведен пример с двумя зависимыми последовательностями. Рекуррентные формулы для вычисления членов этих последовательностей образуют систему.

Пример 22. Последовательность точек M_i, $i = 0, 1, 2, \ldots$ с координатами (X_i, Y_i) вычисляются по формулам:

$$\begin{cases} X_i = \sqrt{\dfrac{X_{i-1}}{2 \cdot (Y_{i-1} + 5)}}, \; X_0 = 3.5, \\ Y_i = \sqrt{X_{i-1} + 1.6}\;, \quad Y_0 = 2.2. \end{cases} \tag{2-22}$$

Составить программу и вычислить координаты X_{20} и Y_{20} точки M_{20}.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
i	I	Номер итерации.	integer
X_{i-1}	X1	Член послед. X с номером i-1.	real
Y_{i-1}	Y1	Член послед. Y с номером i-1.	real
X_i	X	Член последовательности X с номером i.	real
Y_i	Y	Член последовательности Y с номером i.	real

В условиях задачи все исходные данные определены, поэтому ввод с клавиатуры не требуется.

Структурограмма

X1 = 3.5, Y1 = 2.2;
Для I от 1 до 20 с шагом 1 делать:
$X = \sqrt{\dfrac{X1}{2 \cdot (Y1 + 5)}}, Y = \sqrt{X1 + 1.6}$
X1 = X; Y1 = Y;
Вывод X, Y.

```
PROGRAM PR22;
```

```
VAR     X1, Y1, X, Y: REAL; I: INTEGER;
BEGIN
X1 := 3.5; Y1 := 2.2;
FOR I := 1 TO 20
DO      BEGIN
        X := SQRT(X1/2/(Y1+5)); Y := SQRT(X1+1.6);
        X1 := X; Y1 := Y
        END;
WRITELN('X20 =', X:10:8, ', Y20 =', Y:10:8)
END.
```

Вложенный арифметический цикл

Под вложенным арифметическим циклом понимают такую алгоритмическую структуру, при которой в тело одного цикла с параметром включен другой цикл со своим параметром. Другими словами, составная инструкция:

```
FOR     I := ...
DO      FOR     J := ...
        DO      ...
```

является признаком вложенного арифметического цикла.

Пример 23. Вычислить $S = \sum_{k=1}^{10} k^3 \sum_{p=1}^{15} (k-p)^2$.

Структурограмма

S = 0;
Для k от 1 до 10 с шагом 1 делать:
R = 0;
Для p от 1 до 15 с шагом 1 делать:
$R = R + (k-p)^2$
$S = S + k^3 \cdot R$
Вывод S.

Для решения этой задачи необходима дополнительная переменная, как иногда говорят, рабочая ячейка R для накопления в процессе вычисления S вложенной суммы $R = \sum_{p=1}^{15} (k-p)^2$.

```
PROGRAM PR23;
VAR     R, S: REAL; K, P: INTEGER;
BEGIN
S := 0;
FOR     K := 1 TO 10
DO      BEGIN
        R := 0;
        FOR     P := 1 TO 15
        DO      R := R + SQR(K - P);
        S := S + K * K * K * R
        END;
WRITELN('S =', S:10:8)
END.
```

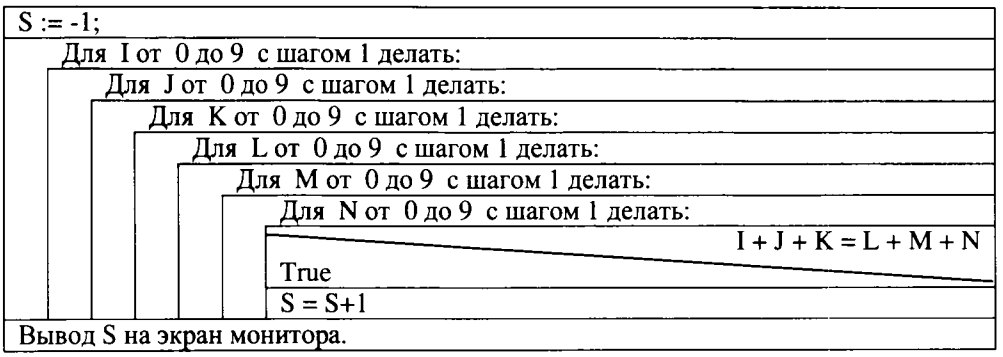
Количество уровней вложения арифметических циклов может быть более

трех десятков. В следующем примере используется алгоритмическая конструкция, имеющая шесть уровней вложения арифметических циклов.

Пример 24. Используя вложенный цикл, определить число счастливых билетов S, номера которых меняются от 000001 до 999999.

В основе алгоритма решения этой задачи лежит принцип десятичного счетчика, имеющего шесть разрядов. Роль разрядов играют индексы в следующем порядке I, J, K, L, M, N. Счастливым называется такой номер, у которого три левых разряда в сумме равны сумме трех правых разрядов, то есть I + J + K = L + M + N.

Структурограмма



В связи с тем, что номер 000000 в катушке билетов отсутствует, этот номер нужно вычесть из найденного числа. Это можно сделать разными способами. В предложенном алгоритме это реализовано так S := -1. Это решение логично. При исходном состоянии счетчика: I = 0, J = 0, K = 0, L = 0, M = 0, N = 0, условие I + J + K = L + M + N принимает значение True, и S становится равным нулю S = -1 + 1 = 0. Таким образом, все переменные приняли исходное значение для дальнейших расчетов.

```
PROGRAM PR24;
VAR    S: REAL; I, N, J, K, L, M: INTEGER;
BEGIN
S := -1;
FOR I := 0 TO 9 DO
        FOR    J := 0 TO 9 DO
                FOR    K := 0 TO 9 DO
                        FOR    L := 0 TO 9 DO
                                FOR    M := 0 TO 9 DO
                                        FOR N := 0 TO 9 DO
                                            IF I + J + K = L + M + N
                                                THEN S := S + 1;
WRITELN('Число счастливых билетов = ', S:6:0)
END.
```

Основным достоинством вложенного цикла является возможность в выражениях (в заголовке цикла или его теле) использовать параметры внешних циклов. Например, в описанном выше примере, в теле цикла с параметром N используются также текущие значения параметров I, J, K, L, M внешних циклов по отношению к этому циклу. К параметрам внутренних циклов из внешнего цикла не должно быть обращений. Транслятор с Паскаля такие обращения не проверяет, но для внешних циклов значения параметров внутренних циклов не определено. Допускается выход из тела цикла FOR … DO … любого уровня вложения на любой

предыдущий уровень до полного завершения цикла с помощью оператора GOTO, но не рекомендуется это делать. При выходе из цикла (досрочном или нормальном) значение параметра цикла становится неопределенным. Можно непосредственно из цикла завершить работу программы, Для этой цели используют оператор HALT [(Код)], где код - это необязательный параметр, представляющий собой целое число типа WORD, которое является кодом возврата EXE модуля.

2.4. Итерационные циклы

Если число повторений заранее не известно, и решение о завершении цикла принимается на основе анализа некоторого условия, то такой повторяющийся вычислительный процесс называется итерационным циклом.

В Паскале для организации итерационных циклов предусмотрено две алгоритмические структуры. Первая структура называется "цикл с предусловием" и использует оператор WHILE ... DO (см. рис. 11.а, 12.а). Вторая структура носит название "цикл с постусловием" и реализуется оператором REPEAT ... UNTIL (см. рис. 11.б, 12.б).

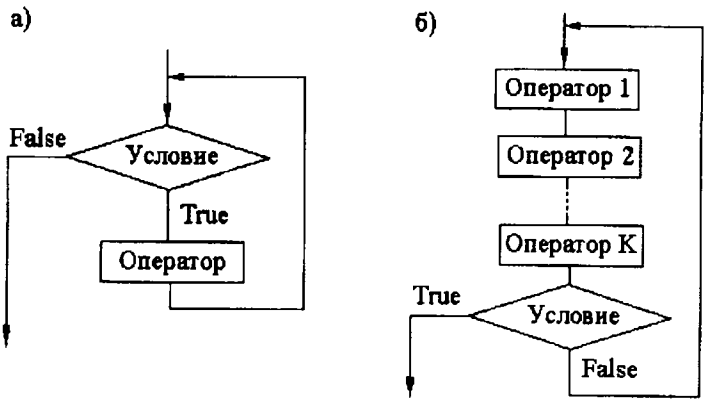


Рис. 11. Блок схема оператора: а) WHILE ... DO б) REPEAT ... UNTIL

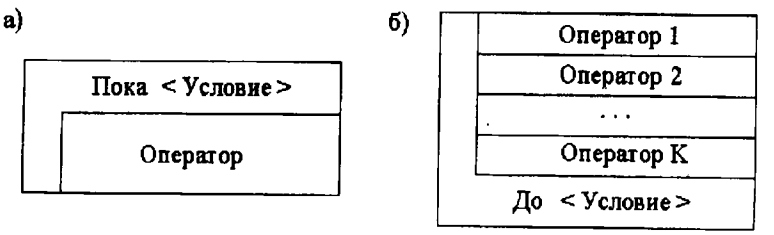


Рис. 12. Структурограмма оператора: а) WHILE ... DO б) REPEAT ... UNTIL

Цикл с предусловием. Оператор WHILE ... DO

Синтаксическая структура оператора цикла с предусловием имеет вид:

```
WHILE  < Логическое выражение, или переменная >
DO         < Оператор >;
```

Как видно из блок-схемы (рис. 11.а), перед каждым выполнением цикла

анализируется логическое выражение или переменная. При значении TRUE выполняется Оператор, составляющий тело цикла. При значении FALSE управление передается следующему за циклом оператору. Если условие ложно с самого начала, то оператор не выполняется ни разу.

Пример 25. Вычислить $Y = \sum_{i=1}^{\infty} \frac{i}{2^i}$. Вычисления остановить при выполнении условия $\frac{i}{2^i} < \varepsilon$.

Для решения этой задачи удобно использовать оператор цикла с предусловием WHILE ... DO.

Структурограмма

Ввод E; i = 1; Y = 0.5;
Пока $i \cdot 2^{-i} > E$ делать:
i = i + 1; Y := Y + $i \cdot 2^{-i}$;
Вывод Y.

```
PROGRAM PR25;
VAR     Y, E: REAL; I: INTEGER;
BEGIN
WRITELN('ВВЕДИТЕ E'); READLN(E);
I := 1; Y := 0.5;
WHILE I * EXP(-I * LN(2)) > E
DO      BEGIN
        I := I + 1;
        Y := Y + I * EXP(-I * LN(2));
        END;
WRITELN('Y =', Y:12:8)
END.
```

Цикл с постусловием. Оператор REPEAT ... UNTIL

Недостатком оператора WHILE является то, что в цикле можно выполнить только один оператор, поэтому приходится в большинстве случаев использовать операторные скобки BEGIN ... END. Этого недостатка лишен оператор цикла с постусловием (иногда его называют оператором "повтора" см. рис. 11.б, 12.б), имеющим следующий вид:

```
REPEAT
        <Оператор 1;>
        <Оператор 2;>
          ....
        <Оператор K >
UNTIL  <Условие>;
```

Ключевые слова REPEAT и UNTIL этого оператора играют роль операторных скобок BEGIN ... END. Поэтому эта конструкция тоже один оператор.

Пример 26. Вычислить с точностью ε квадратный корень из величины X

$Y \approx \sqrt{X}$.

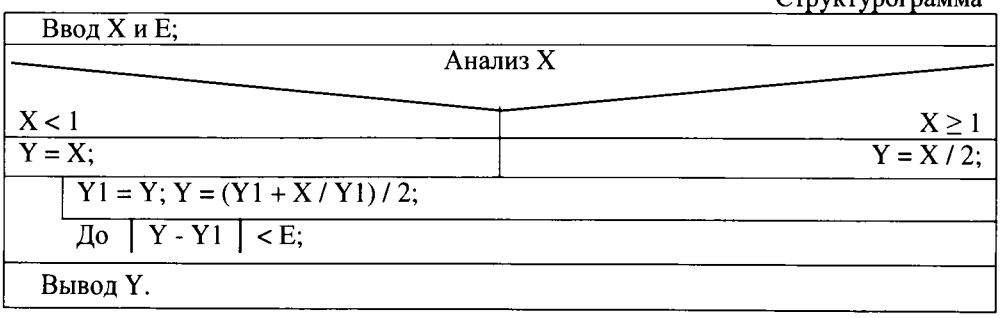
Вычисление проводить по рекуррентной формуле $Y_i = 0.5 \cdot (Y_{i-1} + \frac{X}{Y_{i-1}})$, выбрав в качестве начального приближения величину $Y_0 = \begin{cases} X, & X < 1, \\ 0.5 \cdot X, & X \geq 1. \end{cases}$

При решении подобных задач, условие остановки вычислительного процесса формулируется следующим образом: $|Y_i - Y_{i-1}| < \varepsilon$.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
i	I	Номер итерации.	integer
Y_{i-1}	Y1	Член послед. Y с номером i-1.	real
Y_i	Y	Член последовательности Y с номером i.	real
X	X	Величина X, квадратный корень которой мы ищем.	real
ε	E	Требуемая точность расчетов.	real

Структурограмма



С клавиатуры вводятся величины X и E. Далее вычисляется первое приближение Y. Если X < 1, то Y принимается равным X, в противном случае за Y принимается величина X/2. Далее, на основании Y нужно найти следующее приближение. Поэтому вычисленное значение записывается в ячейку с именем Y1 и с этого момента времени считается предыдущим значением. Текущее значение Y рассчитывается по рекуррентной формуле на основании Y1 и X. Этот циклический процесс повторяется до тех пор, пока не выполнится условие | Y - Y1 | < E. После чего Y считается равным значению корня из X с точностью E и выводится на экран монитора.

```
PROGRAM PR26;
VAR     X, Y, Y1, E: REAL;
BEGIN
WRITELN('ВВЕДИТЕ X, E'); READLN(X, E);
Y := X;
IF X >= 1
THEN Y := Y / 2;
REPEAT
```

```
        Y1 := Y;
        Y := (Y1 + X / Y1) / 2
UNTIL ABS(Y - Y1) < E;
WRITELN('Y =', Y:12:8)
END.
```

Вычисление предела последовательности

Вычисление предела последовательности является типичной задачей на использование итерационного цикла.

Пример 27. Последовательность $\{X_n\}$ определена следующим образом $X_n = \frac{n^2+2}{3 \cdot n^2 - n + 1}$, n = 1, 2, 3,... Найти предел последовательности $\{X_n\}$, принимая за него такое X_n при котором $|X_n - X_{n-1}| < \varepsilon$.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
n	N	Номер итерации.	integer
X_{n-1}	X1	Член послед. X с номером n-1.	real
X_n	X	Член последовательности X с номером n.	real
ε	E	Требуемая точность расчетов.	real

Структурограмма

Ввод E; N = 1; X = 1;
X1 = X; N = N + 1; X = (N * N + 2)/(3 * N * N - N + 1);
До | X - X 1 | < E;
Вывод X.

```
PROGRAM PR27;
VAR     X, X1, E: REAL; N: INTEGER;
BEGIN
WRITELN('ВВЕДИТЕ E'); READLN(E);
N := 1; X := 1;
REPEAT
        X1 := X; N := N + 1;
        X := (N * N + 2)/(3 * N * N - N + 1)
UNTIL ABS(X - X1) < E;
WRITELN('Предел последовательности равен ', X:12:8)
END.
```

Пример 28. Найти $\lim\limits_{n\to\infty} \frac{n}{\sqrt{n^2+3} + 2 \cdot \sqrt{n^2-1}}$ с точностью до ε.

Этот пример, несмотря на кажущееся отличие, является алгоритмической копией предыдущей задачи. Таблицы имен совпадают полностью. Структуро-

граммы и тексты программ тоже очень похожи. Отличие состоит только в формуле (2-23) для вычисления n-ого члена последовательности:

$$X_n = \frac{n}{\sqrt{n^2+3}+2\cdot\sqrt{n^2-1}},\ n = 1, 2, 3, \ldots \qquad (2\text{-}23)$$

Структурограмма

Ввод E; N = 1; X = 0.5;
X1 = X; N = N + 1; X = N /(SQRT(N * N +3) + SQRT(N * N - 1));
До | X - X 1 | < E;
Вывод X.

```
PROGRAM PR28;
VAR     X, X1, E: REAL; N: INTEGER;
BEGIN
WRITELN('ВВЕДИТЕ  E'); READLN(E);
N := 1; X := 0.5;
REPEAT
        X1 := X; N := N + 1;
        X := N /(SQRT( N * N +3) + SQRT(N * N - 1))
UNTIL ABS(X - X1) < E;
WRITELN('Предел последовательности равен ', X:12:8)
END.
```

Пример 29. Найти предел функции $\lim\limits_{Z\to 1}\dfrac{Sin\left|Z-1\right|}{Z-1}$ с точностью до ε.

Этот пример существенно отличается от предыдущих двух примеров, и в первую очередь тем, что у функции при Z = 1 существует два предела: левый предел L, равный -1, и правый предел R, равный 1. Во-вторых, вещественный аргумент функции Z стремится к конечному значению 1, в то время как в предыдущем примере целочисленное n, отражающее номер члена последовательности, стремилась к бесконечности. Однако алгоритм нахождения предела будет использован старый. Для этого нужно сделать преобразование исходной формулы путем замены переменных. Вместо исходного предела функции мы введем в рассмотрение два предела L и R:

$$Z = 1-\frac{1}{A^2};\ \Rightarrow\ L = \lim_{A\to\infty}\frac{Sin\left|1-\frac{1}{A^2}-1\right|}{1-\frac{1}{A^2}-1} = -\lim_{A\to\infty} A^2\cdot Sin\left(\frac{1}{A^2}\right); \qquad (2\text{-}24)$$

$$Z = 1+\frac{1}{A^2};\ \Rightarrow\ R = \lim_{A\to\infty}\frac{Sin\left|1+\frac{1}{A^2}-1\right|}{1+\frac{1}{A^2}-1} = \lim_{A\to\infty} A^2\cdot Sin\left(\frac{1}{A^2}\right); \qquad (2\text{-}25)$$

На втором шаге преобразований, вещественная величина A будет заменена на целочисленную переменную N. И, тем самым, осуществится переход от предела функции к пределу последовательности (2-26):

$$L = -\lim_{N\to\infty} N^2\cdot Sin\left(\frac{1}{N^2}\right);\quad R = \lim_{N\to\infty} N^2\cdot Sin\left(\frac{1}{N^2}\right). \qquad (2\text{-}26)$$

Из полученных формул (2-26) видно, что достаточно вычислить величину

правого предела R, а левый предел можно найти, положив L = - R. Чтобы окончательно свести рассматриваемую задачу к предыдущим, нужно ввести в рассмотрение последовательность $\{X_n\}$, которая определена следующим образом:

$$X_N = N^2 \cdot Sin\left(\frac{1}{N^2}\right), \quad N = 1, 2, 3, \ldots \qquad (2\text{-}27)$$

Предел этой последовательности будет численно равен $X_n \approx R$ с точностью ε, если $|X_n - X_{n-1}| < \varepsilon$.

После проделанных преобразований алгоритм задачи подобен предыдущим двум, таблица имен точно такая же. Приведем только структурограмму.

Структурограмма

Ввод E; N = 1; X = SIN(1);
X1 = X; N := N + 1; X = N * N * SIN(1 / N / N);
До | X - X 1 | < E;
Вывод "L=", X, "R=", - X.

```
PROGRAM PR29;
VAR     X, X1, E: REAL; N: INTEGER;
BEGIN
WRITELN('ВВЕДИТЕ E'); READLN(E);
N := 1; X := SIN(1);
REPEAT
        X1 := X; N := N + 1;
        X := N * N * SIN(1 / N / N)
UNTIL ABS(X - X1) < E;
WRITELN('Предел функции в точке Z = 1 равен:');
WRITELN('левосторонний: ', X:12:8, 'правосторонний: ', -X:12:8)
END.
```

Вычисление суммы бесконечного ряда с использованием рекуррентной формулы

Для вычисления на компьютере сумм бесконечного ряда часто используют рекуррентные формулы, с помощью которых друг за другом вычисляют значения членов бесконечной последовательности. Рекуррентные формулы существенно сокращают время работы программы, упрощают процесс написания программы и ее отладки. Как правило, рекуррентные формулы программист должен составить сам. В этом и состоит искусство программирования вычислительных процессов. Рекуррентная формула может и отсутствовать. В этом случае каждый член ряда придется рассчитывать "в лоб" по полной формуле.

Есть определенные признаки, которые помогают выявить наличие рекуррентных формул. К таким признакам относятся выражения $(-1)^n, X^n, n!$ и подобные этим выражения, присутствующие в формуле общего члена бесконечного ряда. Часто рекуррентная формула для бесконечного ряда находится путем деления соседних членов ряда друг на друга.

Пример 30. Вычислить $Y = \sum_{i=1}^{\infty} (-1)^i \cdot \frac{X^{2i}}{(2i)!}$. Вычисление ряда окончить при выполнении условия $\left|\frac{X^{2i}}{(2i)!}\right| < \varepsilon$.

Для решения этой задачи необходимо использовать рекуррентную формулу. А найти ее можно следующим способом. Необходимо сделать преобразование исходного ряда в следующий вид: $Y=\sum_{i=1}^{\infty}A_i$, $A_i=(-1)^i\cdot\frac{X^{2i}}{(2i)!}$. Тогда условие окончания вычислений будет выглядеть так $|A_i|<\varepsilon$. Это условие либо выполнится для некоторого i = n, и вычислительный процесс будет завершен, или не выполнится. Во втором случае используют термин "зависание программы". Оператор ЭВМ искусственно останавливает программу и выясняет причину зависания: неправильные исходные данные, например, комбинация X и ε, или допущена ошибка в тексте программы, а может быть получена неправильная рекуррентная формула, или другая причина имеет место.

В этом примере интерес представляет нормальный режим работы программы, а это означает, что существует такое n, для которого справедливы следующие формулы:

$$Y\approx\sum_{i=1}^{n}A_i,\quad A_i=(-1)^i\cdot\frac{X^{2i}}{(2i)!},\quad |A_i|<\varepsilon. \tag{2-28}$$

Эти формулы и будут исходными для этой задачи. На этом первый этап подготовки бесконечного ряда к нахождению его суммы Y с погрешностью ε на компьютере завершается. Если рекуррентную формулу найти невозможно или нет в этом необходимости, то можно ограничится только приведенными выше преобразованиями.

Но в данном случае нужен второй этап преобразования, а именно, нахождение рекуррентной формулы. Для этого необходимо поделить два соседних члена A_i, A_{i-1}.

$$\frac{A_i}{A_{i-1}}=\frac{(-1)^i\cdot\frac{X^{2i}}{(2i)!}}{(-1)^{i-1}\cdot\frac{X^{2\cdot(i-1)}}{(2\cdot(i-1))!}}=-\frac{X^2}{2\cdot i\cdot(2\cdot i-1)}. \tag{2-29}$$

Из (2-29) и находится рекуррентная формула:

$$A_i=-\frac{X^2}{2\cdot i\cdot(2\cdot i-1)}\cdot A_{i-1};\ i=2,3,\ldots,n;\ A_1=-\frac{X^2}{2}. \tag{2-30}$$

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
i	I	Номер итерации.	integer
X	X	Параметр бесконечного ряда.	real
Y	Y	Искомая сумма.	real
A_i	A	Член последовательности A с номером i.	real
ε	E	Требуемая точность расчетов.	real

Структурограмма

Ввод X, E с клавиатуры; i = 1; A = - $X^2/2$; Y = A;
Пока | A | ≥ *E* делать:
i = i + 1; A = -X*X/2/i/(2*i-1); Y := Y + A;
Вывод Y на экран монитора.

```
PROGRAM PR30;
VAR    Y, E, A, X: REAL; I: INTEGER;
BEGIN WRITELN('Введите X, E'); READLN(X, E);
      I := 1; A:= -X*X/2; Y := A;
      WHILE  ABS(A) >= E
      DO     BEGIN
             I:=I+1; A := -X*X / 2 / I / (2*I - 1)*A; Y := Y + A;
             END;
      WRITELN('Y = ', Y:10:6)
END.
```

Вложенные итерационные циклы

Под вложенным итерационным циклом понимают такую алгоритмическую структуру, при которой в тело одного итерационного цикла включен другой итерационный цикл. Другими словами, любая составная инструкция, из перечисленных ниже, является признаком вложенного итерационного цикла.

```
WHILE ...      | WHILE ...      | REPEAT ...     | REPEAT ...
DO    WHILE    | DO    REPEAT   |        REPEAT  |        WHILE
...            |         ...    |          ...   | ...
        DO ... |       UNTIL ...|        UNTIL ...|          DO ...
                                | UNTIL ...      |            ...
                                |                | UNTIL ...
```

Пример 31. Составить программу для нахождения числа слов в предложении и количества букв в самом длинном слове.

Задача решается в предположении: в конце предложения стоит точка, слова не имеют знака переноса и написаны на русском языке. В теле предложения могут стоять знаки препинания, но после каждого из них должен быть один пробел, который для предлагаемой программы является разделителем слов. Переменные, используемые в программе, и их тип описаны в табл. 25.

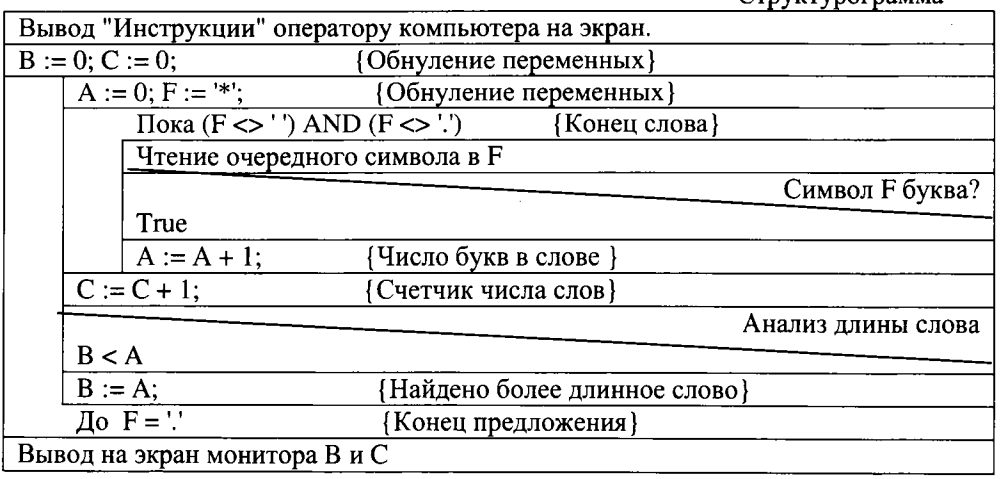
Таблица 25

Идентификатор	Содержательный смысл	Тип переменной
A	Число букв в новом слове	INTEGER
B	Число букв в самом длинном слове	INTEGER
C	Число слов	INTEGER
F	Последний символ, введенный с клавиатуры	CHAR

В программе используется три счетчика A, B, C и переменная F для чтения из входного файла INPUT текущего символа (литеры). Алгоритм содержит два итерационных цикла. Внешний цикл с постусловием используется для под-

счета числа слов в предложении С, а также для выявления самого длинного слова (формирование значения переменной В). Внутренний цикл обеспечивает чтение очередной литеры F, отделяет русские буква от знаков препинания и латинских букв, определяет конец слова и подсчитывает количество букв А в текущем слове. Концом слова считается пробел или точка.

Структурограмма




```
PROGRAM PR31;
VAR    A, B, C: INTEGER; F: CHAR;
BEGIN
WRITELN ('Введите слова, разделенные пробелами.');
WRITELN ('В конце предложения поставьте точку.');
B := 0; C := 0;
REPEAT
        A := 0; F := '*';
        WHILE (F <> ' ') AND (F <> '.')              { Конец слова }
        DO      BEGIN READ(F);
                IF (('A' <= F) AND (F <= 'Я')) OR ((('a' <= F) AND
                   (F <= п')) OR (('р' <= F) AND (F <= 'я')))
                THEN A := A + 1  END;
        C := C + 1;                                  { Новое слово }
        IF B < A THEN B := A
UNTIL F = '.';                                       { Конец предложения }
WRITELN ('Наибольшее число букв в слове =', B);
WRITELN ('Число слов в предложении = ', C)
END.
```

В этой программе оператор READ использован для ввода потока символов произвольной длины. Реализован логический вложенный цикл, в котором применяются как оператор постусловия REPEAT для анализа конца предложения, так и оператор предусловия WHILE для выделения слов в предложении.

Замечание. В версии Турбо-Паскаль 7.0 в циклах FOR, REPEAT, WHILE можно использовать процедуры BREAK и CONTINUE, подробно описанные в приложении 2.

3. Программирование данных

Любые данные, то есть константы, переменные, выражения, значения функций характеризуются своими типами. Тип определяет множество допустимых значений, которые может иметь программируемый объект, множество операций и функций, которые применимы к нему. Тип данных полностью определяет структуру внутреннего хранения информации в памяти компьютера. Большинство распространенных процедурно ориентированных языков программирования предоставляют пользователю только набор *стандартных* типов данных, то есть заранее *предопределенных*.

Паскаль - один из первых языков, разрешивших программисту определять, конструировать новые пользовательские типы данных. На рис. 13 приведена классификация типов данных, используемых в языке Турбо-Паскаль. Различают две большие группы данных *простые* типы данных и *структурные (сложные)*.

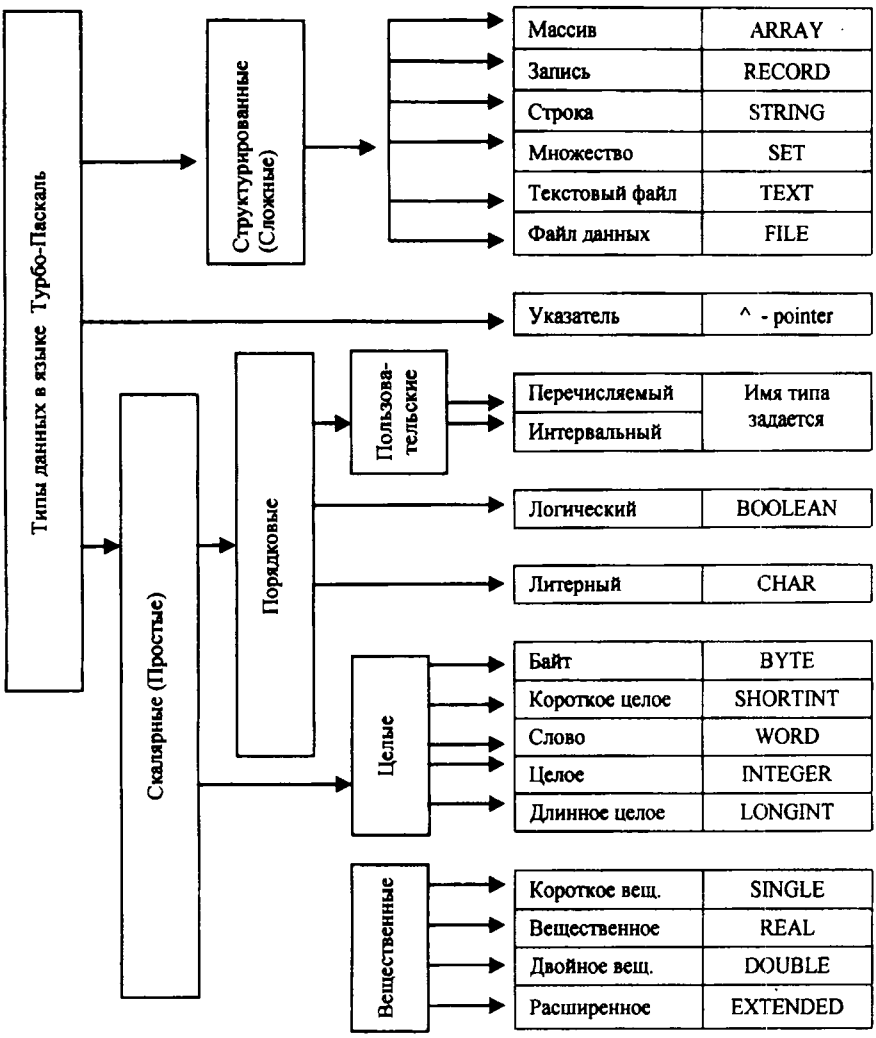


Рис. 13. Классификация типов данных

Простые типы являются основой *(базой)* для определения сложных типов. Типов данных в Турбо-Паскале очень много, поэтому рассмотрение будет производиться в соответствие с ниже приведенной классификацией. Если учесть тот факт, что Паскаль разрешает создавать такие структурированные типы как массив строк, массив записей, массив множеств, массив массивов или запись, элементами которой являются массивы, записи, множества и т.д., то общее количество определяемых программистом типов будет сколь угодно велико.

В предыдущей главе основной упор был сделан на классификацию вычислительных процессов и выработке навыков - определять по внешнему виду задачи требуемых операторов Паскаля, необходимых и достаточных для ее решения. Основной целью этой главы является выбор наиболее удачной комбинированной структуры данных для решения типовых задач, стоящих перед программистом, а так же описание основных приемов для работы с этими типами данных.

3.1. Конструирование простых пользовательских типов

Данные простого типа (константы, переменные, выражения) имеют только одно значение, поэтому этот тип часто называют *скалярным*. К стандартным скалярным типам относятся следующие наиболее известные и часто употребляемые типы: REAL, INTEGER, BOOLEAN, CHAR. Последние три типа содержат ограниченное число значений, поэтому их называют стандартными *перечисляемыми* типами. Вещественные типы данных, в том числе и REAL, к перечисляемым типам не относится. Пользователь на базе стандартных перечисляемых типов может создать свои пользовательские типы. Различают два простых пользовательских типа данных: пользовательские перечисляемые типы и интервальные (иногда говорят ограниченные) типы данных.

Пользовательские перечисляемые типы

Пользователь может конструировать новые скалярные перечисляемые типы, явно описываемые в разделе TYPE:

TYPE <имя типа> = (значение 1, значение 2, ... , значение N);

При этом понимают, что "значение i" - это идентификатор элемента с номером i.

Допускается неявное определение перечисляемого типа непосредственно в разделе VAR:

VAR <список переменных> : (значение 1, значение 2, ..., значение N);

Пример явного задания перечисляемых типов пользователя:

```
TYPE        GAZ = (C, O, N, F);
VAR         G1, G2: GAZ;
```

Данные указанных типов можно использовать в операторах FOR и CASE, в функциях SUCC, PRED и ORD. К сожалению, в стандартных процедурах READ, READLN, WRITE, WRITELN эти данные не поддерживаются. Наиболее часто данные пользовательских перечисляемых типов используются при конструировании сложных типов данных (индексы массивов, указателей и т.д.).

Пример 32. Для заданного года вычислить количество дней.

```
PROGRAM PR32;
TYPE  MONTH = (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT,
              NOV, DEC);     { Пользовательский перечисляемый тип данных }
VAR   S, YEAR: INTEGER; MEC: MONTH;
BEGIN
WRITELN ('ВВЕДИТЕ ГОД'); READ (YEAR); S := 0;
```

```
FOR MEC := JAN TO DEC        { Просмотр всех месяцев по порядку }
DO CASE MEC OF
    APR, JUN, SEP, NOV: S := S+30; { Список месяцев, содержащих 30 дней }
    FEB: IF (YEAR MOD 4 = 0) AND (YEAR MOD 100 <> 0)
            OR (YEAR MOD 400 = 0)
        THEN S := S+29                { Апрель, високосный год }
        ELSE { IF }  S := S+28;       { Апрель, не високосный год }
    ELSE       { CASE }  S := S+31
    END;       { CASE }
WRITELN ('ЧИСЛО ДНЕЙ: ', S)
END.
```

Перечисляемые типы характеризуются упорядоченностью значений (элементов) этого типа. Существует порядковый номер для каждого идентификатора. Первый в списке идентификатор имеет номер 0, второй - 1 и т.д. Исключение составляет тип INTEGER, где номер совпадает с числом. В пользовательском перечисляемом типе номера присваиваются в порядке перечисления идентификаторов при определении типа слева - направо, сверху - вниз. Оператор FOR осуществляет последовательный перебор значений порядковых элементов в соответствии с их упорядоченностью.

На перечисляемых типах данных (стандартных и пользовательских) определены встроенные функции SUCC, PRED, ORD (см. табл. 15).

Все идентификаторы перечисляемых типов упорядочены по номерам. Примеры:

JAN < FEB < MAR < ... < DEC, (тип пользователя - MONTH);
FALSE < TRUE, (тип - BOOLEAN);
-32768 < -32767 < ... < 32767, (тип - INTEGER);
'a' < 'b' < 'c' < ... < 'z', (тип - CHAR).

Допустимо использовать отношения <, >, =, <>, <=, >= для сравнения переменных, выражений или констант одного перечисляемого типа. Примеры:

'a' < 'b' - имеет значение TRUE;
MAR > DEC - имеет значение FALSE.

Интервальные типы

Для переменных порядкового типа можно создавать *интервальный (ограниченный)* тип, используя свойство упорядоченности его значений (элементов). Определение интервального типа:

TYPE <Имя типа> = <Левая граница> .. <Правая граница>;

Левая и правая границы представляют собой константы, или идентификаторы значений базового типа, и задают порядковые номера элементов, ограничивающие диапазон значений переменных создаваемого интервального типа.

Примеры задания интервальных типов:

```
TYPE
  DAY =  (MO, TU, WE, TH, FR, SA, SU);
  WORK_DAY = MO .. FR;            { Интервальный пользовательский тип }
  YEAR    = 1900 .. 2000;         { Интервальные
  LAT_ALFABETH  =  'A' .. 'Z';    { Стандартные типы }
```

Над переменными интервального типа допускаются все операции и функции, которые используются на базовом для него порядковом типе.

Особенности программирования скалярных типов пользователя

В Паскале существует два способа описания типов данных *явный* с использованием раздела TYPE и *неявный*, когда тип данного описывается непосредственно в разделе VAR. Примеры тождественного описания данных:

- Явное задание пользовательских типов.
  ```
  TYPE DAY    = (MO, TU, WE, TH, FR, SA, SU);
  YEAR   = 1900 .. 2000;
  NUMERIC = '0' .. '9';
  VAR  D1, D2 : DAY; G : YEAR; INTG : NUMERIC;
  ```
- Неявное задание тех же типов.
  ```
  VAR  D1, D2 : (MO, TU, WE, TH, FR, SA, SU);
  G    : 1900 .. 2000;
  INTG  : '0' .. '9';
  ```

Паскаль не поддерживает операции ввода-вывода значений пользовательских перечисляемых типов.

3.2. Массивы. Регулярные типы

В простых типах данных каждое данное имеет свое название (идентификатор). В этом разделе вводится структурная взаимосвязь между данными, хранимыми в оперативной памяти путем организации массива, состоящего из непрерывно расположенных данных, не снабженных отдельными именами. Эти данные, в свою очередь, могут быть простыми или сложными и называются элементами массива.

Основное преимущество массива состоит в том, что его элементы не имеют отдельных имен, и нет необходимости описывать каждый элемент по отдельности. Достаточно описать весь массив. Объявление массива содержит следующее описание:

TYPE <имя типа> = ARRAY [тип индекса] OF <тип элементов массива>;

Каждый элемент именуется путем указания имени массива и порядкового номера, определяющего его позицию в массиве, то есть индексом. Если тип данных определен с помощью конструкции ARRAY ... OF ..., то он называется *регулярным* типом.

Паскаль предоставляет пользователю широкие возможности по заданию типа индекса, которым может быть любой порядковый или интервальный типы данных, в том числе и определенные пользователем. Тип элемента массива иногда называется базовым. Он может быть как любым скалярным, так и структурированным типом данных. Правомерно существование массивов-массивов, записей, множеств. Однако не существует массивов файлов. Число компонентов массива неявно определяется через тип индекса при его объявлении и в дальнейшем не меняется.

Одномерные массивы. Вектора

Если в описании массива типом элемента является простой тип данных, то такой массив называется вектором. Поскольку в таком массиве для идентификации величины используется только один индекс, то он называется одномерным. Такие одномерные массивы представляют собой простейшие структурированные данные. Обращение к элементам одномерного массива осуществляется с помощью индексированных переменных, например X[i]. Здесь X - имя массива, а i -

константа, переменная или выражение того же типа, что и тип индекса в объявлении массива.

Пример 33. Определить частоту появления латинских прописных букв в тексте, вводимом с клавиатуры. Ввод данных завершить символом '*'.

В этом примере в качестве индекса удобно использовать ограниченный литерный тип 'A' .. 'Z', что обеспечивает, с одной стороны, мнемонический смысл индекса, соответствующего счетчику частоты появления литеры в тексте, а с другой стороны, легкость перебора значений индекса.

В Паскале нет средств ввода - вывода массива целиком, поэтому эти действия приходится выполнять как циклические процессы над отдельными элементами массива, используя (в частности, в данном примере) оператор FOR. В примере 33 при выводе результатов с помощью форматного вывода реализован перевод целочисленного выражения COUNTER [CH] * 100 / N в вещественную форму числа с фиксированной десятичной точкой.

```
PROGRAM PR33;
VAR    COUNTER: ARRAY ['A' .. 'Z'] OF INTEGER;
       CH: CHAR; N: INTEGER;
BEGIN { Инициализация массива счетчиков букв  COUNTER,
                  то есть - присвоение его элементам значения 0 }
FOR CH := 'A' TO 'Z' DO COUNTER[CH] := 0;   {Обнуление счетчиков литер}
N := 0;              { Обнуление счетчика числа символов в тексте}
REPEAT                    { Повторять для каждой новой литеры }
READ(CH);                 { Ввод очередного символа с клавиатуры }
N := N + 1;               { Увеличение счетчика символов на единицу }
IF (CH >= 'A') AND (CH <= 'Z')
THEN  COUNTER[CH] := COUNTER[CH] + 1;
       { Наращивается только элемент массива с индексом,
         соответствующим коду вводимого символа }
UNTIL CH = '*';           { Если True, то прочитана * - признак конца текста }
WRITELN('Всего прочитано символов: ', N);
WRITELN('буква:' :10, 'частота:' :10, 'процент:' :10);
FOR CH := 'A' TO 'Z'                { Вывод элементов массива на экран }
DO     WRITELN(CH:8, COUNTER[CH]:10, COUNTER[CH]*100/N:10:2)
END.
```

Инициализация одномерного массива

Отличительной особенностью Паскаля от большинства процедурных языков является то, что все переменные должны быть инициализированы. То есть в разделе VAR переменным отводится место, а начальное значение этих величин специально не устанавливается. Поэтому после объявления массива необходимо его элементам задать конкретные значения. Широко используется три способа инициализации одномерного массива:

- Если значения элементов массива определены до начала работы программы, то есть известны на этапе формулировки задания на программирование, то можно использовать следующий способ, характерный исключительно для Турбо-Паскаля:

```
CONST A: ARRAY [1..10] OF REAL = (0.1, -15.3, 7, 0, -11.89, 4,-78, 11.2, 1,0.01);
```

При таком объявлении массива в разделе констант, заносится в одномерный массив A по порядку A[1] = 0.1, A[2] = -15.3, ... A[10] = 0.01 вещественные

числа, перечисленные в круглых скобках. При этом массив является массивом переменных, то есть в процессе работы программы можно менять содержимое любого разряда одномерного массива. Этот способ, конечно, является нарушением по стандарту Паскаля, однако, очень часто используется на практике.

- Второй способ применяется в том случае, если исходные данные необходимо внести с клавиатуры в процессе выполнения программы. Поскольку одномерный массив представляет собой конечный набор однотипных элементов, пронумерованных с помощью индекса (переменной перечисляемого типа), то удобно использовать арифметический цикл (оператор FOR) для ввода значений непосредственно с клавиатуры. При этом можно предложить два равноценных приема. Предположим, в программе сделаны объявления:

```
CONST M = 1; N = 15;
VAR          A: ARRAY[M .. N] OF REAL;
```

где M - нижняя, а N верхняя границы индексов. Первый способ ввода будет иметь инструкцию:

```
WRITELN('Введите массив A, из 15 вещественных чисел');
FOR I := M TO N DO READ(A[I]);
```

При таком способе оператор может ввести все 15 чисел через пробел в одну строку и только затем нажать на клавишу Enter. Если он считает нужным, то он может вводить группы чисел (например, по 3 числа, чтобы не ошибиться) через пробелы, и нажимать Enter. А можно вводить на каждой строке только по одному числу.

Второй способ ввода имеет вид:

```
FOR   I := M TO N
DO    BEGIN
      WRITE('A[', I:1, '] = ');
      READLN(A[I])
      END;
```

Этот фрагмент программы позволяет вводить число непосредственно за подсказкой компьютера, курсор для ввода стоит через пробел за знаком равенства.

- Третий способ заполнения используется для массивов малых размерностей и заключается в прямом присвоении в теле программы значений элементам массива. Например, допустимы следующие операторы:

```
FOR I := M TO N DO A[I] := 0;
```

Пример 34. В результате измерения случайного параметра сформирован массив из N вещественных чисел.

Вычислить эмпирическую среднюю $\tilde{X} = \frac{1}{N} \cdot \sum_{i=1}^{N} X_i$ и среднее

квадратическое отклонение $\sigma = \sqrt{\frac{\sum_{i=1}^{N} (X_i - \tilde{X})^2}{N-1}}$, где N = 10.

Обозначим M = $\tilde{X}$ и S = σ, тогда алгоритм программы будет иметь вид:

Структурограмма

Ввод массива X[1..N] с клавиатуры; M = 0; S = 0;
Для I от 1 до N с шагом 1 делать:
M := M + X[I];
M = M / N;
Для I от 1 до N с шагом 1 делать:
$S = S + (X[I] - M)^2$
$S = \sqrt{\frac{S}{N-1}}$; Вывод M и S на экран монитора.

Ввод массива это инструкция, содержащая насколько операторов, в том числе оператор цикла FOR. Но в этой структурограмме и во всех последующих примерах не будет уточняться способ ввода одномерного массива, оставляя выбор за программистом.

```
PROGRAM PR34;
CONST N = 10;
VAR          X: ARRAY [1 .. N] OF REAL; I: INTEGER; S, M: REAL;
BEGIN
WRITELN('Введите массив X, из ', N:2, ' вещественных чисел');
FOR I := 1 TO N DO READ(X[I]);
M := 0; S := 0;
FOR I := 1 TO N DO M := M + X[I];
M := M / N;
FOR I := 1 TO N DO S := S + (X[I] – M) * (X[I] – M);
S := SQRT(S / (N – 1));
WRITELN('M = ', M :10 :6, ', S = ', S :9 :6);
END.
```

Отображение на экране значений одномерного массива

Если в результате работы программы массив изменил свое состояние, и необходимо значения каждого из его элементов отобразить на монитор, то можно воспользоваться любым из двух способов, описанных ниже. Предположим, в программе сделаны объявления:

```
CONST M = 1; N = 15;
VAR          A: ARRAY[M .. N] OF REAL;
```

Тогда первый способ вывода элементов массива в строку будет иметь инструкцию:

```
WRITELN('Элементы массива A имеют значения: ');
FOR I := M TO N DO WRITE(A[I]: C: D, ' ');
WRITELN;
```

В этой инструкции первый оператор WRITELN сообщает оператору ЭВМ, какую информацию он увидит на экране. Второй оператор сформирует цепочку вещественных чисел, разделенных пробелами в формате : C: D. Третий оператор WRITELN переведет курсор на новую строку.

Второй способ обеспечивает вывод значений элементов массива в столбец, причем каждый из элементов будет идентифицирован:
FOR I := M TO N DO WRITELN('A[', I :2, '] = ', A[I]: C: D);

Пример 35. Найти сумму двух векторов $\vec{Z} = \vec{X} + \vec{Y}$, $\vec{X} = |X_1, X_2, ..., X_N|$, $\vec{Y} = |Y_1, Y_2, ..., Y_N|$, $\vec{Z} = |Z_1, Z_2, ..., Z_N|$. Элементы вектора $\vec{Z}$ находятся по известной формуле $Z_i = X_i + Y_i$.

Структурограмма

Ввод размерности массива N; Ввод массивов X [1..N] и Y [1..N];
Для I от 1 до N с шагом 1 делать:
Z [I] := X [I] + Y [I];
Вывод массива Z [1..N] на экран монитора.

Отличительная особенность этой задачи по сравнению с примером 34 состоит в том, что не известна величина N. В отличие от интерпретаторов типа БЕЙСИК, Паскаль требует, чтобы размерность массива была задана явно константами еще на этапе составления текста программы. Эти константы, определяющие диапазон изменения индексов, могут быть описаны явно в разделе CONST или неявно при описании массива. Когда величина N неизвестна, необходимо выяснить наибольшее из возможных значений G, которые по смыслу задачи может принять величина N:

$$G = Sup\ N, \Rightarrow N \le G.$$

После определения G, следует действовать следующим образом. В разделе констант объявить величину G. При описании массива верхнюю границу индекса положить равной G. Затем ввести с клавиатуры значение переменной N, и это значение использовать в программе, как верхнюю границу индекса. Очевидно, что элементы массива с индексами N+1, N+2, ..., G использоваться в конкретной реализации программы не будут.

```
PROGRAM PR35;
CONST G = 10; {Предполагается, что размерность векторов
                        не превышает 10}
VAR          X, Y, Z: ARRAY [1 .. G] OF REAL; I, N: INTEGER;
BEGIN
WRITELN('Введите размерность массивов X, Y и Z, величину N');
READLN(N);
WRITELN('Введите массив X, из ', N:2, ' вещественных чисел');
FOR I := 1 TO N DO READ(X[I]);
WRITELN('Введите массив Y, из ', N:2, ' вещественных чисел');
FOR I := 1 TO N DO READ(Y[I]);
FOR I := 1 TO N DO Z [I] := X [I] + Y [I];
WRITELN('Элементы вектора Z имеют значения: ');
FOR I := 1 TO N DO WRITE(Z[I]: 8: 4, ' ');
WRITELN
END.
```

Работа с индексами одномерного массива

Существует класс задач, в которых индекс массива используется для формализации вычислительного процесса путем сведения исходных формул к конечным суммам и произведениям. Преобразованные таким образом формулы программируются с помощью арифметических циклов. При обращении к элементам массива в качестве индексов можно использовать выражения перечисляемого типа.

Пример 36. Дана последовательность вещественных чисел X_1, X_2, X_3, ..., X_{24}. Требуется вычислить $U = X_1 \cdot X_2 \cdot X_3 \cdot X_4 + X_5 \cdot X_6 \cdot X_7 \cdot X_8 + \ldots + X_{21} \cdot X_{22} \cdot X_{23} \cdot X_{24}$.

Для программирования необходимо линейную формулу U преобразовать к следующему виду:

$$U = \sum_{i=1}^{6} \prod_{j=1}^{4} X_{4\cdot(i-1)+j}\,. \qquad (3\text{-}1)$$

Из (3-1) нетрудно заметить, что задача сведена к двойному арифметическому циклу.

Структурограмма

```
Ввод массива X [1..24]; U = 0;
    Для i от 1 до 6 с шагом 1 делать:
        P = 1;
            Для j от 1 до 4 с шагом 1 делать:
                P = P * X [4*(i - 1) + j];
        U = U + P;
Вывод U.
```

Для накопления суммы по I используется переменная U, исходное состояние которой равно 0. Для накопления произведения используется рабочая переменная P, которая рассчитывается шесть раз для значений индекса I = 1, 2, ..., 6. Для накопления произведения начальное значение J принимается равным 1.

```
PROGRAM PR36;
VAR          X: ARRAY [1 .. 24] OF REAL; I, J: INTEGER; U, P: REAL;
BEGIN
WRITELN('Введите массив X, из 24 вещественных чисел');
FOR    I := 1 TO 24 DO READ(X[I]); U := 0;
FOR    I := 1 TO 6
DO     BEGIN
       P := 1;
       FOR    J := 1 TO 4
       DO     P := P * X[4*(I - 1) + J];
       U := U + P
       END;
WRITELN('U = ', U:10:2)
END.
```

Пример 37. Дана последовательность $A_1, A_2, A_3, ..., A_{30}$. Вычислить следующие суммы:

$$\begin{aligned} S_1 &= A_1 + A_6 + A_{11} + A_{16} + A_{21} + A_{26} \\ S_2 &= A_2 + A_7 + A_{12} + A_{17} + A_{22} + A_{27} \\ &\vdots \\ S_5 &= A_5 + A_{10} + A_{15} + A_{20} + A_{25} + A_{30} \end{aligned} \qquad (3\text{-}2)$$

Для решения этой задачи можно также использовать двойной вложенный арифметический цикл. Для этого нужно сделать следующие преобразования исходных (3-2) формул:

$$S_i = \sum_{j=1}^{6} A_{5\cdot(j-1)+i};\ \mathrm{i} = 1, 2, 3, 4, 5. \qquad (3\text{-}3)$$

Для хранения в памяти компьютера и организации вычислительного процесса пять независимых величин S_i удобно рассматривать, как компоненты вектора $\vec{S} = \left| S_1, S_2, S_3, S_4, S_5 \right|$.

Структурограмма

```
Ввод массива A [1..30];
    Для I от 1 до 5 с шагом 1 делать:
        S[I] = 1;
            Для J от 1 до 6 с шагом 1 делать:
                S[I] = S[I] + A[5*(J - 1) + I];
        Вывод S[I] на экран монитора;
```

Внешний арифметический цикл с параметром I используется для перебора пяти элементов S_i и вывода на монитор их значений. Внутренний цикл с параметром J используется для накопления суммы $A_{5\cdot(j-1)+i}$, являющейся значением соответствующего элемента S_i.

```
PROGRAM PR37;
VAR          A: ARRAY [1 .. 30] OF REAL; I, J: INTEGER;
BEGIN
WRITELN('Введите массив A, из 30 вещественных чисел');
FOR   I := 1 TO 30 DO READ(X[I]);
FOR   I := 1 TO 5
DO    BEGIN
      S[I] = 1;
      FOR    J := 1 TO 6
      DO     S[I] = S[I] + A[5*(J - 1) + I];
      WRITELN('S', I:1, ' = ', S[I]:8:2)
      END
END.
```

В этой программе с целью сокращения исходного текста для вывода элементов массива S не строится самостоятельный цикл, а используется тот же цикл, что и для расчета элементов S_i.

Нахождение минимального и максимального элементов массива

Пример 38. В массиве X из 20 вещественных чисел поменять местами наибольшие и наименьшие элементы.

Уточним пространство решений. В массиве X может присутствовать несколько максимальных и минимальных элементов. Возможен неординарный случай для этой задачи, который состоит в том, что все элементы массива равны между собой.

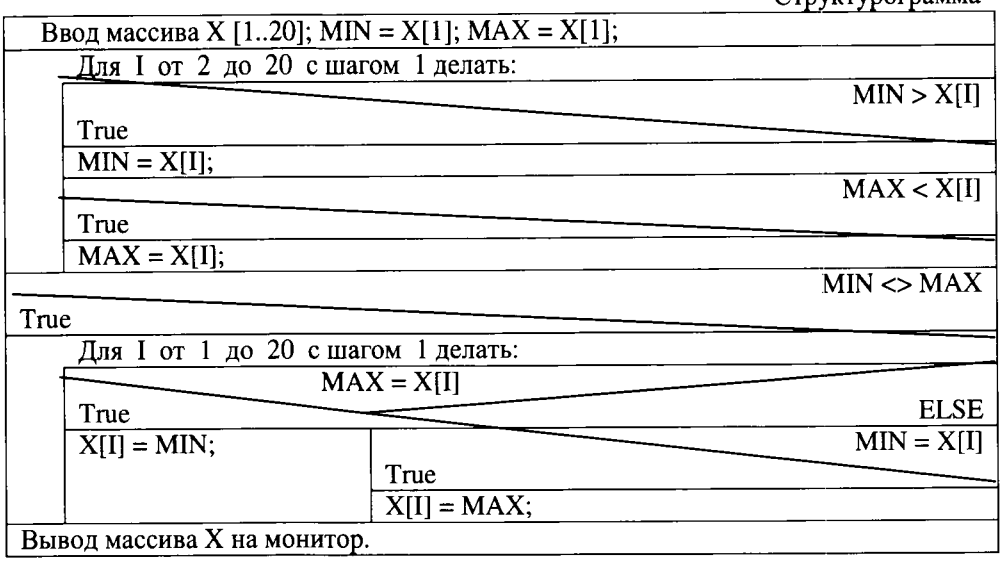
Переменные, используемые в программе, и их тип описаны в табл. 26.

Таблица 26

Идентификатор	Содержательный смысл	Тип переменной
X[I]	Элемент с индексом I массива X	REAL
MIN	Значение наименьшего элемента из X	REAL
MAX	Значение наибольшего элемента из X	REAL

Эту задачу нужно решать с помощью двух последовательных просмотров массива X. Целью первого просмотра является вычисление наибольшего MAX и наименьшего MAX значений элементов массива X. В начале просмотра значение первого элемента X[1] считается одновременно наибольшим и наименьшим, что справедливо в том случае, если в массиве всего один элемент. Далее со второго элемента X[2] и до последнего X[20] сравниваются значение текущего элемента с MIN. Если значение текущего элемента меньше, то оно с этого момента считается минимальным. По окончании цикла в рабочей ячейке MIN окажется число, равное значению наименьшего элемента. Аналогично необходимо поступить для нахождения MAX.

Структурограмма



Теперь наступило время сравнить между собой MIN и MAX. Если эти величины равны между собой, то массив состоит из 20 равнозначных элементов. Следовательно, переставлять их местами нет необходимости. Если MIN ≠ MAX, то нужно наименьшим элементам присвоить значение MAX, а наибольшим эле-

ментам присвоить значение MIN. Эти действия являются основой для второго просмотра массива X.

```
PROGRAM PR38;
VAR          X: ARRAY [1 .. 20] OF REAL; I: INTEGER; MIN, MAX: REAL;
BEGIN
WRITELN('Введите массив X, из 20 вещественных чисел');
FOR   I := 1 TO 20 DO READ(X[I]);
MIN := X[1]; MAX := X[1];
FOR   I := 2 TO 20
DO    BEGIN
      IF      MIN > X[I]
      THEN  MIN := X[I];
      IF      MAX < X[I]
      THEN MAX := X[I];
      END;
IF      MIN <> MAX
THEN  FOR  I := 1 TO 20
        DO    BEGIN
              IF       MAX = X[I]
              THEN  X[I] := MIN
              ELSE  IF          MIN = X[I]
                        THEN  X[I] := MAX;
              END;
WRITELN('Элементы массива X имеют значения: ');
FOR I := 1 TO 20 DO WRITE(X[I]: 4: 1, ' ');
WRITELN
END.
```

3.3. Сортировка одномерного массива

Сортировка - перестановка местами объектов в определенном порядке. Известно несколько сотен алгоритмов сортировки и их модификаций. Любой программист от начинающего до самого известного пробует свои силы именно при разработке алгоритмов этого класса (20).

Пусть дана последовательность элементов - $A_1, A_2, ..., A_n$. Сортировка означает перестановку этих элементов в новом порядке $A_{k_1}, A_{k_2}, ..., A_{k_n}$. Этот порядок соответствует значениям некоторой функции F, такой, что справедливо отношение $F(A_{k_1}) \le F(A_{k_2}) \le ... \le F(A_{k_n})$.

Как у любого вычислительного процесса у сортировки есть свои критерии для сравнительной оценки качества программы и соответствующего ей алгоритма (9). Такими критериями являются: объем используемой памяти и время работы программы.

Хорошей по критерию памяти считается такая сортировка, при которой данные сортируются в том же самом массиве, где находились исходные данные.

При оценке времени сортировки используют два показателя: число сравнений ключей (C), число пересылок данных (M). Хорошими по времени считают-

ся сортировки, в которых число сравнений $C \approx N \cdot Ln(N)$. К *простым*, то есть не очень хорошим, относятся такие сортировки, в которых число сравнений пропорционально квадрату размерности N исходного массива $C \approx N^2$. Следует отметить, что показатели C и M существенно зависят от первоначальной упорядоченности сортируемого массива. Наиболее тяжелым (Max) считается случай, когда массив упорядочен в обратном порядке. Ниже рассматривается три наиболее известных способа сортировки одномерного массива. Сравнительные временные характеристики этих способов приведены в табл. 27.

Таблица 27

Способ	Min	Средний	Max
Метод Пузырька	$C=(N^2-N)/2$, $M=0$	$C=(N^2-N)/2$, $M=0.75\cdot(N^2-N)$	$C=(N^2-N)/2$, $M=1.5\cdot(N^2-N)$
Простой выбор	$C=(N^2-N)/2$, $M=3\cdot(N-1)$	$C=(N^2-N)/2$, $M=N\cdot(Ln(N)+0.57)$	$C=(N^2-N)/2$, $M=N^2/4+3\cdot(N-1)$
Простое включение	$C=N-1$, $M=2\cdot(N-1)$	$C=(N^2+N-2)/4$, $M=(N^2-9N-10)/4$	$C=(N^2-N)/2-1$, $M=(N^2+3N-4)/2$

Сортировка простым обменом. Метод пузырька

Пример 39. Методом пузырька упорядочить (отсортировать) в порядке возрастания массив из 8 целых чисел (44, 55, 12, 42, 94, 18, 06, 67).

Концептуальную модель сортировки рассматривается на примере восьми целых чисел, которые располагаются в первом вертикальном столбце. Вертикальное расположение сортируемого массива наглядно иллюстрирует «всплывание легких элементов (чисел) вверх к поверхности» по мере сортировки массива.

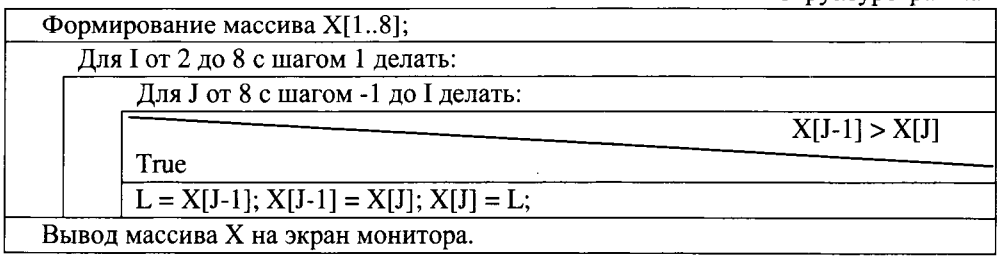
Элементы массива рассматриваются попарно снизу-вверх. Если нижний элемент меньше, то они меняются местами. При первом просмотре (проходе) самый «легкий» элемент оказывается самым верхним. Поэтому при втором просмотре его можно уже не рассматривать. В табл. 28 стрелками показаны перемещения элементов массива после каждого прохода.

Таблица 28

Индекс элемента массива	Номер прохода сортировки (I)							
	1	2	3	4	5	6	7	8
J = 1	44	**06**	06	06	06	06	06	06
2	55	44	**12**	12	12	12	12	12
3	12	55	44	**18**	18	18	18	18
4	42	12	55	44	**42**	42	42	42
5	94	42	**18**	55	44	**44**	44	44
6	18	94	42	42	55	55	**55**	55
7	06	18	94	67	67	67	67	**67**
8	67	67	67	94	94	94	94	94

Элементы всплывают вверх, к поверхности в соответствии с их весами, пока не встретят элемент с более малым весом. С каждым проходом наиболее легкий элемент из оставшихся поднимается на свое место, показанное жирными цифрами в табл. 28. Таким образом, нет необходимости на каждом проходе проверять элементы, стоящие выше выделенных, поскольку они уже отсортированы ранее. Начиная с 6 прохода, состояние массива не изменяется, что объясняется начальной упорядоченностью элементов в исходном массиве.

Структурограмма



Эта программа предназначена для изучения сортировки методом пузырька, поэтому взят массив из восьми целых чисел. Массив заранее известен, следовательно, инициализировать X[1..8] проще всего в разделе констант. Ввод данных с клавиатуры в этой программе не требуется. Алгоритм программы представляет собой двойной вложенный цикл. Индекс I внешнего арифметического цикла соответствует номеру прохода сортировки. Индекс J - это номер элемента в массиве. Для перестановки двух соседних элементов X[J-1] и X[J] местами используется промежуточная рабочая ячейка с идентификатором L.

```
PROGRAM PR39;
CONST X: ARRAY [1 .. 8] OF INTEGER = (44, 55, 12, 42, 94, 18, 06, 67);
VAR          L, I, J: INTEGER;
BEGIN
{ Вывод на экран исходного массива X }
FOR I := 1 TO 8 DO WRITE(X[I]:5); WRITELN;
FOR    I := 2 TO 8
DO     FOR    J := 8 DOWNTO I
       DO     IF X[J-1] > X[J]
              THEN BEGIN { Переставить X[J-1] с X[J] местами }
                     L := X[J-1]; X[J-1] := X[J]; X[J] := L
                     END; { IF }
{ Вывод на экран отсортированного массива X }
FOR I := 1 TO 8 DO WRITE(X[I]:5); WRITELN
END.
```

Сортировка простым включением

Пример 40. Методом простого включения упорядочить (отсортировать) в порядке возрастания массив из 8 целых чисел (44, 55, 12, 42, 94, 18, 06, 67).

Таблица 29

Номер прохода сортировки	Индекс элемента массива (J)							
	1	2	3	4	5	6	7	8
Нач. полож.	44	**55**	12	42	94	18	06	67
I = 2	44	55	**12**	42	94	18	06	67
3	12	44	55	**42**	94	18	06	67
4	12	42	44	55	**94**	18	06	67
5	12	42	44	55	94	**18**	06	67
6	12	18	42	44	55	94	**06**	67
7	06	12	18	42	44	55	94	**67**
8	06	12	18	42	44	55	67	94

Этот метод используют игроки в карты, перебирая все карты по очереди, начиная со второй и ставя ее на свое место по старшинству среди стоящих левее карт. В табл. 29 стрелками показаны перемещения элементов массива после каждого прохода. Жирный шрифт соответствует выбираемому элементу.

Структурограмма

```
Формирование массива X[1..8];
    Для I от 2 до 8 с шагом 1 делать:
        J = I - 1; L = X(I);
            Пока (J > 0 ) и (L < X(J) делать:
                X[J+1] = X[J]; J = J - 1;
        X[J+1] = L;
Вывод массива X на экран монитора.
```

```
PROGRAM  PR40;
CONST X: ARRAY [1 .. 8] OF INTEGER = (44, 55, 12, 42, 94, 18, 06, 67);
VAR              L, I, J: INTEGER;
BEGIN { Вывод на экран исходного массива  X }
FOR I := 1 TO 8 DO WRITE(X[I]:5); WRITELN;
FOR     I := 2 TO 8
DO      BEGIN
        J := I-1; L := X[I];
        WHILE (J > 0) AND (L < X[J])
        DO      BEGIN { Переставить X[J] на место X[J+1] }
                X[J+1] :=X [J]; J := J-1
                END;
        X[J+1] := L
        END;
{ Вывод на экран отсортированного массива  X }
FOR I := 1 TO 8 DO WRITE(X[I]:5); WRITELN
END.
```

Сортировка простым выбором

Пример 41. Методом простого выбора упорядочить (отсортировать) в порядке возрастания массив из 8 целых чисел (44, 55, 12, 42, 94, 18, 06, 67).

Таблица 30

Номер прохода сортировки	Индекс элемента массива (J)							
	1	2	3	4	5	6	7	8
I = 1	44	55	12	42	94	18	06	67
2	**06**	55	12	42	94	18	44	67
3	06	**12**	55	42	94	18	44	67
4	06	12	**18**	42	94	55	44	67
5	06	12	18	**42**	94	55	44	67
6	06	12	18	42	**44**	55	94	67
7	06	12	18	42	44	**55**	94	67
Результат	06	12	18	42	44	55	**67**	94

Этот метод более предпочтителен, чем сортировка простым включением. Концептуальная модель этого метода состоит в следующем. Начиная с первой позиции, просматриваются все N элементов, и находится номер K наименьшего из элементов. Элемент K ставится на первое место. А элемент, стоявший на втором месте, перемещается на место K. На втором проходе I = 2, первый элемент уже не рассматривается. Рассматриваются оставшиеся N-1 элементы, и среди них находится наименьший элемент, имеющий номер K. Этот элемент ставится на второе место, а элемент со второго места смещается на место K.

Этот процесс продолжается до тех пор, пока не будет просмотрен весь массив X, содержащий N элементов. В табл. 30 стрелками показаны перемещения элементов массива после каждого прохода. Жирный шрифт соответствует элементу, который уже занял свое место, и левее которого все элементы отсортированы в порядке убывания.

Структурограмма

```
Формирование массива X[1..8];
  Для I от 1 до 7 с шагом 1 делать:
    K = I;
      Для J от I + 1 до 8 с шагом 1 делать:
                                  X[J] < X[K];
        True
        K = J;
    L = X[I]; X[I] = X[K]; X[K] = L;
Вывод массива X на экран монитора.
```

```
PROGRAM PR41;
CONST X: ARRAY [1 .. 8] OF INTEGER = (44, 55, 12, 42, 94, 18, 06, 67);
VAR          K, L, I, J: INTEGER;
BEGIN { Вывод на экран исходного массива X }
```

```
FOR    I := 1 TO 8 DO WRITE(X[I]:5); WRITELN;
FOR    I := 1 TO 7
DO     BEGIN
       K := I;
       FOR J := I + 1 TO 8  { Поиск номера K наименьшего X[I] ... X[K] }
       DO IF X[J] < X[K]  THEN K := J;
       { Переставить X[J] на место X[J+1] }
       L := X[I]; X[I] := X[K]; X[K] := L;
       END;
{ Вывод на экран отсортированного массива  X }
FOR I := 1 TO 8 DO WRITE(X[I]:5); WRITELN
END.
```

Программное управление порядком сортировки

В приведенных примерах 39, 40, 41 массив сортировался в порядке возрастания элементов, в соответствии с критерием $F(A_{k_1}) \le F(A_{k_2}) \le ... \le F(A_{k_n})$. Часто используется и обратный порядок - сортировка по убыванию элементов массива в соответствии с критерием $F(A_{k_1}) \ge F(A_{k_2}) \ge ... \ge F(A_{k_n})$.

Пример 42. Методом пузырька отсортировать массив из N вещественных чисел. Порядок сортировки возрастания определить с клавиатуры.

В качестве базовой программы для решения этой задачи нужно взять пример 39. Легко заметить, что порядок сортировки по возрастанию определяется в операторе IF условием X[J-1] > X[J]. Заменяя это условие на ему обратное X[J-1] < X[J], происходит изменение порядка сортировки на противоположный. Управлять порядком сортировки необходимо с помощью логической переменной F, принимающей значения:

$$F = \begin{cases} True, & \text{возрастание,} \\ False, & \text{убывание.} \end{cases}$$

```
PROGRAM  PR42;
CONST N = 20; {Верхний предел размерности массива X (K<=N)}
VAR           X: ARRAY [1 .. N] OF REAL; K, I, J: INTEGER;
              R: REAL; F: BOOLEAN;
BEGIN
WRITELN('Введите размерность K массива X'); READLN(K);
WRITELN('Введите массив X');
FOR I := 1 TO K DO READ(X[I]);
WRITELN('Укажите порядок сортировки: 1 - по возрастанию'); READLN(I);
F := TRUE;
IF I <> 1 THEN F := FALSE;
FOR    I := 2 TO K
DO     FOR    J := K  DOWNTO  I
       DO     IF ((X[J-1] > X[J]) AND F) OR ((X[J-1] < X[J]) AND NOT F)
              THEN  BEGIN  { Переставить X[J-1] с X[J] местами }
                           R := X[J-1]; X[J-1] := X[J]; X[J] := R
                           END; { IF }
```

```
{ Вывод на экран отсортированного массива X }
FOR I := 1 TO K DO WRITE(X[I]:5:1); WRITELN
END.
```

На примере 42 нетрудно заметить одно существенное неудобство, связанное с использованием регулярных типов. Оно состоит в том, что необходимо сразу фиксировать число элементов массива - K. В отдельных случаях заранее не известна размерность массива, подлежащего обработке. Например, может возникнуть необходимость в одном случае отсортировать массив в 20 вещественных чисел, а в другом - 100. Поэтому в программу приходится вносить исправления. Здесь помогает понятие константы, описанной в разделе CONST. Достаточно заменить описание N = 20 на N = 30, или N = 100, и больше никаких исправлений в программе не потребуется. Если бы программа не содержала раздела CONST, а размерность массива была указана в тексте программы, то потребовалось бы внести, по крайней мере, четыре исправления. При этом необходимо разбираться в сути программы, иначе велика вероятность того, что какие-то места все-таки будут пропущены.

3.4. Многомерные массивы

Индексы имеют еще одно свойство - чем больше объем массива, тем менее эффективна с ним работа, поэтому часто используют массивы массивов, то есть с двумя, тремя и более индексами для идентификации элементов. Конструирование многомерных массивов в общем виде выглядит следующим образом:

< имя типа > = ARRAY [тип индекса] OF < тип элементов массива >,

где тип элемента массива, в свою очередь, - массив, содержащий N-1 индекс. Допустима запись:

< имя типа > = ARRAY[тип индекса N] OF ARRAY[тип индекса N-1]
OF ARRAY [тип индекса N-2] OF ... ARRAY [тип индекса 1]
OF < тип элементов массива >;

Так определен массив от N индексов, то есть N-мерный массив. Это же определение массива может быть сделано в сокращенной форме:

< имя типа > = ARRAY [тип индекса N, тип индекса N-1, ... , тип индекса 1]
OF < тип элементов массива >.

Можно использовать и другие формы определения массива:

< имя типа > = ARRAY[тип индекса N, тип индекса N-1]
OF ARRAY [тип индекса N-2, ... , тип индекса 1]
OF <тип элементов массива>;

Все указанные формы описания типов могут быть приведены как в явном виде в разделе TYPE, так и в неявном в разделе VAR. Обращение к многомерному массиву осуществляется с помощью индексированной переменной вида X[I1, I2, ..., IN], где X - имя массива, а I1, I2,..., IN - константы, переменные или выражения, типы которых должны соответствовать типам индексов в определении массива. Указанная форма записи индексированной переменной называется сокращенной и наиболее часто используется. Хорошим стилем программирования считается использование в программе либо сокращенной, либо полной формы описания массивов.

Ниже описаны многомерные массивы, содержащие два индекса. Правила работы с такими массивами полностью распространяются на массивы, имеющие три и более индексов.

Двумерные массивы. Матрицы

Массивы массивов, содержащие два индекса (N = 2), называют двумерными. Если элементами таких массивов являются простые числовые данные, то эти массивы часто называют матрицами.

Обращение к элементам двумерного массива осуществляется с помощью индексированных переменных, например A[I, J]. Здесь A - имя массива, а I и J - константы, переменные или выражения того же типа, что и типы индексов в объявлении массива. Двумерный массив обычно используют для представления в памяти компьютера матриц:

$$A = \begin{vmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{vmatrix}.$$

Связь между элементами матрицы A_{IJ}; $I = \overline{1,m}$; $J = \overline{1,n}$ и соответствующими компонентами двумерного массива простая и очевидная $A_{IJ} \Leftrightarrow A[I,J]$.

Объявление и инициализация матрицы

Объявление и инициализация матрицы аналогичны описанным выше способам работы с векторами. То есть в разделе VAR переменным отводится место, а начальное значение этих величин специально не устанавливается. Поэтому после объявления массива необходимо его элементам задать необходимые значения. Используется три способа инициализации двумерного массива:

- Если значения элементов массива определены до начала работы программы, то есть известны на этапе формулировки задания на программирование, то можно использовать следующий способ:

```
CONST A: ARRAY [1..5, 1..2] OF REAL = ((0.1, -15.3), (7, 0), (-11.89, 4),
                                       (-78, 11.2), (1, 0.01));
```

При таком объявлении массива в разделе констант, заносится в двумерный массив A по порядку A[1, 1] = 0.1, A[1, 2] = -15.3, A[2, 1] = 7, A[2, 2] = 0, ... A[5, 2] = 0.01 вещественные числа, перечисленные в круглых скобках. При этом массив является массивом переменных, то есть в процессе работы программы можно менять содержимое любого элемента матрицы.

- Второй способ применяется в том случае, если исходные данные необходимо внести с клавиатуры в процессе выполнения программы. Так как матрица представляет собой конечный набор однотипных элементов, пронумерованных с помощью двух индексов I и J, то удобно использовать вложенный арифметический цикл (операторы FOR) для ввода значений непосредственно с клавиатуры. При этом можно предложить два приема. Пусть, в программе сделаны объявления:

```
CONST M = 3; N = 4;
VAR          A: ARRAY[1 .. M, 1 .. N] OF REAL;
```

где M - количество строк, а N - количество столбцов в матрице. Первый способ ввода будет иметь инструкцию:

```
WRITELN('Введите массив A, из 12 вещественных чисел');
FOR  I := 1 TO M
DO   FOR  J := 1 TO N
     DO   READ(A[I, J]);
```

При таком способе оператор может ввести все 12 чисел в форме матрицы. Через пробел в одну строку вводится четыре числа первой строки и затем нажимается на клавишу Enter. Вторая и третья строки матрицы **A** вводятся аналогично. На экране монитора остается изображение матрицы в удобочитаемом виде, близком к оригиналу.

Второй способ ввода имеет вид:

```
FOR  I := 1 TO M
DO   FOR   J := 1 TO N
     DO    BEGIN
           WRITE('A[', I:1, J:1 '] = ');
           READLN(A[I, J])
           END;
```

Этот фрагмент программы позволяет вводить число непосредственно за подсказкой компьютера, курсор для ввода стоит через пробел за знаком равенства.

- Третий способ заполнения заключается в прямом присвоении в теле программы значений элементам массива. Например, следующие операторы вложенного арифметического цикла в совокупности с оператором присваивания обеспечивают обнуление всех 12 компонентов массива:

```
FOR  I := 1 TO M
DO   FOR J := 1 TO N
     DO A[I, J] := 0;
```

Пример 43. Дана матрица $A_{7\times9}$ и вектор $\vec{S} = \left| S_1, S_2, S_3, \cdots, S_7 \right|$ требуется вычислить: $Y_1 = S_1 \cdot \sum_{j=1}^{9} A_{1j}$, $Y_2 = S_2 \cdot \sum_{j=1}^{9} A_{2j}, \cdots, Y_7 = S_7 \cdot \sum_{j=1}^{9} A_{7j}$.

Для решения этой задачи нужно преобразовать расчетные формулы к виду, удобному для программирования. Пусть дан, вектор $\vec{Y} = \left| Y_1, Y_2, Y_3, \cdots, Y_7 \right|$, компоненты которого вычисляются по формулам:

$$Y_i = S_i \cdot \sum_{j=1}^{9} A_{ij},\ i = \overline{1,7}. \qquad (3\text{-}4)$$

Для решения этой задачи с клавиатуры нужно ввести матрицу **A** размерности 7×9 и вектор $\vec{S}$, содержащий 9 элементов. Вычисления результирующего вектора $\vec{Y}$ носят циклический характер. Внешний цикл по I обеспечивает расчет семи координат вектора $\vec{Y} = \left| Y_1, Y_2, Y_3, \cdots, Y_7 \right|$. Внутренний цикл по J поддерживает накопление суммы $\sum_{j=1}^{9} A_{ij}$ для конкретного значения индекса I. По окончании расчетов одномерный массив Y выводится на экран монитора.

Структурограмма

Формирование массива A[1..7,1..9]; Формирование массива S[1..9];
Для I от 1 до 7 с шагом 1 делать:
Y[I] = 0;
Для J от 1 до 9 с шагом 1 делать:
Y[I] = Y[I] + A[I, J];
Y[I] = S[I] * Y[I];
Вывод массива Y на экран монитора.

```
PROGRAM PR43;
VAR          S: ARRAY [1 .. 9] OF REAL; Y: ARRAY [1 .. 7] OF REAL;
             A: ARRAY [1 .. 7,1 .. 9] OF REAL; I, J: INTEGER;
BEGIN
WRITELN('Введите матрицу A');
FOR I := 1 TO 7 DO FOR J := 1 TO 9 DO READ(A[I, J]);
WRITELN('Введите вектор S');
FOR I := 1 TO 9 DO READ(S[I]);
FOR   I := 1 TO 7
DO    BEGIN Y[I] := 0;
      FOR   J := 1 TO 9
      DO    Y[I] := Y[I] + A[I, J];
      Y[I] := S[I] * Y[I];
      END; { IF }
FOR I := 1 TO 7 DO WRITELN('Y[', I:1, '] =', Y[I]:8:3)
END.
```

Отображение на экране значений двумерного массива

Если в результате работы программы необходимо значения каждого элемента двумерного массива отобразить на экране монитора, то можно воспользоваться любым из двух способов, описанных ниже. Пусть, в программе сделаны объявления:

```
CONST M = 3; N = 4;
VAR          A: ARRAY[1 .. M, 1 .. N] OF REAL;
```

Тогда первый способ вывода элементов массива в виде матрицы будет иметь инструкцию:

```
WRITELN('Элементы матрицы A имеют значения: ');
FOR I := 1 TO M
DO    BEGIN
      FOR J := 1 TO N DO WRITE(A[I, J]: C: D, ' '); {Вывод строки}
      WRITELN              {Переход к новой строке}
      END;
```

В этой инструкции первый оператор WRITELN сообщает оператору, какую информацию он увидит на экране. Второй оператор WRITE сформирует цепочку (строку) вещественных чисел, разделенных пробелами в формате :C :D. Третий оператор WRITELN переведет курсор на новую строку.

Второй способ обеспечивает вывод значений элементов двумерного мас-

сива в столбец, причем каждый из элементов будет идентифицирован парой индексов I и J:

```
FOR I := 1 TO M
DO      FOR J := 1 TO N
        DO WRITELN('A[', I :1, ', ', J :1, '] = ', A[I, J]: C: D);
```

Пример 44. Найти сумму двух матриц **C = A + B** размерностью m × n. Элементы C_{ij} искомой матрицы **C** вычисляются по формулам:

$C_{ij} = A_{ij} + B_{ij}$; $i = \overline{1,m}$; $j = \overline{1,n}$.

Эта задача отличается от предыдущего примера тем, что неизвестна размерность матриц. Поэтому значения m и n необходимо ограничить сверху константами $GM = Sup\ m, GN = Sup\ n$.

Структурограмма

Ввод размерностей M и N матриц A и B;
Формирование массивов A[1..GM, 1..GN] и B[1..GM, 1..GN];
Для I от 1 до M с шагом 1 делать:
Для J от 1 до N с шагом 1 делать:
C[I, J] = A[I, J] + B[I, J];
Вывод массива C на экран монитора.

```
PROGRAM PR44;
CONST GM = 8; GN = 8;
VAR             A, B, C: ARRAY [1 .. GM, 1 .. GN] OF REAL; M, N, I, J: INTEGER;
BEGIN
WRITELN('Введите количество строк M и столбцов N матрицы A');
READLN(M, N);
WRITELN('Введите матрицу A');
FOR I := 1 TO M DO FOR J := 1 TO N DO READ(A[I, J]);
WRITELN('Введите матрицу B');
FOR I := 1 TO M DO FOR J := 1 TO N DO READ(B[I, J]);
FOR     I := 1 TO M                     { Вычисление матрицы C }
DO      FOR     J := 1 TO N
        DO      C[I, J] := A[I, J] + B[I, J];
WRITELN('Матрица C имеет вид:');
FOR I := 1 TO M
        DO      BEGIN
                FOR J := 1 TO N DO WRITE(A[I, J]: 5: 2, ' ');
                WRITELN
                END
END.
```

Работа с индексами двумерного массива

Пример 45. Задана квадратная матрица $A_{n \times n}$. Найти суммы двух главных диагоналей D_1 и D_2 матрицы **A**.

Очевидны следующие расчетные формулы:

$$D_1 = A_{11} + A_{22} + A_{33} + A_{44} + A_{55} + \cdots + A_{nn}, \qquad (3\text{-}6)$$

$$D_2 = A_{1n} + A_{2,n-1} + A_{3,n-2} + A_{4,n-3} + \cdots + A_{n1}.$$

Для решения этой задачи необходимо использовать индексы массива A для формализации вычислительного процесса. При этом формулы (3-6) для расчета D_1 и D_2 примут вид:

$$D_1 = \sum_{k=1}^{n} A_{kk}, \quad D_2 = \sum_{k=1}^{n} A_{k,n-k+1}. \qquad (3\text{-}7)$$

Построение этих формул сводится к следующим действиям: пересчитать количество слагаемых (элементов матрицы A) и составить выражения для первого и второго индекса.

Структурограмма

Ввод размерности N матрицы A;
Формирование матрицы A[1..GN, 1..GN]; D1 =0; D2 = 0;
Для K от 1 до N с шагом 1 делать:
D1 = A[K, K] + D1; D2 = A[K, N-K+1] + D2;
Вывод D1 и D2 на экран монитора.

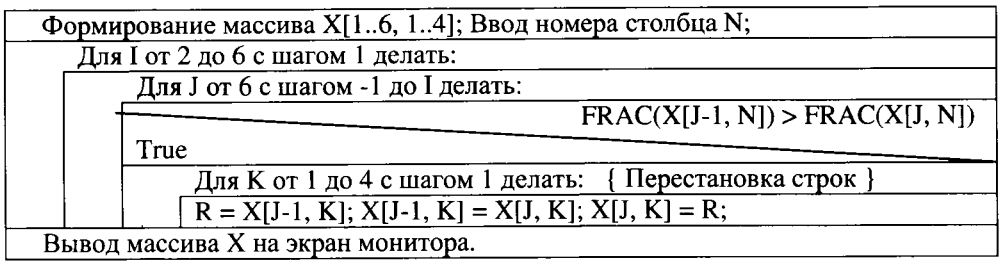
```
PROGRAM PR45;
CONST GN = 20;          {Верхняя граница размерности матрицы A}
VAR             A: ARRAY [1 .. GN, 1 .. GN] OF REAL;
                K, N, I, J: INTEGER; D1, D2: REAL;
BEGIN
WRITELN('Укажите количество строк и столбцов N матрицы A'); READLN(N);
WRITELN('Введите матрицу A');
FOR I := 1 TO N DO FOR J := 1 TO N DO READ(A[I, J]);
D1 := 0; D2 := 0;
FOR     K := 1 TO N
DO      BEGIN
        D1 := A[K, K] + D1;      { Вычисление диагонали D1}
        D2 := A[K, N-K+1] + D2;  { Вычисление диагонали D2}
        END;
WRITELN('D1 = ', D1:5:2, ', D2 = ', D2:5:2);
END.
```

Сортировка двумерного массива

Пример 46. Задан двумерный массив X из 6 строк и 4 столбцов. Упорядочить массив X по возрастанию элементов дробной части столбца с номером N. Отсортированный массив X вывести на экран монитора.

За основу алгоритма необходимо взять алгоритм сортировки одномерного массива методом пузырька. Отличие этой задачи от сортировки одномерного массива в том, что переставлять местами нужно не два соседних элемента одномерного массива, а две соседние строки двумерного массива. Перестановка двух строк реализована с помощью арифметического цикла, третьего уровня вложения, с параметром цикла K.

Структурограмма

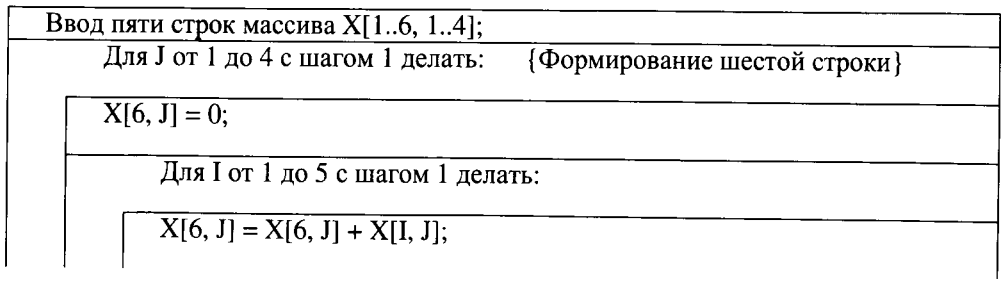



```
PROGRAM  PR46;
VAR              X: ARRAY [1..6, 1..4] OF REAL; N, K, I, J: INTEGER; R: REAL;
BEGIN
WRITELN('Введите матрицу X');
FOR I := 1 TO 6 DO FOR J := 1 TO 4 DO READ(X[I, J]);
WRITELN('Укажите номер столбца N матрицы X'); READLN(N);
FOR      I := 2 TO 6
DO       FOR      J := 6  DOWNTO  I
         DO       IF FRAC(X[J-1, N]) > FRAC(X[J, N])
                  THEN  FOR  K := 1 TO 4          { Перестановка строк }
                           DO BEGIN
                                 R := X[J-1, K]; X[J-1, K] := X[J, K]; X[J, K] := R;
                                 END;
{ Вывод на экран отсортированного массива  X }
WRITELN('Матрица X имеет вид:');
FOR I := 1 TO 6
         DO       BEGIN
                  FOR J := 1 TO 4 DO WRITE(X[I, J]: 6: 3, ' '); WRITELN
                  END
END.
```

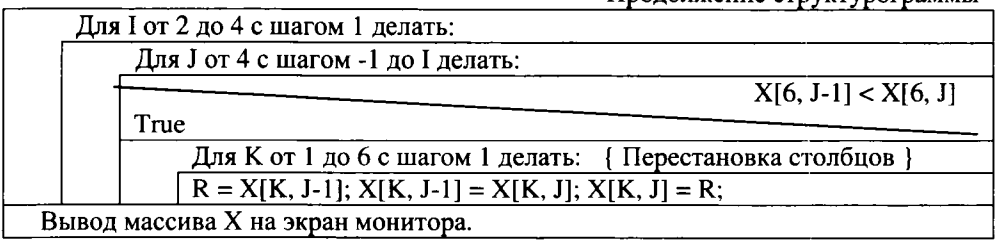
Пример 47. Задан двумерный массив X из 5 строк и 4 столбцов. Упорядочить массив X по убыванию сумм элементов столбцов.

Для решения этой задачи можно использовать дополнительную шестую строку матрицы X, в которой нужно хранить сумму остальных пяти элементов каждого столбца матрицы. Перестановку столбцов матрицы можно делать именно по этой строке в порядке убывания значений ее элементов. При выводе матрицы на экран монитора эту строку можно не показывать.

Структурограмма



Продолжение структурограммы




```
PROGRAM  PR47;
VAR              X: ARRAY [1..6, 1..4] OF REAL; K, I, J: INTEGER; R: REAL;
BEGIN
WRITELN('Введите матрицу X');
FOR I := 1 TO 5 DO FOR J := 1 TO 4 DO READ(X[I, J]);
FOR J := 1 TO 4          {Формирование шестой строки}
DO      BEGIN
        X[6, J] := 0;
        FOR     I := 1 TO 5
        DO      X[6, J] := X[6, J] + X[I, J]
        END;
FOR     I := 2 TO 4
DO      FOR     J := 4  DOWNTO  I
        DO      IF X[6, J-1] < X[6, J]
                THEN  FOR  K := 1 TO 6   { Перестановка столбцов }
                        DO BEGIN
                                R := X[K, J-1]; X[K, J-1] := X[K, J]; X[K, J] := R;
                                END;
{ Вывод на экран отсортированного массива  X }
WRITELN('Матрица X имеет вид:');
FOR I := 1 TO 5
        DO      BEGIN
                FOR J := 1 TO 4 DO WRITE(X[I, J]: 6: 3, ' '); WRITELN
                END
END.
```

Операции над массивами

В Паскале допускается работа с массивом как единым целым. В этом случае используется идентификатор массива без указания индекса в квадратных скобках.

Пусть есть два произвольных идентичных массива A и B, например: VAR A, B: ARRAY [1 .. M, 1 .. N] OF REAL; тогда определено действие присвоения A := B; (копирование массива A в B). При этом все элементы массива A становятся равными соответствующим элементам массива B.

В условиях (логических выражениях) часто требуется использовать две операции отношения над массивами:

- A = B - результат: TRUE, если элементы A[i, j] = B[i, j], и FALSE, если есть

хотя бы один элемент A[i, j] ≠ B[i, j];

- A <> B - результат: TRUE, если хотя бы один элемент A[i, j] отличен от B[i, j], и FALSE в противном случае.

Эти операции, хотя и понятны, и логичны, однако, не реализованы в рассматриваемой версии языка (за исключением одномерного массива литер). Поэтому в тех случаях, когда в задаче необходимо сравнить два массива, приходится организовывать арифметический цикл попарного сравнения элементов двух массивов. В приведенном ниже примере реализовано отношение A = B. Булевская переменная FLAG принимает значение TRUE, если элементы этих массивов попарно равны и FALSE в противном случае.

Пример 48. Вектор параметров $\vec{A}$ поддерживаемого технологического режима должен иметь нормативное значение - $\vec{A} = |1, 2.5, 4.2, 14|$. Если вектор $\vec{B}$ фактических значений параметров соответствует нормативным значениям, значит на исполнительные механизмы необходимо выдать команду $\vec{C}_1 = |1, 0, 1, 1, 1|$. При нарушении режима требуется сформировать команду $\vec{C}_2 = |1, 1, 1, 0, 0|$.

Эта задача имитирует фрагмент алгоритма управления гипотетической технологической установкой, на вход которой подается команда управления пятью дискретными исполнительными механизмами (вектор $\vec{C}$), а состояние контролируется с помощью четырех аналоговых датчиков (вектор $\vec{B}$). Моделируя работу этой установки, значения вектора $\vec{B}$ будем вводить с клавиатуры. А выбранное управляющее значение - вектор $\vec{C}_1$ или $\vec{C}_2$ будем отображать на экране монитора.

```
PROGRAM PR48;
TYPE        CX = ARRAY[1..5] OF BYTE;        {Выходные воздействия}
            AB = ARRAY[1..4] OF REAL;        { Входные параметры  }
CONST A: AB = (1, 2.5, 4.2, 14);
            C1: CX = (1, 0, 1, 1, 1); C2: CX = (1, 1, 1, 0, 0);
VAR   B: AB; I: INTEGER; C: CX; FL: BOOLEAN;
BEGIN
WRITELN('Введите показания 4-х датчиков');
FOR I := 1 TO 4 DO READ(B[I]);
FL := TRUE;
FOR I:=1 TO 4        {Сравнение двух массивов A и B}
DO  IF A[I] <> B[I] THEN FL := FALSE;
IF FL
THEN  C := C1          {Копирование массива C1 в массив C}
ELSE  C := C2;         {Копирование массива C2 в массив C}
FOR   I := 1 TO 5 DO WRITE(C[I]:1, ' '); WRITELN
END.
```

Замечание. Для копирования массивов желательно типы этих массивов объявить в разделе TYPE в явном виде. В этом примере это тип CX. Тип AB объявлен в явном виде для сокращения исходного текста программы, которая стала нагляднее и компактнее.

4. Вычислительные процессы линейной алгебры

Методы линейной алгебры лежат в основе многих вычислительных процессов, на базе которых строятся программы обращения матриц, вычисление собственных чисел и векторов, умножения матриц и векторов, решения систем линейных алгебраических уравнений.

Во всех приведенных в дальнейшем примерах процессов и программах предполагается, что векторы представлены одномерными массивами. Величины **n**, **m** определяют количество строк и столбцов прямоугольной матрицы соответственно, при **n = m** рассматриваются квадратные матрицы. Если рассматриваются вектора, то величина n определяет число компонент вектора. Эти же величины определяют верхние границы размерностей массивов.

Целочисленные переменные **i, j, k, l** соответственно могут использоваться для перечисления строк, столбцов и (или) компонент векторов.

4.1. Операции над матрицами и векторами

В этом разделе программируются наиболее часто встречающиеся математические операции над векторами и матрицами.

Скалярное произведение векторов

Скалярное произведение векторов **X** и **Y** вычисляется по формуле:

$$S = \sum_{i=1}^{n} x_i \cdot y_i , \qquad (4\text{-}1)$$

где x_i, y_i - соответственно компоненты векторов **X, Y;** n - размерность векторов.

Пример 49. Разработать программу вычисления скалярного произведения двух векторов $S = \vec{X} \cdot \vec{Y}$ по формуле (4-1).

При решении задач линейной алгебры желательно использовать таблицы имен, которые раскрывают связь между математической моделью вычислительного процесса и внутренним представлением данных в компьютере.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
x_i	X[i]	Элемент массива X, i-ая компонента вектора X.	real
y_i	Y[i]	Элемент массива Y, i-ая компонента вектора Y.	real
n	n	Размерность вектора.	integer
S	S	Скалярное произведение.	real
i	i	Текущий номер компоненты.	integer

Структурограмма

Ввод размерности векторов N; S = 0;
Для I от 1 до N с шагом 1 делать:
Ввести элементы X[I] и Y[I]; S = S + X[I] * Y[I];
Вывод S на экран монитора.

В предлагаемом алгоритме используется всего один цикл для ввода элементов двух векторов X и Y и расчета элементов нового вектора C. Этот прием достаточно часто используется в задачах линейной алгебры и других численных методах.

Пусть максимальная размерность векторов в программе задается константой NMAX и для определенности равняется 25.

```
PROGRAM PR49;
CONST NMAX = 25; {Верхняя граница количества элементов X и Y}
VAR   I, N: INTEGER; S: REAL; X, Y: ARRAY [1..NMAX] OF REAL;
BEGIN
WRITELN('Введите размерность векторов X и Y'); READLN(N);
S := 0;
WRITELN('Введите попарно компоненты двух векторов X и Y');
FOR   I := 1 TO N
DO    BEGIN READLN(X[I], Y[I]); S := S + X[I]*Y[I]  END;
WRITELN('S = ', S:10:4);
END.
```

Умножение вектора на матрицу

Умножение вектора на матрицу $\vec{C} = \vec{B} \times A$ выполняется по формуле:

$$C_j = \sum_{i=1}^{m} b_i \cdot a_{ij}\ ,\quad j = \overline{1,n}\ , \tag{4-2}$$

где a_{ij} - элементы прямоугольной матрицы $A_{m\times n}$; **n** - число столбцов матрицы $A_{m\times n}$; **m** - число строк матрицы $A_{m\times n}$ и число компонент вектора $\vec{B}$; b_j - компоненты вектора **B;** C_i - компоненты вектора **C**, получаемые в результате умножения.

Пример 50. Разработать программу вычисления произведения вектора $\vec{B} = |\, b_1, b_2, \cdots, b_m \,|$ на матрицу $A_{m\times n}$ по формуле (4-2).

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
a_{ij}	A [i, j]	Элемент двухмерного массива **A**, элемент i-ой строки и j-ого столбца матрицы $A_{m\times n}$.	real
b_i	B [i]	i-ый элемент одномерного массива B, i-ая компонента вектора $\vec{B}$.	real

Продолжение табл. имен

C_i	C [i]	i-ый элемент одномерного массива C, i-ая компонента вектора $\vec{C}$.	real
m	m	Количество строк матрицы $A_{m\times n}$, и элементов вектора $\vec{B}$.	integer
n	n	Количество столбцов матрицы $A_{m\times n}$.	integer

Структурограмма

```
Ввод размерности матрицы M и N; Ввод элементов матрицы A;
Ввод элементов вектора B;
    Для I от 1 до N с шагом 1 делать:
        C[I] = 0;
            Для J от 1 до M с шагом 1 делать:
                C[I] = C[I]+A[J, I]*B[J];
        Вывод элемента C[I] вектора C на экран монитора;
```

Программа процесса умножения предполагает поэлементный ввод массивов **A**, **B** и вычисление $\mathbf{C_i}$ как скалярного произведения вектора $\vec{B}$ на i-ый вектор-столбец матрицы $A_{m\times n}$.

```
PROGRAM PR50;
CONST NM = 20; MM = 20;
VAR   I, J, N, M: INTEGER; B: ARRAY[1..NM] OF REAL;
C: ARRAY[1..MM] OF REAL; A: ARRAY[1..MM, 1..NM] OF REAL;
BEGIN
WRITELN('Укажите  M - размерность вектора B и N - столбцов матрицы A');
READLN(M, N);
WRITELN('Введите матрицу A');
FOR   I := 1 TO M
DO    FOR J := 1 TO N DO READ(A[I, J]);
WRITELN('Введите вектор B');
FOR   J := 1 TO M DO READ(B[J]);
FOR   I := 1 TO N                {Вычисление вектора C и вывод на монитор}
DO    BEGIN C[I] := 0;
      FOR   J := 1 TO M DO C[I] := C[I]+A[J, I]*B[J];
      WRITELN('C[', I:1, '] = ', C[I]:10:5)
      END
END.
```

Умножение матрицы на матрицу

Умножение матрицы на матрицу **C = A × B** выполняется по формуле:

$$C_{ij} = \sum_{k=1}^{p} A_{ik} \cdot B_{kj}\ ;\ i = \overline{1, m}\ ;\ j = \overline{1, n}\ . \qquad (4\text{-}3)$$

где A_{ik} - элементы прямоугольной матрицы $A_{m\times p}$; **m** - число строк матрицы $A_{m\times p}$; **p** - число столбцов $A_{m\times p}$ и число строк матрицы $B_{p\times n}$; B_{kj} - компоненты матрицы $B_{p\times n}$; **n** - число столбцов матрицы $B_{p\times n}$; C_{ij} - компоненты матрицы $C_{m\times n}$, получаемые в результате умножения.

Пример 51. Найти произведение двух вещественных матриц $A_{2\times 3}$ и $B_{3\times 4}$ по формуле (4-3).

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
A_{ik}	A [i, k]	Элемент двухмерного массива **A**, элемент i-ой строки и k-ого столбца матрицы $A_{m\times p}$.	real
B_{kj}	B [k, j]	Элемент двухмерного массива **B**, элемент k-ой строки и j-ого столбца матрицы $B_{k\times n}$.	real
C_{ij}	C [i, j]	Элемент двухмерного массива **C**, элемент i-ой строки и j-ого столбца матрицы $C_{m\times n}$.	real
m = 2	m	Количество строк матриц $A_{m\times p}$, $C_{m\times n}$.	integer
p = 3	p	Количество столбцов матрицы $A_{m\times p}$, и строк матрицы $B_{k\times n}$.	integer
n = 4	n	Количество столбцов матрицы $C_{m\times n}$.	integer

Эту конкретную задачу необходимо решать в общем виде. Для этого в разделе констант нужно сделать объявление: CONST M = 2; P = 3; N = 4;

Сами матрицы определены в неявной форме в разделе переменных VAR. Для ввода и вывода элементов матриц A, B, C использованы двойные циклы по индексам I, J.

Структурограмма

```
Ввод матрицы A; Ввод элементов матрицы B;
    Для I от 1 до M с шагом 1 делать:
        Для J от 1 до N с шагом 1 делать:
            C[I, J] = 0;
            Для K от 1 до P с шагом 1 делать:
                C[I, J] := C[I, J] + A[I, K]*B[K, J];
Вывод матрицы C на экран монитора.
```

```
PROGRAM PR51;
CONST M = 2; P = 3; N = 4;
VAR   A: ARRAY [1 .. M, 1 .. P] OF REAL; B: ARRAY [1 .. P, 1 .. N] OF REAL;
      C: ARRAY [1 .. M, 1 .. N] OF REAL; I, J, K: INTEGER;
BEGIN {Ввод матриц A и B}
WRITELN('Введите матрицу A');
FOR I := 1 TO M DO FOR J := 1 TO P DO READ (A[I, J]);
WRITELN('Введите матрицу B');
FOR I := 1 TO P DO FOR J := 1 TO N DO READ (B[I, J]);
{Умножение матриц A и B}
FOR I := 1 TO M
DO    FOR J := 1 TO N
      DO    BEGIN
            C[I, J] := 0;
            FOR   K := 1 TO P
            DO    C[I, J] := C[I, J] + A[I, K]*B[K, J]
            END;
{ Отображение матрицы C }
WRITELN('Матрица C:');
FOR I := 1 TO M
DO    BEGIN
      FOR J := 1 TO N DO WRITE(C[I, J]:10:5);
      WRITELN
      END
END.
```

4.2. Умножение транспонированной прямоугольной матрицы на исходную матрицу

Прямоугольная матрица $A_{m\times n}$ после перестановки строк на место столбцов превращается в транспонированную матрицу $A^{\mathrm{T}}_{n\times m}$ по отношению к исходной $A_{m\times n}$.

В некоторых задачах линейной алгебры возникает необходимость перемножения слева транспонированной матрицы на исходную. Результирующая квадратная матрица $AR_{n\times n}$ определяется по формуле:

$$AR_{n\times n} = A^{\mathrm{T}}_{n\times m} \cdot A_{m\times n} \qquad (4\text{-}4)$$

Элементы матрицы $AR_{n\times n}$ целесообразно вычислять, пользуясь формулой:

$$ar_{ij} = \sum_{k=1}^{m} a_{ki} \cdot a_{kj} \ , \ i = \overline{1,n} \ , \ j = \overline{1,n} \ , \qquad (4\text{-}5)$$

которая учитывает равенство элементов транспонированной и исходной матриц.

Пример 52. Разработать алгоритм и программу ввода прямоугольной матрицы $A_{m\times n}$ и перемножения транспонированной матрицы $A^{\mathrm{T}}_{n\times m}$ с исходной по формуле (4-5). Вывести на экран элементы результирующей матрицы в форме квадратной таблицы.

Структурограмма

Ввод количества строк m и столбцов n матрицы A; Ввод элементов матрицы A
Для i от 1 до n делать
Для j от 1 до n делать
$ar_{ij} = 0$;
Для k от 1 до m делать
$ar_{ij} = ar_{ij} + a_{ki} * a_{kj}$;
Вывод матрицы AR

```
PROGRAM PR52;
CONST  NM = 20;
VAR    I, J, K, N, M: INTEGER; AR, A: ARRAY[1..NM, 1..NM] OF REAL;
BEGIN
WRITELN('Введите M, N'); READLN(M, N);
WRITELN('Введите матрицу A');
FOR    I := 1 TO M DO  FOR J := 1 TO N DO  READ(A[I, J] );
{Вычисление матрицы AR по формуле (4-5)}
FOR I := 1 TO N
DO   FOR J := 1 TO N
     DO  BEGIN
           AR[I, J] := 0;
           FOR K := 1 TO M DO AR[I, J] := AR[I, J] + A [K, I]*A[K, J]
           END;
WRITELN('Результирующая матрица AR:');
FOR I := 1 TO N
DO FOR J := 1 TO N
   DO  BEGIN
         WRITE(AR[I, J]:8:2, ' ');
         IF J = N THEN WRITELN
         END
END.
```

4.3. Обращение квадратной матрицы

Для всякой невырожденной матрицы $A_{n\times n}$ может быть найдена обратная матрица $A^{-1}{}_{n\times n}$ такая, что

$$A^{-1}_{n\times n} \cdot A_{n\times n} = A_{n\times n} \cdot A^{-1}_{n\times n} = E_{n\times n},$$

где $E_{n\times n}$ - единичная матрица с элементами $\mathbf{e}_{ij}$

$$e_{ij} = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases} \qquad i, j = \overline{1,n}.$$

Для вычисления элементов обратной матрицы используются вычислительные формулы Жордановых преобразований, один шаг которых выполняется по формулам:

$$ap_{kj} = -\frac{a_{kj}}{a_{kk}} \qquad j \neq k,\ j = \overline{1,n} \tag{4-6}$$

$$ap_{ik} = \frac{a_{ik}}{a_{kk}} \qquad i \neq k,\ i = \overline{1,n} \tag{4-7}$$

$$ap_{ij} = a_{ij} - \frac{a_{ik} \cdot a_{kj}}{a_{kk}} \quad i \neq k,\ j \neq k,\ i = \overline{1,n},\ j = \overline{1,n} \tag{4-8}$$

$$ap_{kk} = \frac{1}{a_{kk}} \tag{4-9}$$

Формулы (4-6, 4-7, 4-8, 4-9) соответствуют преобразованию элементов a_{ij} исходной матрицы относительно разрешающего элемента a_{kk}, расположенного на пересечении разрешающей строки и разрешающего столбца. Номера строки и столбца совпадают, поэтому разрешающий элемент располагается на главной диагонали. Первый шаг преобразования (k = 1) выполняется относительно разрешающего элемента a_{11}, второй шаг преобразования (k = 2) выполняется относительно элемента a_{22} над матрицей, полученной в результате первого преобразования. Каждый k-ый шаг преобразования выполняется относительно элемента a_{kk} над матрицей, полученной в результате (k-1)-ого преобразования. Всего выполняется **n** шагов преобразования ($k = \overline{1,n}$).

Пример 53. Разработать алгоритм и программу обращения квадратной матрицы $A_{n\times n}$ и проверить обращение посредством умножения обратной матрицы на исходную.

Алгоритм и программа, приводимые в данном примере, предполагают ввод элементов квадратной матрицы, вычисление элементов обратной матрицы, умножение обратной матрицы на исходную, вывод обратной матрицы, вывод единичной матрицы.

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
		Количество строк (столбцов) матрицы, количество шагов преобразования.	integer
j	j	Текущий номер строки матрицы.	integer
j	j	Текущий номер столбца матрицы.	integer
k	k	Текущий номер шага преобразования, номер разрешающей строки, номер разрешающего столбца.	integer
a_{ij}	a[i, j]	Элемент i-ой строки, j-ого столбца матрицы a, элемент двумерного массива a.	real
a_{kk}	a[k, k]	Разрешающий элемент матрицы, элемент двумерного массива a.	real

a_{kj}	a[k, j]	Элемент разрешающей строки.	real
a_{ik}	a[i, k]	Элемент разрешающего столбца.	real
e_{ij}	e[i, j]	Элемент единичной матрицы e, элемент двумерного массива e.	real
ap_{kj}	ap[k, j]	Элемент k-ой строки преобразований матрицы, элемент двумерного массива ap.	real
ap_{ik}	ap[i, k]	Элемент k-ого столбца преобразований матрицы, элемент двумерного массива ap.	real
ap_{ij}	ap[i, j]	Элемент i-ой строки, j-ого столбца преобразований матрицы, элемент двумерного массива ap.	real
e	e	Текущий номер элемента строки и столбца перемножаемых матриц.	integer
ais_{ij}	ais[i, j]	Элемент i-ой строки, j-ого столбца исходной матрицы, элемент двумерного массива ais.	real

Структурограмма

Ввод размерности n матрицы A

Ввод исходной матрицы ais; a = ais;

Для k от 1 до n с шагом 1 делать: {Вычисление обратной матрицы AP}

Для j от 1 до n с шагом 1 делать:

$j \neq k$

True

ap[k, j]:= - a[k, j] / a[k, k]; {Формула (4-6)}

Для i от 1 до n с шагом 1 делать:

$i \neq k$

True

ap[i, k]:= a[i, k] / a[k, k]; {Формула (4-7)}

Для i от 1 до n с шагом 1 делать:

Для j от 1 до n с шагом 1 делать:

$i \neq k$ и $j \neq k$

True

ap[i, j]:= a[i, j] - a[i, k] * a[k, j] / a[k, k]; {Формула (4-8)}

ap[k, k]:= 1/ a [k, k]; {Формула (4-9)}

a : = ap;

Для i от 1 до n с шагом 1 делать: {Вычисление единичной матрицы E}

Для j от 1 до n с шагом 1 делать:

e [i, j] = 0;

Для l от 1 до n с шагом 1 делать:

e[i, j] := e[i, j] + ap[i, l] * ais[l, j];

Вывод обратной матрицы ap;

Вывод единичной матрицы E.

```
PROGRAM PR53;
CONST NM = 20;
VAR             I, J, K, N, L: INTEGER;
                E, A, AIS, AP: ARRAY[1..NM, 1..NM] OF REAL;
BEGIN
WRITELN('Введите N');
READLN(N);
WRITELN('Введите исходную матрицу AIS:');
FOR    I := 1 TO N
DO     FOR J := 1 TO N
       DO READ(AIS[I, J]);
A := AIS;
FOR K := 1 TO N
DO BEGIN
    FOR J := 1 TO N
    DO IF (J <> K)
        THEN AP[K, J] := -A[K, J]/A[K, K];                    {Формула (4-6)}
    FOR I := 1 TO N
    DO IF (I <> K)
        THEN AP[I, K]:=A[I, K]/A[K, K];                       {Фор-
мула (4-7)}
    FOR I := 1 TO N
    DO FOR J := 1 TO N
        DO IF (I <> K) AND (J <> K)
            THEN AP[I, J] := A[I, J]-A[I, K]*A[K, J]/A[K, K]; {Формула (4-8)}
    AP[K, K] := 1/A[K, K];                                    { Фор-
мула (4-9)}
    A := AP;
    END;
FOR I := 1 TO N          {Умножение исходной и обратной матриц}
DO FOR J := 1 TO N
    DO BEGIN E[I, J] := 0;
        FOR L := 1 TO N
        DO  E[I, J] := E[I, J] + AP[I, L]*AIS[L, J]
        END;
WRITELN('Обратная матрица AP:');
FOR I := 1 TO N
DO BEGIN
    FOR J := 1 TO N
    DO WRITE(AP[I, J]:5:2,' ');
    WRITELN
    END;
WRITELN('Единичная матрица E:');
FOR I := 1 TO N
DO BEGIN
    FOR J := 1 TO N
    DO WRITE(E[I, J]:5:2,' ');
    WRITELN
    END
END.
```

5. ПОДПРОГРАММЫ, ОПРЕДЕЛЕННЫЕ ПОЛЬЗОВАТЕЛЕМ

Технология программирования с использованием подпрограмм предусматривает разбиение программ на логически связанные, но функционально-замкнутые, компоненты, имеющие свои имена, что дает следующие преимущества:

- появляется возможность поблочной отладки больших программ, возможно, создаваемых несколькими программистами, с последующим объединением отлаженных подпрограмм в единое целое;
- экономится оперативная память, так как многократно используемые компоненты (подпрограммы) заносятся в память ЭВМ один раз;
- облегчаются изменения программы, так как изменение одной программы не вызывает корректировку других.

Подпрограммы могут использоваться пятью способами:

- основная программа и подпрограммы располагаются в одном файле (программном, а после трансляции - в объектном модуле);
- тексты подпрограмм расположены в различных файлах и подключаются директивами компилятора;
- подпрограммы организуются как оверлейные структуры и поочередно загружаются на одно и тоже место в оперативной памяти ЭВМ;
- подпрограммы пишутся на другом языке программирования и подключаются одним из вышеописанных способов;
- подпрограммы оформляются как внешние и вызываются из основной программы.

В языке Паскаль подпрограммы реализуются в виде процедур или функций.

5.1. Описание функций

Функцией называется часть программы, имеющая уникальное имя, предназначенная для решения определенной задачи и возвращающая в точку вызова скалярное значение как значение имени этой функции.

Функция, определенная пользователем, содержит заголовок и тело. Заголовок функции имеет вид:

FUNCTION <имя функции> (<список параметров>): <тип результата>;

где FUNCTION - служебное слово; *имя функции* - идентификатор функции; *список параметров* - перечень формальных параметров (то есть исходных данных) с указанием их типов; *тип результата* - данное скалярного типа, значение которого должно приобрести имя функции. На рис. 14 представлена в виде "Черного ящика" функция, имеющая список входных параметров и использующая переменные, описанные на более высоком уровне.

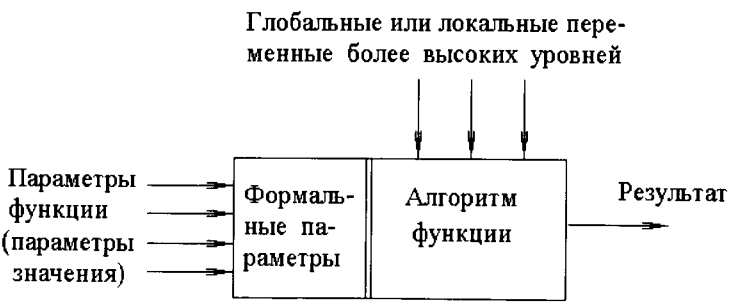


Рис. 14. Структура функции, имеющей входные параметры

Выходной параметр у функции всегда один, его называют результатом. Как правило, это данное любого скалярного типа или строка типа STRING.

Допускается описание функции и без формальных параметров, Примерами встроенных функций без входных параметров являются: PI (возвращает значение числа π) и RANDOM (возвращает случайное вещественное число в интервале от 0 до 1). Заголовок функции без параметров имеет вид:

FUNCTION <имя> : <тип результата>;

Такая функция может использовать глобальные или иные переменные более высокого уровня, а может обходиться и без них. Пример такой функции приведен на рис. 15.

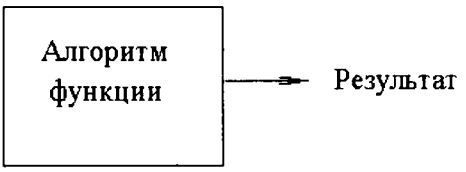


Рис. 15. Структура функции без входных параметров

Тело функции аналогично по структуре основной программе. Функция содержит те же самые разделы описаний, что и основная программа, и раздел операторов.

Примеры применения арифметических функций

Пример 54. Вычислить выражение:

$$Y = \frac{A^{X+1} + \left| X^2 + 1 \right|^B \cdot Cos\left(\sqrt[3]{\frac{X}{2} + \frac{\pi}{8}} \right)}{\sqrt{(X^2 + B^2)^3}} \qquad (5\text{-}1)$$

В формуле (5-1) многократно используются операции возведения в степень и извлечения корней, которые реализуются с помощью показательной функции. Такая функция не входит в список встроенных функций языка Паскаль. Поэтому для решения этой задачи предлагается создать пользовательскую, показательную функцию PW(Z, W), где Z - основание, а W - показатель степени. Эта функция реализует вычисление по формуле Z^W.

```
PROGRAM PR54;
VAR  X, Y, A, B: REAL;
FUNCTION PW(Z, W: REAL): REAL;
BEGIN
PW := EXP(Z * LN(W))
END;
BEGIN
WRITELN('Введите значения X, A, B'); READ(X, A, B);
Y := (PW(A, X+1) + PW(ABS(X*X + 1), B) * COS(PW(X/2, 1/3) + PI/8))
        / PW(X*X + B*B, 1.5);
WRITELN('Y= ', Y)
END.
```

На этом примере хорошо виден важнейший принцип использования функций, а именно, работа механизма передачи в тело функции значений параметров для осуществления расчетов. При вычислении Y функция PW вызывается четыре раза. Первый раз фактическими параметрами являются A и X+1. При вызове функции PW происходит подстановка Z = A, W = X+1. Теперь, когда формальные параметры Z и W определены, управление передается операторам тела функции PW. По завершении расчетов результат (вещественное число) помещается в переменную с именем PW, то есть становится доступным в исходном выражении Y, откуда и был произведен вызов. При втором вызове функции PW опять происходит вычисление и подстановка значений фактических параметров во входные переменные функции (называемые формальными параметрами) Z = ABS(X*X + 1) и W = B и т.д.

Пример 55. Вычислить выражение:

$$Z = F(A \cdot B, A + B, |A - B|) - 2.5 \cdot F(Sin(|A| + |B|), A \cdot B, 1.5 \cdot B), \quad (5\text{-}2)$$

где

$$F(X,Y,Z) = \begin{cases} X \cdot Y \cdot Z, & X \cdot Y \cdot Z > 0, \\ X + Y + Z, & X \cdot Y \cdot Z < 0, \\ X^2 + Y^2 + Z^2, & X \cdot Y \cdot Z = 0. \end{cases} \quad (5\text{-}3)$$

В этом примере непосредственно из самой постановки задачи видна необходимость в программировании пользовательской функции F, имеющей три формальных параметра X, Y, Z. Из математических формул (5-2, 5-3), подлежащих вычислению, очевиден тип функции F и величин X, Y, Z - это вещественные переменные.

```
PROGRAM PR55;
VAR  A, B, Z: REAL;
FUNCTION  F(X, Y, Z: REAL): REAL;
VAR C: REAL;
BEGIN
C := X * Y * Z; F := C;
IF C < 0 THEN F := X +Y + Z;
IF C = 0 THEN F := X * X +Y * Y + Z * Z
END;
BEGIN
WRITELN('Введите значения A, B'); READ(A, B);
Z := F(A*B, A + A, ABS(A-B))-2.5*F(SIN(ABS(A) + ABS(B)), A*B, 1.5*B);
WRITELN('Z = ', Z: 12: 6)
END.
```

При программировании функции можно обратить внимание на то, что выражение $X \cdot Y \cdot Z$ встречается четыре раза. Обычно в таком случае вводят рабочую переменную, в настоящем примере C = $X \cdot Y \cdot Z$. Эта переменная используется только в теле функции, а в основной программе не нужна. Поэтому переменная C объявляется в разделе VAR функции и поэтому доступна только из операторов этой функции. Такая переменная называется *локальной* переменной. Локальная переменная существует только во время выполнения функции и недоступна из тела основной программы. Переменные, объявленные в разделе описания VAR основной программы, называются *глобальными* они доступны из основной программы и из всех функций этой программы. В данном примере таких пе-

ременных три A, B, Z. Если бы была такая необходимость, то в теле функции можно было бы прочитать значение этих переменных и даже присвоить им новые значения, что, впрочем, делать не рекомендуется из соображений трудностей, возникающих при отладке.

Обратные тригонометрические функции

В языке Паскаль из учебных целей реализована только функция арктангенса ARCTAN, остальные функции арксинус, арккосинус, арккотангенс - должны вычисляться по известным формулам, через арктангенс.

Пример 56. Вычислить выражение: $Y = \arcsin(2 - \sqrt{X^2 + A^2})$ по формуле (5-4).

$$\arcsin(X) = \begin{cases} \frac{\pi}{2}, & X = 1; \\ \operatorname{arctg}\left(\frac{X}{\sqrt{1 - X^2}}\right), & -1 < X < 1; \\ -\frac{\pi}{2}, & X = -1. \end{cases} \qquad (5\text{-}4)$$

Графики функций arcsin (X) и arctg (X) приведены на рис. 16.

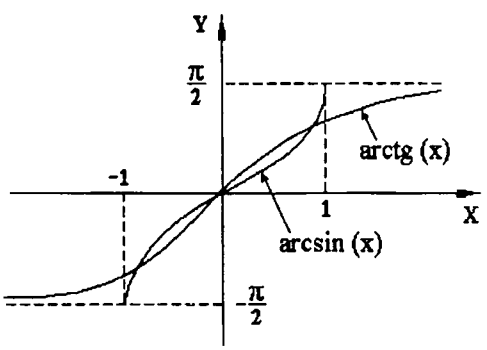


Рис. 16. График функции arcsin(X)

Из графика видно, что арксинус определен на интервале от -1 до 1. На границах интервала, в точках 1 и -1, арксинус равен $\frac{\pi}{2}$ и $-\frac{\pi}{2}$. Это предельные значения арктангенса при $X \to +\infty$ и $X \to -\infty$.

```
PROGRAM PR56;
VAR  K, A, X: REAL;
FUNCTION ARCSIN(Z: REAL): REAL;
BEGIN
IF Z = 1
THEN  ARCSIN := PI/2
ELSE  IF Z = -1
          THEN ARCSIN := -PI/2
          ELSE ARCSIN := ARCTAN(Z/SQRT(1 - Z*Z))
END;
```

```
BEGIN
WRITELN('Введите значения A, X'); READ(A, X);
K := 2 - SQRT(X*X + A*A);
IF ABS(K) <= 1
THEN WRITELN('Y = ', ARCSIN(K) :8 :6)
ELSE WRITELN('Аргумент за пределами ОДЗ арксинуса!')
END.
```

Пример 57. Вычислить выражение: $Y = \arccos(2 - \sqrt{X^2 + A^2})$ по формуле (5-5).

$$\arccos(X) = \begin{cases} \operatorname{arctg}\left(\dfrac{\sqrt{1-X^2}}{X}\right), & 0 < X \le 1; \\ \dfrac{\pi}{2}, & X = 0; \\ \pi - \operatorname{arctg}\left(\dfrac{\sqrt{1-X^2}}{X}\right), & -1 \le X < 0. \end{cases} \qquad (5\text{-}5)$$

Графики функций arccos(X) и arctg (X) приведены на рис. 17. Из графика видно, что арксинус определен на интервале от -1 до 1. В середине интервала, в точке 0, арккосинус равен $\frac{\pi}{2}$. Это предельное значения арктангенса при $X \to +\infty$ и второго арктангенса при $X \to -\infty$.

Очевидно утверждение: $arctg_2(x) \equiv \pi + arctg_1(x)$

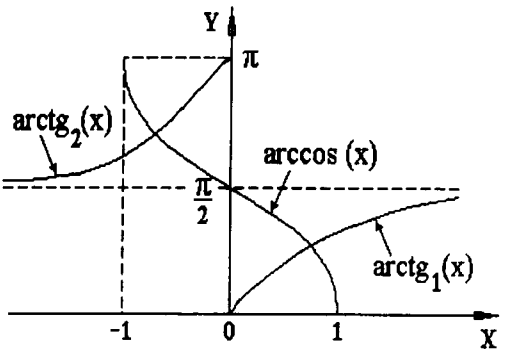


Рис.17. График функции arccos(X)

```
PROGRAM PR57;
VAR  K, A, X: REAL;
FUNCTION ARCCOS(Z: REAL): REAL;
BEGIN
IF Z = 0
THEN  ARCCOS := PI/2
ELSE  IF (Z >= -1) AND (Z < 0)
          THEN ARCCOS := PI - ARCTAN(SQRT(1 - Z*Z) / Z)
          ELSE ARCCOS := ARCTAN(SQRT(1 - Z*Z) / Z)
END;
```

```
BEGIN
WRITELN('Введите значения A, X'); READ(A, X);
K := 2 - SQRT(X*X + A*A);
IF ABS(K) <= 1
THEN WRITELN('Y = ', ARCCOS(K):8:6)
ELSE WRITELN('Аргумент за пределами ОДЗ арккосинуса!')
END.
```

Пример 58. Вычислить выражение: $Y = arcctg\,(X - \sqrt{X^2 + A^2})$ по формуле (5-6).

$$arcctg\,(X) = \begin{cases} \operatorname{arctg}\left(\dfrac{1}{X}\right), & 0 < X; \\ \dfrac{\pi}{2}, & X = 0; \\ \pi - \operatorname{arctg}\left(\dfrac{1}{X}\right), & X < 0. \end{cases} \qquad (5\text{-}6)$$

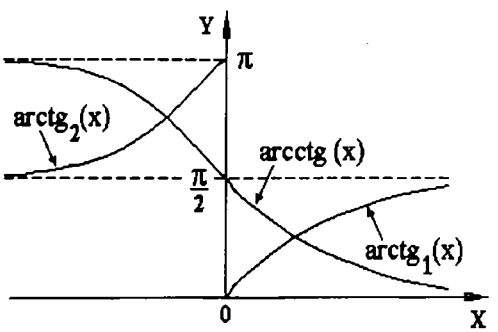


Рис. 18. График функции arcctg(X)

Графики функций arcctg(X) и arctg (X) приведены на рис. 18. Из графика видно, что арккотангенс определен на интервале от $-\infty$ до $+\infty$. В точке 0 арккотангенс равен $\dfrac{\pi}{2}$. Это предельное значения арктангенса при $X \to +\infty$ и второго арктангенса при $X \to -\infty$.

В программе вычисления арккотангенса областью допустимых значений (ОДЗ) X является любое вещественное число. Поэтому в теле основной программы проверку аргумента функции ARCCTG(X) делать нет необходимости, в отличие от двух предыдущих программ.

```
PROGRAM PR58;
VAR  K, A, X: REAL;
FUNCTION ARCCTG(Z: REAL) : REAL;
BEGIN
IF Z = 0
THEN  ARCCTG := PI/2
```

```
ELSE  IF Z < 0
        THEN ARCCTG := PI - ARCTAN(1/Z)
        ELSE ARCCTG := ARCTAN(1/Z)
END;
BEGIN
WRITELN('Введите значения A, X'); READ(A, X);
K := X - SQRT(X*X + A*A);
WRITELN('Y = ', ARCCTG(K):8:6)
END.
```

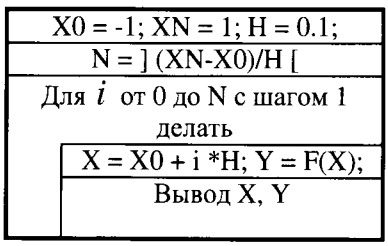
Использование функций в циклических процессах

Использование функций при решении многих задач упрощает алгоритмическую структуру программ и делает программы универсальными.

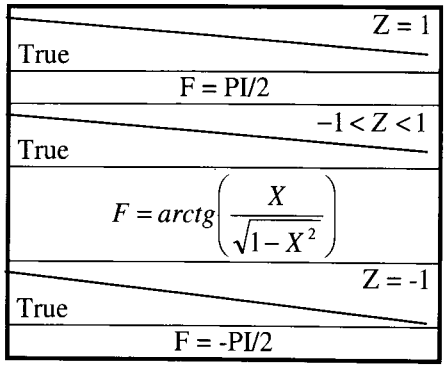
Пример 59. Табулировать функцию F(X) = arcsin(X) с шагом табуляции 0.1, заданную на промежутке [-1, 1].

Программа табуляции функции подробно описана выше, во втором разделе, примеры 18 и 19, а функция arcsin(X) описана в примере 53. Отличительная особенность этой программы в том, что предлагается функцию F(X) описать не в теле программы, а вынести в раздел описаний процедур и функций. После такого преобразования тело программы приобретает функциональную законченность. Тело программы становится универсальным, а особенности конкретной задачи табуляции учитываются при программировании пользовательской функции. Теперь программа содержит два алгоритмических блока: алгоритм функции и алгоритм программы. Эти блоки разнесены в коде программы и связаны механизмом вызова функции и, возможно, глобальными переменными. Поэтому для построения алгоритма необходимы две структурограммы. Одна структурограмма описывает тело основной программы, а другая структурограмма предназначена для описания пользовательской функции, вызываемой из тела программы. Ниже приведены две структурограммы, рассматриваемой задачи.

Структурограмма программы



Структурограмма функции F(X)




```
PROGRAM PR59;
CONST X0 = -1.0; XN = 1.0; H =
0.1;
VAR   I, N: INTEGER;  X, Y: REAL;
FUNCTION F(Z: REAL): REAL;
BEGIN
IF Z = 1 THEN F := PI/2;
```

```
IF (Z > -1) AND (Z < 1) THEN F := ARCTAN(Z/SQRT(1 - Z*Z));
IF Z = -1 THEN F := -PI/2
END;
BEGIN
N := TRUNC((XN - X0)/H);
FOR I := 0 TO N
DO BEGIN
    X := X0 + I *H; Y := F(X);
    WRITELN (X: 4: 1, ' ', Y: 9: 6)
    END
END.
```

Пример 60. Последовательность $\{A_n\}$ вычисляется по формуле (5-7).

$$A_n = \frac{1}{n} \cdot \sum_{i=1}^{n} Cos\left(\frac{B^{\lfloor\sqrt{i}\rfloor} \cdot X^n}{n} \right);\ 0 < X < 1. \qquad (5\text{-}7)$$

Найти предел последовательности, принимая за него такое A_n, при котором $|A_n - A_{n-1}| < \varepsilon$. Дополнительные условия: B = 1.3; n = 2, 3, 4, …

При решении подобных задач необходимо ответить на следующий вопрос: можно ли построить рекуррентную формулу $A_n = F(A_{n-1}, A_{n-2}, \cdots)$ для расчета величины A_n. Если такую формулу удалось найти, то построить программу удобнее по образцу примера 30. Для (5-7) рекуррентной формулы не существует, поэтому удобно использовать приведенный ниже, достаточно универсальный, алгоритм.

Поставить в соответствие величине A_n вещественную функцию $A(n)$ целочисленного аргумента n. Формула (5-8) для расчета значения A(n) тождественна (5-7).

$$A(n) = \frac{1}{n} \cdot \sum_{i=1}^{n} Cos\left(\frac{B^{\lfloor\sqrt{i}\rfloor} \cdot X^n}{n} \right);\ 0 < X < 1. \qquad (5\text{-}8)$$

Таблица имен

Математ. величина	Обозначение в программе	Содержательный смысл	Тип переменной
B	B	Константа B = 1.3	real
X	X	Параметр члена последовательности A_n	real
A_n	A(N)	Член последовательности $\{A_n\}$ с номером N, значение функции A(N)	real
ε	E	Погрешность приближения	real
-	M	Максимальный номер вычисляемого члена последовательности N ≤ M	integer
n	N	Номер члена последовательности $\{A_n\}$	integer

В алгоритме программы предусматривается дополнительное условие N ≤ M. Это условие предохранит данную программу от зацикливания. Значение M произвольное, но из разумных значений наиболее часто используется M = 1000.

Структурограмма программы

M = 1000; Ввод X и E; K = 2;
Пока $\lvert A(n) - A(n-1) \rvert > \varepsilon \wedge n < M$ делать:
K := K + 1
Вывод N, Y

Структурограмма функции A(n)

B = 1.3; S := 0;
Для I от 1 до N с шагом 1 делать:
$S = Cos\left(\frac{B^{\sqrt{i}} \cdot X^n}{n}\right) + S$;
A = S / N;

```
PROGRAM PR60;
CONST M = 1000;
VAR X, E: REAL; K: INTEGER;
FUNCTION A(N: INTEGER): REAL;
CONST B = 1.3;
VAR I: INTEGER; S: REAL;
BEGIN
S := 0;
FOR I := 1 TO N
DO S := S + COS(EXP(TRUNC(SQRT(I))*LN(B) + N *LN(X))/N);
A := S/N
END;
BEGIN
WRITELN('Введите X, E'); READ(X, E); K := 2;
WHILE (ABS(A(K) - A(K - 1)) > E) AND (K < M)
DO K := K + 1;
WRITELN('N = ', K: 3, ', A = ', A(K): 6: 4)
END.
```

В функции A(n) при расчете S использовано преобразование:

$$B^{\sqrt{i}} \cdot X^n = e^{\sqrt{i}\cdot Ln(B)} \cdot e^{n\cdot \mathrm{Ln(X)}} = e^{\sqrt{i}\cdot Ln(B) + n\cdot \mathrm{Ln(X)}} \qquad (5\text{-}9)$$

Тело функции A(n) включает разделы описаний CONST и VAR, в которых описаны локальная константа B и локальные переменные I, S. Они определены только в теле функции A(n), и их значения недоступны в теле основной программы. Глобальные переменные X, E описаны в разделе VAR основной программы, и их значения могут быть использованы в любом месте программы, в том числе и в пользовательской функции A(n).

Если функций несколько, то в них могут быть идентификаторы переменных и констант с одинаковыми наименованиями, но область их действия будет распределяться только на тот блок, где они описаны, поэтому их описания (типы данных) могут различаться. Если в функции локальная константа или переменная имеют тот же идентификатор, что и глобальная константа или переменная, то глобальная из этой функции становится невидимой (недоступной).

5.2. Передача имени объекта в качестве параметра функции

Формальным параметром функции может быть не только переменная скалярного типа, но и идентификатор переменной структурированного типа данных.

Использование массива в качестве параметра

Пример 61. Массивы X(k), Y(m), Z(n) представляют собой три выборки

значений случайной величины из общей генеральной совокупности. Требуется для каждой из трех выборок X, Y и Z вычислить их эмпирическую среднюю M и несмещенную дисперсию D.

Для решения этой задачи нужно запрограммировать две групповые пользовательские функции: Avg и VarD, которые описываются с помощью формул:

$$Avg(p,W)=\overline{W}=\frac{1}{p}\cdot\sum_{i=1}^{p}W_i,\quad VarD(p,W)=\frac{1}{p-1}\cdot\sum_{i=1}^{p}(W_i-\overline{W})^2,\ p>1\ . \qquad (5\text{-}10)$$

Функции Avg и VarD имеют в качестве параметра имя массива, содержащего значения случайной величины, и объем выборки.

Структурограмма функции Avg(p, W)

S = 0;
Для J от 1 до p с шагом 1 делать:
S = S + W[J];
Avg = S / p;

Структурограмма функции VarD(p, W)

M = Avg(p, W); S = 0;
Для J от 1 до p с шагом 1 делать:
S = S + (W[J] - M)^2;
VarD = S / (p-1);

Структурограмма программы

Вод размерности K массива X; Ввод массива X[1..K] с клавиатуры;
Вод размерности M массива Y; Ввод массива Y[1..M] с клавиатуры;
Вод размерности N массива Z; Ввод массива Z[1..N] с клавиатуры;
Вывод M(X), D(X), M(Y), D(Y), M(Z), D(Z).

Для описания этого примера понадобилось целых три структурограммы, однако, они просты, наглядны, компактны и легки для понимания. Для составления текста программы можно сделать предположение (5-11), что объемы выборок не превосходят целого числа G:

$$G = Sup\{K, M, N\}. \qquad (5\text{-}11)$$

Для передачи имени объекта в качестве параметра этот объект нужно в явном виде описать в разделе TYPE. В качестве параметра передается имя одномерного массива, поэтому идентификатор VEK присваивается этому типу.

```
PROGRAM PR61;
CONST G = 50;
TYPE        VEK = ARRAY [1 .. G] OF REAL;
VAR         X, Y, Z: VEK; I, K, M, N: INTEGER;
FUNCTION AVG(P: INTEGER; W: VEK): REAL;
VAR J: INTEGER; S: REAL;
BEGIN
S := 0;
FOR J := 1 TO P DO S := S + W[J];
AVG := S/P
END;
FUNCTION VARD(P: INTEGER; W: VEK): REAL;
VAR J: INTEGER; M, S: REAL;
```

```
BEGIN
M := AVG(P, W); S := 0;
FOR J := 1 TO P DO S := S + SQR(W[J] - M);
VARD := S/(P - 1)
END;
BEGIN
WRITELN('Укажите размерность K массива X '); READLN(K);
WRITELN('Введите массив X, из ', K:2, ' вещественных чисел');
FOR I := 1 TO K DO READ(X[I]);
WRITELN('Укажите размерность M массива Y '); READLN(M);
WRITELN('Введите массив Y, из ', M:2, ' вещественных чисел');
FOR I := 1 TO M DO READ(Y[I]);
WRITELN('Укажите размерность N массива Z '); READLN(N);
WRITELN('Введите массив Z, из ', N:2, ' вещественных чисел');
FOR I := 1 TO N DO READ(Z[I]);
WRITELN('M(X) = ', AVG(K, X) :10 :6, ', D(X) = ', VARD(K, X) :10 :6);
WRITELN('M(Y) = ', AVG(M, Y) :10 :6, ', D(Y) = ', VARD(M, Y) :10 :6);
WRITELN('M(Z) = ', AVG(N, Z) :10 :6, ', D(Z) = ', VARD(N, Z) :10 :6)
END.
```

Пример 62. Даны три вещественных массива X(k), Y(m), Z(n). Вычислить выражение (5-12):

$$S = \frac{\left| Max(X) + Max(Y) + Max(Z) \right|}{\left| Min(X) + Min(Y) + Min(Z) \right|} \qquad (5\text{-}12)$$

По аналогии с предыдущей программой нетрудно заметить, что в этой задаче можно использовать две групповые функции: Max(p, W) и Min(p, W). Эти функции возвращают численные значения наибольшего и наименьшего элементов массива W объемом p. Задача нахождения минимального и максимального элементов одномерного массива была подробно описана ранее в примере 38. Структура программы очень похожа на предыдущий пример 61, поэтому ограничимся только текстом программы.

```
PROGRAM PR62;
CONST G = 50;
TYPE        VEK = ARRAY [1 .. G] OF REAL;
VAR         X, Y, Z: VEK; I, K, M, N: INTEGER; S: REAL;
FUNCTION MAX(P: INTEGER; VAR W: VEK): REAL;
VAR J: INTEGER; S: REAL;
BEGIN
S := W[1];
FOR J := 2 TO P
DO IF S < W[J] THEN S := W[J];
MAX := S
END;
FUNCTION MIN(P: INTEGER; W: VEK): REAL;
VAR J: INTEGER; S: REAL;
BEGIN
S := W[1];
FOR J := 2 TO P
DO IF S > W[J] THEN S := W[J];
```

```
MIN := S
END;
BEGIN
WRITELN('Укажите размерность K массива X '); READLN(K);
WRITELN('Введите массив X, из ', K:2, ' вещественных чисел');
FOR I := 1 TO K DO READ(X[I]);
WRITELN('Укажите размерность M массива Y '); READLN(M);
WRITELN('Введите массив Y, из ', M:2, ' вещественных чисел');
FOR I := 1 TO M DO READ(Y[I]);
WRITELN('Укажите размерность N массива Z '); READLN(N);
WRITELN('Введите массив Z, из ', N:2, ' вещественных чисел');
FOR I := 1 TO N DO READ(Z[I]);
S := ABS(MIN(K, X) + MIN(M, Y) + MIN(N, Z));
IF S = 0
THEN WRITELN('Знаменатель равен 0 в формуле (5-7)')
ELSE WRITELN('S = ', ABS(MAX(K, X)+MAX(M, Y)+MAX(N, Z))/S :10 :6)
END.
```

Организация этой программы лучше предыдущей. В предыдущей программе массив копировался вместе с кодом функции в стек, где и происходило выполнение функции. В этой программе, объявив массив VAR W: VEK, как параметр-переменную, используется основной массив, не создавая его копию. Таким образом, происходит ускорение программы и снижение объема необходимой памяти для ее выполнения.

Одномерные массивы открытого типа. Функции LOW, HIGH

В версии Турбо-Паскаль 7.0 в качестве параметров переменных разрешено использовать одномерные массивы открытого типа, у которого не заданы размеры. Единственное ограничение состоит в том, что компоненты массива фактического параметра и массива формального параметра должны совпадать.

Значение границ индексов передаваемого массива можно найти с помощью двух стандартных функций: LOW(W) и HIGH(W), где W - тип массива. Целочисленная функция LOW(W) возвращает значение нижней границы значения индекса массива W. Целочисленная функция HIGH(W) возвращает значение верхней границы значения индекса массива W. При передаче имени *открытого массива* в качестве параметра переменной есть ограничение, нижний индекс всех массивов должен быть равен нулю.

Пример 63. Даны три вещественных массива X(k), Y(m), Z(n). Вычислить выражение $S = Max(X) + Max(Y) + Max(Z)$.

По аналогии с предыдущей программой нетрудно заметить, что в этой задаче следует использовать групповую функцию Max(W). Пусть эта функция имеет только один параметр - имя массива, и возвращает численные значения наибольшего из элементов массива W. Задача нахождения максимального элемента одномерного массива была подробно описана ранее в примерах 38 и 62. Отличительная черта этого примера состоит в том, что нижняя граница массива всегда 0, а верхняя граница не передается как второй параметр, а вычисляется с помощью функции HIGH(W). Структура программы очень похожа на предыдущий пример, поэтому нет необходимости ее строить, можно ограничиться только текстом программы.

```
PROGRAM PR63;
TYPE            VEK = ARRAY[0..30] OF REAL;
VAR             X, Y, Z: VEK; I, K, M, N: INTEGER; S: REAL;
FUNCTION  MAX(VAR W: ARRAY OF REAL): REAL; {Тип параметра W}
VAR J, P: INTEGER; S: REAL;          {открытый массив}
BEGIN
S := W[0]; P := HIGH(W);
IF P > 0                  {Массив W содержит более 1-го элемента}
THEN  FOR J := 1 TO P
        DO IF S < W[J] THEN S := W[J];
MAX := S
END;
BEGIN
WRITELN('Укажите размерность K массива X '); READLN(K);
WRITELN('Введите массив X, из ', K + 1: 2, ' вещественных чисел');
FOR I := 0 TO K DO READ(X[I]);
WRITELN('Укажите размерность M массива Y '); READLN(M);
WRITELN('Введите массив Y, из ', M + 1: 2, ' вещественных чисел');
FOR I := 0 TO M DO READ(Y[I]);
WRITELN('Укажите размерность N массива Z '); READLN(N);
WRITELN('Введите массив Z, из ', N + 1: 2, ' вещественных чисел');
FOR I := 0 TO N DO READ(Z[I]);
S := MAX(X) + MAX(Y) + MAX(Z);
WRITELN('S = ', S :10 :6)
END.
```

Использование функций в качестве параметра

Идентификатор функции или процедуры может использоваться в качестве параметра другой функции. Если такая необходимость появилась, то требуется соответствующим образом специфицировать каждый такой формальный параметр. Спецификацией в этом случае являются заголовок передаваемой функции, включающей имя функции, список формальных параметров и тип результата. Спецификация функции описывается в разделе TYPE следующим образом:

```
TYPE
    <имя спецификации> = FUNCTION(<список параметров>): <тип результата>;
```

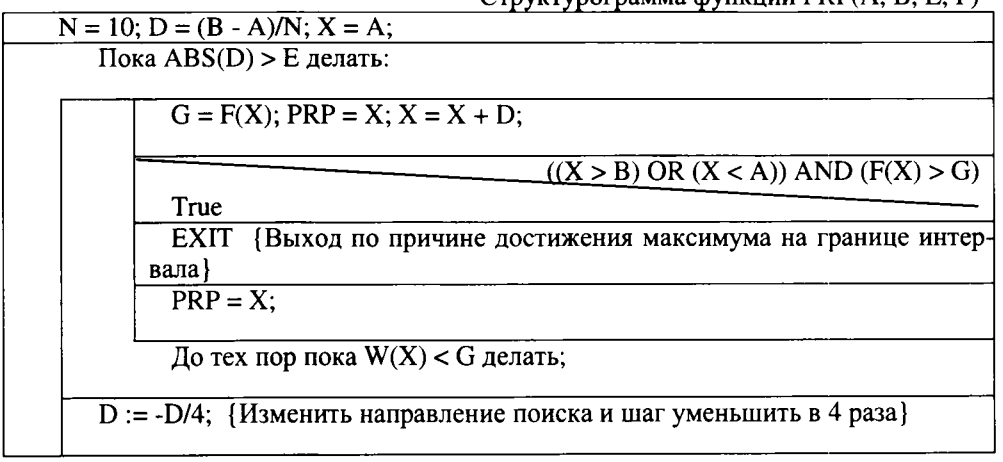
Пример 64. Найти максимум функции $F(X) = 0.1 \cdot X^3 - 2 \cdot X^2 + 10 \cdot X$ на интервале [A, B] методом поразрядного приближения (9).

Формульно-словесный алгоритм метода состоит в следующем:

1. Задается начальное приближение $X_0 = A$ слева от максимума F(X) и вычисляется $F(X_0)$. Задается D = h, где h = ΔX – начальный шаг поиска.
2. Предполагается, что $G = F(X_n)$, где вначале $F(X_n) = F(X_0)$, задается X = X + D и вычисляется $F(X_{n+1}) = F(X)$.
3. Проверяется условие $F(X_{n+1}) > G$; если оно выполняется, то осуществляется переход к п. 2, если нет - к п. 4.
4. Полагается, что D = - D/4. Проверяется условие | D | >E/4, где E - заданная погрешность вычисления X_n в точке максимума. Если она выполняется, то осуществляется переход к п. 2, то есть обеспечиваем поиск максимума в другом направлении с шагом в четыре раза меньше прежнего. Если данное условие выполняется, заканчивается счет, принимая за максимум X.

В этом известном алгоритме не учитывается тот факт, что в заданном интервале [A, B] нет экстремума (максимума). В этом случае максимум будет на границе интервала либо в точке A, либо в точке B. Учитывая простоту функции F(X) и основной программы, приведем структурограмму только функции PRP(A, B, E, F), которая осуществляет поиск максимума функции F на интервале [A, B], с погрешностью приближения E.

Структурограмма функции PRP(A, B, E, F)




```
PROGRAM PR64;
TYPE  FUN = FUNCTION(X: REAL):REAL; {Спецификация функции}
VAR               X, A, B, E: REAL; N: INTEGER;
{$F+}  {Директива. Компиляция функций с дальним типом обращения }
FUNCTION F(Z: REAL): REAL; {Исследуемая на экстремум функция}
BEGIN
F := ((0.1*Z-2)*Z+10)*Z
END;
FUNCTION  PRP(A, B, E: REAL; W: FUN): REAL;
CONST  N = 10;  { Число участков, на которые разбивается [A, B]}
VAR I: INTEGER; D, G, X: REAL;
BEGIN
D := (B - A)/N; X := A;              {D - шаг поиска}
WHILE ABS(D) > E
DO  BEGIN
     REPEAT
     G := W(X); PRP := X; X := X + D;
     IF ((X > B) OR (X < A)) AND (W(X) > G)    {Максимум на границе интервала}
     THEN EXIT;                      {Досрочное завершение функции}
     PRP := X;
     UNTIL  W(X) < G;
     D := -D/4
    END
END;
{$F-}
BEGIN
```

```
WRITELN('Укажите границы диапазона A, B'); READLN(A,B);
WRITELN('Задайте погрешность расчетов E'); READLN(E);
X := PRP(A, B, E, F);                    {Вычисление Xmax}
WRITELN('Xmax = ', X:11:9, ', F(Xmax) =', F(X):13:9);
END.
```

Задав значения [A, B] = [2, 3], E = 10^{-9}, получится X = 3,0; F(X) = 14.7. Очевидно, что экстремума на этом интервале нет, так как X_{max} на границе интервала в точке B. Изменив исходные данные: [A, B] = [2, 5], E = 10^{-9}, получится X_{max} = 3,333332149; F(X_{max}) = 14.814814815.

В теле функции использован оператор EXIT, который обеспечивает нормальный выход из функции в том случае, когда нет экстремума, и максимальное значение на границе интервала.

Замечание. Как правило, компилятор Турбо-Паскаля автоматически подбирает метод адресации подпрограмм. Если подпрограмма находится в одном тексте с телом основной программы, то она компилируется с "ближним" (near) адресом входа и возврата, содержащем только величину смещения адреса внутри текущего сегмента памяти. Если подпрограмма находится в другом программном модуле, то генерируется "дальний" (far) адрес, содержащий адрес сегмента и смещения в сегменте. При передаче имени функции в качестве программы необходим "дальний" адрес, но компилятор это не понимает. Поэтому необходимо использовать директиву компилятора {$F+}. Действие этой директивы распространяется на все подпрограммы, описанные ниже по тексту программы, или до директивы {$F-}.

Пример 65. Найти сумму $S = \sum_{i=m}^{n} U_i$, где $U_i = F(i)$. Функция $F(i)$ может быть различной: $F_1(i) = \frac{1}{i}$; $F_2(i) = \frac{i}{i+1}$; $F_3(i) = Cos\left(\frac{\pi}{i}\right)$.

При решении этой задачи следует помнить, что передача в качестве параметров предопределенных (стандартных) функций, к которым относится и Cos(X), запрещена. Это ограничение обходится путем переопределения стандартной функции в пользовательскую функцию.

```
PROGRAM PR65;
TYPE  FUN = FUNCTION(A: INTEGER): REAL;
VAR   S1: REAL; M, N, K: INTEGER;
{$F+}
FUNCTION  F1(I: INTEGER): REAL;
BEGIN  F1 := 1/I          END;
FUNCTION  F2(I: INTEGER): REAL;
BEGIN  F2 := I/(I + 1)    END;
FUNCTION  F3(I: INTEGER): REAL;     { Переопределение функции COS }
BEGIN  F3 := COS(PI/I)  END;
FUNCTION SUM(F : FUN): REAL;
VAR I: INTEGER; S: REAL;
BEGIN
S := 0;
FOR I := M TO N
DO S := S + F(I);
```

```
SUM := S
END;
{$F-}
BEGIN WRITELN('Задайте M, N и номер функции (1, 2, 3)'); READ(M, N, K);
CASE K OF
1: S1:= SUM(F1);
2: S1:= SUM(F2);
3: S1:= SUM(F3)
END;
WRITELN('S = ', S1:8:5)
END.
```

Параметры M и N передаются в функцию SUM в качестве глобальных переменных. Фактическим параметром для этой функции является идентификатор одной из трех функций F1, F2, F3.

5.3. Рекурсивные вычислительные процессы

Функция может вызывать другую функцию, та, в свою очередь третью и т.д. В результате программы приобретают иерархическую структуру.

Простая рекурсия

В Паскале допускается, чтобы функция вызывала саму себя. Такой способ вызова называется простой РЕКУРСИЕЙ. Классическим примером использования рекурсии является вычисление факториала целого положительного числа - n!.

Пример 66. Найти факториал N!, используя рекурсивную функцию FACT.

Вычисление факториала осуществляется по рекуррентной формуле (5-13).

$$n! = \begin{cases} 1, & n = 0; \\ 1, & n = 1; \\ n \cdot (n-1)!, & n > 1. \end{cases} \qquad (5\text{-}13)$$

В этой программе рекурсивный процесс с каждым шагом упрощает задачу, сводя n! с помощью рекуррентной формулы к (n-1)! и далее до 1!, который по определению равен единице. Это событие и является не рекурсивным решением, обеспечивающим завершение процесса вычисления факториала.

```
PROGRAM PR66;
VAR     N: REAL;
FUNCTION  FACT(A: REAL): REAL;
BEGIN
IF (A = 0) OR (A = 1) THEN FACT := 1
ELSE  FACT := A * FACT(A - 1)          {Рекуррентная формула}
END;
BEGIN
WRITELN('Введите число N');
READLN (N);
WRITELN(N: 3:0, '!=', FACT(N): 7: 0)
END.
```

С помощью графических алгоритмов, в том числе и структурограммы,

работу рекурсивной функции объяснить невозможно по той причине, что в оперативной памяти возникает и параллельно существует семейство копий такой функции с разными исходными данными. Пусть пример 66 решается для случая, когда n=5 (5!). Временная диаграмма вычислительного процесса представлена на рис. 19.

При вводе параметра N = 5 в момент времени T_1 происходит вызов рекурсивной функции FACT(5) из тела основной программы. Управление передается функции. В соответствии с рекуррентной формулой в теле рекурсивной функции при вызове функции выполняется оператор FACT := A * FACT(A - 1) для A = 5, то есть вызывается функция FACT(4). Это происходит в момент времени T_2. В оперативной памяти создается еще один образ функции FACT и в качестве входного параметра ей дается значение A = 4. С момента времени T_2 по T_9 функция FACT(5) находится в *пассивном состоянии*, то есть существует, но не работает. С момента времени T_2 по T_3 выполняются операторы функции FACT(4).

Происходит анализ параметра A, и поскольку A = 4, вызывается функция FACT(3) в момент времени T_3. В оперативной памяти создается еще один образ функции FACT, и в качестве входного параметра ей дается значение A = 3. С момента T_3 и до момента T_8 функция FACT(4) становится пассивной, поскольку управление она передала функции FACT(3). Функция FACT(3) вызывает функцию FACT(2), а та, в свою очередь, функцию FACT(1). Функция FACT(1) является последней в семействе, то есть она возвращает значение равное 1 в функцию FACT(2) в момент времени T_6. В этот момент код функции FACT(1) удаляется из оперативной памяти. Функция FACT(2) становится активной и вычисляет значение равное 2, и в момент времени T_7 передает управление FACT(3). Функция FACT(2) в момент времени T_7 удаляется из памяти. Этот процесс продолжается до момента времени T_{10}, когда рекурсивная функция завершает свою работу, возвращая вычисленное значение коду основной программы.

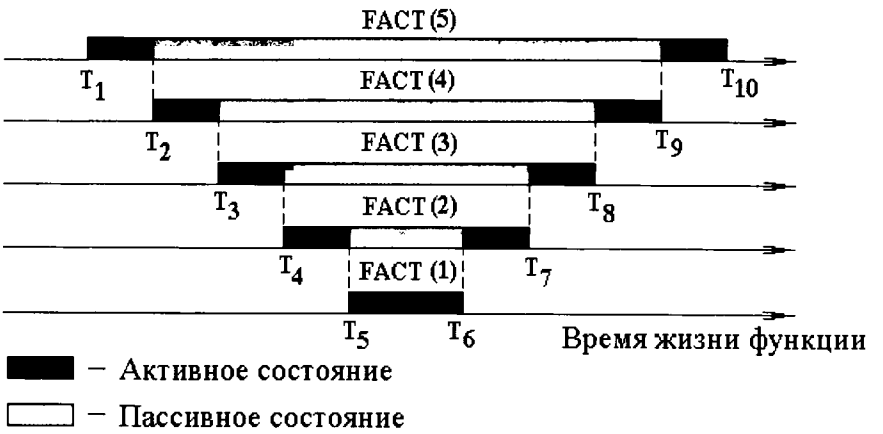


Рис. 19. Семейство копий для рекурсивной функции FACT(5)

Опыт изучения рекурсивных функций показывает, что, несмотря на приведенное выше подробное описание, программист все же плохо чувствует работу

рекурсивных программ, поскольку принцип их действия выходит за рамки интуитивного понимания процессов программирования. Следующая демонстрационная программа, которая называется музыкальный факториал, поможет прочувствовать работу рекурсивной программы. Суть музыкального факториала состоит в том, чтобы каждая ступень временной диаграммы звучала полсекунды с определенной частотой, отличной от соседней ступеньки на величину в 100 герц. В работе программы используются три процедуры SOUND, NOSOUND, DELAY стандартного модуля CRT (см. табл. 31).

Таблица 31

Процедура	Параметр	Содержание
SOUND(N)	Частота, в герцах.	Включить звуковой сигнал.
NOSOUND	Нет параметра.	Отключить звуковой сигнал.
DELAY(T)	Время в миллисекундах.	Ничего не делать T миллисекунд.

Процедуры SOUND и NOSOUND обеспечивают работу со встроенным в каждый компьютер динамиком, который генерирует звуковой сигнал в диапазоне от 37 до 32767 гц. Процедура DELAY использует таймер компьютера.

Пример 67. Демонстрационная программа "Музыкальный факториал".

```
PROGRAM PR67;
USES CRT;                {Подключение библиотеки CRT}
VAR     N: REAL;
FUNCTION  FACT(A: REAL): REAL;
BEGIN
SOUND(TRUNC(A + 1)*100); DELAY(500);
IF (A = 0) OR (A = 1)
THEN  FACT := 1
ELSE  FACT := A * FACT(A - 1);              {Рекуррентная формула}
SOUND(TRUNC(A + 1)*100); DELAY(500); NOSOUND
END;
BEGIN
WRITELN('Введите число N');
READLN (N);
WRITELN(N:3:0, '!=', FACT(N): 12: 0)
END.
```

Программирование рекуррентных формул с помощью рекурсивных функций

Большинство рекуррентных формул может быть запрограммировано с помощью рекурсивных функций.

Пример 68. Пользуясь рекурсивной функцией, для заданного N, вычислить Y_N, если известны Y_0, Y_1, Y_2, а $Y_i\,(i \geq 3)$ вычисляется по формуле $Y_i = Y_{i-1} + 2 \cdot Y_{i-2} - Y_{i-3}$.

Принцип построения рекурсивной функции состоит в следующем. Величине Y_N ставится в соответствие рекурсивная функция Y(N). В тело рекурсивной функции (см. текст программы PR67) встраивается оператор CASE N OF, который полностью обрабатывает все варианты N, начиная с 0, 1, 2. Этот список отражает глубину памяти рекуррентной формулы. В принципе, память может быть достаточно большой. Ветка ELSE содержит саму рекуррентную формулу. Если N > 2, то для этого случая (память 3 члена последовательности) идет понижение

N в ветке ELSE до тех пор, пока все ветки дерева вычислительного процесса не выйдут на списки 0, 1, 2 оператора CASE. На рис. 20.а приведено дерево вычислений для Y(5).

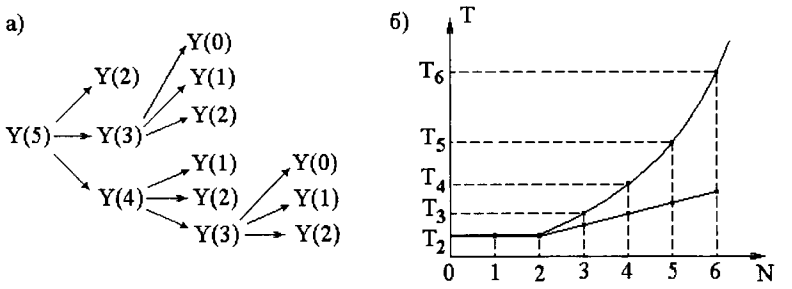


Рис. 20. Характеристики рекурсивного процесса:
а) дерево вычислений для Y(5), б) временные характеристики Y(6)

```
        PROGRAM PR68;
VAR Y0, Y1, Y2: REAL; N: INTEGER;
FUNCTION Y(K: INTEGER): REAL
BEGIN CASE K OF
                0: Y := Y0;
                1: Y := Y1;
                2: Y := Y2
                ELSE Y := Y(K-1) + 2*Y(K-2) - Y(K-3)
                END
END;
BEGIN
WRITELN('Введите Y0, Y1, Y2, N); READLN(Y0, Y1, Y2, N);
WRITELN('Yn =', Y(N):8:3)
END.
```

Анализируя дерево вычислений и структуру программы, можно заметить, что формула для оценки времени T вычисления величины Y_N будет иметь вид:

$$T(Y_N) \approx T_2 + 3 \cdot 2^{N-3} \cdot T_f, \quad N > 2, \tag{5-14}$$

где T_2 - время работы рекурсивной функции при $N \le 2$; T_F - время активного состояния рекурсивной функции при $N > 2$, включая время создания ее копии. Число 3 в формуле (5-14) соответствует глубине памяти рекуррентной формулы. У программируемой формулы глубина памяти равна 3. Таким образом время работы рекурсивной функции возрастает по экспоненциальному закону с ростом номера N члена рекуррентного ряда. Качественная оценка времени приведена на рис. 20.б. Величины T_k соответствуют временам вычисления с помощью рекурсивной функции членов последовательности Y_k. Линейная функция соответствует временным характеристикам программы, реализующей вычисления по рекуррентной формуле с помощью цикла FOR (см. пример 21).

Заключение. Программирование рекуррентных зависимостей с помощью рекурсивных функций требует существенных затрат по времени и памяти при N >> K, где N - номер члена последовательности, K - глубина памяти рекуррентной формулы. Если N близка к K, то имеет смысл рассмотреть возможность при-

менения рекурсивной функции.

Порядок описания подпрограмм. Косвенная рекурсия

При формировании раздела процедур и функций большое значение имеет порядок описания подпрограмм. Основной принцип состоит в том, что описание должно предшествовать вызову этой подпрограммы.

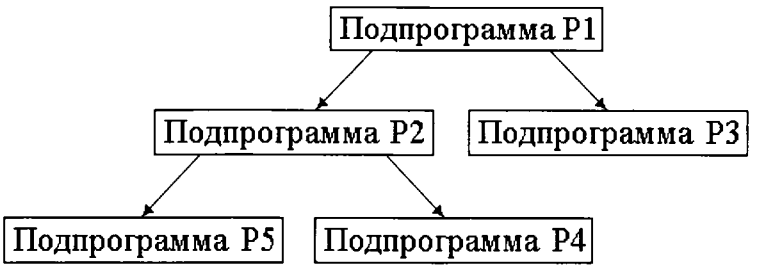


Рис. 21. Схемы вызовов подпрограмм

Для примера на рис. 21. представлена иерархическая (вложенная) структура взаимодействия пяти подпрограмм с условными идентификаторами P1, P2, P3, P4, P5. Стрелками показаны вызовы (передача управления) одной подпрограммой другой подпрограммы.

Корректное описание этих подпрограмм должно соблюдать следующую последовательность: ((P4 и P5), P2) и P3), P1. То есть к моменту описания подпрограммы P2 подпрограммы P4 и P5 должны быть уже описаны. Аналогично, к моменту описания подпрограммы P1 должны быть описаны подпрограммы P4, P5, P2 и P3. Подпрограммы P4 и P5 при описании можно менять местами. Можно также менять местами группы описаний подпрограмм (P4, P5, P2) и P3.

Однако не всегда удается соблюсти указанный принцип. Иногда (очень редко) встречается такой вариант рекурсии, когда одна подпрограмма вызывает другую, которая к моменту вызова еще не определена. Подобная ситуация представлена на рис. 22, где функция P1 обращается к процедуре P2, а та, в свою очередь, вызывает P3. Но P3 опять требует функцию P1. Такая структура взаимосвязи подпрограмм называется *косвенной рекурсией* (см. рис. 22).

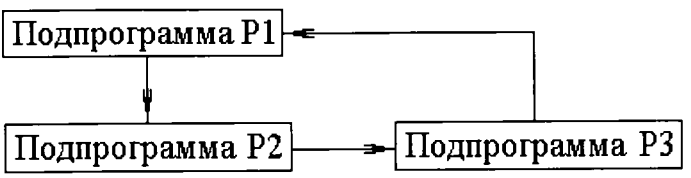


Рис. 22. Пример косвенной рекурсии

Для реализации косвенной рекурсии используется предварительное описание функции с помощью директивы FORWARD. В следующем фрагменте программы приведено описание структуры, изображенной на рис. 22.

Пример 69. Разработать структуру (скелет) программы, использующей косвенную рекурсию в соответствии с рис. 22.

```
PROGRAM PR69;
VAR  S1 : REAL; M, N, K: INTEGER;
FUNCTION P1(L: INTEGER): REAL; FORWARD;     {Предварит. объявление P1}
FUNCTION P3(I: INTEGER): REAL;                {Описание P3}
BEGIN   . . .
          S1:=P1(M); {Вызов функции P1, предварительно объявленной выше}
              . . .
END;
PROCEDURE P2(J: INTEGER);          {Описание P2}
BEGIN   . . .
        S1:=P3(N);     {Вызов функции P3}
         . . .
END;
FUNCTION P1;              {Описание P1, с учетом предварительного объявления}
BEGIN      . . .
             P2(K);  {Вызов процедуры P2}
           . . .
END;
```

5.4. Описание процедур

Процедуры являются основой модульного программирования (МП). Модульное программирование - это процесс построения программы, разделенной на логические части, называемые модулями, и последовательное программирование каждого модуля.

Принципы модульного программирования

Целью МП является: возможность независимого написания и отладки отдельного модуля; возможность замены модуля без изменения всей программы; облегчение тестирования (проверки) модулей программы.

Если соблюдается системный принцип проектирования задачи сверху вниз, то она разбивается на подзадачи, которые возможно отразить программными модулями. При этом преследуют две цели:

1. Нужно добиться, чтобы программный модуль был правильным и не зависел от контекста, в котором этот модуль будет использоваться.

2. Следует стремиться к тому, чтобы из модулей можно было формировать программу без каких-либо знаний о внутренней работе отдельно взятого модуля.

Для модуля должно быть известно: алгоритм решения задачи; область допустимых входных значений; область возможных выходных значений.

Каждый модуль должен решать одну задачу, то есть быть «кирпичиком», из которого строится программа. Как правило, использование модулей сокращает число глобальных переменных программы.

На рис. 23 в виде блок-схемы приведен проект будущей программы, имеющей линейную структуру.

Эта программа должна обеспечить ввод с клавиатуры двух матриц A и B, осуществить их умножение, результатом которого является матрица C, и вывести матрицу C на экран монитора. Тело программы содержит вызовы трех подпрограмм MOD1, MOD2, MOD3, имеющих один вход и один выход.

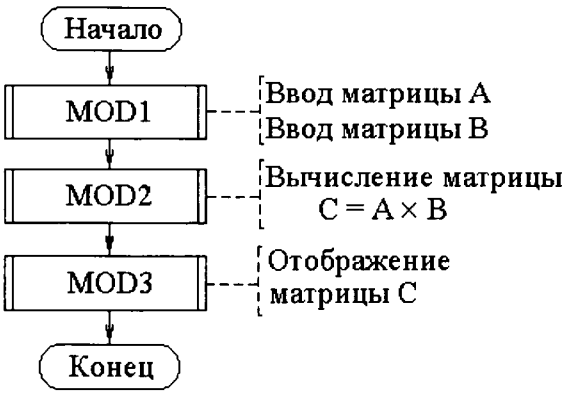


Рис. 23. Тело программы PR70

В таком представлении (рис. 23) процесс разработки программы сводится к последовательному программированию модулей, один за другим, если один разработчик, и параллельно всех трех, если разработчика три. Пусть MOD2 и MOD3 уже разработаны, а MOD1 - нет. В этом случае вместо MOD1 ставят «заглушку», то есть модуль имитирующий процесс работы MOD1.

Принципы модульного программирования реализуются в Паскале с помощью пользовательских процедур, представляющих собой оператор с уникальным идентификатором и списком формальных параметров, если они нужны.

Структура и синтаксис процедуры

Процедура пользователя представляет собой именованную группу операторов, реализующую часть общей задачи. Процедура отличается от функции способом вызова и возврата управления. Процедура работает с параметрами значениями (исходными данными) и параметрами переменными, которые используются для возвращения в качестве результатов переменных скалярного или структурированного типа. Схематично (в виде черного ящика) процедуру можно отобразить следующим образом (рис. 24.):

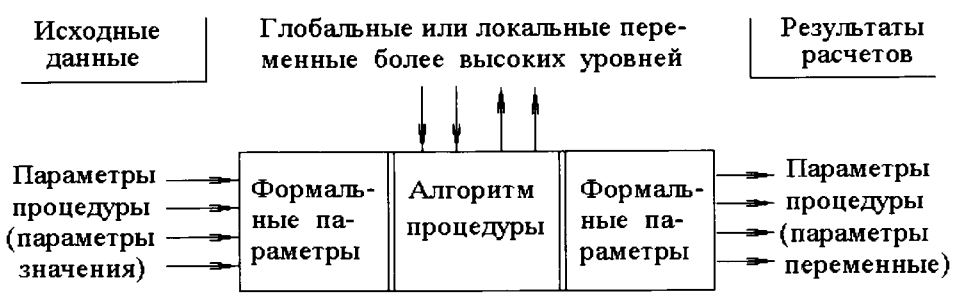


Рис. 24. Схема процедура

Допускаются описания процедур, выполняющих некоторые действия и не формирующих никаких результатов расчетов. Например: вывод на экран строки из звездочек или часто встречающихся сообщений.

Процедура, описанная пользователем, в общем случае имеет синтаксис:

```
PROCEDURE < имя > [(< список формальных параметров >)];   { Заголовок }
< Разделы описаний >                  { Тело процедуры }
BEGIN
< Раздел операторов >
END;
```

При обращении к процедуре указывается имя процедуры и список фактических параметров, порядок которых очень существенен, так как он должен строго соответствовать порядку, принятому в списке формальных параметров заголовка процедуры. Фактические параметры отделяются друг от друга запятыми и заключаются в круглые скобки. В общем случае список параметров процедур может отсутствовать.

Подстановка фактических параметров вместо формальных осуществляется механизмом замены, который обеспечивает многократное выполнение процедуры с различными исходными данными и помещает результаты вычислений в переменные, задаваемые пользователем.

Пример 70. Используя принцип модульного проектирования программ, разработать программу вычисления произведения двух матриц **C = A × B** по формуле (5-15):

$$C_{ij} = \sum_{k=1}^{p} A_{ik} \cdot B_{kj}\ ;\ \mathrm{i} = \overline{1, \mathrm{m}}\ ;\ \mathrm{j} = \overline{1, \mathrm{n}}\ . \qquad (5\text{-}15)$$

где A_{ik} - элементы прямоугольной матрицы $A_{m\times p}$; **m** - число строк матрицы $A_{m\times p}$; **p** - число столбцов $A_{m\times p}$ и число строк матрицы $B_{p\times n}$; B_{kj} - компоненты матрицы $B_{p\times n}$; **n** - число столбцов матрицы $B_{p\times n}$; C_{ij} - компоненты матрицы $C_{m\times n}$, получаемые в результате умножения.

Эта задача достаточно близка к условию задачи 51, и внешний (пользовательский) интерфейс этих задач один и тот же, а вот программная реализация отличается и соответствует модульной структуре, представленной на рис. 23.

```
PROGRAM PR70;
TYPE   MAT = ARRAY [1 .. 10, 1 .. 10] OF REAL;
VAR            AF, BF, CF: MAT; MF, NF, PF: INTEGER;
PROCEDURE   MOD1(CH: CHAR; VAR M, N: INTEGER; VAR W: MAT);
VAR I, J: INTEGER;
BEGIN { Ввод матрицы W, размерностью M на N }
WRITELN('Введите размерность M и N матрицы ', CH); READLN(M, N);
WRITELN('Введите матрицу ', CH);
FOR I := 1 TO M DO FOR J := 1 TO N DO READ (W[I, J]);
END; { MOD1 }
PROCEDURE   MOD2(M, P, N: INTEGER; A, B: MAT; VAR C: MAT);
VAR I, J, K: INTEGER;
BEGIN {Умножение матриц A и B}
FOR I := 1 TO M
DO      FOR J := 1 TO N
        DO      BEGIN
                C[I, J] := 0;
                FOR     K := 1 TO P
```

```
                DO      C[I, J] := C[I, J] + A[I, K]*B[K, J]
                END
END; { MOD2 }
PROCEDURE   MOD3(CH: CHAR; M, N: INTEGER; W: MAT);
VAR I, J: INTEGER;
BEGIN { Отображение матрицы W, размерностью M на N }
WRITELN('Матрица ', CH, ':');
FOR I := 1 TO M
DO      BEGIN
        FOR J := 1 TO N DO WRITE(W[I, J]: 10: 5);
        WRITELN
        END
END; { MOD3 }
BEGIN           {Тело программы}
MOD1('A', MF, PF, AF); { Ввод матрицы A, размерностью MF на PF }
MOD1('B', PF, NF, BF); { Ввод матрицы B, размерностью PF  на NF }
MOD2(MF, PF, NF, AF, BF, CF);           { Умножение матриц A и B }
MOD3('C', MF, NF, CF)  { Отображение матрицы C, размерностью MF на NF }
END.
```

Спецификация процедуры

Процедура представляет собой программу в миниатюре, являясь, в свою очередь, частью основной программы или другой процедуры. Синтаксис процедуры полностью повторяет синтаксис программы. Отличие состоит только в заголовках. Заголовок процедуры всегда начинается ключевым словом PROCEDURE, а программы - PROGRAM, которое, впрочем, писать необязательно. Процедуры могут быть простыми и маленькими, всего с десяток операторов, а могут быть очень сложными и большими - несколько сотен операторов. Процедуры могут содержать свои собственные процедуры и функции. Поэтому на процедуры, как и на программы, распространяется методология проектирования программных систем. Эта методология предусматривает разработку спецификации на каждую программную единицу, в том числе и на процедуру.

Спецификация - это точное, однозначное, недвусмысленное описание, написанное постановщиком задачи для программиста. Спецификация включает в себя концептуальное описание программной единицы, описание потоков данных, подробный алгоритм и, возможно, другие разделы, которые зависят от выбранного метода проектирования.

Построение спецификации для процедуры MOD1 программы PR70 (см. пример 70) производится следующим образом.

Назначение процедуры. Ввод с клавиатуры двумерного массива размерностью M × N.

Описание потоков данных. Существуют специальные диаграммы Варнье-Орра для описания потоков данных, которые применяются в информационных системах. Однако данный пример достаточно прост, поэтому можно ограничится табличным описанием данных, с которыми работает описываемая процедура. Эти данные сведены в табл. 32.

Параметры-переменные в списке формальных параметров процедуры отличаются от параметров-значений наличием ключевого слова VAR, расположенным перед перечнем переменных. Переменные I и J являются локальными для процедуры MOD1, поэтому в табл. 32 не представлены.

Таблица 32

№ п/п	Идентификатор	Данное	Тип переменной	Содержательный смысл
1	CH	Параметр-значение	CHAR	Наименование массива W: литера "A" или "B".
2	M	Параметр-переменная	INTEGER	Нижняя граница вводимого с клавиатуры массива W.
3	N	Параметр-переменная	INTEGER	Верхняя граница вводимого с клавиатуры массива W.
4	W	Параметр-переменная	ARRAY [1..10, 1..10] OF REAL	Значения элементов массива W.

Алгоритм. Алгоритм подпрограммы достаточно прост и описан ниже с помощью структурограммы.

Структурограмма процедуры MOD1(CH, M, N, W)

Ввод размерности M, N массива CH; Ввод элементов массива W;
Для I от 1 до M с шагом 1 делать:
Для J от 1 до N с шагом 1 делать:
Ввести с клавиатуры значение элемента массива W[I, J];

Спецификации процедур MOD2 и MOD3 аналогичны процедуре MOD1. Рекомендуется их составить самостоятельно.

Процедура вывода даты и времени. Процедуры GetTime и GetDate

Очень часто пользовательские процедуры и функции используют для программирования часто встречающихся действий или процессов, если при этом необходимы специальные знания. Например, существуют редко используемые системные процедуры и функции. Такие процедуры имеют достаточно "тяжелый" интерфейс, и их применение в прикладных программах, как правило, вызывает затруднение. Поэтому программист для себя один раз создаст процедуру, использующую эти системные подпрограммы, или перепишет ее из книги, или возьмет из другого источника. Эту процедуру теперь можно использовать во всех разработках, где она уместна. Для этого необходимо хранить текст и спецификацию своей процедуры. В качестве примера можно привести пользовательскую программу, содержащую две системные процедуры GetTime и GetDate модуля DOS.

Пример 71. Вывести на экран дисплея текущее время, день недели и дату.

```
PROGRAM PR71;
USES DOS;
VAR  TIME, DATE :STRING[8];          { Глобальные переменные }
       NAME_DAY :STRING[11]; D, T : INTEGER;
PROCEDURE DAT(K, L: INTEGER);
VAR    I : INTEGER;    { Локальные переменные процедуры DAT}
       HOUR, MIN, SEC, SEC100: WORD;
       YEAR, MONTH, DAY, DAYOFWEEK: WORD;
       GG, MM, DD, HH, MN, SS: STRING[2]; ST: STRING[4];
BEGIN          { Формирование даты }
GETDATE(YEAR, MONTH, DAY, DAYOFWEEK);
STR(YEAR:4, ST); GG := ST[3] + ST[4]; STR(MONTH:2, MM); STR(DAY:2, DD);
```

```
IF MONTH < 10 THEN MM[1] := '0'; IF DAY < 10 THEN DD[1] := '0';
CASE  K OF
1 : DATE := MM + '/' + DD + '/' + GG;     { Американский формат }
2 : DATE := DD + '.' + MM + '.' + GG      { Немецкий формат }
ELSE DATE := GG + '.' + MM + '.' + DD { Системный формат ANSI }
END; { CASE }
CASE  DAYOFWEEK OF                        { Формирование дня недели }
0 : NAME_DAY := 'Воскресенье';
1 : NAME_DAY := 'Понедельник';
2 : NAME_DAY := 'Вторник';
3 : NAME_DAY := 'Среда';
4 : NAME_DAY := 'Четверг';
5 : NAME_DAY := 'Пятница';
6 : NAME_DAY := 'Суббота'
END;    { Case }
        { Формирование времени суток }
GETTIME(HOUR, MIN, SEC, SEC100);
STR(HOUR:2,HH); STR(MIN:2,MN); STR(SEC:2,SS);
IF HOUR < 10 THEN HH[1] := '0';
IF MIN < 10 THEN MN[1] := '0';
IF SEC < 10 THEN SS[1] := '0';
CASE L OF
1 : TIME := HH + ':' + MN;          { Формат: <Часы : Минуты> }
2 : Time := MN + "" + SS + "";      { Формат: <Минуты ' Секунды "> }
ELSE Time := HH + ':' + MN + ':' + SS     { Формат: 'Часы: Минуты: Секунды' }
END     { Case }
END;                {Конец процедуры DAT}
BEGIN
WRITELN('Укажите формат даты: 1, 2, 3'); READLN (D);
WRITELN('Задайте формат времени: 1, 2, 3'); READLN (T);
DAT(D, T);        { Вызов процедуры }
WRITELN('Сейчас : ' + Time + ', '+ Name_Day + ', ' + Date);
END.
```

Переменные TIME, NAME_DAY, DATE используются для формирования текстовых строк Время, День недели, Дата. Эти переменные могут формироваться в разных форматах. Так, дата в привычном нам виде записывается дд.мм.гг (Немецкий формат). Во многих случаях пользователь ЭВМ встречается с датой мм/дд/гг - это "Американский формат". Разработчикам баз данных наиболее удобен "Системный формат" гг.мм.дд, который снимает проблему индексации, и сортировки данных по дате.

В этом примере предлагается универсальная процедура, позволяющая сформировать "Дату" и "Время" в нескольких форматах. Выбор формата осуществляется с помощью кодов (параметров-значений), передаваемых при вызове процедуры DAT(D, T). Кроме даты и времени в процедуре определяется день недели. Переменные TIME, NAME_DAY, DATE являются глобальными, то есть доступными и в основной программе и в процедуре DAT.

Размещение процедур и функций в оперативной памяти

В языке Паскаль используется два способа передачи параметров: по значению и ссылке (адресу переменной). Соответственно, различают параметры-значения и параметры-переменные.

При обращении в программе к процедуре или функции в оперативной памяти ЭВМ (рис. 25.) в области "Рекурсивного стека" создается "КОПИЯ" рабочих полей этой подпрограммы, содержащая всю локальную для этой подпрограммы информацию, необходимую для ее выполнения.

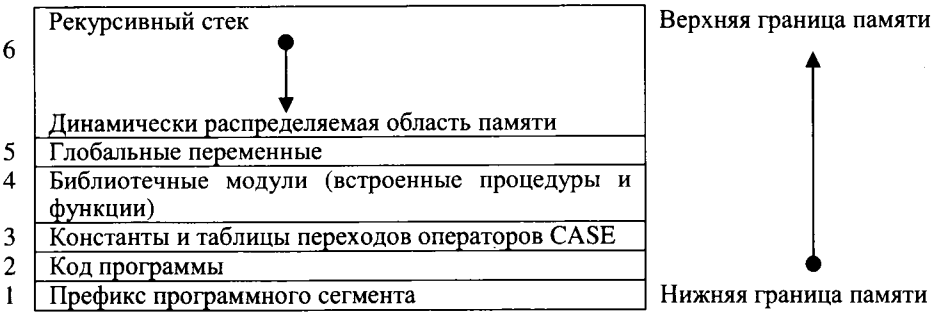


Рис. 25. Карта распределения оперативной памяти

Префикс программного сегмента строится операционной системой при загрузке .EXE модуля в оперативную память ЭВМ, занимает 256 байт и представляет собой совокупность данных, важных для выполнения программы под управлением DOS. Остальная память выделяется для программы на время ее выполнения. Рекурсивный стек размещается в памяти непосредственно перед верхней границей памяти и заполняется страницами по мере вызова процедур и функций при выполнении программы. Структура страницы представлена на рис. 26.

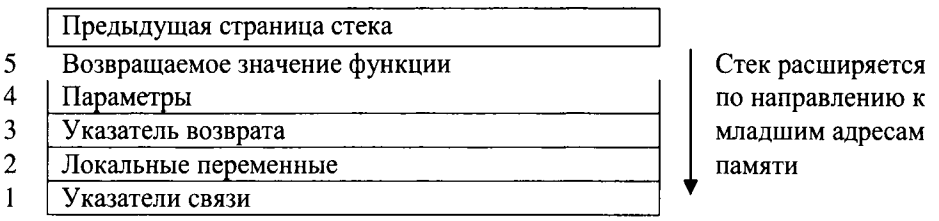


Рис. 26. Структура страницы стека

Поле 5 присутствует только в странице, отводимой при вызове функции. В это поле записывается результат (значение) функции, которое будет возвращено в тело вызывающего программного модуля. Параметры (поле 4) - это значения переменных, переданных в функцию или процедуру как параметры-значения. Для параметров-переменных здесь хранятся только адреса, а сами переменные находятся либо в поле 5 «Глобальные переменные» (рис. 26.), либо в поле 2 «Локальные переменные» одной из предыдущих страниц стека, если данная процедура (функция) была вызвана другой подпрограммой. Указатель возврата (адрес) обеспечивает выход из подпрограммы после ее завершения на оператор, следующий за оператором вызова подпрограммы. Если в подпрограмме используются переменные более высоких уровней вложения, то для связи с ними необходимо поле 1, содержащее адреса этих переменных.

Некоторые соображения по использованию подпрограмм

1. Именно такая реализация подпрограмм определяет то, что локальные переменные недоступны из более высоких уровней, так как они создаются только на время обращения к подпрограмме и уничтожаются при ее завершении (стирается

страница стека).

2. Величина параметра-значения хранится в странице стека только в процессе выполнения подпрограммы и стирается при ее завершении, а значение параметра-переменной хранится на более высоком уровне и меняется подпрограммой во время ее выполнения. Когда страница стека стирается, то уничтожается лишь адрес этой переменной, а значение сохраняется.

3. Все идентификаторы в подпрограмме (основной программе) должны быть уникальными. Если программист использует идентификаторы, которые совпадают с именами встроенных функций или процедур, то последние становятся недоступными в пределах области действия подпрограммы. Аналогично для переменных. Если программист вводит локальную переменную с именем уже описанной ранее глобальной переменной, то в данной подпрограмме эта глобальная переменная становится недоступной.

4. Динамическая организация выполнения подпрограмм делает возможным вызов подпрограммой самой себя с новыми значениями фактических параметров. Это называется рекурсией.

5. Вложенность описания подпрограмм не должна превышать 10. Это не относится к рекурсиям, то есть вложенности при выполнении, которая ограничивается только физическими размерами поля ОЗУ, отводимого под стек.

6. Использование подпрограмм приводит к существенным затратам времени, так как осуществляется формирование стека и связывание формальных и фактических параметров. Поэтому, если вызовов подпрограмм слишком много, то временные характеристики могут быть существенно хуже той же программы, но без подпрограмм.

ГЛАВА 3. МИРОВЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

1. Общие положения

Internet - всемирное объединение компьютерных сетей и отдельных компьютеров, созданное для совместного пользования информацией или обмена электронной почтой с помощью согласованных протоколов передачи данных. Internet - не единая аппаратная система, как, например, локальная сеть, а совокупность компьютеров и локальных сетей (ЛС), добровольно придерживающихся соглашений о совместном использовании Internet - протоколов, используемых в качестве Lingua Frank ("единого языка") для обмена информацией.

1.1. Этапы развития Internet

Идея создания Internet была предложена RAND Corporation USA в связи с необходимостью построения коммуникационной отказоустойчивой сети, которая могла бы продолжить свою работу в условиях, когда большая ее часть разрушена. Тем не менее, без централизованного контроля информационные пакеты самостоятельно перемещаются от одного узла к другому, пока не попадут в пункт назначения. Необходимо только контролировать ошибки при передаче пакетов и процесс доставки информации.

Разработкой Internet в начале 70-х годов занималось Агентство передовых проектов национальной безопасности США (United States Defense Advanced Research Project Agency, сокращенно - DARPA). В 1969г. в Калифорнийском университете был установлен узел ARPAnet. Тем самым была создана основа построения сети с коммутацией пакетов.

К началу 1971г. ARPAnet насчитывала уже 20 узлов, этой сетью было охвачено 30 университетов США. В начале 80-х годов к ARPAnet стали подключать первые ЛС, и с этой целью был выбран набор протоколов: Transmission Control Protocol / Internet Protocol (TCP/IP). Данное семейство протоколов стали называть протоколами Internet.

Существует ряд причин, по которым именно протокол TCP/IP был взят в основу сети Internet:

1. Возможность работы этих протоколов в локальных (LAN) и глобальных (WAN) сетях.
2. Способность этих протоколов управлять большим количеством стационарных и мобильных пользователей.
3. Удобство использования людьми или организациями, желающими подсоединиться прямо к Internet или через фирмы, предоставляющие этот сервис.
4. Высокий уровень взаимодействия между различными операционными системами: предоставление средств для разработки на их основе приложений, использующих современный программный интерфейс.

В 1986г. на базе существующей опорной сети ARPAnet производительностью 56 Кбит, состоящей из небольшого количества соединенных между собой 6-ти суперкомпьютерных центров США, решено было начать строительство новой сети. Заказ был дан консорциуму Merit, MCI, IBM. и, спустя 8 месяцев, была построена новая система на базе магистральных каналов T1 с производительностью 1,54 Мбит/с. Она содержала 13 коммуникационных узлов, каждый из которых представлял из себя 9 - 11 компьютеров IBM под управлением Berkeley UNIX,

соединенных локальной сетью. Они занимались маршрутизацией пакетов и сбором сетевой информации. Узлы коммутированы так, что если одна из машин выходит из строя, ее функции на себя берут другие машины.

Далее развитие Internet пошло подобно "снежному кому": ядро стало обрастать большим количеством хостов (компьютеров, которые принимают и передают информацию), соединений и подключенных сетей. Уже в 1989г. загрузка опорных линий T1 достигла насыщения.

В 1991г. была очередная модернизация ядра Internet. С магистрали T1 перешли на магистраль T3 с производительностью 45 Мбит/с. И компьютер установили IBM RT/6000. В сети было 16 узлов и 3600 подключенных ЛС. По прогнозам специалистов в 2001 году все грамотные люди будут работать в Internet.

1.2. Наиболее распространенные сервисы протокола TCP/IP.

В настоящий момент времени эти протоколы Internet больше не модифицируются. Все изменения происходят с протоколами прикладного уровня.

Сервис разрешения имен. (Механизм именования ресурсов сети).

Этот сервис необходим для того, чтобы каждый компьютер мог получить доступ к любому ресурсу Internet. О каждом компьютере, подключенном к Internet и предоставляющем свои ресурсы для использования другим, следует знать не только адрес, но и вид услуг. Поэтому возникают проблемы, где хранить и как управлять именами ресурсов сетей, а также их адресами. Проблема соответствия имен и адресов очень сложная - в географически разнесенных компьютерах нужно хранить и обновлять имена и адреса, что является задачей распределенной БД. Это обеспечивает сервис DNS, именно он управляет распределенной БД.

Электронная почта

Позволяет обмениваться сообщениями с пользователем другого компьютера. Чтобы включиться в систему электронной почты, необходимо зарегистрировать свой почтовый ящик на почтовом сервере, куда будет складываться корреспонденция. При поступлении нового сообщения система дописывает его в почтовый ящик (файл). Таким образом, чтобы послать сообщение по почте, нужно присоединиться к компьютеру, где находится почтовый ящик получателя, и туда послать сообщение. В настоящее время пользователей электронной почтой стало много, и появилась необходимость в использовании почтовых серверов (пересылочных пунктов сообщений), которые передают сообщения по назначению. Чтобы получить присланные сообщения, достаточно запросить свой почтовый ящик.

Система новостей Use Net

Система ведения дискуссий, состоящих из нескольких групп, каждая из которых, являясь группой новостей, в свою очередь, делится на темы. В каждой теме до 1000 подгрупп, которые могут иметь свои структуры. Группы новостей позволяют пользователям с общими интересами обмениваться информацией между собой. Можно отправлять статьи, отвечать на вопросы, задаваемые в статьях другого пользователя. Принципиальное отличие новостей от почты заключается в том, что следует централизованно хранить все новости на сервере Use Net; именно к этому банку данных должны обращаться все пользователи сети. Система Use Net позволяет вести дискуссию, осуществлять фильтрацию статей по ключевым словам, просматривать специальные файлы, подключенные к статьям (картинки, презентации).

Всемирная паутина - WWW (Word Wide Web)

WWW состоит из компьютеров, объединенных в единую сеть, которые предоставляют вполне определенный тип доступа к хранящейся информации. Предоставляется возможность тиражирования рекламной, образовательной и подобной ей информации. Работу с сервером WWW поддерживает протокол HTTP (Hypertext Transfer Protocol) – самый последний, широко используемый протокол, обеспечивает:

- навигацию по миру WWW;
- аутентификацию - при подключении хоста нужно предъявить имя – user name, пароль – password;
- формирование информационного запроса;
- возвращение отобранной информации пользователю.

Пример 1: http://www.microsoft.com – WEB адрес корпорации Microsoft.

Пример 2: http://www.whitehouse.gov/wh/Family/life.html - история Белого дома.

Адрес включает http:// - сервис Internet; www – имя сервера Internet, к которому осуществляется подключение; whitehouse – имя домена; gov/wh/Family – каталог; life.html – имя документа.

Передача файлов FTP (File Transfer Protocol)

Самый ранний сервис Internet позволяет пользователю, работающему на одном компьютере, считывать файлы с другого или пересылать файлы на другой компьютер, если там установлена программа FTP.

Протокол FTP содержит два упрощенных протокола - SFTP и TFTP. Некоторые компьютеры используют Internet только для передачи файлов, их называют FTP-серверы. С такого компьютера можно брать все необходимые файлы. Если к FTP-серверу может присоединиться любой пользователь, то такой сервер называют *анонимным FTP-сервером.*

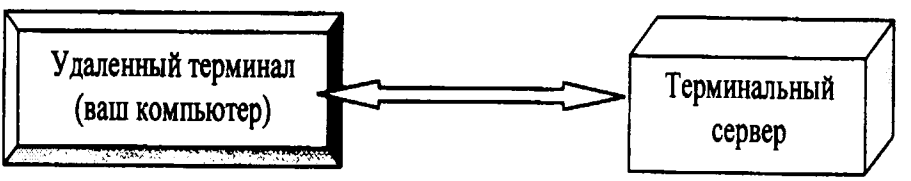


Рис. 3.1. Удаленный терминал и терминальный сервер. Протокол Telnet.

Имеется возможность работы на другом компьютере сети, который поддерживает сессию Telnet. В начале сессии подсоединяетесь к компьютеру и, по необходимости, указываете свое имя и пароль. После этого исходные данные передаются на удаленный компьютер. Вводите свои данные в компьютер, но его больше «не видно», и кажется, что работа идет с хостом, а компьютер только поддерживает сеанс связи. Раньше это был основной сервис Internet, сейчас он используется для удаленного администрирования компьютерных систем и для доступа к хостам.

Если компьютерная система не может долго находиться в терминальном соединении, то используют специальные терминальные серверы. Такой сервер представляет собой компьютер, который умеет запускать протокол Telnet для входа в удаленную систему. Если компьютер подсоединен к такому серверу, то

следует указать имя того хоста, с которым вы желаете работать. Терминальный сервер одновременно открывает несколько протоколов Telnet и может управлять передачей нескольких потоков данных. Терминальные серверы поддерживают сервис разрешения имен и другие сетевые протоколы.

Сетевая файловая система NFS

Позволяет одному компьютеру работать с файлами и устройствами других компьютеров. Сервис берет начало от UNIX, где все устройства представлены в виде файлов dev/*. Создается иллюзия того, что устройства, диски, порты других компьютеров подключены непосредственно к компьютеру. Поэтому несколько человек одновременно могут работать с одним файлом, что существенно упрощает организацию работы в системе с распределенными данными.

Мониторинг и управление сетью.

SNMP (Simple Network Management Protocol) - протокол прикладного уровня, предназначенный для обмена информацией между сетевыми устройствами, формирует справочную информацию, в том числе показатель числа пакетов в секунду, а также коэффициент сетевых ошибок. Используя эти данные, администраторы могут управлять производительностью сети и решать различные сетевые проблемы. Как правило, SNMP работает на базе протокола транспортного уровня UDP (User Datagram Protocol). Агент SNMP – это прикладной программный модуль, который работает в управляемой системе и собирает информацию об устройствах, делая эту информацию доступной для системы управления сетями NMS (Network Management Systems). SNMP позволяет осуществлять мониторинг сети, учитывая возможности и полномочия агента.

Удаленное выполнение процедур

Этот сервис позволяет запустить на удаленном компьютере определенную программу, например, для решения большой сложной задачи. Сейчас он находится в стадии развития, поэтому общепринятые протоколы отсутствуют, а есть множество очень далеких от совершенства реализаций программ и связей.

2. Приложение MICROSOFT INTERNET EXPLORER

В состав Office 97 входит приложение Internet Explorer, позволяющий просматривать огромный объем сведений, хранящихся по всему миру на WWW-страницах. Можно знакомится с их содержанием на своем компьютере и использовать найденную информацию в своих документах, или сохранить в файлах компьютера. На рабочем столе монитора компьютера расположен ярлык Internet Explorer. После щелчка появится окно браузера (см. рис. 1).

Просмотр WWW-страницы

Нужно выбрать любую ссылку на начальной странице (она открывается первой при запуске Internet Explorer). Начальная страница может быть страницей Internet или страницей рабочего компьютера; ее изменение возможно в любое время. Ссылка может быть рисунком, трехмерным изображением или текстом другого цвета (обычно с подчеркнутым шрифтом).

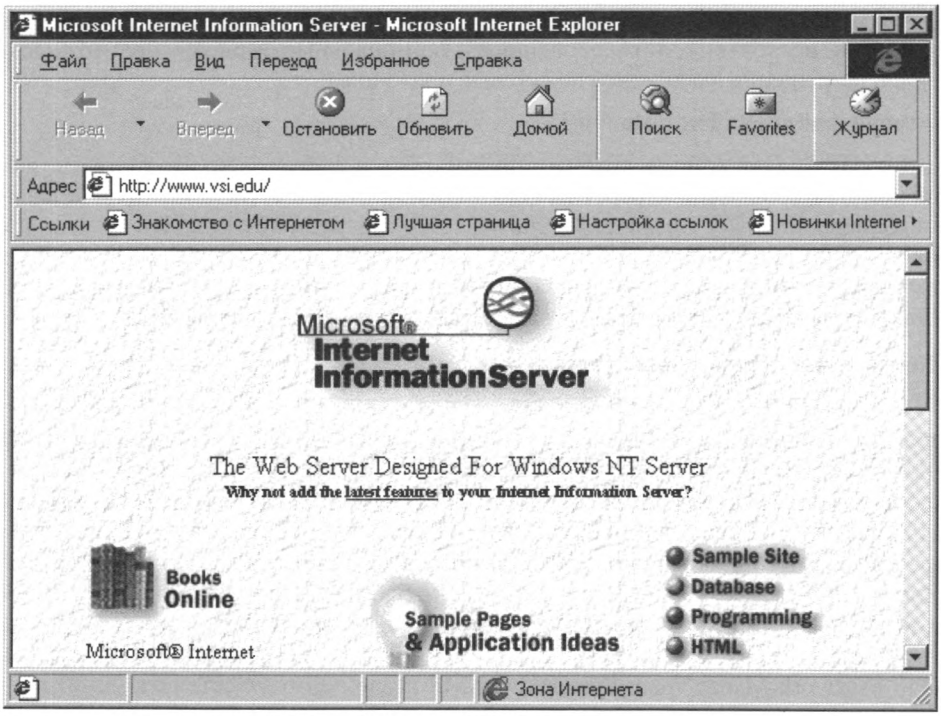


Рис. 1. Окно Microsoft Internet Explorer (начальная страница)

Для перехода от страницы к странице использует следующие кнопки.

Переход на следующую страницу

Переход на предыдущую страницу

Чтобы узнать, является ли элемент страницы ссылкой, следует подвести к нему указатель. Если он принимает вид кисти руки с указательным пальцем, значит эта строка или рисунок является ссылкой.

Переход в определенное место или на определенную страницу.

Нужно ввести в адресную строку адрес страницы, которую нужно открыть, или нажать кнопку со стрелкой для выбора адреса из списка, а затем нажать клавишу **ENTER**. Адрес узла Internet обычно начинается с имени протокола, за которым следует обслуживающая узел организация; суффикс обозначает тип организации. Например, "http://www.yale.edu/" указывает сервер WWW Йельского университета. Начало "http://www" указывает, что это сервер WWW, который использует протокол http. Суффикс ".edu" определяет общеобразовательную организацию. Обычно адреса коммерческих узлов заканчиваются на ".com", а адреса правительственных организаций - на ".gov".

Если адрес указывает на определенную страницу, то включаются допол-

нительные сведения, такие как имя порта, каталог, в котором находится страница и имя файла страницы. Страницы WWW, разработанные с использованием языка HTML (Hypertext Markup Language), часто заканчиваются расширением ".htm" или ".html".

Для отображения страницы в новом окне нужно выполнить команду **Файл/Создать окно** и задать адрес.

Имеется также возможность перейти на определенную страницу. Нужно выполнить для этого команду **Файл/Открыть**, а затем ввести адрес.

Возвращение к просмотренной ранее странице.

Выберите страницу, на которую нужно перейти, из списка меню **Переход.** Этот список очищается при выходе из Microsoft Internet Explorer.

Для просмотра полного списка страниц, уже просмотренных во всех сеансах связи, нужно выполнить команду **Переход/Открыть папку журнала**. Чтобы открыть нужную страницу, необходимо выбрать ее и дважды щелкнуть кнопкой мыши.

Поиск в Internet

Для поиска нажмите на панели инструментов кнопку. Доступная программа поиска зависит от вашего поставщика услуг Internet. Страницу поиска можно изменить в любое время.

Возвращение на начальную страницу

Для возвращения на начальную страницу нужно щелкнуть на панели инструментов по кнопке. Если панель инструментов не отображается, необходимо выбрать пункт **Панель инструментов** в меню **Вид**.

Настройка советника

- Необходимо выбрать подпункт **Вид/Свойства обозревателя** для отображения свойств Internet.
- Необходимо выбрать вкладку **Содержание**, а затем нажать кнопку **Включить** в группе **Оценки.**
- Если специальный пароль еще не был задан для данного компьютера, выдается запрос на создание такого пароля. Если пароль был задан, выдается запрос на ввод этого пароля.

Нужно записать специальный пароль, поскольку он требуется при каждом изменении параметров советника по содержимому.

Поиск текста на странице

- Нужно выполнить команду **Найти** (на данной странице) меню **Правка**.
- Необходимо ввести образец текста для поиска.
- Нужно изменить параметры поиска, если нужно.
- Нужно нажать кнопку **Найти далее**.

Копирование сведений страницы в документ

Копирование сведений с одной страницы WWW на другую невозможно. Можно скопировать информацию в документ WORD:

- Нужно выбрать сведения для копирования. При необходимости копировать содержимое всей страницы, выполнить команду **Выделить все** меню **Правка**.
- Выполнить команду **Копировать** меню **Правка**.
- Открыть документ, в который необходимо добавить эти сведения, а затем выберите место, куда их нужно поместить.
- Выполнить команду **Вставка** меню **Правка** этого документа.

Изменение вида страницы

- Выполните команду **Вид / Свойства обозревателя**.
- Измените необходимые параметры на вкладке **Общие (Цвета, Шрифты, Языки, Специальные возможности)**.

Для получения справки по любому из этих элементов окна, нужно нажать кнопку  в верхней части диалогового окна, а затем выберите нужный элемент. Второй способ: навести курсор мыши на интересующий элемент, щелкнуть правой кнопкой и подтвердить вопрос "Что это такое?".

Некоторые страницы используют заданные цвет и шрифт, вместо установленных в данном окне.

Изменение вида панели инструментов

Имеется возможность перемещать разделы адресов и ссылок панели инструментов - вверх, вниз, влево или вправо.

Чтобы освободить дополнительное место на экране, можно скрыть подписи кнопок панели инструментов. Перетащить для этого нижнюю грань широкой панели инструментов вверх, пока подписи кнопок не исчезнут.

Использование панели ссылок

Выберите на панели инструментов слово **Ссылки,** а затем - одну из них. Раздел ссылок панели инструментов позволяет открыть в Internet определенные страницы WWW, сведения которых регулярно обновляются.

Чтобы увеличить или уменьшить раздел ссылок панели инструментов, выберите команду **Избранное/Добавить в избранное/Ссылки**. Можно изменить порядок ссылок с помощью команды **Избранное/Упорядочить; Избранное/Ссылки.**

Отображение текста с помощью другого шрифта

- Выполнить команду **Вид/Шрифты**.
- Второй способ – выбрать команду **Вид/Свойства** обозревателя на вкладке **Общие**, щелкнуть по кнопке **Шрифты** и выберите **Размер шрифта (Самый крупный, Крупный, Средний, Мелкий, Самый мелкий**); установить цвет символов.

Некоторые страницы используют заданные цвета и шрифты, вместо заданных в этом окне.

Изменение начальной страницы

- Перейти на страницу, которую необходимо открывать при запуске Microsoft Internet Explorer.

- Выполнить команду **Вид/Свойства обозревателя/Адрес начальной страницы**, с которой следует начинать обзор.

При необходимости можно попробовать заняться разработкой страниц WWW, создавая свою собственную начальную страницу со ссылками на наиболее интересные страницы WWW.

Чтобы восстановить исходную начальную страницу, нужно нажать кнопку **С исходной, С текущей, С пустой.**

Использование графического изображения в качестве фонового рисунка рабочего стола.

- Перейти на страницу, содержащую нужное изображение.
- Выбрать его с помощью правой кнопки мыши и выполнить команду контекстного меню **Сделать рисунком рабочего стола**.

Использование клавиатуры для просмотра документов.

Переход на следующую страницу	SHIFT+BACKSPACE
Переход на предыдущую страницу	BACKSPACE
Отображение контекстного меню для ссылки	SHIFT+F10
Перемещение между рамками	SHIFT+CTRL+TAB
Прокрутка по направлению к началу документа	UP ARROW
Прокрутка по направлению к концу документа	DOWN ARROW
Прокрутка по направлению к началу документа через больший интервал	PAGE UP
Прокрутка по направлению к концу документа через больший интервал	PAGE DOWN
Переход к началу документа	HOME
Переход к концу документа	END

Работа с Internet Explorer

Обновление текущей страницы	F5
Остановка загрузки страницы	ESC
Переход в новое место	CTRL+O
Переход в новое окно	CTRL+N
Сохранение текущей страницы	CTRL+S
Печать текущей страницы или активной рамки	CTRL+P
Активизация выбранной ссылки	ENTER

Использование устройства IntelliMouse

В Internet Explorer имеется возможность использовать IntelliMouse для выполнения следующих действий:

– Переходы по ссылкам. Имеется возможность перейти на любую ссылку, указывая на нее, а затем, выполняя операцию "прокрутки данных", вперед. Чтобы вернуться к предыдущему разделу, нужно выполнить "прокрутки данных" назад. Чтобы выполнить операцию "прокрутки данных", необходимо нажать и удерживайте клавишу **SHIFT** при вращении колесика.

– Прокрутка страницы WWW выполняется вращением колесика вперед или назад. Для скольжения или непрерывной прокрутки текущей страницы WWW нужно удерживать нажатой кнопку колесика при перемещении мыши.

– Для получения дополнительных сведений о программном обеспечении IntelliPoint и IntelliMouse нужно обратиться к документации по IntelliMouse.

Создание ярлыка страницы на рабочем столе

- Перейти на страницу, для которой необходимо создать ярлык.
- Выполнить команду **Файл/Отправить/Ярлык на рабочий стол.**

Если окно Internet Explorer раскрыто не полностью, имеется возможность создать ярлык, перетаскивая ссылку из окна Internet Explorer в нужное место, например, на рабочий стол.

Сохранение текущей страницы на компьютере

- Выполнить команду **Файл/Сохранить как/Файл HTML.**
- Выбрать папку для сохранения страницы и дважды нажмите кнопку мыши.
- В поле **Имя файла** нужно задать имя файла страницы и нажать кнопку **Сохранить**. Следует отметить, что в этом случае сохраняется только текст без объектов (рисунков, клипов).
- Для сохранения копии ссылки на WWW-страницу достаточно перетащить ссылку в нужное место, например, на рабочий стол. В результате создается ярлык страницы WWW.

3. Приложение MICROSOFT OUTLOOK

Приложение Outlook было разработано фирмой Microsoft для выполнения функций персонального организатора. Outlook используется среди владельцев компьютеров, на которых установлены Windows95 или Windows NT. Outlook – единый инструмент для управления самыми различными типами информации: документами и файлами, сообщениями, которые приходят по электронной почте и факсу, событиями календаря, личными задачами, базой данных контактов и т.п.

Средствами Outlook можно управлять файловой системой, иметь доступ к локальным и сетевым ресурсам. Outlook позволяет копировать, перемещать, удалять, переименовывать папки и файлы, запускать приложения.

После установки приложения Outlook нет необходимости использовать Microsoft Exchange, поскольку Outlook отправляет, получает и хранит сообщения электронной почты различных типов.

Outlook позволяет планировать личное время аналогично тому, как это выполняется при помощи приложения Schedule+. Если же компьютер связан с компьютерами других пользователей Outlook, то возможно согласовать время собрания со всеми его участниками.

С помощью Outlook возможно поддерживать базы данных таких информационных элементов, как заметки, адреса и телефоны корреспондентов, дневниковые записи и т.п., а также осуществлять экспорт и импорт информации приложения Outlook в файлы различных форматов и обратно.

Объединение вышеперечисленных возможностей позволяет рассматривать Outlook как приложение, которое следует запускать в начале рабочего дня и закрывать в конце.

Электронная почта

Microsoft Outlook можно использовать автономно, например, для организации личной деятельности. Тем не менее, только коллективная работа с Outlook позволяет использовать все его возможности, одной из которых является обмен сообщениями электронной почты.

Outlook позволяет обмениваться сообщениями по электронной почте как внутри рабочей группы, так и с пользователями, имеющими доступ к вашей ло-

кальной или корпоративной сети. При наличии подключения к Интернет круг общения можно расширить до масштабов всей планеты.

Сообщения электронной почты могут быть не только текстовыми, с применением шрифтов разнообразных размеров, цветов и стилей. Можно вставлять графические изображения, объекты OLE, а также любые файлы в качестве вложения.

Преимуществом использования Outlook для обмена сообщениями является наличие множества дополнительных возможностей, упрощающих работу с электронной почтой. Microsoft Outlook позволяет организовать архив корреспонденции в личных папках и структурировать его по различным критериям. Воспользовавшись средством поиска элементов Outlook, можно быстро найти нужное сообщение. Дневник Outlook позволяет отслеживать моменты поступления корреспонденции.

Отправка сообщения

Для отправки сообщения нужно нажать кнопку **Отправить (Send)** на панели инструментов формы сообщения. Форма исчезнет. Outlook сохранит сообщение в папке **Отправленные (Sent Items)** и перешлет его в почтовое отделение Microsoft Mail (или, в случае использования другой информационной службы, на сервер электронной почты). Чтобы проверить, было ли отправлено сообщение, нужно перейти в папку **Отправленные**. Верхний элемент списка – сообщение, посланное самому себе. Присутствие сообщения в списке подтверждает отправку сообщения, но не означает, что сообщение было доставлено конечному адресату и было им прочтено. Подтверждение получения сообщения адресатами описано далее в разделе «Контроль за доставкой сообщений».

Получение сообщения

Обычно Outlook проверяет приход новой почты через равные промежутки времени. В случае прямого подключения к системе электронной почты (по локальной сети) можно установить короткий промежуток (например, 10 минут).

Один из способов получить посланное самому себе сообщение – подождать некоторое время, по истечении которого Microsoft Outlook начнет проверку поступления новых сообщений. Тогда оно появится в окне просмотра элементов папки **Входящие**.

Если же нужно просмотреть сообщение, можно обратиться к почтовому отделению немедленно. Для начала следует убедиться, что Outlook настроен на проверку новой почты на нужных информационных службах. Для этого стоит выбрать команду **Сервис/Доставить почту через (Tools, Check for New Mail On)**. Появится диалоговое окно **Проверка новых сообщений (Check for New Mail On)** (рис. 2). Следует убедиться, что для нужных вам информационных служб установлены флажки. Затем нажать кнопку **ОК**. Теперь можно проверить наличие новой почты. Нужно выбрать команду **Сервис/Доставить почту** (или нажмите клавишу <F5>). В процессе опроса проверки и извлечения сообщений Outlook выводит анимированное окно с летающим конвертом.

После получения сообщения оно появится в папке **Входящие.** Outlook по умолчанию выводит сообщения в представлении **Сообщения с автопросмотром**, поэтому для каждого непрочитанного сообщения выводятся заголовок и первые три строки, а для уже прочитанного сообщения - только заголовок. Необходимо дважды щелкнуть левой кнопкой мыши по значку или первым трем строкам сообщения для вывода полного текста сообщения, включая внедренную графику в форме сообщения.

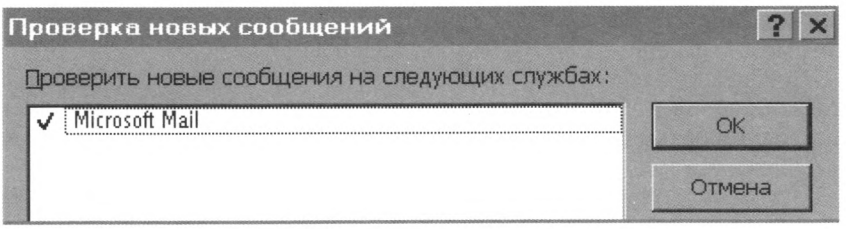


Рис. 2. Диалоговое окно **Проверка новых сообщений**

Подготовка простого сообщения

Нужно перейти в папку **Входящие**, щелкнув по ее ярлыку на панели Outlook. Затем нажать кнопку **Создать сообщение (New Mail Message)** на панели инструментов или выбрать команду **Сообщение/Создать сообщение (Compose, New Mail Message)**. Появиться форма нового сообщения (рис. 3).

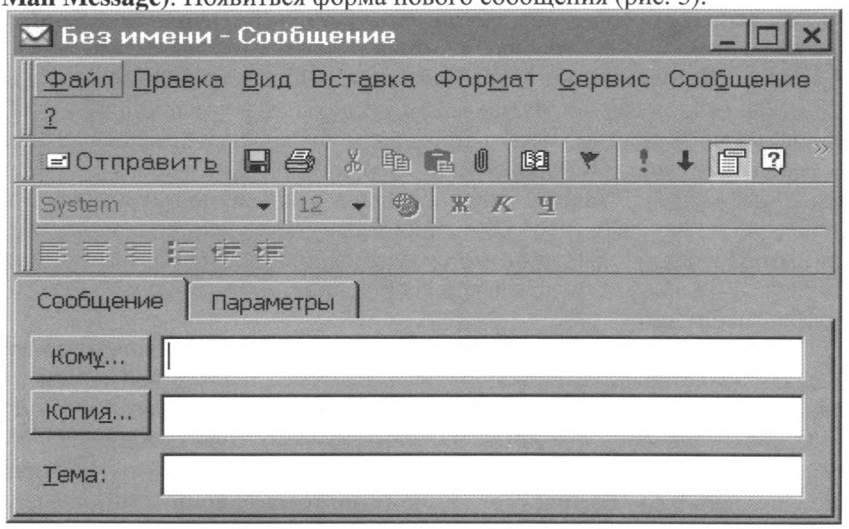


Рис. 3. Форма нового сообщения

Если форма **Сообщение** не содержит полей **Копия и Тема**, то для их вывода следует нажать кнопку **Полный заголовок (Message Header)** на панели инструментов.

Первый шаг создания сообщения – задание имен получателей. Для этого нужно нажать кнопку **Кому (To).** Появиться диалоговое окно **Выбор имен (Select Names)** (см. рис. 4).

Диалоговое окно **Выбор имен** будет содержать один из имеющихся списков адресов, скорее всего - список контактов. Чтобы послать сообщение кому-нибудь из заданной группы, нужно выбрать имя из списка адресов почтового отделения. Для этого в раскрывающемся списке **Источник адресов (Show Names from the …)** следует выбрать элемент **Список адресов п/о**.

Примечание. Адреса электронной почты, отличные от Microsoft Mail, обычно хранятся в **Личной адресной книге** или в папке **Контакты**. Эти адреса также доступны из диалогового окна **Выбор имен** – достаточно выбрать нужный элемент в раскрывающемся списке **Источник адресов**. Процедура выбора имени получателя, приведенная ниже, не зависит от используемой адресной книги.

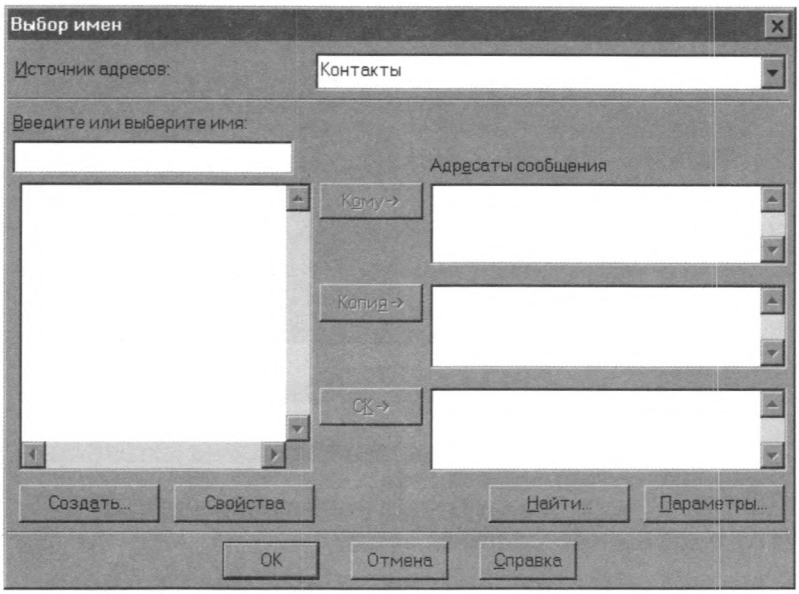


Рис. 4. Диалоговое окно **Выбор имен**

Процедура выбора имени

- Выделить нужное имя в списке.
- Нажать кнопку **Кому (To)**, чтобы скопировать имя в список **Адресаты сообщения (Message Recipients)** в правой части диалогового окна **Выбор имен.**
- Если нужно, повторить ту же процедуру для остальных получателей. Чтобы не добавлять по одному имени, можно выделить одно имя в списке и выделять остальные, удерживая при этом в нажатом состоянии клавишу <Ctrl>.
- Последовательность имен можно выделить, начав с первого и выделив последнее, удерживая клавишу <Shift>. Выделив все нужные имена, нажать кнопку **Кому**, чтобы скопировать их в список **Адресаты сообщения**.
- Имена получателей копии сообщения можно аналогичным образом выделить, а затем скопировать в нужное поле, нажав кнопку **Копия** или **СК** для получателей слепой копии. Имена получателей копии фигурируют в общем списке у всех адресатов, а имена получателей слепой копии – нет.
- Определив имена адресатов, нужно нажать кнопку **ОК** для возврата в форму сообщения, содержащую теперь имена всех получателей.

Следующий шаг в создании сообщения – *заголовок.* Поместть курсор ввода в текстовое поле **Тема** и ввести краткое описание предмета сообщения.

Теперь можно ввести текст сообщения. Поместить курсор ввода в большое текстовое поле и ввести текст. После этого сообщение будет выглядеть примерно так, кА показано на рис. 5.

Форматирование текста

Встроенный редактор электронной почты может форматировать текст сообщения.

Шрифт и размер шрифта, используемые по умолчанию для текста отправляемых и принимаемых сообщений, устанавливается на вкладке **Отправка (Sending)** диалогового окна **Параметры (Options)** (рис. 6). Выбирается команда **Сервис/Параметры** и раскрывается вкладка **Отправка** для настройки этого параметра.

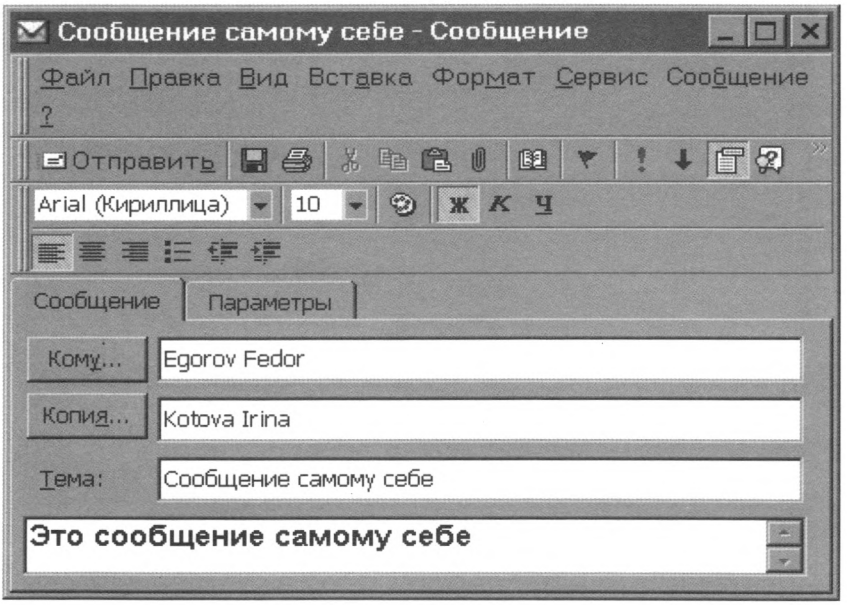


Рис. 5. Готовое сообщение

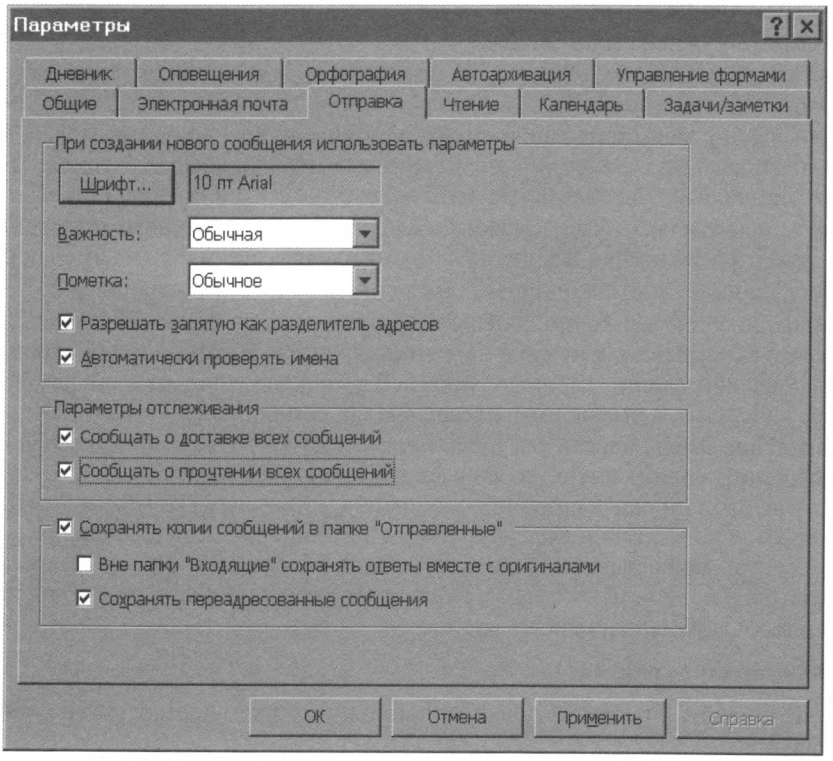
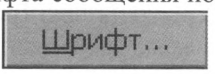


Рис. 6. Вкладка **Отправка** диалогового окна **Параметры**

Изменить настройку шрифта сообщения по умолчанию (10 пт Arial) можно, нажав кнопку **Шрифт.**



Появится стандартное диалоговое окно **Шрифт:** любые доступные на компьютере значения параметров **Шрифт, Начертание, Размер**, **Зачеркнутый, Подчеркнутый** и **Цвет**. Нужно нажать кнопку **ОК,** закончив настройку параметров.

Примечание. Шрифт сообщения при отсутствии его у получателя, будет замещен, поэтому лучше выбирать стандартные шрифты Windows и быть уверенным, что они есть у получателя.

Форма сообщения имеет две панели инструментов: верхняя – **Стандартная** и нижняя – **Формат** с кнопками для форматирования текста сообщения (рис. 7). Эта панель активна, если курсор ввода находится в текстовом поле внизу окна сообщения.

Рис. 7. Панели инструментов **Стандартная** и **Формат**

В табл. 1 приведен список кнопок панели инструментов **Формат** встроенного редактора сообщений (кнопки панели форматирования WordMail несколько отличаются).

Кнопки панели инструментов Формат Таблица 1.

Кнопка	**Назначение**
Шрифт	Выбрать шрифт из установленных на компьютере
Размер	Выбрать размер шрифта
Цвет текста	Выбрать цвет текста
Полужирный	Установить или отменить полужирное начертание
Курсив	Установить или отменить курсив
Подчеркнутый	Установить или отменить подчеркивание
По левому краю	Установить или отменить выравнивание по левому краю
Кнопка	**Назначение**
По центру	Установить или отменить выравнивание по центру
По правому краю	Установить или отменить выравнивание по правому краю
Маркированный список	Установить или отменить маркированный список
Уменьшить отступ	Уменьшить отступ абзаца
Увеличить отступ	Увеличить отступ абзаца

Форматировать текст сообщения можно и с помощью команд меню. Для форматирования символов следует выбрать команду **Формат, Шрифт**, а для форматирования абзаца – команду **Формат, Абзац**.

Присоединение файлов к сообщению

Для вставки файла в сообщение нужно нажать кнопку **Добавить файл (Insert File)** на панели инструментов формы сообщения или выбрать команду **Вставка, Файл**. Появится диалоговое окно.

Найти в списке файлов нужный и выделите его. В группе **Вставить как** выбрать один из трех типов вставки и нажмите кнопку **ОК,** в сообщении появится соответствующий значок. На рис. 8 файл Word вставлен дважды: как вложение и как ярлык.

Примечание. Вложенные элементы представлены в виде значков с именем элемента внизу. Ярлыки представлены таким же символом с наложенной стрелкой. Стрелка как индикатор ярлыка – обычный прием, используемый в приложении.

Дважды щелкнув левой кнопкой мыши по значку вложения, адресат откроет связанное (в реестре Windows) с файлом приложение, а в нем – присланный файл. При двойном щелчке по ярлыку приложение проследит путь в ярлыке, найдет и откроет исходный файл.

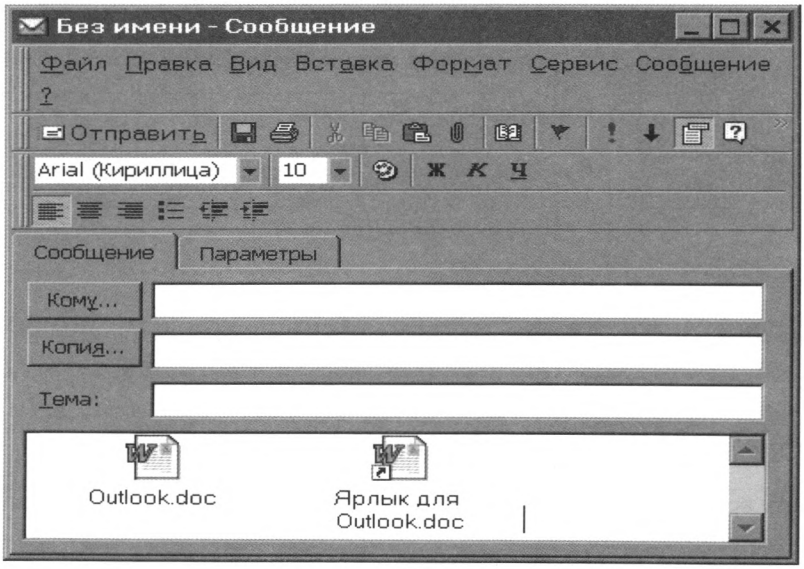


Рис. 8. Сообщение с файлом, вставленным как вложение и как ярлык

Чтобы послать в сообщении элемент Outlook, например, элемент из папки **Контакты,** следует выполнить ряд действий.

1. В форме сообщения выбрать команду **Вставка/Документ**. Появится диалоговое окно **Вставка элемента** (рис. 9).
2. В списке **Папки** вверху диалогового окна выбрать папку, содержащую элемент, который нужно послать, например, **Контакты**.
3. В списке **Элементы** внизу диалогового окна выбрать нужный элемент.
4. В группе **Вставить как** установить нужный переключатель **- Вложение** или **Ярлык.**

5. Нажмите кнопку **ОК**, чтобы вставить элемент в сообщение.

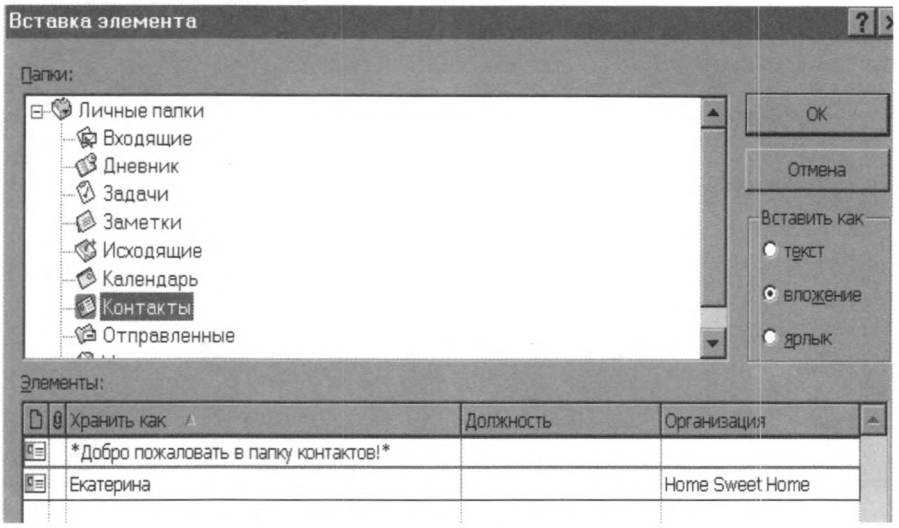


Рис. 9. Диалоговое окно **Вставка элемента**

Адресат, получив сообщение, содержащее значки - вставленные элементы Outlook, может дважды щелкнуть по ним левой кнопкой мыши, чтобы увидеть подробности. Естественно, у получателя должен быть установлен Outlook, Exchange Client или Windows Messaging.

Примечание. Outlook, естественно, может открыть любой элемент Outlook, а Exchange Client – только элементы некоторых типов.

Внедрение объектов OLE

Помимо файлов и элементов Outlook в сообщение можно вставить и объекты OLE, например, графику или звуки, причем как существующий объект, так и созданный прямо в Outlook. Далее будет рассмотрена сначала процедура внедрения существующего объекта.

Пусть необходимо внедрить в сообщение существующий рисунок, созданный в редакторе Paint. Для этого нужно:

1. Вывести форму сообщения и выбрать команду **Вставка объекта (Insert Object)**. Появится диалоговое окно **Вставка объекта** (рис. 10). Содержание поля **Тип объекта** зависит от установленных приложений Windows.
2. Установить переключатель **Создать из файла (Create From File).** При этом вид диалогового окна изменится (рис. 11).
3. Нажать кнопку **Обзор (Browse),** найти папку или файл, содержащие нужный объект, выделить его и нажать кнопку **ОК** для возврата в диалоговое окно **Вставка объекта**, которое теперь содержит имя выбранного файла.
4. Не устанавливая флажки **Связь (Link)** и **В виде значка (Display as Icon),** нажать кнопку **ОК**, появится форма сообщения со вставленным объектом или его частью. Нажать кнопку **Отправить** для отправки сообщения.

Когда адресат получит сообщение и дважды щелкнет по нему левой кнопкой мыши, оно появится со вставленным объектом. При этом у получателя должно быть установлено приложение, в котором был создан объект.

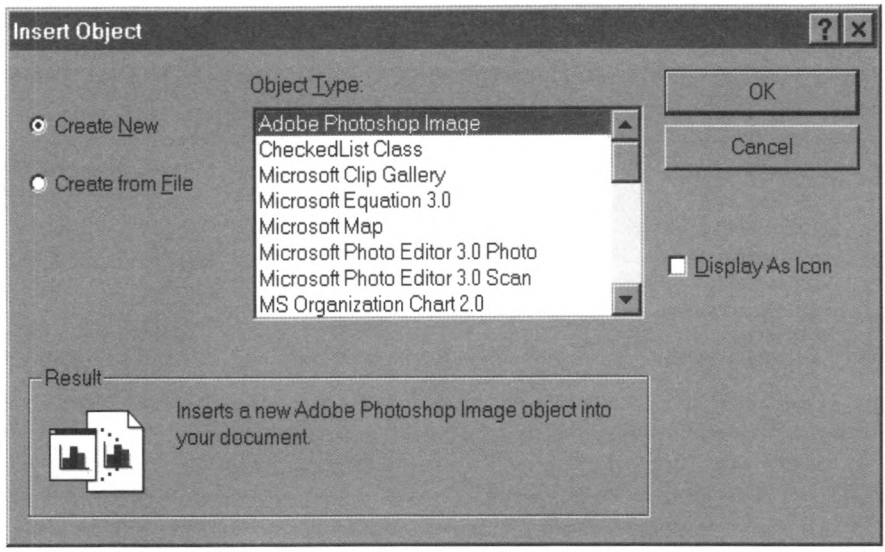


Рис. 10. Диалоговое окно **Вставка объекта**

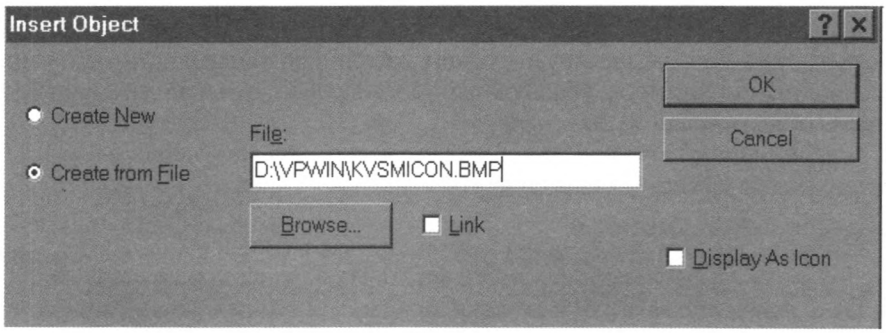


Рис. 11. Окно поиска вставляемого объекта

Примечание. Если объект вставлен как значок, то адресат сможет открыть его двойным щелчком – но только при наличии у него установленного приложения.

Вместо вставки уже существующего рисунка Paint можно создать его прямо в сообщении. Для этого следует:

1. Вывести форму сообщения и выбрать команду **Вставка**, **Объект**. Появится диалоговое окно **Вставка объекта** (рис. 11).
2. Выбрать переключатель **Создать новый (Create New)**, в поле **Тип объекта (Object Type)** выбрать нужный тип и нажать кнопку **ОК** для открытия приложения.

Теперь прямо в поле сообщения можно создавать новый объект средствами соответствующего приложения. После создания объекта нужно щелкнуть вне объекта. Приложение закроется, и сообщение можно будет отправить. Когда адресат получит сообщение и откроет его, выведенное сообщение будет содержать внедренный объект. Приложение, в котором объект был создан (или совмести-

мое), потребуется адресату только в том случае, если он собирается редактировать объект.

Добавление подписи

В Microsoft Outlook существует понятие **Автоподпись (AutoSignature)** – некоторый текст, помещаемый в сообщение автоматически или вручную. Обычно **Автоподпись** состоит из имени и, возможно, должности и номера телефона, вообще ее текст может быть любым. Нужно выбратьте команду **Сервис/Автоподпись (Tools, AutoSignature)**. Появится диалоговое окно с соответствующим названием.

Ввести текст в большом текстовом поле, его можно форматировать соответствующими кнопками.

Если нужно, чтобы Outlook автоматически добавлял **Автоподпись** в конце создаваемых сообщений, установить флажок: **Добавлять эту подпись в конце новых сообщений (Add this Signature to the End of New Messages).** Если нужно добавлять "Автоподпись" в конце ответов или пересылаемых сообщений, то следует снять этот флажок. Нажатием кнопки **ОК** сохраняется **Автоподпись**.

При выборе автоматического добавления, она появляется всякий раз при открытии формы сообщения. Вводимый текст должен быть расположен выше **Автоподписи.** При желании ее можно просто удалить.

Если флажок автоматического добавления снять, **Автоподпись** можно добавить вручную. Следует поместить курсор ввода в конце сообщения и выбрать команду **Вставка**.

Определение дополнительных атрибутов

Microsoft Outlook позволяет указать дополнительные атрибуты сообщений. Например, можно установить важность отправляемого сообщения как *высокую, обычную* или *низкую*; пометить его как *обычное, личное, частное* или *ДСП*. По умолчанию важность и пометка сообщения - обычные. Изменить атрибуты, используемые по умолчанию, можно на вкладке **Отправка** диалогового окна **Параметры** (рис. 6).

Панель инструментов формы сообщения содержит кнопки **Важность: высокая** и **Важность: низкая**, обычно отключенные, если установка обычной важности не была изменена. Ту или иную из этих кнопок можно нажать для изменения важности сообщения, при этом кнопка фиксируется. Когда сообщение будет доставлено получателю и выделено в окне просмотра папки **Входящие** в представлении **Сообщения (Messages)**, левый столбец, скорее всего, окажется пустым, указывая на обычную важность сообщения. Если отправитель сообщения установит высокую важность, в этом столбце будет восклицательный знак, а если низкую – синяя стрелка вниз. Эти символы соответствуют кнопкам на панели инструментов формы сообщения. Установка пометки аналогична установке важности, за двумя исключениями:

- для установки пометки не существует кнопки на панели инструментов, она устанавливается в раскрывающемся списке **Пометка (Sensitivity)** на вкладке **Параметры** формы сообщения;
- получатель не может изменить пометку сообщения.

При получении помеченного сообщения и двойном щелчке по нему появится форма сообщения, вверху которого будет текст, извещающий об уровне пометки.

Контроль за доставкой сообщений

Microsoft Outlook позволяет отслеживать доставку сообщений и открытие сообщения адресатом (приходится предполагать, что открытие означает прочтение сообщения получателем). Для установки параметров отслеживания нужно выбрать команду **Сервис, Параметры** и раскрыть вкладку **Отправка** (рис. 6), которая содержит два флажка отслеживания.

Сообщать о доставке сообщений (Tell Me When All Messages Have Been Delivered). Установить этот флажок для получения подтверждения доставки сообщения.

Сообщать о прочтении всех сообщений (Tell Me When All Messages Have Been Read). Установить этот флажок для получения подтверждения открытия получателем посланного сообщения.

Уведомления о доставке и прочтении сообщения – это тоже сообщение электронной почты. Поэтому при поступлении уведомления Microsoft Outlook помещает его в папку **Входящие**.

Ответ на сообщение

При получении сообщения на него можно ответить или переслать кому-нибудь. Обычно с ответом Outlook отсылает копию первоначального сообщения. К нему также можно добавить комментарии.

Для ответа на сообщение необходимо дважды щелкнуть левой кнопкой мыши по сообщению. Появится форма сообщения. Теперь на панели инструментов можно нажать следующие кнопки:

- **Ответить (Reply).** Создает ответ и отсылает его отправителю.
- **Ответить всем (Reply to All).** Создает ответ и отсылает его не только отправителю, но и всем получателям копий.
- **Переслать (Forward).** Пересылает исходное сообщение другому адресату.

При нажатии кнопки **Ответить всем** Outlook открывает похожую форму сообщения, но с полем **Копия,** с именами всех получателей копии (кроме получателей слепой копии) оригинального сообщения. В этом случае можно нажать кнопку **Кому** для вывода диалогового окна **Выбор имен**, в котором можно добавить или удалить имена в списках **Кому, Копия, СК.**

При нажатии кнопки **Переслать** Outlook открывает такую же форму сообщения, как при ответе, но без имени адресатов. Далее следует нажать кнопку **Кому**, чтобы открыть диалоговое окно **Выбор имен.** Выбрав имя, нужно нажать кнопку **ОК,** закрыть окно, добавить комментарии к пересылаемому сообщению и нажать кнопку **Переслать** для отправки сообщения.

СЛОВАРЬ ОСНОВНЫХ ТЕРМИНОВ

А **ASCII -код** - двоичный равномерный восьмиразрядный код, который используется в вычислительной технике для кодирования набора из 256 символов, представляющих собой буквы, цифры, знаки препинания, управляющие символы и символы псевдографики для визуального отображения информации. ASCII (American Standard Code for Information Interchange) – стандартный американский код для обмена информацией или его национальная разновидность. Например, русифицированный код, вторая половина кодовой таблицы, имеет следующую кодировку букв кириллицы:

10001111 - **П**
10010000 - **Р**
10001000 - **И**
10001100 - **М**
10000101 - **Е**
10010000 – **Р**

Б **Бит** - единица количества информации, равная количеству информации, содержащееся в сообщении, имеющем вероятность возникновения равную 0,5.

Г **Графический метод** - метод построения неравномерных кодов, основанный на построении кодового дерева от корня к листьям. Этот метод гарантирует выполнение свойства самосинхронизации. Коды, полученные этим методом, не всегда являются наилучшими.

Д **Длина кодовой комбинации или длина кода** - количество символов в кодовой комбинации.

В **равномерных кодах** все кодовые комбинации имеют одну и ту же длину n.

В **неравномерных кодах** длины кодовых комбинаций различных сообщений разные, поэтому неравномерные коды принято характеризовать средней длиной кода, которая определяется с учетом статистических свойств источника сообщений по следующей формуле:

$$N_{cp} = \sum_{i=1}^{n} N_i \cdot P_{oi} ;$$

где n_i - длина i-той кодовой комбинации, P_{oi} - вероятность i-того сообщения, M - количество сообщений

Е **Единица количества информации** - это количество сведений, которое передается двумя равновероятными символами или сообщениями. Эта единица называется двоичной единицей информации или битом.

З **Защитный отказ.** Если принята одна из запрещенных кодовых комбинаций, то фиксируется наличие ошибки, и при приёме сообщения дается *защитный отказ*. Для уменьшения вероятности трансформации сообщений при одном и том же уровне избыточности всю избыточность можно использовать только для обнаружения ошибок, а не на их коррекции.

И **Избыточный код** - код, содержащий дополнительные (контрольные) позиции в кодовых комбинациях по сравнению с неизбыточным, предназначенные для обнаружения (и/или исправления) ошибок, возникающих в коде при его передаче, хранении или обработке.

Избыточность информации - это наличие в данных некоторой дополнительной информации, которая может быть вредной или полезной, в зависимости от возможностей ее использования. От вредной (или бесполезной) избыточности стремятся избавиться, чтобы сократить время преобразования информации и

уменьшить объемы запоминающих устройств. Однако, для борьбы с возникающими в информации при ее обработке ошибками часто специально вводят информационную избыточность - это полезная избыточность.

Избыточный код - код содержащий дополнительные (контрольные) позиции в кодовых комбинациях по сравнению с неизбыточным, предназначенные для обнаружения (и/или исправления) ошибок, возникающих в коде при его передаче, хранении или обработке.

Информация - сведения об окружающем мире и протекающих в нем процессах, воспринимаемые человеком или специальными устройствами.

Информационные позиции - это основные позиции, представляющие неизбыточную часть кода, используемую непосредственно для представления кодируемых сообщений. Информационные позиции можно выделить только в систематических кодах. В несистематических кодах основные и контрольные позиции не имеют четкого разграничения. Количество информационных позиций **m** в коде должно быть таким, чтобы при основании кода **K** можно было бы построить столько различных кодовых комбинаций, сколько сообщений должно быть закодировано. Иными словами:

$$m \geq \log_K M;$$

где M - количество кодируемых сообщений.

К **Канал связи** - совокупность аппаратных и программных средств, осуществляющих передачу данных от источника к получателю сообщений. Канал связи характеризуется определенной скоростью передачи, степенью искажения передаваемых сигналов и пропускной способностью. Каналы связи могут иметь различную физическую природу: электрические, радио, оптические, механические, гидравлические и другие каналы связи.

Канал связи без шума - канал связи, в котором вероятность искажения информации при ее передаче пренебрежимо мала. Основной задачей при передаче информации по каналам связи без шума является обеспечение максимально возможной скорости передачи. Решение этой задачи связано с использованием неравномерных оптимальных кодов, строящихся с учетом статистических свойств источника сообщений.

Канал связи обратный - канал связи, по которому осуществляется передача служебных сигналов в системах передачи с обратной связью. Обратный канал связи в таких системах используется для реализации специальных алгоритмов повышения верности передачи информации (для борьбы с возникающими в передаваемой информации ошибками). Основная идея использования обратного канала связи состоит в передаче по нему сигналов либо подтверждающих правильный прием информации, либо запрашивающих повторную ее передачу при обнаружении в ней ошибок.

Канал связи прямой - канал связи, по которому в системах передачи информации с обратной связью осуществляется передача основной информации. Обычно в таких системах для передачи основной информации используются обнаруживающие коды. В зависимости от результата приема кодовой комбинации по прямому каналу (комбинация принята правильно или в ней обнаружены ошибки) по обратному каналу связи передается сигнал либо подтверждающий правильный прием данных, либо запрашивающий повторную передачу той же самой информации.

Канала связи пропускная способность - это верхний предел количества информации, которую может передать канал связи. Пропускная способность канала связи определяется физическими характеристиками помех, действующих в канале, статистическими свойствами источника сообщений и выбранными методами кодирования и приема информации.

К-ичный код - код с основанием *К*, то есть строящийся с использованием *К* символов, которые принято обозначать цифрами от *0* до *К-1*. Если *К > 10*, то к цифрам добавляется недостающее количество первых букв латинского алфавита.

Код - совокупность знаков (символов) и системы определенных правил, при помощи которых информация может быть представлена в виде набора из таких символов для передачи, хранения и обработки.

Кодирование - процедура перехода от сообщения к кодовой комбинации, представляющей это сообщение, в соответствии с правилами построения этого кода. Процедура обратного перехода - от кода к сообщению - называется декодированием.

Количество информации - это количество сведений, содержащихся в том или ином информационном элементе (в символе кода или в сообщении). В статистической теории информации в качестве меры количества информации используют энтропию.

Количеством собственной информации I, содержащееся в сообщении X_{oj}, определяется по формуле:

$$I(X_{0j}) = -\log_d p(X_{0j});\ j = 1,2,3,\ldots,M;$$

где $p(X_{0j})$ - вероятность сообщения X_{oj}; d – основание логарифма, определяющее единицу измерения информации. В большинстве случаев d = 2, реже d = e (e = 2,71828182845904523536028747135 27…). В соответствии с основанием логарифма, в первом случае единица измерения количества информации называется binary digit, а по Русски, сокращённо, – бит:

$$I(X_{0j}) = -\log_2 p(X_{0j})\,[\text{бит}]\ .$$

Во втором случае для оценки количества информации используются натуральные логарифмы, и единицей информации является natural digit, или – нат:

$$I(X_{0j}) = -\ln p(X_{0j})\,[\text{нат}]\ .$$

По умолчанию всегда единицей количества информации является двоичная единица информации или бит.

Контрольные позиции - дополнительные позиции в избыточном коде, используемые для обнаружения (и/или коррекции) ошибок, возникающих при передаче или обработке кодов. Чем больше контрольных позиций в коде, тем более мощными обнаруживающими и корректирующими свойствами он обладает. Однако, при этом уменьшается скорость его передачи и обработки, а также увеличивается аппаратные затраты на реализацию его обработки и хранения.

Корректирующий код - избыточный код, который использует введенную избыточность для исправления возникающих ошибок. Доля избыточной информации (количество контрольных позиций) в корректирующих кодах существенно превышает избыточность обнаруживающих кодов, что приводит к значительному снижению скорости передачи и обработки данных и к увеличению аппаратных затрат на их обработку и хранение.

Л **Линейный код** - линейный блочный (n, m) код представляет собой последовательность длины n кодовых символов, из которых m являются информационными, а остальные n-m - проверочными. Любая линейная комбинация такого рода также является комбинацией этого кода. Линейные коды задаются с помощью порождающих и проверочных матриц. Комбинации линейного кода получаются как различные линейные комбинации строк порождающей матрицы. Для любой комбинации линейного кода ее синдром, то есть произведение этой комбинации на транспонированную проверочную матрицу, равен 0. Циклические коды, являющиеся подмножеством линейных кодов, могут также задаваться порождающими и проверочными многочленами.

Линия связи - физическая среда, по которой происходит распространение электромагнитной энергии, и технические устройства, обеспечивающие качественную передачу сигналов сообщения от передатчика к приемнику.

М **Метод дополнительных групп** - метод построения неравномерных кодов, основанный на последовательном объединении самых маловероятных сообщений в группы. Этот метод гарантирует выполнение свойства самосинхронизации и получение наилучшего неравномерного кода.

Минимальное расстояние кода $d_{\min}$ - наименьшее из попарных расстояний Хемминга между различными кодовыми словами.

Модем (modem) - специальное устройство, включаемое между компьютером и телефонной линией. Это устройство при передаче цифровой информации по каналу связи преобразовывает поток битов в аналоговые сигналы, а при приеме информации из канала связи в ЭВМ выполняет обратное действие – преобразовывает аналоговые сигналы в поток битов, которые может обрабатывать ЭВМ.

Слово "модем" собирательное, оно происходит от сочетания "МОдулятор/ДЕМодулятор" и используется для обозначения широкого спектра устройств передачи цифровой информации при помощи аналоговых сигналов путем их модуляции - изменения во времени одной или нескольких характеристик аналогового сигнала: частоты, амплитуды и (или) фазы.

Н **Неизбыточный код** - код, не содержащий дополнительных контрольных позиций. При заданном основании кода **К** количество различных кодовых комбинаций М неизбыточого кода связано с длиной кода m (количеством позиций или разрядов) следующим равенством:

$$M = K^m.$$

Построение неизбыточного кода можно осуществить путем перевода десятичных чисел от 0 до М-1 в К-ичную систему счисления. Эти К-ичные числа и будут представить кодовые комбинации К-ичного неизбыточного кода.

Неравномерный код - код, в котором разные сообщения представлены кодовыми комбинациями различной длины. Неравномерные коды используются для обеспечения максимальной скорости передачи по каналам связи без шума или для минимизации объемов устройств для хранения информации. Для построения неравномерных кодов используют метод дополнительных групп или графический метод. Оба метода основаны на использовании статистических свойств источника сообщений. Неравномерные коды обладают важным свойством - свойством самосинхронизации.

Несистематический код - избыточный код, в котором нет четкого разграничения основных (информационных) и контрольных позиций. Декодирование таких кодов и исправление в них ошибок основано на сопоставлении принятой кодовой комбинации с набором известных, что снижает эффективность процедур декодирования по сравнению с систематическими кодами, использующими алгоритмические процедуры декодирования и исправления ошибок.

О **Основание кода** - количество различных символов, используемых при построении кода. Основание кода К всегда больше или равно 2. Основание кода К и длина кодовых комбинаций n определяют количество различных комбинаций кода N.

$$N = K^n.$$

Основные позиции или информационные позиции - это позиции, представляющие неизбыточную часть кода, используемую непосредственно для представления кодируемых сообщений. Информационные позиции можно выделить только в систематических кодах. В несистематических кодах основные и контрольные позиции не имеют четкого разграничения. Количество информаци-

онных позиций m в коде должно быть таким, чтобы при основании кода K можно было бы построить столько различных кодовых комбинаций, сколько сообщений должно быть закодировано. Иными словами:

$$m \geq \log_K M$$

Оптимальная длина кода - минимально возможная при заданной статистике источника сообщений и основании кода средняя длина кодовых комбинаций. Достигается путем неравномерного кодирования, обеспечивающего равновероятное появление символов кода на выходе кодирующего устройства, и, следовательно, максимальную энтропию источника символов. Для построения таких неравномерных кодов используют метод дополнительных групп или графический метод. Оптимальная длина кода равна:

$$N_{опт} = H(S) / \log_2 K;$$

где H(S) - энтропия источника сообщений, K - основание кода.

Обнаруживающий код - избыточный код, который использует введенную в него избыточность только для обнаружения ошибок. Обнаруживающие коды используются, как правило, в сочетании с дополнительными методами обеспечения требуемой верности информации. Например, в системах передачи информации с обратной связью, или в системах на заданное число повторений (системы с дублированием).

Оптимальный код - чаще всего это неравномерный код, средняя длина кодовых комбинаций которого равна оптимальной длине. Строго оптимальные коды получаются достаточно редко - только для некоторых распределений вероятностей возникновения сообщений. Для достижения оптимальности необходимо обеспечить равновероятное появление символов кода на выходе кодирующего устройства (при этом достигается максимум энтропии источника символов). Чаще удается построить код близкий к оптимальному, у которого средняя длина несколько превышает оптимальную. Метод дополнительных групп при соблюдении всех его правил гарантирует построение наилучшего (наиболее близкого к оптимальному) кода.

П **Помехи** - воздействия, искажающие сигнал, несущий полезную информацию в устройствах связи, управления, измерения, вычислительной техники. Под воздействием помех в информации возникают ошибки. Для борьбы с помехами используют различные физические методы (увеличение мощности передаваемых сигналов, специальные методы их приема и т.д.). Наряду с ними широко используются информационные методы борьбы с ошибками, основанные на использовании специальных методов кодирования и алгоритмов передачи информации (обнаруживающие и корректирующие ошибки коды, системы передачи с обратной связью и т.д.).

Правило кодирования - система правил однозначного преобразования сообщений в последовательность символов кода. Эти правила могут представлять собой некоторые алгоритмы кодирования или задаваться в виде таблиц соответствия "сообщение - код". Алгоритмические методы кодирования, как правило, используются при построении систематических кодов. Табличные - при работе с несистематическими кодами. Алгоритмические методы более эффективны, чем табличные.

Порождающая матрица - в линейных кодах - матрица размерности m строк на n столбцов, строки которой являются линейно независимыми комбинациями линейного (n, m) кода. Используется для получения всех $N = 2^m$ комбинаций линейного кода путем суммирования строк в различных комбинациях. Для получения систематических линейных кодов порождающую матрицу с помощью элементарных преобразований преобразуют к приведенно-ступенчатому виду. Порождающая матрица G связана с проверочной матрицей H следующим соот-

ношением:

$$G \times H^T = H \times G^T = \mathbf{0};$$

где **0** - нулевая матрица.

Порождающий многочлен - в циклических (n, m) кодах - многочлен степени n - m, являющийся делителем многочлена $X^n + 1$. Комбинация кода длины n является комбинацией циклического кода, если она делится без остатка на порождающий многочлен. С помощью порождающего многочлена по информационным позициям можно построить всю комбинацию кода. На основе порождающего многочлена строятся порождающая матрица и декодирующий регистр циклического кода. Порождающий многочлен g(x) и проверочный многочлен h(x) циклического (n, m) кода связанны между собой следующим соотношением:

$$G(x) \times g(x) = x^n + 1.$$

Проверка на четность - способ формирования проверочных позиций путем подсчета количества единичных символов на определенных позициях кодовой комбинации. Если количество единиц четно, то говорят, что проверка выполняется. Значение проверочной позиции должно дополнять значения проверяемых позиций до четного числа единиц. Формально проверка на четность выполняется суммированием по модулю 2 позиций, входящих в проверку. Эта сумма должна быть равна 0. При кодировании и декодировании кодов составляют систему проверочных уравнений (каждое из которых является проверкой на четность), решение которой позволяет либо сформировать проверочные позиции, либо обнаружить или исправить ошибки в кодовой комбинации. Эта система уравнений задается строками проверочной матрицы.

Проверочная матрица - в линейных (n, m) кодах - матрица размерности (n - m) строк на n столбцов, строки которой задают систему проверок на четность для этого кода. Проверочную матрицу H линейного кода можно получить по порождающей матрице G из соотношения:

$$G \times H^T = H \times G^T = 0$$

Однако, значительно проще проверочная матрица определяется, если порождающая матрица задана в приведенно-ступенчатой форме. С помощью проверочной матрицы определяется синдром ошибки, который равен

$$S = V \times H^T;$$

где V - принятая из канала комбинация.

Если синдром равен 0, то комбинация не содержит ошибок. Столбцы проверочной матрицы являются синдромами одиночных ошибок. Для получения синдрома, например, двойной ошибки надо сложить столбцы матрицы H, соответствующие позициям ошибок.

Проверочные позиции - дополнительные позиции в избыточном коде, используемые для обнаружения (и/или коррекции) ошибок, возникающих при передаче или обработке кодов. Чем больше проверочных позиций в коде, тем более мощными обнаруживающими и корректирующими свойствами он обладает. Однако, при этом уменьшается скорость его передачи и обработки, а также увеличивается аппаратные затраты на реализацию его обработки и хранения. Синдром - контрольные позиции.

Проверочный многочлен - в циклических (n, m) кодах - многочлен h(x) степени m, связанный с порождающим многочленом g(x) соотношением:

$$H(x) \times g(x) = x^n + 1.$$

На основе проверочного многочлена строятся проверочная матрица и кодирующий регистр циклического кода.

Р **Равномерный код** - код, у которого все кодовые комбинации имеют одну и ту же длину. В общем случае длина кодовой комбинации n складывается из m информационных (основных) и l контрольных пози-

ций. При l = 0 код является неизбыточным. При l > 0 - код избыточный. В избыточном коде из всех $N = K^n$ кодовых комбинаций, которые могут быть построены при основании кода K и длине n, для представления сообщений используются только $M = K^m$ комбинаций (разрешенные). На каждую разрешенную приходится K^l запрещенных комбинаций. При воздействии ошибок разрешенные комбинации могут переходить в запрещенные. За счет этого и осуществляется обнаружение и коррекция ошибок.

С **Самосинхронизация** - свойство неравномерных кодов, которое позволяет однозначно выделить переданные сообщения из непрерывного принимаемого потока кодовых символов. Это свойство обеспечивается тем, что ни одна кодовая комбинация неравномерного кода не является началом никакой другой комбинации кода. Использование метода дополнительных групп или графического метода гарантирует выполнение этого свойства. В равномерных кодах это свойство выполняется автоматически.

Символ кода - отдельный знак из набора условных обозначений, используемых для построения кода. Если строится код с основанием K, то это означает, что символ кода может принимать одно из K возможных значений кодового алфавита. Принято в качестве значений кодовых комбинаций символов использовать цифры от 0 до K-1. Если же K > 10, то кодовый алфавит дополняется до K символов первыми буквами латинского алфавита. Это правило не является обязательным. Так, например, в азбуке Морзе в качестве символов кода используются символы + и -.

Синдром - в линейных (n, m) кодах - двоичный вектор длины n-m, компоненты которого являются результатами проверок на четность, заданными строками проверочной матрицы кода. Если линейный код задан проверочной матрицей H, то синдром S равен:

$$S = V \times H^T,$$

где V - вектор принятой из канала связи комбинации.

В циклическом коде, заданном порождающим многочленом g(x), синдром определяется как остаток от деления принятой комбинации V(x) на g(x). Синдром не зависит от переданной комбинации кода, а определяется только вектором ошибки. Если синдром комбинации равен 0, то комбинация ошибок не содержит. С помощью синдрома можно обнаруживать и исправлять ошибки.

Систематический код - избыточный код, в котором информационные и контрольные позиции четко разделены. За счет такого разделения в систематических кодах используются эффективные алгоритмические методы кодирования, декодирования, обнаружения и исправления ошибок, что затруднено при использовании несистематических кодов, в которых используются процедуры сопоставления кодовых комбинаций с набором известных.

Т **Трансформация сообщения** - событие состоящее в том, что одно сообщение под воздействием возникших в нем ошибок переходит в некоторое другое сообщение. Для уменьшения вероятности трансформации сообщений переходят к избыточным кодам. При одном и том же уровне избыточности вероятность трансформации сообщений будет меньше, если вся избыточность будет направлена только на обнаружение ошибок, а не на их исправление.

Х **Хемминга код** - линейный блочный (n, m) код, в котором проверочные символы располагаются на позициях, номера которых соответствуют целым степеням двойки (1,2,4,….). *Обычный код Хемминга* имеет минимальное кодовое расстояние d = 3 и в режиме коррекции способен исправлять любые ошибки, а в режиме только обнаружения - обнаруживать двойные. За счет специальной организации проверочных уравнений десятичный эквивалент синдрома в коде Хемминга указывает номер ошибочной позиции в комбинации. *В*

модифицированном коде Хемминга к обычному коду добавляется еще одна проверочная позиция, формируемая как общая проверка всей комбинации на четность. За счет этого минимальное кодовое расстояние становится равным 4, и код в режиме коррекции способен не только исправлять одиночные ошибки, но и обнаруживать двойные. В режиме только обнаружения этот код способен обнаруживать до трех ошибок.

Ц **Циклический код** - подмножество линейных блочных кодов. Если для любой комбинации кода ее циклический сдвиг также является комбинацией этого кода, то такой код называется циклическим. Циклический (n, m) код обычно задается порождающим многочленом g(x) степени n-m, являющимся делителем многочлена $x^n + 1$. Все комбинации циклического кода делятся на порождающий многочлен без остатка. Если остаток (его называют синдромом) не равен 0, то комбинация содержит ошибки. Анализ синдрома может помочь исправить ошибки в принятой из канала связи комбинации. С порождающим многочленом связан проверочный многочлен. С помощью порождающего и проверочного многочленов можно построить порождающие и проверочные матрицы циклического кода, а также построить кодирующие и декодирующие регистры.

Ш **Шум** - воздействия, искажающие сигнал, несущий полезную информацию в устройствах связи, управления, измерения, вычислительной техники. Синонимом этого термина является слово помехи. Под воздействием помех в информации возникают ошибки. Для борьбы с помехами используют различные физические методы (увеличение мощности передаваемых сигналов, специальные методы их приема и т.д.). Наряду с ними широко используются информационные методы борьбы с ошибками, основанные на использовании специальных методов кодирования и алгоритмов передачи информации (обнаруживающие и корректирующие ошибки коды, системы передачи с обратной связью и т.д.).

Э **Энтропия** - мера неопределенности состояния системы. В статистической теории информации энтропия используется в качестве меры количества информации. Энтропия ансамбля из N событий, имеющихся вероятности p_i (i = 1, 2, ….N), равна:

$$H = -\sum_{i=1}^{N} P_i \cdot \log_2 P_i .$$

Максимального значения энтропия достигает при равновероятных событиях. При этом она равна:

$$H_{max} = \log_2 N$$

Энтропия сообщения - или энтропия источника сообщений - количество информации, в среднем приходящееся на каждое из M сообщений с вероятностями p_{oi} (i = 1, 2, . . ., M). Энтропия сообщения равна:

$$H(X_o) = -\sum_{i=1}^{k} P_{oi} \cdot \log_2 P_{oi}$$

Энтропия источника или энтропия источника символов - это количество информации, в среднем приходящееся на каждый из K символов, появляющееся на выходе кодирующего устройства с вероятностями $p(X_i)$ Энтропия источника символов равна:

$$H(X) = -\sum_{i=1}^{k} P(X_i) \cdot \log_2 P(X_i) .$$

Энтропия источника достигает максимума при равновероятных символах и равна:

$$H_{max}(X) = \log_2 K.$$

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. WINDOWS 2000: server и professional, русские версии. Александр Андреев, Егор Беззубов и др. под общей редакцией Алексеея Чекмарева и Дмитрия Вишнякова. СПб. – C^bhv, 2000. – 1056 с.
2. Аппаратные средства и организация персонального компьютера. Учебное пособие. Дудкин Г.А., Кондратьев Д.Д., Неклюдов С.Ю. / Под редакцией С.Ю. Неклюдова - СПб.: СПбГУВК, 2004. - 585с.
3. Баричев С., Плотников О. Ваш Offis 2000 – М.:КУДИЦ – ОБРАЗ, 2000 – 320с.
4. Белунцов В. Компьютер для музыканта. Самоучитель. - СПб.: "Питер", 2001.
5. Белунцов В. Музыкальные возможности компьютера: справочник. - СПб.: "Питер", 2000.
6. Бэрри Нанс. Компьютерные сети. - М.: БИНОМ" 1996.
7. В.Байков. Интернет от E-mail к WWW. – СПб.: BHV, 2000.
8. Воген Т. Мультимедиа. - Минск: ООО "Попурри", 1997.
9. Вычислительные системы, сети и телекоммуникации: Учебник/ А.П. Пятибратов, Л.П. Гудыно, А.А. Кириченко; Под ред. А.П. Пятибратова. – М: Финансы и статистика, 2002, 512 с.
10. Гаскаров Д.В., Истомин Е.П., Фролов А.К. Информационная поддержка систем экологического контроля и управления – СПб.:СПГУВК, 1999 – 256с.
11. Грибов Д.Е. Macromedia Flash 4. Интерактивная веб-анимация - М.: ДМК Пресс, 2000.
12. Гук М. Аппаратные средства IBM PC. Энциклопедия. – СПб:, Питер, 2003. 928с.
13. Гук М. Дисковая подсистема ПК. СПб: Питер, 2001.
14. Гук М. Интерфейсы ПК. Справочник. СПб: Питер Ком, 1999.
15. Гук М., Юров В. Процессоры Pentium 4, Pentium III, Athlon и Duron. СПб: Питер, 2001.
16. Гук М., Юров В. Процессоры Pentium III, Athlon и другие. СПб: Питер, 2000.
17. Дарнелл Р.. JavaScript: справочник. – СПб: Питер, 1998.
18. Дж. Мейнджер. JavaScript: основы программирования. – Киев: BHV, 1999.
19. Джой Крейнан, Джой Хебрейкен. Энциклопедия Интернет. - СПб: Питер, 1999.
20. Дудкин Г.А., Д.Д. Кондратьев, С.Ю. Неклюдов, Б.Н. Попов. Выполнение лабораторных работ по курсу: «Мировые информационные системы»: Методическое пособие. - СПб.: СПбГУВК, 2000 – 156 с.
21. Дудкин Г.А., Д.Д. Кондратьев, С.Ю. Неклюдов. Аппаратные средства и организация персонального компьютера. Учебное пособие. / Под редакцией С.Ю. Неклюдова - СПб.: СПбГУВК, 2004. - 585с.
22. Дудкин Г.А., Кондратьев Д.Д., Неклюдов С.Ю. Архитектура IBM-совместимых персональных компьютеров: Учебное пособие. /Под ред. С.Ю. Неклюдова. – СПб: СПГУВК, 2001, 301, с.
23. Дэйв Гиббонс, Дэвид Фокс, Дик Крэвенс, Эндрю Б. Шафран. Работа в E-Mail. - М.: БИНОМ, 1996.
24. Евсеев Г. Мауа 3.0 Анимация и специальные эффекты - М.: "ДЕСС-КОМ", 2001.
25. Евсеев Г. Мауа 3.0 Трехмерная графика и анимация. - М.: "ДЕСС-КОМ", 2001.
26. Зельднер Г., А. Пешков, В. Юдаков. Компьютер на связи (Факс-модемы, мо-

демы, глобальные сети, E-Mail). - М.: ABF, 1996.
27. Золотов С. Протоколы Internet. – М. - СПб.: BHV 1998.
28. Информационно-измерительная техника: Учеб. пособие для вузов/ Н.Г. Вострокнутов, Н.Н. Евтихиев. – М: Высшая школа, 1977, 232 с.
29. Информационно-телеметрические системы. Ч.2. Помехоустойчивость аналоговых и цифровых систем передачи телеметрической информации: Учеб. пособие/ А.А. Чертков, Г.И. Козырев. – СПб: ВИКУ, 2001, 54 с.
30. Истомин Е.П., Гаскаров Д.В., Кутузов О.И. Сетевые модели распределенных автоматизированных систем - СПб.: Энергоатомиздат Санкт-Петербургское отделение, 1998 – 354с.
31. Истомин Е.П., Неклюдов С.Ю. Программирование на алгоритмических языках высокого уровня. Учебник. - СПб.: Изд-во Михайлова, 2003. - 719с.
32. Карпенко С. Internet в вопросах и ответах. - СПб.: BHV, 2000.
33. Каскадные коды. Форни Д. – М: Мир, 1970, 380 с.
34. Кирсанов Д. Понятный Интернет. - СПб: Символ, 1998.
35. Кларк Дж. мл., Кейн Дж. Кодирование с исправлением ошибок в системах цифровой связи: Пер. с англ. – М: Радио и связь, 1987, 392 с.
36. Кобурн Ф., Маккормик П. Эффективная работа с CorelDRAW 8. - СПб.: "Питер", 1998.
37. Коды исправляющие ошибки. Питерсон У., Уэлдон Э. – М: Мир, 1976, 360с.
38. Колесниченко О.В., Шишигин И.В. Аппаратные средства PC. 3-издание, переработанное и дополненное. – СПб.: BHV – Санкт-Петербург, 1999, 800с.
39. Компьютер на связи (Факс-модемы, модемы, глобальные сети, E-Mail). Г. Зельднер, А. Пешков, В. Юдаков. М.: "ABF", 1996, 344с.
40. Кондратьев Д.Д., Неклюдов С.Ю. Локальные вычислительные сети. Учебное пособие.– СПб: СПбГУВК, 2002, 342 с.
41. Космические радиотехнические комплексы: Учебник для вузов/ В.В. Гладченко, А.А. Корниенко, И.Ю. Лютынский В.С., Марков и др. Под ред. Г.В. Стогова. – МО СССР, 1986, 626 с.
42. Кречман Д.Л., Пушков А.И. Мультимедиа своими руками. - СПб.: БХВ, 1999.
43. Кристиан Крамлиш. INTERNET для занятых. - СПб: Питер, 1997.
44. Ксенакис Д., Лондон Ш. PhotoShop 5 для профессионалов. - СПб.: "Питер", 1999.
45. Кузнецов И. Р. Анимация для Интернета: краткий курс. - СПб.: "Питер", 2001.
46. Кулаков В. Программирование на аппаратном уровне. Специальный справочник. СПб: Питер, 2001.
47. Кулибанов Ю.М., Неклюдов С.Ю., Сорокин Л.М., Полякова Н.А.. Приложения интегрированной программной среды MS-OFFICE 97: EXCEL, ACCESS, WORD. Тексты лекций. - СПб.: СПбГУВК, 1999.
48. Курс теории информации. В.Д. Колесник, Г.Ш. Полтырёв. – М.: Наука, 1982, 416 с.
49. Кэролайн Холидей. Секреты ПК. Киев: Изд-во "Диалектика", 1995, 413с.
50. Лагутенко О.И. Современные модемы.– М: Эко-Трендз, 2002, 346 с.
51. Луиза Паттерсон. Использование HTML. Ясно. Кратко. Надежно. - М. – СПб. – Киев: BHV,:1998.
52. Лэмонт Вуд. WEB -графика. Справочник. - СПб: Питер, 1998.
53. Манылов М. Macromedia Flash 4: учебный курс. - СПб.: "Питер", 2000.
54. Маров М. 3D Studio MAX 2.5: справочник. - СПб.: "Питер", 1999.
55. Маров М. 3D Studio MAX 3: учебный курс. - СПб.: "Питер", 2000.

56. Математическое обеспечение сетей передачи данных/ Ю.М. Мартынов, А.М. Крюков, В.Л. Разгон; Под ред. Ю.М. Мартынова. – М: Радио и связь, 1986, 288 с.
57. А. Матросов, А. Сергеев, М. Чаунин. HTML 4.0. – СПб.: BHV, 2000.
58. Миронов Д. CorelDRAW 9: учебный курс. - СПб.: "Питер", 1999.
59. Мишель Петровски. Microsoft INTERNET INFORMATION SERVER 4.0 - Киев: BHV, 1998.
60. Мультимедиа / Под ред. А. И. Петренко. Киев, 1994.
61. Неклюдов С.Ю., Чертков А.А. Системы передачи информации: Учебное пособие. - СПб.: СПбГУВК, 2003.- 298с.
62. Нортон П. Персональный компьютер: аппаратно-программная реализация. Книга 1. – СПб.: BHV – Санкт-Петербург, 2000, 848с.
63. Орлов А. М. Виртуальная реальность. Пространство экранных культур как среда обитания. - М., 1998.
64. Основы теории и расчета информационно-измерительных систем. Новоселов О.Н., Фомин А.Ф. – М: Машиностроение, 1991, 336 с.
65. Основы теории информации и кодирования: /Учебник для вузов/ И.В. Кузьмин, В.А. Кедрус. – Киев: "Вища школа", 1977, 280 с.
66. Пасько В.. Word 97 (русифицированная версия). - Киев: BHV, 1998.
67. Паттон Б., Франклин Д. Flash 4. Анимация в Интернете - СПб.: "Символ-Плюс", 2000.
68. Пенин П.И., Филиппов Л.И. Радиотехнические системы передачи информации: Учеб. пособие для вузов. – М.: Радио и связь, 1984. – 256 с.
69. Передающее устройство радиотехнических систем: Учеб. пособие/ Л.С. Дмитриев, Ю.В. Петелин, Е.П. Ряховский, А.А. Чертков и др. – МО РФ, 1993, 204 с.
70. Петелин Р. Звуковая студия в PC. - СПб.: БХВ, 1998.
71. Петелин Р. Персональный оркестр в PC. - СПб.: БХВ, 1998.
72. Петерсон М., Ларри М. Эффективная работа с 3D Studio MAX 2. СПб.: "Питер", 1999.
73. Помехоустойчивость цифровых систем передачи телемеханической информации. Юргенсон Р.И. – Л.: 1971, 152 с.
74. Преобразование и передача информации в АСУ. /Учебник для вузов/ Четвериков В.Н. – М: Высшая школа, 1974, 320 с.
75. Приписнов Д. Моделирование в 3D Studio MAX 3.0. - СПб.: БХВ, 2000.
76. Проектирование и техническая эксплуатация радиопередающих устройств: Учеб. пособие для вузов/ М.А. Сиверс, Г.А. Зейтленок, Ю.Б. Несвижский и др. – М: Радио и связь, 1989, 367 с.
77. Проектирование и техническая эксплуатация систем передачи: Учеб. пособие для вузов/ И.Р. Берганов, В.Н. Гордиенко, В.В. Крухмалев. – М: Радио и связь, 1989, 272 с.
78. Пятибратов А.П., Гудыно Л.П., Кириченко А.А. Вычислительные системы, сети и телекоммуникации: Учебник/ Под ред. А.П. Пятибратова. – М: Финансы и статистика, 2002, 512 с.
79. Р. Борланд. Эффективная работа с Microsoft Word 97-98. – СПб.: Питер, 1998.
80. Работа в E-Mail. Д. Гиббонс, Д. Фокс, Д. Крэвенс, Э. Б. Шафран. М.: "БИНОМ", 1996, 333с.
81. Радиопередающие устройства. Учебник для вузов/ Под ред. В.В. Шахгильдяна. – М: Радиосвязь, 1994, 430с.
82. Резников Ф.А. Видеомонтаж на персональном компьютере. Adobe Premiere

5.5 и Adobe After Effects 4.1: практическое пособие. - М.: "Триумф", 2000.
83. Роберт Муллен. HTML 4 для Internet Explorer 4 и Netscape Communicator 4. Справочник. - СПб: Питер, 1998.
84. Рудометов Е., Рудометов В. Аппаратные средства и мультимедиа: справочник. - СПб.: "Питер", 1999.
85. Рудометов Е., Рудометов В. Архитектура ПК, комплектующие, мультимедиа. - СПб.: "Питер", 2000. - 416 с.
86. Рудометов Е., Рудометов В. Материнские платы и чипсеты, 2-е издание. СПб: Питер, 2001.
87. Рудометов Е., Рудометов В. Устройство мультимедийного компьютера. - СПб.: "Питер", 2001.
88. Сборник работ по теории информации и кибернетики. Шеннон К. – М: Иностранная литература, 1963, 438с.
89. Системы передачи информации от терминалов к ЦВМ. Советов Б.Я., Рухман Е.Л., Яковлев С.А. – Л: Изд-во ЛГУ, 1978, 240 с.
90. Системы передачи информации: Учебное пособие. С.Ю. Неклюдов, А.А. Чертков. - СПб.: СПбГУВК, 2003.- 298с.
91. Системы радио-, радиорелейной и космической связи: Учебник для вузов/ А.А. Ланко, В.А. Емельянов, И.Т. Осташов. – Л: ВИКИ им. А.Ф. Можайского, 1981, 366 с.
92. Советов Б.Я. Информационные технологии. – М: Высшая школа, 1994.
93. Советов Б.Я. Теория информации.– Л: Изд-во ЛГУ, 1977, 184 с.
94. Современные модемы. Лагутенко О.И. – М: Эко-Трендз, 2002, 346 с.
95. Справочник по математике для научных работников и инженеров. Корн Г., Корн Т. – М: Наука, 1984, 832 с.
96. Спутниковая связь и вещание. Справочник. – 2-е изд. Г.Б. Аскинази, В.Л. Быков, М.Н. Дьячкова и др. Под ред. Кантора. – М: Радио и связь, 1988, 344 с.
97. Средства мобильной связи. Андрианов В.И., Соколов А.В. – СПб: БХВ-Петербург, 2001, 256 с.
98. Статистический контроль каналов связи. Коричнев Л.П., Королев В.Д. – М: Радио и связь, 1989, 240 с.
99. Стефен Л. Нельсон. INTERNET и WINDOWS 95. Иллюстрированный краткий справочник. Microsoft Corporation. – Вашингтон, 1996.
100. Страсницкас М. Эффективная работа с Photoshop 5. - СПб.: "Питер", 1999.
101. Тайц А. Adobe Illustrator 8: учебный курс. - СПб.: "Питер", 1999.
102. Теоретические основы информационной техники: Учеб. пособие для вузов/ Ф.Е. Темников, В.А. Афонин, В.И. Дмитриев. – М: Энергия, 1979, 512 с.
103. Теоретические основы информационных процессов: Учеб. пособие для вузов/Л.Ф. Куликовский, В.В. Мотов. – М: Высшая школа, 1987, 248 с.
104. Теоретические основы многоканальной связи. /Учебник для электротехнических институтов связи/ А.Ю. Лев. – М: Связь, 1978, 192 с.
105. Теория информации и надёжная связь. Галлагер Р. – М: Советское радио, 1974.
106. Теория информации. Электронный учебник. Воронов Ю.В., Советов Б.Я., Цехановский В.В. – СПб: СПГТУ, 1994.
107. Теория передачи сигналов: Учебник для вузов/ А.Г. Зюко, Д.Д. Кловский, М.В. Назаров, Л.М. Финк. – М: Радио и связь, 1986, 304 с.
108. Томпсон С., Элишер К. Осваиваем мультимедиа: Пер. с англ. М,, 1997.
109. Три подхода к определению понятия количества информации. Колмогоров

А.Н. – Проблемы передачи информации. 1965, т. 1, № 1.
110. Утилова Н. И. Монтаж как средство художественной выразительности. М., 1994.
111. Федорчук А. Как создаются Web-сайты: краткий курс. – СПб.: Питер, 2000.
112. Фролов М. Мультимедиа в примерах. - СПб.: БХВ, 1998.
113. Холмский Е.Г. Maya 3.0 Моделирование и анимация - М.: "Солон-Р", 2001.
114. Хоумер А., К. Улмен. Dynamic HTML: справочник – СПб.: Питер, 1999.
115. Цифровые методы в космической связи. Под ред. С. Голомба. Пер. с англ. под ред. В.И. Шляпоберского. – М: Связь, 1969, 271 с.
116. Цифровые методы в спутниковой связи. В.Л. Банкет, В.М. Дорофеев. – М: Радио и связь, 1988, 240 с.
117. Частотные характеристики букв: Методы прикладной математики в транспортных системах/ Сб. научных трудов. Выпуск 6. Кондратьев Д.Д., Неклюдов С.Ю. СПб: СПбГУВК, 2002.
118. Черил Кирк. Internet: Книга ответов. - СПб: Питер, 1998.
119. Шагурин И.И., Бердышев Е.М. Процессоры семейства INTEL P6 – Pentium II, Pentium III, Celeron и др. Архитектура, программирование, интерфейс. М: Изд-во "Горячая линия - Телеком", 2000, 248с.
120. Шарыгин М.Е. Сканеры и цифровые камеры. СПб: BHV-Санкт-Петербург, 2000, 382 с.

ИСТОМИН Евгений Петрович, 1953 года рождения, доктор технических наук, профессор по кафедре вычислительных систем и информатики. Имеет около 100 научных трудов, в том числе: 3 монографии, 5 учебников, 12 учебных пособий, 5 изобретений и более 70 научных статей и тезисов докладов. Область научных интересов – современные информационные технологии, теория управления государством и принятие решений в социально-экономических системах. Хобби – путешествие с семьей на автомобиле, литература и политика.

НЕКЛЮДОВ Сергей Юрьевич, 1951 года рождения, кандидат технических наук, профессор Санкт-Петербургского государственного гидрометеорологического университета. Имеет свыше 100 научных трудов, в том числе: 3 монографии, 4 учебника, 6 учебных пособий и более 80 научных статей и тезисов докладов. Читает курсы: Информатика, Вычислительная техника и программирование, Численные методы, Управление данными, Мультимедиа технологии и другие.

РОМАНЧЕНКО Владимир Иванович, 1960 года рождения, кандидат экономических наук, профессор морской государственной академии им. Адмирала Ушакова. Имеет около 30 научных трудов, в том числе: 1 монографию, 1 учебник, 3 учебных пособия и свыше 20 статей и тезисов докладов. Преподает дисциплины экономического и управленческого циклов, информационные технологии. Хобби – туризм, театр, литература.
