MINISTERUL EDUCAȚIEI AL REPUBLICII MOLDOVA

Анатол ГРЕМАЛЬСКИ
Юрие МОКАНУ
Лудмила ГРЕМАЛЬСКИ

ИНФОРМАТИКА

Учебник для 10 класса

Ştiinţa, 2012

CZU 004(075.3)
Г 80

Elaborat conform curriculumului disciplinar în vigoare şi aprobat prin Ordinul ministrului educaţiei (nr. 211 din 11 aprilie 2012). Editat din sursele financiare ale *Fondului Special pentru Manuale*.

Comisia de experţi: *Teodora Gherman*, dr. în pedagogie, conferenţiar, şef Catedră Tehnologii Informaţionale Aplicate, Academia de Administrare Publică de pe lîngă Preşedintele Republicii Moldova; *Gheorghe Chistruga*, prof. şc., grad did. superior, Liceul Teoretic „Mihai Eminescu”, Drochia; *Mihai Chigai*, prof. şc., grad did. I, Liceul Teoretic Caplani, rn. Ştefan-Vodă

Recenzenţi: *Gheorghe Ciocanu*, dr. habilitat în informatică, profesor universitar, Universitatea de Stat din Moldova; *Valeriu Cabac*, dr. în fizică şi matematică, conferenţiar universitar, Universitatea de Stat „Alecu Russo”, Bălţi; *Mihai Şleahtiţchi*, dr. în psihologie şi pedagogie, conferenţiar universitar, Universitatea Liberă Internaţională din Moldova; *Tatiana Cartaleanu*, dr. în filologie, conferenţiar universitar, Universitatea Pedagogică de Stat „Ion Creangă”, Chişinău; *Alexei Colîbneac*, Maestru în Arte, profesor universitar, Academia de Muzică, Teatru şi Arte Plastice, Chişinău

Traducere din limba română: *Arcadie Malearovici* (capitolele 2–7), *Veronica Musteaţă* (capitolul 1)
Responsabil de ediţie: *Valentina Rîbalchina*
Redactor: *Tatiana Bolgar*
Redactor tehnic: *Nina Duduciuc*
Machetare computerizată: *Anatol Andriţchi*
Copertă: *Vitaliu Pogolşa*

Întreprinderea Editorial-Poligrafică *Ştiinţa*,
str. Academiei, nr. 3; MD-2028, Chişinău, Republica Moldova;
tel.: (+373) 022-73-96-16; fax: (+373) 022-73-96-27;
e-mail: prini@stiinta.asm.md

DIFUZARE:

ÎM Societatea de Distribuţie a Cărţii *PRO-NOI*
str. Alba-Iulia, nr. 23/1A; MD-2051, Chişinău;
tel.: (+373) 022-51-68-17, 51-57-49; fax: (+373) 022-50-15-81;
e-mail: info@pronoi.md; www.pronoi.md

Descrierea CIP a Camerei Naţionale a Cărţii

Гремальски, Анатол

Информатика: Учеб. для 10 класса / Анатол Гремальски, Юрие Мокану, Лудмила Гремальски; trad. din lb. rom.: Arcadie Malearovici, Veronica Musteaţă; Min. Educaţiei al Rep. Moldova. – Ch.: Î.E.P. *Ştiinţa*, 2012 (Tipografia „SEREBIA” SRL). – 188 p.

ISBN 978-9975-67-821-6
004(075.3)

ISBN 978-9975-67-821-6

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Впечатляющие достижения в области информатики, создание суперкомпьютеров и персональных компьютеров, появление киберпространства, виртуальной реальности и ИНТЕРНЕТа предполагают углубленные знания принципов работы и устройства современных компьютеров. Именно эти знания будут надежным путеводителем в постоянно меняющемся мире.

Данный учебник предназначен для усвоения учащимися знаний, необходимых для понимания процессов автоматизированной обработки данных с помощью цифровых компьютеров.

Глава 1 содержит теоретические и прикладные знания по определению и обработке составных типов данных: массивов, строк, записей, множеств и файлов. В ней также представлены методы создания и обработки файлов: связывание файловых переменных ПАСКАЛЯ с внешними файлами, запись и считывание компонент файлов.

Глава 2 содержит изложение фундаментальных знаний из теории информации: количество информации, содержащейся в непрерывных и дискретных сообщениях, кодирование и декодирование информации, представление информации в компьютере.

В **главе 3** изложены фундаментальные сведения из области компьютерной арифметики: системы счисления и арифметические операции в двоичной системе счисления, представление натуральных, целых и вещественных чисел в компьютере.

Глава 4 содержит основы булевой алгебры. В ней излагаются понятия булевой переменной, константы и функции, а также изучаются часто используемые логические функции.

В **главе 5** рассматриваются комбинационные и последовательностные схемы, широко применяемые в любом цифровом компьютере: сумматоры, компараторы, шифраторы и дешифраторы, регистры, счетчики, генераторы импульсов.

Устройство и работа компьютера описаны в **главе 6**. Материал изложен таким образом, чтобы устройство современного цифрового компьютера было понято и усвоено методически: от логических элементов, устройств и блоков до вычислительной системы в целом. Особое внимание уделяется взаимной зависимости математических концепций и методов физической реализации цифровых устройств современных вычислительных систем, взаимосвязи технического и программного обеспечения компьютера.

В **главе 7** рассматриваются компьютерные сети. Изложены технологии взаимодействия в сети, топология и архитектура локальных, региональных и глобальных сетей. Для облегчения исследования киберпространства данная глава содержит фундаментальные сведения об ИНТЕРНЕТе и услугах сети: передаче файлов, электронной почте, *Web*-странице.

Учебник разработан в соответствии с *Куррикулумом по информатике*, утвержденным приказом Министерства Образования Республики Молдова № 244 от 27 апреля 2010 года. Распределение материала по профилям – гуманитарный и реальный – приведено в содержании учебника.

Глава 1

СОСТАВНЫЕ ТИПЫ ДАННЫХ

1.1. Тип данных *массив* (array)

Множество значений типа данных **array** состоит из массивов (таблиц). Массивы составляются из фиксированного числа компонент одного и того же типа, который называется **базовым**. Ссылка на компоненты осуществляется с помощью **индексов**.

Тип данных *массив* определяется конструкцией вида

```
type <Имя типа> = array [T1] of T2;
```

где T_1 – тип индекса, который должен быть порядковым, а T_2 – тип компонент (базовый), который может быть любым.

Примеры:

1)
```
type Vector = array [1..5] of real;
var  x : Vector;
```

2)
```
type Zi = (L, Ma, Mi, J, V, S, D);
     Venit = array [Zi] of real;
var  v : Venit;
     z : Zi;
```

3)
```
type Ora = 0..23;
     Grade = -40..40;
     Temperatura = array [Ora] of Grade;
var  t : Temperatura;
     h : Ora;
```

Структура данных, используемых в этих примерах, представлена на *рис. 1.1*.

Доступ к компонентам переменной типа *массив* осуществляется явно через имя переменной, за которой следует соответствующий индекс, заключенный в квадратные скобки.

Примеры:

1)
```
x[1], x[4];
```

2)
```
v[L], v[Ma], v[J];
```

3)
```
t[0], t[15], t[23];
```

4)
```
v[z], t[h].
```

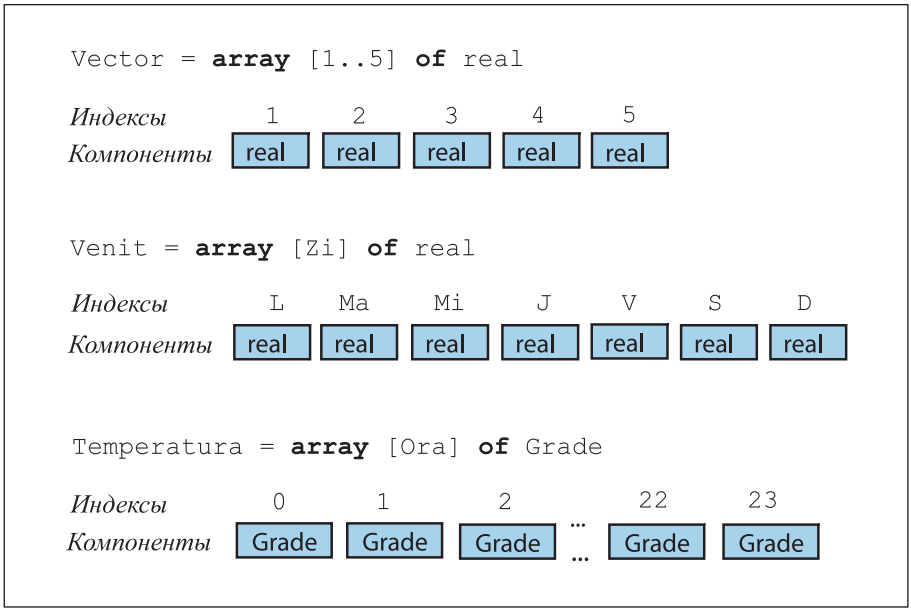


Рис. 1.1. Структура данных типа `Vector`, `Venit` и `Temperatura`

К компонентам данных типа *массив* можно применять все операции, допустимые для соответствующего базового типа. Следующая программа выводит на экран сумму компонент переменной `x` типа `Vector`. Значения переменных `x[1]`, `x[2]`, ..., `x[5]` вводятся с клавиатуры.

```
Program P77;
{ Сумма компонент переменной x типа Vector }
type Vector = array [1..5] of real;
var x : Vector;
    i : integer;
    s : real;
begin
  writeln('Введите 5 чисел:');
  for i:=1 to 5 do readln(x[i]);
  writeln('Были введены:');
  for i:=1 to 5 do writeln(x[i]);
  s:=0;
  for i:=1 to 5 do s:=s+x[i];
  writeln('Сумма=', s);
  readln;
end.
```

Для того чтобы расширить область применения программы, количество компонент данных типа **array** рекомендуется указывать через константы.

Например, программу `P77` можно изменить таким образом, чтобы она вычисляла сумму n действительных чисел, $n \le 100$:

```
Program P78;
{ Расширение области применения программы P77 }
const nmax = 100;
type Vector = array [1..nmax] of real;
var x : Vector;
    n : 1..nmax;
    i : integer;
    s : real;
begin
  write('n='); readln(n);
  writeln('Введите ', n, ' чисел:');
  for i:=1 to n do readln(x[i]);
  writeln('Были введены:');
  for i:=1 to n do writeln(x[i]);
  s:=0;
  for i:=1 to n do
    s:=s+x[i];
  writeln('Сумма=', s);
  readln;
end.
```

Двумерные массивы определяются с помощью конструкции вида:

type <*Имя типа*> **= array** [T_1, T_2] **of** T_3;

где T_1 и T_2 указывают тип индексов, а T_3 – тип компонент.

В качестве примера на *рис. 1.2* представлена структура данных типа:

```
Matrice = array [1..3, 1..4] of real
```

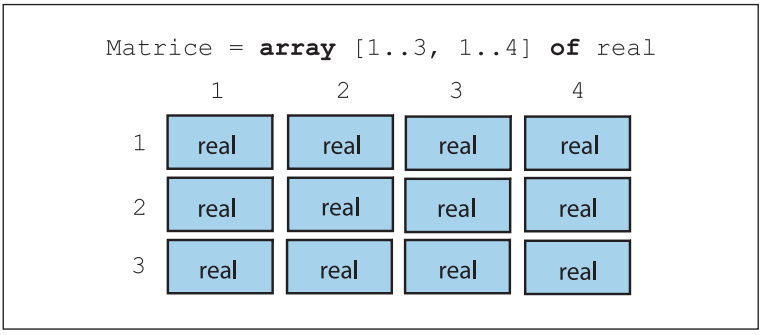


Рис. 1.2. Структура данных типа `Matrice`

Доступ к компонентам переменной типа *двумерный массив* осуществляется явно через имя переменной, за которой следуют соответствующие индексы, разделенные запятой и заключенные в квадратные скобки.

Например, при описании

```
var m : Matrice;
```

обозначение m[1,1] означает ссылку на компоненту, расположенную в первой строке и в первом столбце (см. *рис. 1.2*); обозначение m[1,2] означает ссылку на компоненту, расположенную в первой строке и втором столбце; обозначение m[i,j] означает ссылку на компоненту, расположенную в строке *i* и в столбце *j*.

Следующая программа выводит на экран сумму компонент переменной m типа Matrice. Значения компонент m[1,1], m[1,2], ..., m[3,4] вводятся с клавиатуры.

```
Program P79;
{ Сумма компонент переменной m типа Matrice }
type Matrice = array [1..3, 1..4] of real;
var m : Matrice;
    i, j : integer;
    s : real;
begin
  writeln('Введите компоненты m[i,j]:');
  for i:=1 to 3 do
    for j:=1 to 4 do
      begin
        write('m[', i, ',', j, ']=');
        readln(m[i,j]);
      end;
  writeln('Были введены:');
  for i:=1 to 3 do
    begin
      for j:=1 to 4 do write(m[i,j]);
      writeln;
    end;
  s:=0;
  for i:=1 to 3 do
    for j:=1 to 4 do
      s:=s+m[i,j];
  writeln('Сумма=', s);
  readln;
end.
```

В общем виде тип ***n*-мерный массив** (*n* = 1, 2, 3 и т.д.) определяется с помощью синтаксических диаграмм, приведенных на *рис. 1.3*. Слово **packed** (упакованный) указывает компилятору, что область памяти для элементов типа **array** должна быть выделена с применением оптимизации. Отметим, что в большинстве компьютеров использование этого префикса необязательно, так как оптимизация осуществляется автоматически.

Если даны две переменные типа *массив* одного и того же базового типа, то имена этих переменных могут встречаться в операциях присваивания. Такое присваивание означает копирование всех компонент массива, расположенного в правой части, в массив, расположенный в левой части.

Например, при описании

```
var a, b : Matrice;
```

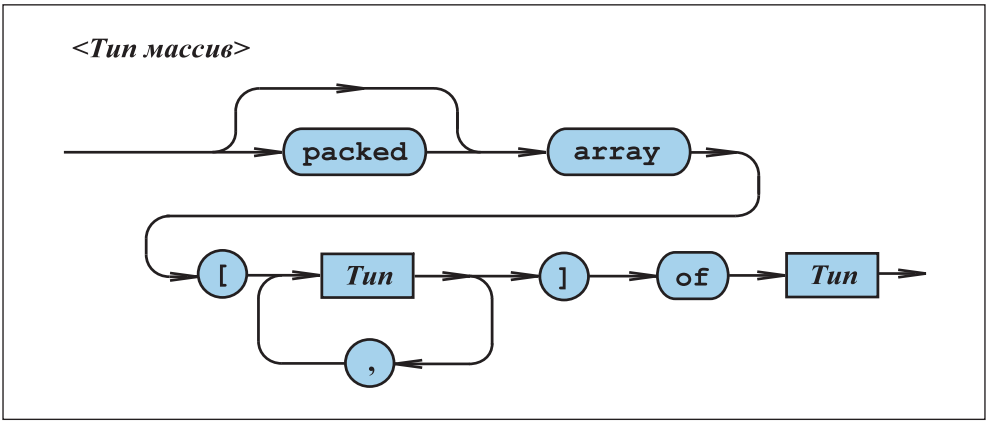


Рис. 1.3. Синтаксическая диаграмма <*Тип массив*>

оператор

```
a:=b
```

является правильным.

В примерах, рассмотренных выше, базовый тип (тип компонент) всегда был простым. Так как базовый тип в большинстве случаев может быть любым, то не исключена возможность определения массивов, компоненты которых относятся к составному типу. Рассмотрим пример, в котором базовым типом является сам тип **array**.

```
Type Linie = array [1..4] of real;
     Tabel = array [1..3] of Linie;
var L : Linie;
    T : Tabel;
    x : real;
```

Переменная `T` состоит из 3-х компонент: `T[1]`, `T[2]` и `T[3]` типа `Linie`.

Переменная `L` состоит из 4-х компонент: `L[1]`, `L[2]`, `L[3]` и `L[4]` типа `real`.

Следовательно, операции присваивания

```
L[1]:=x; x:=L[3]; T[2]:=L; L:=T[1]
```

являются правильными.

Доступ к элементам переменной `T` может осуществляться через `T[i][j]` или `T[i,j]`. Здесь индекс `i` означает номер компоненты типа `Linie` переменной `T`, а `j` – номер компоненты типа `real` компоненты `T[i]` типа `Linie`.

Отметим, что объявления вида

array [T_1, T_2] **of** T_3

и

array [T_1] **of array** [T_2] **of** T_3

определяют различные типы данных.

Первое объявление определяет двумерные массивы с компонентами типа T_3. Второе – определяет одномерные массивы с компонентами типа **array** [T_2] **of** T_3.

В программах на языке ПАСКАЛЬ массивы используются для группировки под одним именем нескольких переменных, обладающих одинаковыми характеристиками.

Вопросы и упражнения

❶ Определите тип индексов и тип компонент в следующих объявлениях:

```
type P = array [1..5] of integer;
     Culoare = (Galben, Verde, Albastru, Violet);
     R = array [Culoare] of real;
     S = array [Culoare, 1..3] of boolean;
     T = array [boolean] of Culoare;
```

Нарисуйте структуры данных типа P, R, S и T (*рис. 1.1* и *1.2*).

❷ Укажите на синтаксической диаграмме *рис. 1.3* пути, которые соответствуют объявлениям из упражнения 1.

❸ Напишите металингвистические формулы, которые соответствуют синтаксической диаграмме *<Тип массив> рис. 1.3*.

❹ Даны описания:

```
type Vector = array [1..5] of real;
var x, y : Vector;
```

Напишите арифметическое выражение, значением которого является:
а) сумма первых трех компонент переменной x;
б) сумма всех компонент переменной y;
в) произведение всех компонент переменной x;
г) абсолютное значение третьей компоненты переменной y;
д) сумма первых компонент переменных x и y.

❺ Даны описания:

```
type Zi = (L, Ma, Mi, J, V, S, D);
     Venit = array [Zi] of real;
var  v : Venit;
```

Компоненты переменной v представляют собой ежедневный доход предприятия. Напишите программу, которая:
а) вычисляет еженедельный доход предприятия;
б) подсчитывает средний ежедневный доход;
в) определяет день, когда был получен наибольший доход;
г) определяет день, когда был получен наименьший доход.

❻ Даны описания:

```
type Ora = 0..23;
     Grade = -40..40;
     Temperatura = array [Ora] of Grade;
var  t : Temperatura;
```

Компоненты переменной t представляют собой значения температуры, измеряемой каждый час в течение 24 часов. Напишите программу, которая:
а) вычисляет среднюю температуру;
б) определяет минимальное и максимальное значения температуры;
в) определяет час (часы), в который была зарегистрирована максимальная температура;
г) определяет час (часы), в который была зарегистрирована минимальная температура.

❼ Даны описания:

```
type Oras = (Chisinau, Orhei, Balti, Tighina, Tiraspol);
     Zi = (L, Ma, Mi, J, V, S, D);
     Consum = array [Oras, Zi] of real;
var  C : Consum;
     r : Oras;
     z : Zi;
```

Компонента C[r,z] переменной C представляет собой потребление электроэнергии города r в день z. Напишите программу, которая:
а) вычисляет количество электроэнергии, потребляемой каждым городом за неделю;
б) вычисляет количество электроэнергии, потребляемой данными городами ежедневно;
в) определяет город с максимальным еженедельным потреблением электроэнергии;
г) определяет город с минимальным еженедельным потреблением электроэнергии;
д) определяет день, в который было потреблено наибольшее количество электроэнергии;
е) определяет день с наименьшим потреблением электроэнергии.

❽ Даны описания:

```
type Vector = array [1..5] of real;
     Matrice = array [1..3, 1..4] of real;
     Linie = array [1..4] of real;
     Tabel = array [1..3] of Linie;
var  V : Vector;
     M : Matrice;
     L : Linie;
     T : Tabel;
     x : real;
     i : integer;
```

Какие из следующих операций присваивания корректны?

a) `T[3]:=T[1];`

b) `M:=T;`

c) `L:=V;`

d) `L[3]:=x;`

e) `x:=i;`

f) `i:=x;`

g) `L[3]:=i;`

h) `i:=M[1,2];`

i) `x:=V[4];`

j) `L[3]:=V[4];`

k) `T[1]:=4;`

l) `T[2]:=V;`

m) `L:=T[3];`

n) `T[1,2]:=M[1,2];`

o) `T[1,2]:=M[1,2];`

p) `M[1]:=4;`

q) `M[1,3]:=L[2];`

r) `x:=T[1][2];`

s) `x:=M[1];`

t) `L:=M[1];`

u) `V[5]:=M[3,4];`

v) `L:=M[3,4].`

❾ Используя тип данных *массив*, напишите программу, которая реализует алгоритм Эратосфена для вычисления простых чисел, меньших заданного n ($n \le 200$).

1.2. Тип данных *строка символов*

В стандартном языке тип данных *строка символов* является частным случаем типа **array** и определяется конструкцией вида

```
<Имя типа> ::= packed array [1..n] of char;
```

Множеством значений данного типа являются все строки, содержащие ровно n символов.

Пример:

```
Program P80;
{ Строки символов с фиксированной длиной }
type Nume = packed array [1..8] of char;
     Prenume = packed array [1..5] of char;
var  N : Nume;
     P : Prenume;
begin
  N:='Munteanu';
  P:='Mihai';
  writeln(N);
  writeln(P);
  readln;
end.
```

Результат, выводимый на экран:

```
Munteanu
Mihai
```

Так как строки различной длины принадлежат разным типам данных, в программе P80 не допустимы присваивания вида:

```
N:= 'Olaru';
P:= 'Ion'.
```

В таких случаях программисту необходимо заполнить соответствующее пространство пробелами для того, чтобы в строке было ровно n символов, например:

```
N:= 'Olaru ';
P:= 'Ion '.
```

Значения любой переменной v типа **packed array** [1..n] **of** char можно ввести с клавиатуры только путем поочередного считывания соответствующих компонент:

```
read(v[1]); read(v[2]); ...; read(v[n]).
```

Однако вывести всю строку на экран можно с помощью одного вызова write(v) или writeln(v).

Особо выделим тот факт, что строки символов типа **packed array** [1..n] **of** char содержат ровно n символов, т. е. являются **строками постоянной длины**. Их длину нельзя изменять в процессе выполнения соответствующей программы. Это усложняет создание программ, предназначенных для обработки строк символов произвольной длины.

Чтобы устранить указанный недостаток, современные версии языка ПАСКАЛЬ разрешают использование строк символов произвольной длины.

В версии Turbo PASCAL тип данных *строка символов*, множеством значений которого являются **строки произвольной длины**, определяется с помощью конструкции вида:

```
type <Имя типа> = string;
```

или

```
type <Имя типа> = string [nmax];
```

где *nmax* – максимальная длина, которую могут иметь соответствующие строки. При отсутствии параметра *nmax* максимальная длина устанавливается по умолчанию, как правило – 255 символов.

К строкам типа **string** можно применять операцию конкатенации (склеивания), обозначаемую знаком «+». Текущую длину любой переменной v типа **string** можно узнать с помощью стандартной функции length(v), которая возвращает значение типа integer. Независимо от длины все строки символов типа **string** являются совместимыми.

Пример:

```
Program P81;
{ Строки символов произвольной длины }
type Nume = string [8];
     Prenume = string [5];
     NumePrenume = string;
var N : Nume;
    P : Prenume;
    NP : NumePrenume;
    L : integer;
```

```
begin
  N:='Munteanu';   L:=length(N);    writeln(N, L:4);
  P:='Mihai';      L:=length(P);    writeln(P, L:4);
  NP:=N+' '+P;     L:=length(NP);   writeln(NP, L:4);
  N:='Olaru';      L:=length(N);    writeln(N, L:4);
  P:='Ion';        L:=length(P);    writeln(P, L:4);
  NP:=N+' '+P;     L:=length(NP);   writeln(NP, L:4);
  readln;
end.
```

Результаты, выводимые на экран:

```
Munteanu   8
Mihai   5
Munteanu Mihai  14
Olaru   5
Ion   3
Olaru Ion 9
```

Отметим, что в процессе выполнения данной программы длина строк символов N, P и NP изменяется.

К строкам символов можно применять операции отношения <, <=, =, >=, >, < >. Строки сравниваются посимвольно, слева направо, в соответствии с порядковыми номерами символов типа данных char. Оба операнда должны относиться к типу **packed array** [1..n] **of** char с одинаковым числом компонент либо к типу **string**. Естественно, что операнды типа **string** могут быть различной длины.

Например, результатом операции

```
'AC' < 'BA'
```

является true, а результатом операции

```
'AAAAC' < 'AAAAB'
```

является false.

Переменную типа *строка символов* можно использовать полностью или частично, обращаясь к отдельному символу строки.

Например, в строке P='Mihai' имеем P[1]='M', P[2]='i', P[3]='h' и т. д. После выполнения последовательности операторов

```
P[1]:='P';
P[2]:='e';
P[3]:='t';
P[4]:='r';
P[5]:='u'
```

переменная P примет значение 'Petru'.

Следующая программа вводит с клавиатуры произвольные строки символов и выводит на экран количество пробелов в соответствующей строке. Работа программы завершается после введения строки 'Sfîrşit'.

```
Program P82;
{ Количество пробелов в строке символов }
var S : string;
    i, j : integer;
begin
  writeln('Введите строку символов:');
  repeat
    readln(S);
    i:=0;
    for j:=1 to length(S) do
      if S[j]=' ' then i:=i+1;
    writeln('Количество пробелов=', i);
  until S='Sfarsit';
end.
```

Вопросы и упражнения

❶ Как определяется тип данных *строка символов*?

❷ Какие операции можно применять к строкам символов?

❸ Прокомментируйте следующую программу:

```
Program P83;
{ Ошибка }
var S : packed array [1..5] of char;
begin
  S:='12345';
  writeln(S);
  S:='Sfat';
  writeln(S);
end.
```

❹ Напишите программу, которая:

а) определяет, сколько раз встречается в строке символ 'A';
б) заменяет символ 'A' символом '*';
в) удаляет из строки символ 'B';
г) определяет, сколько раз встречается в строке слог 'MA';
д) заменяет слог 'MA' слогом 'TA';
е) удаляет из строки слог 'TO'.

❺ Определите результаты операций отношения:

a) `'B' < 'A';`
b) `'BB' > 'AA';`
c) `'BAAAA' < 'AAAAA';`
d) `'CCCCD' > 'CCCCA';`
e) `'A A' = 'AA';`
f) `'BB ' < 'B B';`
g) `'A' = 'a';`
h) `'Aa' > 'aA';`
i) `'123' = '321';`
j) `'12345' > '12345'.`

❻ Даны строки символов, состоящие из прописных букв латинского алфавита и пробелов. Напишите программу, которая выводит на экран данные строки согласно следующим правилам:
– все буквы от 'A' до 'Y' заменяются последующими буквами алфавита;
– каждая буква 'Z' заменяется буквой 'A';
– все пробелы заменяются знаком '-'.

❼ Напишите программу, которая расшифровывает строки символов, зашифрованные согласно правилам из упражнения 6.

❽ Дано m ($m \le 100$) строк, состоящих из строчных букв латинского алфавита. Напишите программу, которая выводит на экран данные строки в алфавитном порядке.

❾ Строка S составлена из нескольких предложений, каждое из которых заканчивается точкой, восклицательным или вопросительным знаком. Напишите программу, которая выводит на экран количество предложений в данной строке.

1.3. Тип данных *запись* (record)

Множество значений типа данных **record** состоит из записей. Записи состоят из компонентов, называемых ***полями***. В отличие от элементов массива поля могут относиться к разным типам. Каждое поле имеет свое имя (идентификатор поля).

Тип данных *запись* определяется структурой вида

```
type <Имя типа> = record
                    <Имя поля 1> : T1;
                    <Имя поля 2> : T2;
                    ...
                    <Имя поля n> : Tn;
                  end;
```

где T_1, T_2, ... T_n указывают тип соответствующих полей. Тип любого поля может быть произвольным, значит, поле, в свою очередь, может относиться к типу *запись*. Таким образом можно определять вложенные типы.

Примеры:

1)
```
type Elev = record
              Nume : string;
              Prenume : string;
              NotaMedie : real;
            end;
var E1, E2 : Elev;
```

2)
```
type Punct = record
               x : real; { координата x }
               y : real; { координата y }
             end;
var P1, P2 : Punct;
```

3)
```
type Triunghi =  record
                    A : Punct; { вершина A }
                    B : Punct; { вершина B }
                    C : Punct; { вершина C }
                 end;
var T1, T2, T3 : Triunghi;
```

Структура данных из приведенных выше примеров представлена на *рис. 1.4*.

Если две переменные относятся к одному и тому же типу *запись*, то между ними разрешена операция присваивания. При таком присваивании все поля переменной, стоящей в правой части, копируются в переменную, стоящую в левой части. Например, для типов данных и переменных, определенных выше, следующие операции корректны:

```
E1:=E2;
T2:=T3;
P2:=P1
```

К каждому элементу любой переменной типа **record** можно обращаться явно, по имени переменной и названию поля, которые разделяются точкой.

Примеры:

1) `E1.Nume, E1.Prenume, E1.NotaMedie;`

2) `E2.Nume, E2.Prenume, E2.NotaMedie;`

3) `P1.x, P1.y, P2.x, P2.y;`

4) `T1.A, T1.B, T1.C, T2.A, T2.B, T2.C;`

5) `T1.A.x, T1.A.y, T2.B.x, T2.B.y.`

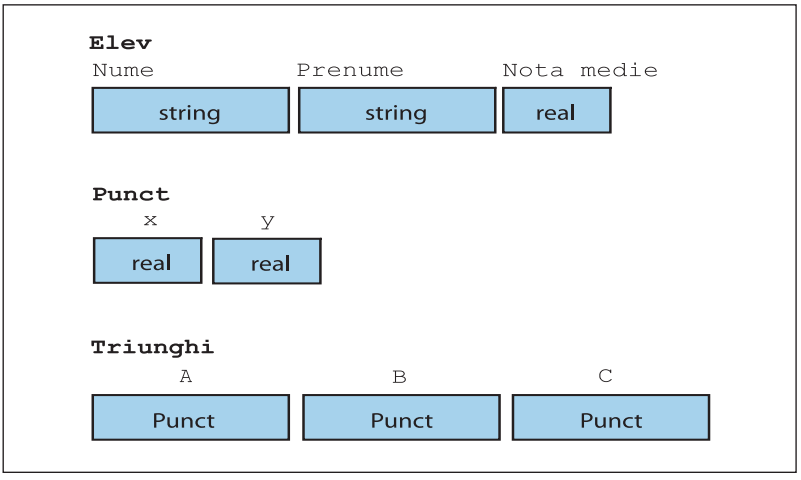


Рис. 1.4. Структура данных типа `Elev`, `Punct` и `Triunghi`

Очевидно, элемент E1. Nume относится к типу **string**; элемент P1.x – к типу real; элемент T1.A – к типу Punct; элемент T1.A.x – к типу real и т. д.

К элементам данных типа *запись* можно применять все операции, допустимые в типе соответствующего поля. Следующая программа сравнивает средние баллы двух учеников и выводит на экран имя и фамилию ученика с более высоким средним баллом. Считается, что средние баллы учеников различны.

```
Program P84;
{ Данные типа Elev }
type Elev = record
              Nume : string;
              Prenume : string;
              NotaMedie : real;
            end;
var E1, E2, E3 : Elev;
begin
  writeln('Введите данные о первом ученике:');
  write('Фамилия:');      readln(E1.Nume);
  write('Имя:');          readln(E1.Prenume);
  write('Средний балл:'); readln(E1.NotaMedie);

  writeln('Введите данные о втором ученике:');
  write('Фамилия:');      readln(E2.Nume);
  write('Имя:');          readln(E2.Prenume);
  write('Средний балл:'); readln(E2.NotaMedie);

  if E1.NotaMedie > E2.NotaMedie then E3:=E1 else E3:=E2;

  writeln('Ученик с наиболее высоким средним баллом:');
  writeln(E3.Nume, ' ', E3.Prenume, ':', E3.NotaMedie : 5:2);
  readln;
end.
```

Любой тип данных **record** может служить базовым типом для формирования других составных типов.

Пример:

```
type ListaElevilor = array [1..40] of Elev;
var  LE : ListaElevilor;
```

Очевидно, обозначение LE[i] указывает на i-го ученика из списка; обозначение LE[i]. Nume указывает на имя данного ученика и т.д. Следующая программа вводит с клавиатуры данные об n учениках и выводит на экран имя, фамилию и средний балл лучшего ученика. Считается, что средние баллы учеников различны.

```
Program P85;
{ Массив с элементами типа Elev }
type Elev = record
              Nume : string;
              Prenume : string;
              NotaMedie : real;
            end;
     ListaElev = array [1..40] of Elev;
var E : Elev;
    LE : ListaElev;
    n : 1..40;
    i : integer;
begin
  write('n='); readln(n);
  for i:=1 to n do
    begin
      writeln('Введите данные об ученике  ', i);
      write('Фамилия:  '); readln(LE[i].Nume);
      write('Имя:  ');
      readln(LE[i].Prenume);
      write('Средний балл:  ');
      readln(LE[i].NotaMedie);
    end;
  E.NotaMedie:=0;
  for i:=1 to n do
    if LE[i].NotaMedie > E.NotaMedie then E:=LE[i];
  writeln('Лучший ученик:');
  writeln(E.Nume, ' ', E.Prenume, ':', E.NotaMedie : 5:2);
  readln;
end.
```

В общем случае тип данных *запись* определяется с помощью синтаксических диаграмм на *рис. 1.5*. В дополнение к фиксированным записям (с фиксированным количеством полей) язык ПАСКАЛЬ позволяет использование вариантных записей, которые изучаются в более углубленных курсах информатики.

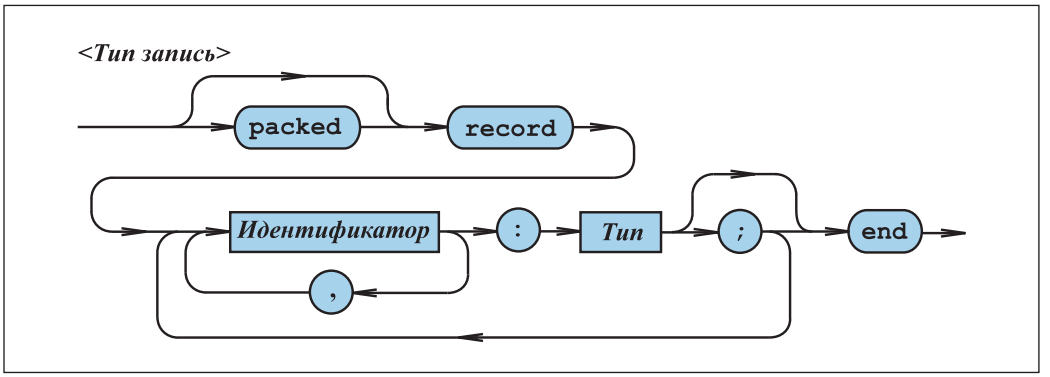


Рис. 1.5. Синтаксическая диаграмма <*Тип запись*>

Вопросы и упражнения

❶ Укажите множество значений типа данных *запись*.

❷ Укажите на синтаксической диаграмме *рис. 1.5* пути, которые соответствуют определениям типов данных *запись* из программ P84 и P85.

❸ Напишите металингвистические формулы для синтаксической диаграммы на *рис. 1.5*.

❹ Даны следующие типы данных:

```
type Data = record
              Ziua : 1..31;
              Luna : 1..12;
              Anul : integer;
            end;
     Persoana = record
                  NumePrenume : string;
                  DataNasterii : Data;
                end;
     ListaPersoane = array [1..50] of Persoana;
```

Напишите программу, которая вводит с клавиатуры данные о n лицах ($n \le 50$) и выводит на экран:

а) фамилии и имена тех, кто родился в день z месяца;

б) фамилии и имена тех, кто родился в месяц l года;

в) фамилии и имена тех, кто родился в год a;

г) фамилии и имена тех, чья дата рождения $z.l.a$;

д) фамилию и имя самого старшего человека;

е) фамилию и имя самого младшего человека;

ж) возраст каждого человека в годах, месяцах и днях;

з) список лиц старше v лет;

и) список лиц в алфавитном порядке;

к) список лиц, упорядоченный согласно дате рождения;

л) список лиц одного возраста (рожденных в одном и том же году).

❺ Дано n ($n \le 30$) точек на евклидовой плоскости. Каждая точка i определяется координатами x_i, y_i. Расстояние между точками i, j вычисляется по формуле

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Напишите программу, которая выводит на экран координаты двух точек, расстояние между которыми максимально.

❻ Площадь треугольника вычисляется по формуле Герона

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

где p – полупериметр, a, b и c – длины соответствующих сторон. Используя типы данных Punct и Triunghi из данного параграфа, напишите программу, которая считывает с клавиатуры информацию о n треугольниках ($n \le 10$) и выводит на экран:

а) площадь каждого треугольника;

б) координаты вершин треугольника, площадь которого максимальна;

в) координаты вершин треугольника, площадь которого минимальна;

г) информацию о всех треугольниках в порядке возрастания их площадей.

1.4. Оператор `with(c)`

К каждому элементу любой переменной типа *запись* можно обращаться явно, по имени переменной и названию поля, которые разделяются точкой.

Например, при следующем описании

```
type Angajat = record
                 NumePrenume : string;
                 ZileLucrate : 1..31;
                 PlataPeZi   : real;
                 PlataPeLuna : real;
               end;
var A : Angajat;
```

к элементам переменной A можно обращаться через A.NumePrenume, A.ZileLucrate, A.PlataPeZi и A.PlataPeLuna.

Так как имя A переменной типа *запись* постоянно повторяется, то такой способ обращения к элементам является очень громоздким. Этих повторений можно избежать при использовании оператора **with** (c).

Синтаксис данного оператора:

<Оператор **with***>* ::= **with** *<Переменная>* {, *<Переменная>*} **do** *<Оператор>*

Синтаксическая диаграмма представлена на *рис. 1.6*.

Внутри оператора **with** к элементам одной или нескольких переменных типа *запись* можно обращаться только по имени соответствующего поля.

Пример:

```
with A do PlataPeLuna:=PlataPeZi*ZileLucrate
```

Этот оператор эквивалентен следующему:

```
A.PlataPeLuna:=A.PlataPeZi*A.ZileLucrate
```

Использование оператора **with** требует особого внимания от программиста, который обязан определять однозначным образом элементы переменных типа *запись*. Внутри оператора **with**, при появлении некоторого идентификатора, вначале проводится проверка того, может ли он быть интерпретирован как имя поля соответствующей записи. Если да, то он будет интерпретирован как имя поля, даже если в данный момент доступна другая переменная под таким же именем.

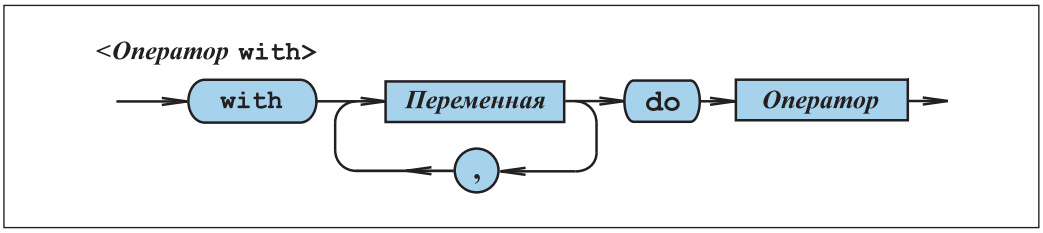


Рис. 1.6. Синтаксическая диаграмма оператора **with**

Пример:

```
type Punct = record
               x : real;
               y : real;
             end;
     Segment = record
                 A : Punct;
                 B : Punct;
               end;
var P : Punct;
    S : Segment;
    x : integer;
```

В нашем случае идентификатор x может указывать на переменную x типа integer или на поле P.x записи P.

В операторе

```
x:=1
```

идентификатор x указывает на переменную x типа integer.

В операторе

```
with P do x:=1
```

идентификатор x указывает на поле P.x переменной P типа Punct.

Так как переменная S типа Segment не содержит поля с именем S.x, то в операторе

```
with S do x:=1
```

идентификатор x будет интерпретирован как переменная x типа integer.

Оператор вида

with $v_1, v_2, ..., v_n$ **do** <*Оператор*>,

где $v_1, v_2, ..., v_n$ – переменные типа *запись*, эквивалентен оператору:

```
with v1 do
with v2 do
 {...}
with vn do <Оператор>.
```

Очевидно, что поля записей $v_1, v_2, ..., v_n$ должны быть определены однозначно.

Например, для переменных P и S, описанных выше, можно записать:

```
with P, S do
begin
  x:=1.0;    { ссылка на P.x }
  y:=1.0;
```

```
  A.x:=0;   { ссылка на S.A.x }
  A.y:=0;
  B.x:=2.0; { ссылка на S.B.x }
  B.y:=2.0;
end;
```

Как правило, оператор **with** используется только в тех случаях, когда это приводит к значительному сокращению текста программы.

Вопросы и упражнения

❶ Укажите на синтаксической диаграмме *рис. 1.6* пути, которые соответствуют операторам **with** из примеров, приведенных в данном параграфе.

❷ В чем назначение оператора **with**?

❸ Используя оператор **with**, исключите из программ P84 и P85, рассмотренных в предыдущем параграфе, повторения типа

```
E1.Nume, E1.Prenume, ...,
LE[i].Nume, LE[i].Prenume.
```

❹ Даны следующие типы данных:

```
type Angajat = record
                 NumePrenume : string;
                 ZileLucrate : 1..31;
                 PlataPeZi : real;
                 PlataPeLuna : real;
               end;
     ListaDePlata = array [1..50] of Angajat;
```

Ежемесячная зарплата каждого работника вычисляется путем умножения ежедневной платы на количество отработанных дней. Напишите программу, которая:
а) вычисляет ежемесячную зарплату каждого работника;
б) вычисляет среднюю зарплату всех работников, включенных в список;
в) выводит на экран данные о всех работниках, ежемесячная зарплата которых максимальна;
г) выводит на экран список работников в алфавитном порядке;
д) выводит на экран список работников в порядке возрастания ежедневной платы;
е) упорядочивает список работников в порядке возрастания ежемесячной зарплаты;
ж) выводит на экран список работников в порядке возрастания количества отработанных дней.

❺ Окружность может быть задана через координаты x, y центра и радиус r. Напишите программу, которая считывает с клавиатуры данные об n окружностях ($n \le 50$) и выводит на экран:
а) координаты центра и радиус окружности, которая описывает круг максимальной площади;

б) количество окружностей, входящих в круг с максимальным радиусом, и координаты соответствующих центров;
в) координаты центра и радиус окружности, которая описывает круг минимальной площади;
г) количество окружностей, в которые входит круг с минимальным радиусом, и координаты соответствующих центров.

1.5. Тип данных *множество* (set)

Тип данных *множество* **(set)** определяется по отношению к базовому типу, который должен быть порядковым:

```
<Тип множество> ::= [packed] set of <Тип>
```

Значениями типа данных **set** являются множества, состоящие из значений базового типа. Если базовый тип имеет n значений, то тип *множество* будет иметь 2^n значений. Значение n ограничено: $n \le 256$.

В языке ПАСКАЛЬ элементы множества могут перечисляться в квадратных скобках „[" и „]", которые являются аналогом фигурных скобок в математике.

Запись `[]` означает пустое множество.

Примеры:

```
type Indice = 1..10;
     Zi = (L, Ma, Mi, J, V, S, D);
     MultimeIndicii = set of Indice;
     ZileDePrezenta = set of Zi;
var MI : MultimeIndicii;
    ZP : ZileDePrezenta;
```

Порядковый тип `Indice` имеет n=10 значений: `1, 2, 3, 4, 5, 6, 7, 8, 9, 10`. Тип `MultimeIndicii` имеет $2^{10} = 1024$ значений, а именно:

```
[], [1], [2], ..., [1, 2], [1, 3], ...,
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10].
```

Таким образом, переменная `MI` может принимать любое из этих значений, например:

```
MI:= [1, 3].
```

Порядковый тип `Zi` имеет n=7 значений: `L, Ma, Mi, J, V, S, D`. Тип `ZileDePrezenta` имеет $2^7 = 128$ значений, а именно:

```
[], [L], [Ma], [Mi], ..., [L, Ma], [L, Mi], ...,
[L, Ma, Mi, J, V, S, D].
```

Переменная `ZP` может принимать любое из этих значений, например:

```
ZP:= [L, Ma, Mi, V]
```

Значения типа *множество* могут определяться через **конструктор** множества. Синтаксическая диаграмма грамматической единицы <*Конструктор множества*> представлена на *рис. 1.7*.

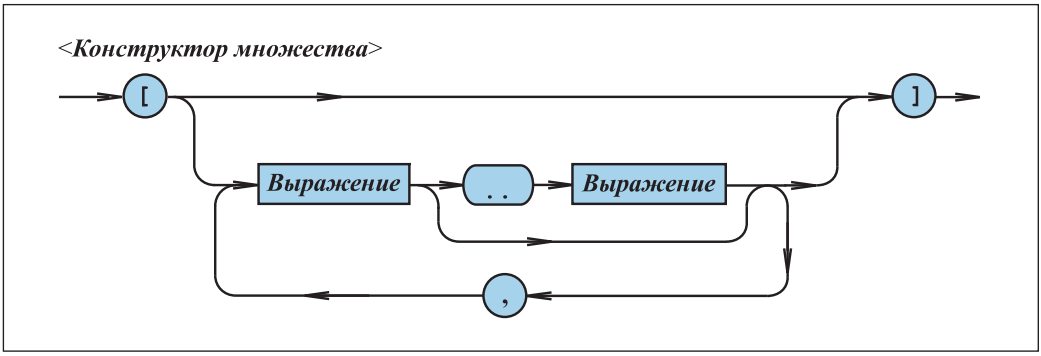


Рис. 1.7. Синтаксическая диаграмма <*Конструктор множества*>

Конструктор содержит значения элементов множества, разделенных запятыми и заключенных в квадратные скобки. Элемент может являться конкретным значением базового типа или интервалом вида:

<*Выражение*>..<*Выражение*>.

Значения данных выражений представляют собой верхний и нижний пределы интервала.

Примеры:

1) `[];`

2) `[1, 2, 3, 8];`

3) `[1..4, 8..10];`

4) `[i-k..i+k];`

5) `[L, Ma, V..D].`

К значениям типа данных *множество* можно применять обычные операции:

`+` объединение;
`*` пересечение;
`–` разность,

результат которых относится к типу *множество*, и операции отношения:

`=` равенство;
`<>` неравенство;
`<=, >=` включение;
`in` принадлежность,

результат которых относится к типу `boolean`.

Следующая программа выводит на экран результаты операций `+`, `*` и `–`, применяемых к переменным типа `MultimeIndicii`.

```
Program P86;
{ Данные типа MultimeIndicii }
type Indice = 1..10;
     MultimeIndicii = set of Indice;
var A, B, C : MultimeIndicii;
    i : integer;
begin
  A:= [1..5, 8];      { A содержит 1, 2, 3, 4, 5, 8 }
  B:= [1..3, 9, 10]; { B содержит 1, 2, 3, 9, 10 }
  C:= [];             { C является пустым множеством }

  C:=A+B;             { C содержит 1, 2, 3, 4, 5, 8, 9, 10 }
  writeln('Объединение');
  for i:=1 to 10 do
    if i in C then write(i:3);
  writeln;

  C:=A*B;             { C содержит 1, 2, 3 }
  writeln('Пересечение');
  for i:=1 to 10 do
    if i in C then write(i:3);
  writeln;

  C:=A-B;             { C содержит 4, 5, 8 }
  writeln('Разность');
  for i:=1 to 10 do
    if i in C then write(i:3);
  writeln;
  readln;
end.
```

В отличие от массивов и записей, к элементам которых существует прямой доступ соответственно через индексы и названия полей, к элементам множества прямого доступа нет. Допускается только проверка на принадлежность элемента множеству (операция отношения **in**). Благодаря этому при использовании типов данных *множество* увеличивается скорость работы и улучшается читабельность программ PASCAL.

Например, оператор:

```
if (c='A') or (c='E') or (c='I') or (c='O') or (c='U') then ...
```

можно заменить более простым оператором:

```
if c in ['A','E','I','O','U'] then ...
```

Другим примером является использование типов данных *множество* для вычисления простых чисел, меньших заданного n, где n – натуральное число. Для этого используется алгоритм *Решето Эратосфена*:

1) в решете находятся числа 2, 3, 4, ..., n;

2) из решета удаляется наименьшее число i;

3) указаное число включается в множество простых чисел;
4) из решета удаляются все числа m кратные числу i;
5) процесс завершается, когда решето становится пустым.

```
Program P87;
{ Решето Эратосфена }
const n = 50;
type MultimeDeNumere = set of 1..n;
var Sita, NumerePrime : MultimeDeNumere;
    i, m : integer;
begin
 {1} Sita:= [2..n];
     NumerePrime:=[];
     i:=2;
     repeat
 {2}    while not (i in Sita) do i:=succ(i);
 {3}    NumerePrime:=NumerePrime+[i];
        write(i:4);
        m:=i;
 {4}    while m<=n do
          begin Sita:=Sita-[m]; m:=m+i; end;
 {5} until Sita=[];
     writeln;
     readln;
end.
```

Цифры 1, 2, 3, 4, 5, стоящие в фигурных скобках в левой части программы, соответствуют пунктам алгоритма Эратосфена.

Вопросы и упражнения

❶ Перечислите допустимые значения переменных, описанных ниже:

```
var V : set of 'A'..'C';
    S : set of (A, B, C);
    I : set of '1'..'2';
    J : set of 1..2;
```

❷ Прокомментируйте следующую программу:

```
Program P88;
{ Ошибка }
type Multime = set of integer;
var M : Multime;
    i : integer;
begin
  M:=[1, 8, 13];
  for i:=1 to MaxInt do
    if i in M then writeln(i);
end.
```

❸ Даны следующие описания:

```
type Culoare = (Galben, Verde, Albastru, Violet);
     Nuanta = set of Culoare;
var NT : Nuanta;
```

Какие значения может принимать переменная NT?

❹ Напишите металингвистическую формулу, которая соответствует синтаксической диаграмме *<Конструктор множества> рис. 1.7*.

❺ Рассматривается тип данных MultimeIndicii, описанный в этом параграфе. Перечислите элементы множеств, определяемых следующими конструкторами:

a) `[];`

b) `[1..10];`

c) `[1..3, 9..10];`

d) `[1+1, 4..7, 9];`

e) `[3, 7..9];`

f) `[4..3];`

g) `[1..3, 7..6, 9];`

h) `[4-2..7+1];`

i) `[7-5..4+4];`

j) `[6, 9, 1..2].`

❻ Напишите программу, которая выводит на экран все подмножества множества: {1, 2, 3, 4}.

❼ Напишите программу, которая выводит на экран все подмножества множества: {'*A*', '*B*', '*C*', '*D*'}.

❽ Дана строка символов, в которой слова разделяются пробелом или символами: *точка, запятая, точка с запятой, восклицательный знак, вопросительный знак*. Напишите программу, которая выводит на экран слова, входящие в состав произвольной строки, считываемой с клавиатуры.

❾ Дана строка символов. Напишите программу, которая выводит на экран количество гласных в строке.

❿ Напишите программу, которая считывает с клавиатуры две строки символов и выводит на экран:
а) символы, которые встречаются хотя бы в одной из строк;
б) символы, которые встречаются в обеих строках;
в) символы, которые встречаются в первой строке и не встречаются во второй.

⓫ Напишите программу, которая проверяет правильность введения в компьютер имени человека (под именем понимается последовательность символов, не содержащая цифр).

⓬ В современных версиях языка количество значений базового типа любого типа *множество* ограничено, обычно $n \le 256$. Отсюда следует, что программа P87 не может находить простые числа больше n. Напишите программу, которая вычисляет простые числа из интервала 8, ..., 10000.
Указание. Решето из алгоритма Эратосфена можно представить в виде массива, элементами которого являются множества.

1.6. Файлы

Под **файлом** понимают структуру данных, которая состоит из последовательности компонент. Все компоненты файла относятся к одному и тому же типу, который называется *базовым*. Число компонентов файла является произвольным, однако конец файла обозначается специальным символом: *EOF* (*End of File* – конец файла). Файл, который не содержит ни одного элемента, называется *пустым файлом*.

Файловый тип данных определяется следующим образом:

```
<Файловый тип> ::= [packed] file of <Тип>;
```

где *<Тип>* является базовым типом. Базовый тип может быть любым, кроме самого *файлового типа* (не существует "файл файлов").

Примеры:

1)
```
type FisierNumere = file of integer;
var FN : FisierNumere;
    n : integer;
```

2)
```
type FisierCaractere = file of char;
var FC : FisierCaractere;
    c : char;
```

3)
```
type Elev = record
               Nume : string;
               Prenume : string;
               NotaMedie : real;
            end;
     FisierElevi = file of Elev;
var FE : FisierElevi;
    E : Elev;
```

Структура данных *файлового типа* представлена на *рис. 1.8*. Отметим, что символ *EOF*, который означает конец файла, не является компонентом файла.

Переменные FN, FC, FE *файлового типа* называются **логическими файлами, файлами языка ПАСКАЛЬ** или просто **файлами**. В отличие от остальных типов данных, значения которых хранятся во внутренней памяти компьютера, данные файлов хранятся на периферийных устройствах – носителях информации (на дисках, магнитных лентах, оптических дисках, бумаге принтера или на устройстве считывания документов и др.). Информация на таких носителях хранится в виде **внешних файлов** в соответствии с требованиями операционной системы. Таким образом, перед использованием переменную *файлового типа* необходимо связать с некоторым внешним файлом. Методы связи зависят от версии языка и операционной системы, установленной на компьютере.

В стандартной версии языка связь осуществляется посредством включения переменных *файлового типа* в заголовок программы в качестве аргументов.

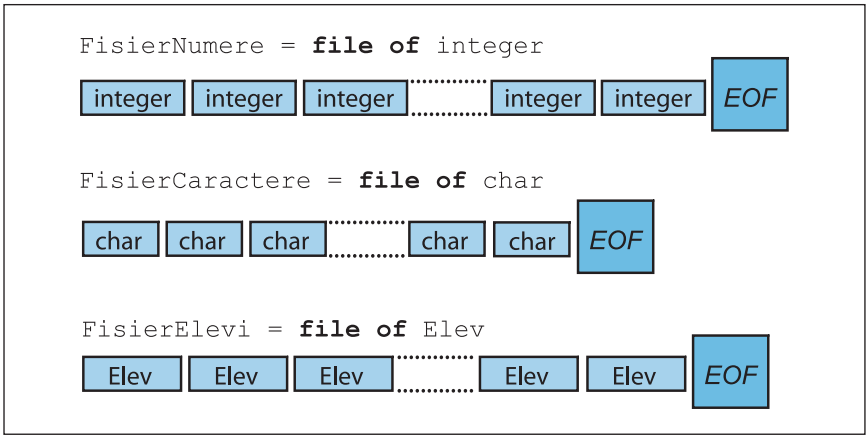


Рис. 1.8. Структура данных типа `FisierNume`, `FisierCaractere` и `FisierElevi`

В Turbo PASCAL связь файловой переменной *f* с внешним файлом осуществляется вызовом процедуры

```
assign(f, s);
```

где *s* – это выражение типа **string**, задающее путь доступа и имя внешнего файла.

Примеры:

1)
```
assign(FN, 'A:\REZULTAT\R.DAT')
```

– файл `FN` связывается с внешним файлом `R.DAT`, находящимся в каталоге `REZULTAT` на диске `A`.

2)
```
assign(FC, 'C:\A.CHR')
```

– файл `FC` связывается с файлом `A.CHR`, находящимся в корневом каталоге диска `C`.

3)
```
write('Введите имя файла:');
readln(str);
assign(FE, str);
```

– файл `FE` связывается с внешним файлом, имя которого считывается с клавиатуры в переменную `str` типа **string**.

После выполнения оператора `assign` (*f*, s) все операции, осуществляемые над файлом *f*, фактически будут выполняться над внешним файлом *s*.

Самыми распространенными операциями, выполняемыми над файлами, являются считывание компонентов из файла и их запись в файл.

Считывание текущей компоненты из файла осуществляется с помощью оператора вызова процедуры

```
read(f, v),
```

где *v* – переменная, которая относится к базовому типу файла *f*.

Запись следующей компоненты в файл осуществляется с помощью оператора вызова процедуры:

```
write(f, e),
```

где *e* – выражение, относящееся к базовому типу файла *f*.

Примеры:

1) `read(FN, n);`

2) `write(FC, c);`

3) `read(FE, E).`

По **типам операций, применяемых** к компонентам, файлы подразделяются на:

– входные (открыты только для чтения);

– выходные (открыты только для записи);

– рабочие (открыты и для чтения и для записи).

По **методу доступа** к компонентам файлы подразделяются на:

– файлы последовательного доступа или последовательные (доступ к компоненте *i* возможен только после считывания или записи компоненты *i*–1);

– файлы прямого доступа (к любой компоненте есть прямой доступ через ее порядковый номер *i* в файле).

Отметим, что в стандартном языке допустимы только входные и выходные файлы последовательного доступа.

Тип файла (входной, выходной или рабочий) и метод доступа (прямой или последовательный) задаются при **открытии файла**. В стандартном языке существуют следующие процедуры для открытия файлов:

`reset(f)` – открывает существующий файл для чтения;

`rewrite(f)` – создает пустой файл для записи.

После завершения обработки компонент файл нужно закрыть. При **закрытии** файла операционная система записывает символ *EOF*; регистрирует только что созданный файл в соответствующем каталоге и т.д.

В стандартном языке по окончании работы программы все файлы закрываются автоматически. В Turbo PASCAL закрытие файла *f* осуществляется явно с помощью оператора процедуры `close (f)`.

В заключение приведем порядок вызова процедур, предназначенных для обработки данных *файлового* типа:

1) `assign(f, s)` – связывание файловой переменной *f* с внешним файлом *s*;

2) `reset(f)/rewrite(f)` – открытие файла *f* для чтения/записи;

3) `read(f, v)/write(f, e)` – чтение/запись текущей компоненты файла *f*;

4) `close(f)` – закрытие файла *f*.

После закрытия файла переменная *f* может быть связана с другим внешним файлом.

Так как значения переменных файлового типа хранятся на внешних носителях информации, в языке ПАСКАЛЬ **операция присваивания файлов запрещена**.

Вопросы и упражнения

❶ Объясните термины *файл языка ПАСКАЛЬ*, *внешний файл*.

❷ Где хранятся данные файла? Для чего нужна процедура `assign`?

❸ Нарисуйте структуру следующих типов данных:

a)
```
type Tabel = array [1..5, 1..10] of real;
     FisierTabele = file of Tabel;
```

b)
```
type Multime = set of 'A'..'C';
     FisierMultimi = file of Multime;
```

c)
```
type Punct = record x, y:real end;
     Segment = record A, B:Punct end;
     FisierSegmente = file of Segment;
```

❹ Для чего нужны процедуры открытия и закрытия файлов? Как выполняются эти процедуры?

❺ Для чего нужны процедуры `read` и `write`? Какого типа должна быть переменная v в вызове процедуры `read`(f, v)? Какого типа должно быть выражение e в вызове процедуры `write`(f, e)?

❻ Как классифицируются файлы в зависимости от допустимых операций и от метода доступа?

❼ Переменные `A` и `B` введены посредством описания

```
var A, B : file of integer;
```

Корректна ли следующая запись

```
A:=B
```

Обоснуйте ваш ответ.

1.7. Файлы с последовательным доступом

Рассмотрим следующие описания:

```
type FT = file of T;
var f : FT; v : T;
```

посредством которых определяются *файловый* тип `FT` с базовым типом `T`, *файловая* переменная f и переменная v типа `T`.

Для открытия **выходного файла последовательного доступа** используется вызов процедуры `rewrite`(f). Затем, в файл записываются соответствующие компоненты. Запись отдельных компонент производится с помощью процедуры:

```
write(f, e),
```

где e – выражение типа `T`.

Оператор вида

write($f, e_1, e_2, ..., e_n$)

эквивалентен последовательности операторов

write(f, e_1); write(f, e_2); ...; write(f, e_n).

После записи последней компоненты файл необходимо закрыть.

Пример:

```
Program P89;
{ Создание файла с компонентами типа Elev }
type Elev = record
              Nume : string;
              Prenume : string;
              NotaMedie : real;
            end;
     FisierElevi = file of Elev;
var FE : FisierElevi;
    E : Elev;
    str : string;
    i, n : integer;
begin
  write('Введите имя файла: ');
  readln(str);

  assign(FE, str);    { связывает FE с именем str }
  rewrite(FE);        { открывает файл FE для записи }

  write('Введите количество учеников: '); readln(n);

  for i:=1 to n do
    begin
      writeln('Введите данные об ученике ', i);

     { Считывает поля переменной E с клавиатуры }
      write('Фамилия: ');        readln(E.Nume);
      write('Имя: ');            readln(E.Prenume);
      write('Средняя оценка: '); readln(E.NotaMedie);

     { Записывает значение переменной E в файл FE }
      write(FE, E);
    end;
  close(FE);          { Закрывает файл FE }
  readln;
end.
```

Для **открытия входного файла последовательного доступа** используется процедура `reset(f)`. Чтение текущего элемента из файла выполняется с помощью вызова процедуры:

`read(f, v).`

Оператор вида

`read(f, v1, v2, ..., vn)`

эквивалентен последовательности операторов:

`read(f, v1); read (f, v2); ..., read (f, vn).`

Конец файла можно обнаружить с помощью булевой функции `eof(f)`, которая возвращает значение `true` после чтения последнего элемента.

Пример:

```
Program P90;
{ Считывание файла с элементами типа Elev }
type Elev = record
              Nume : string;
              Prenume : string;
              NotaMedie : real;
            end;
     FisierElevi = file of Elev;
var FE : FisierElevi;
    E : Elev;
    str : string;
begin
  write('Введите имя файла:  ');
  readln(str);

  assign(FE, str);   { связывает FE с именем str }
  reset(FE);         { открывает файл FE для чтения }

  while not eof(FE) do
    begin
       { считывает E из файла FE }
       read(FE, E);
       { выводит E на экран }
       writeln(E.Nume, ' ', E.Prenume, ':',
               E.NotaMedie : 5:2);
    end;
  close(FE);          { закрывает файл FE }
  readln;
end.
```

Отметим, что число элементов файла заранее неизвестно и не задается в описании соответствующего типа. Следовательно, в выходной файл последо-

вательного доступа теоретически можно записать бесконечное множество элементов. Однако практически количество элементов ограничено емкостью внешнего носителя информации. Считывание элементов любого входного файла последовательного доступа завершается при достижении символа *EOF*.

Вопросы и упражнения

❶ Из скольких элементов может состоять файл? В каком порядке записываются и считываются элементы файла с последовательным доступом?

❷ Даны следующие типы данных:

```
type Data = record
              Ziua : 1..31;
              Luna : 1..12;
              Anul : integer;
            end;
     Persoana = record
                  NumePrenume : string;
                  DataNasterii : Data;
                end;
     FisierPersoane = file of Persoana;
```

Напишите программу, которая считывает с клавиатуры данные об n лицах и записывает их в файл. Создайте файлы: `FILE1.PRS`, `FILE2.PRS`, `FILE3.PRS`, в которых должны содержаться данные соответственно о 2, 7 и 10 лицах.

❸ Напишите программу, которая читает файлы, созданные программой из предыдущего упражнения, и выводит на экран:
а) данные о всех лицах, занесенных в файл;
б) данные о лицах, родившихся в год *a*;
в) данные о лицах, у кого дата рождения *z.l.a*;
г) данные о самом старшем человеке;
д) данные о самом младшем человеке.

❹ Напишите программу, которая выводит на экран среднее арифметическое чисел, записанных в файле типа **file of** `real`.

❺ В файле типа **file of** `char` записаны произвольные символы. Напишите программу, которая выводит на экран количество согласных, содержащихся в файле.

❻ Прокомментируйте следующую программу:

```
Program P91;
{ Ошибка }
type FisierNumere = file of integer;
var FN : FisierNumere;
    i : integer;
    r : real;
    s : string;
begin
  Writeln('Введите имя файла:  ');
```

```
  readln(s);
  assign(FN, s);
  rewrite(FN);
  i:=1;
  write(FN, i);
  i:=10;
  write(FN, i);
  r:=20;
  write(FN, r);
  close(FN);
end.
```

1.8. Текстовые файлы

Известно, что данные файлов хранятся на внешних носителях информации. В случае, когда файлы описываются в виде **file of** *T*, элементы типа *T* записываются на соответствующих носителях в виде **последовательности двоичных цифр**. Такой способ представления данных удобен для внешней памяти (магнитные диски и магнитные ленты, оптические диски и др.). Для устройств ввода/вывода (клавиатура, экран, принтер и т.д.) соответствующие данные должны быть представлены во **внешней форме**, то есть через строки символов.

Для того чтобы упростить общение пользователя и компьютера, в языке ПАСКАЛЬ информация, предназначенная для пользователя, представляется в виде текстовых файлов. Текстовый файл состоит из последовательности символов, разделенных на строки (*рис. 1.9*). Длина строк – произвольна. Конец каждой строки обозначается специальным символом *EOL* (*End Of Line* – конец строки). Так как длина строк произвольна, то позицию некоторой строки в файле нельзя знать заранее. Следовательно, доступ к элементам текстового файла может быть только **последовательным**.

Текстовый файл задается описанием вида:

```
var f : text;
```

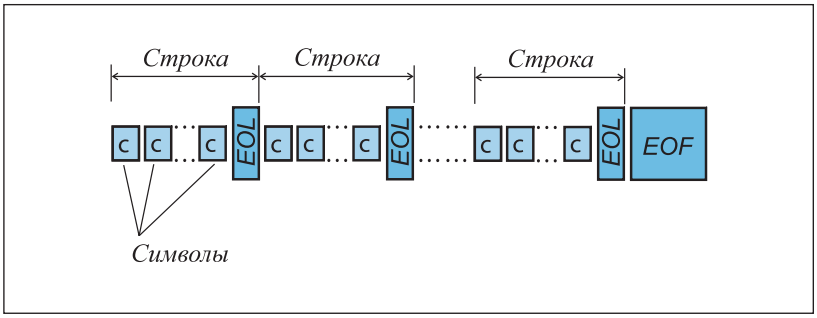


Рис. 1.9. Структура *текстового* файла

где предопределенный тип text известен любой программе на языке ПАСКАЛЬ. Отметим, что типы text и **file of** char различны, так как файл **file of** char не содержит символов *EOL* (*рис.1.8*).

Обработка текстовых файлов может осуществляться с помощью известных процедур, применимых к любым типам файлов: assign, reset, rewrite, read, write, close. В дополнение к ним в языке есть специальные процедуры для обработки элементов *EOL*:

writeln(*f*) – записывает в файл элемент *EOL* (конец строки);

readln(*f*) – переход на следующую строку.

Конец строки определяется с помощью булевой функции eoln(*f*), которая принимает значение true после считывания последнего символа строки.

Оператор вида

writeln(*f*, e_1, e_2, ..., e_n)

эквивалентен последовательности операторов:

write(*f*, e_1, e_2, ..., e_n); writeln(*f*).

Оператор вида

readln(*f*, v_1, v_2, ..., v_n)

эквивалентен последовательности операторов:

read(*f*, v_1, v_2, ..., v_n); readln(*f*).

Для ввода и вывода данных используются, как правило, предопределенные текстовые файлы Input и Output. Файл Input предназначен только для чтения и связан со стандартным устройством ввода операционной системы (как правило, с клавиатурой). Файл Output предназначен только для записи и связан со стандартным устройством вывода (как правило, экраном). Эти файлы открываются и закрываются автоматически в начале и соответственно в конце выполнения программы. Если при вызове подпрограмм имя файла не указано в списке параметров, то предполагается, что текстовым файлом является файл Input или файл Output в зависимости от назначения подпрограммы. Например, read(c) эквивалентен read(Input, c), а write(c) эквивалентен write(Output, c).

В качестве примера приводится программа P93, которая создает на текущем диске текстовый файл FILE.TXT. Строки файла вводятся с клавиатуры (файл Input). Признаком конца строки является нажатие клавиши <*ENTER*>, а признаком конца файла является нажатие клавиш <*CTRL+Z*>, <*ENTER*>.

```
Program P93;
{ Создание текстового файла FILE.TXT }
var F : text;
    c : char;
begin
  assign(F, 'FILE.TXT'); { связывает файл F с FILE.TXT }
```

```
  rewrite(F);          { открывает F для записи }
  while not eof do     { eof(Input) }
    begin
      while not eoln do  { eoln(Input) }
        begin
          read(c);       { считывает c из Input }
          write(F, c);   { записывает c в F }
        end;
      writeln(F);        { записывает EOL в F }
      readln;    { переходит на следующую строку из Input }
    end;
  close(F);              { закрывает F }
end.
```

Следующая программа выводит на экран содержимое файла FILE.TXT.

```
Program P94;
{ Чтение текстового файла FILE.TXT }
 var F : text;
     c : char;
begin
  assign(F, 'FILE.TXT'); { связывает F с FILE.TXT }
  reset(F);              { открывает F для чтения }
  while not eof(F) do
    begin
      while not eoln(F) do
        begin
          read(F, c);    { считывает c из F }
          write(c);      { записывает c в Output }
        end;
      readln(F);         { переходит на следующую строку F }
      writeln;           { записывает EOL в Output }
    end;
  close(F);              { закрывает F }
  readln;
end.
```

Посимвольное чтение и запись *текстовых* файлов являются достаточно сложными в случае, когда последовательности символов, находящиеся в текстовом файле, интерпретируются как данные типа integer, real, boolean, *строка символов*. Перевод данных указанных типов из внутреннего представления во внешнее становится задачей программиста. Поэтому действие процедур чтения/записи расширяется следующим образом.

Для текстовых файлов переменная *v* из процедуры read(*f*, *v*) может относиться к типу: integer, real, char или *строка символов*. При чтении последовательность символов, которая задает значение переменной *v*, будет переведена во внутреннее представление.

За выражением *e* в процедуре write(*f*, *e*) может следовать указатель формата. Значение выражения может относиться к типу integer, real, char или *строка символов*. При записи соответствующее значение переводится из внутреннего представления в последовательность символов.

Последовательности символов, считываемые/записываемые с помощью процедур read/write, соответствуют синтаксису констант типа переменной/выражения *v*/*e*.

В качестве примера приводится программа P95, которая считывает с клавиатуры по три вещественных числа *a, b, c* и записывает их в файл IN.TXT. После чтения указанных чисел, которые представляют собой длины сторон треугольника, программа записывает в файл OUT.TXT числа *a*, *b* и *c*, полупериметр *p* и площадь *s* треугольника. В завершение содержимое файла OUT.TXT выводится на экран.

```
Program P95;
{ Обработка файлов IN.TXT и OUT.TXT }
var F, G : text;
    a, b, c, p, s : real;
    str : string;
begin
  assign(F, 'IN.TXT');   { связывает F с IN.TXT }
  rewrite(F);            { открывает F для записи }

  writeln('Введите вещественные числа a, b, c:');
  while not eof do
    begin
     readln(a, b, c);  { считывает a, b, c с клавиатуры  }
     writeln(F, a:8:2, b:8:2, c:8:2); { scrie a, b, c, in F }
    end;
  close(F);              { закрывает F }

  reset(F);              { открывает F для чтения }
  assign(G, 'OUT.TXT'); { связывает G с OUT.TXT }
  rewrite(G);            { открывает G для записи }

  while not eof(F) do
    begin
      readln(F, a, b, c);  { считывает a, b, c из F }
      write(G, a:8:2, b:8:2, c:8:2); {записывает a, b, c в G }
      p:=(a+b+c)/2;
      s:=sqrt(p*(p-a)*(p-b)*(p-c));
      writeln(G, p:15:2, s:15:4); { записывает p, s в G }
    end;

  close(F);              { закрывает F }
  close(G);              { закрывает G }
```

```
  reset(G);              { открывает G для чтения }
  while not eof(G) do
    begin
      readln(G, str);    { считывает str из G }
      writeln(str);      { выводит str на экран }
    end;
  close(G);              { закрывает G }
  readln;
end.
```

Для входных данных

```
1 1 1 <ENTER>
3 4 6 <ENTER>
<CTRL+Z ><ENTER>
```

программа P95 выводит на экран:

```
1.00 1.00 1.00 1.50 0.4330
3.00 4.00 6.00 6.50 5.3327
```

Вопросы и упражнения

❶ В чем разница между текстовым файлом и файлом **file of** char?

❷ Что обозначают элементы *EOL* и *EOF*?

❸ В чем разница между процедурами read и readln? А между процедурами write и writeln?

❹ Запустите следующую программу на выполнение:

```
Program P96;
{ Связывание файла FN с устройством ввода }
type FisierNumere = file of integer;
var FN : FisierNumere;
    i : integer;
begin
  assign(FN, 'CON');
  rewrite(FN);
  i:=1;
  write(FN, i);
  i:=2;
  write(FN, i);
  i:=3;
  write(FN, i);
  close(FN);
  readln;
end.
```

Объясните результаты, выводимые на экран.

❺ Напишите программу, которая выводит на экран содержание любого *текстового* файла.

❻ Напишите программу, которая выводит на экран количество гласных, содержащихся в *текстовом* файле.

❼ Входные данные некоторой программы записаны в *текстовый* файл. В каждой строке файла содержатся два целых и три вещественных числа, разделенные пробелами. Напишите программу, которая выводит на экран сумму целых и сумму вещественных чисел из каждой строки.

❽ Входные данные некоторой программы записаны в *текстовый* файл. В каждой строке файла содержатся по три вещественных числа, разделенных пробелами, и по одному из слов `ADMIS`, `RESPINS`. Напишите программу, которая:

а) выводит содержимое данного файла на экран;

б) создает резервную копию файла;

в) создает *текстовый* файл, строки которого содержат среднее арифметическое трех вещественных чисел, взятых из соответствующих строк входного файла;

г) выводит на экран строки входного файла таким образом, что перед каждой строкой пишется ее порядковый номер 1, 2, 3 и т. д.

❾ Каждая строка *текстового* файла содержит следующие данные, разделенные пробелами:

– порядковый номер (`integer`);
– фамилия (последовательность символов, не содержащая пробелов);
– имя (последовательность символов, не содержащая пробелов);
– оценка по 1-му предмету (`real`);
– оценка по 2-му предмету (`real`);
– оценка по 3-му предмету (`real`).

Напишите программу, которая:

а) создает резервную копию *текстового* файла;

б) выводит на экран содержимое файла;

в) создает *текстовый* файл, строки которого содержат следующие данные, разделяемые пробелами:

– порядковый номер (`integer`);
– фамилия (`string`);
– имя (`string`);
– средний балл (`real`).

Файл, созданный в пункте *в*, необходимо вывести на экран.

Тест для самопроверки № 1

1. Даны следующие типы данных:

```
type Obiect = (Istoria, Geografia, Matematica,
               Informatica, Fizica);
     Nota = 1..10;
     SituatiaScolara = array [Obiect] of Nota;
```

Изобразите на рисунке структуру данных типа `SituatiaScolara`.

2. Для приведенных ниже описаний укажите тип индексов и тип компонент данных типа `OrarulLectiilor`:

```
type ZiDeScoala = (L, Ma, Mi, J, V, S);
     Lectie = 1..6;
     Obiect = (LimbaRomana, LimbaModerna, Istoria,
               Geografia, Matematica,Informatica, Fizica,
               Chimia);
     OrarulLectiilor = array [ZiDeScoala, Lectie] of Obiect;
```

3. Даны следующие описания и объявления:

```
type Tablou = array [1..10] of integer;
var x, y : Tablou;
```

Напишите арифметическое выражение для вычисления:
a) суммы первых четырех компонент переменной `x`;
b) суммы последних четырех компонент переменной `y`;
c) абсолютного значения третьей компоненты переменной `x`;
d) абсолютного значения шестой компоненты переменной `y`;
e) суммы первой компоненты переменной `x` и последней компоненты переменной `y`.

4. Даны n ($n \leq 50$) целых чисел a_1, a_2, a_3, ..., a_n. Напишите программу на ПАСКАЛЕ, которая вводит с клавиатуры рассматриваемые числа и выводит их на экран в порядке, обратном вводу: a_n, ..., a_3, a_2, a_1.

5. Какие операции могут быть выполнены над строками символов типа `string`? Укажите тип результата этих операций.

6. Напишите программу на ПАСКАЛЕ, которая выводит на экран строку символов в порядке, обратном тому, в котором она была введена с клавиатуры. Например, строка `'soare'` будет выведена на экран как `'eraos'`.

7. Для приведенного ниже описания изобразите на рисунке структуру данных типа `Data` и `Persoana`:

```
type Data = record
              Ziua : 1..31;
              Luna : 1..12;
              Anul : integer;
            end;
     Persoana = record
                  NumePrenume : string;
```

```
                    DataNasterii : Data;
                  end;
```

8. Даны следующие типы данных:

```
type Angajat = record
                 NumePrenume : string;
                 Salariu : real;
               end;
     ListaDePlata = array[1..100] of Angajat;
```

Напишите программу, которая вводит с клавиатуры данные об n ($n \leq 100$) сотрудниках и выводит на экран информацию о сотруднике (сотрудниках), получающем самую высокую заработную плату.

9. Для чего предназначен оператор **with**?

10. Перечислите всевозможные значения переменных из приведенных ниже объявлений:

```
var V : set of 'X'..'Z';
    I : set of 8..9;
```

11. Даны строки символов, состоящие из заглавных букв латинского алфавита. Напишите программу, которая выводит на экран количество гласных букв в считанной с клавиатуры строке символов *S* .

12. Где хранятся данные файловых переменных ПАСКАЛЯ? Для чего предназначена процедура `assign`?

13. Как классифицируются файлы в зависимости от способа доступа и допустимых операций?

14. Даны следующие типы данных:

```
type Angajat = record
                 NumePrenume : string;
                 Salariu : real;
               end;
     FisierAngajati = file of Angajat;
```

Напишите программу, которая создает файл `SALARII.DAT` и записывает в нем данные об n сотрудниках. Соответствующие данные считываются с клавиатуры.

15. Напишите программу на ПАСКАЛЕ, которая выводит на экран содержимое файла `SALARII.DAT` , созданного в предыдущем пункте.

16. Входные данные некоторой программы хранятся в текстовом файле `REZULTAT.TXT`. Каждая строка файла содержит по два вещественных числа, разделенных пробелом, и слово `BUN` или `DEFECT`. Напишите программу, которая:

a) создает текстовый файл `MEDIA.TXT`, каждая строка которого содержит число, представляющее собой среднее арифметическое вещественных чисел и слово `BUN` или `DEFECT` из соответствующей строки входного файла;

b) выводит на экран строки созданного файла.

Глава 2

ИНФОРМАЦИЯ

2.1. Количество информации

Обычный смысл слова **информация** – "новости, устное, письменное или переданное другими способами сообщение об определенных фактах, событиях, деятельности и т.п.", – конкретизируется в специальном разделе математики, который называется **теорией информации.** В соответствии с этой теорией **источник информации** описывается переменной S, которая может принимать значения из конечного множества различных элементов $\{s_1, s_2, \ldots, s_n\}$. Предполагается, что текущие значения переменной S априори не известны. Известно только множество $\{s_1, s_2, \ldots, s_n\}$, называемое **множеством возможных сообщений.**

Например, дорожный светофор можно рассматривать как источник информации, множество возможных сообщений которого: {*зеленый, желтый, красный*}. Телеграфный аппарат представляет собой источник информации, множество возможных сообщений которого включает буквы *A, B, C, ..., Z,* цифры *0, 1, 2, ..., 9* и знаки препинания. Возможными сообщениями клавиатуры являются "*Нажата клавиша A*", "*Нажата клавиша B*", ..., "*Нажата клавиша F1*", "*Нажата клавиша F2*", ..., "*Одновременно нажаты клавиши CTRL и BREAK*" и т.д.

Сообщения передаются от источника к приемнику информации через физическую среду, называемую **каналом передачи** (*рис. 2.1*). Например, телеграфные сообщения передаются по проводам, радиосообщения передаются через эфир, сообщения клавиатуры – через группу проводников. **Помехи** (шумы) упомянутой физической среды могут искажать передаваемые сообщения. Очевидно, что текущее значение переменной S становится известным приемнику только после приема соответствующего сообщения.

Количество информации I, которое содержится в сообщении, переданном источником, определяется соотношением:

$$I = \log_a n,$$

где n – это количество возможных сообщений источника. Конкретное значение константы a устанавливается выбором **единицы измерения количества информации.** Обычно в качестве единицы измерения используется **бит.**

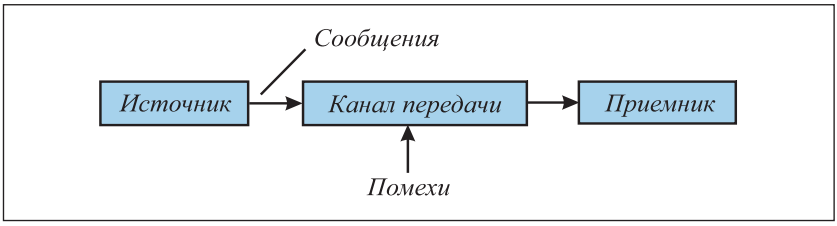


Рис. 2.1. Общая схема системы передачи информации

Бит – это количество информации в отдельном сообщении от источника с множеством только из двух возможных сообщений.

Следовательно, как и в случае других величин (длины, массы, температуры и т.д.), количество информации измеряется сравнением с **эталоном.** Так как для эталонного источника $n = 2$, из уравнения

$$\log_a 2 = 1 \ (бит)$$

получаем $a = 2$. Следовательно, количество информации I, измеряемое в битах, определяется соотношением:

$$I = \log_2 n \ (бит).$$

В *таблице 2.1* представлены часто используемые значения функции $\log_2 n$.

Проанализируем несколько примеров. Количество информации в одном сообщении светофора:

$$I = \log_2 3 \approx 1{,}585 \ (бит).$$

Количество информации одной буквы латинского алфавита $\{A, B, C, \ldots, Z\}$, $n = 26$, составляет

$$I = \log_2 26 \approx 4{,}700 \ (бит).$$

Таблица 2.1

Значения функции $\log_2 n$

n	$\log_2 n$	n	$\log_2 n$
1	0,000	21	4,392
2	1,000	22	4,459
3	1,585	23	4,524
4	2,000	24	4,585
5	2,322	25	4,644
6	2,585	26	4,700
7	2,807	27	4,755
8	3,000	28	4,807
9	3,170	29	4,858
10	3,322	30	4,907
11	3,459	31	4,954
12	3,585	32	5,000
13	3,700	33	5,044
14	3,807	34	5,087
15	3,907	35	5,129
16	4,000	36	5,170
17	4,087	37	5,209
18	4,170	38	5,248
19	4,248	39	5,285
20	4,322	40	5,322

Количество информации одной буквы греческого алфавита $\{A, B, \Gamma, \Delta, \ldots, \Omega\}$, $n = 24$, составляет

$$I = \log_2 24 \approx 4{,}585 \text{ бит}.$$

Если известно количество информации I, содержащееся в отдельном сообщении, то общее **количество информации, переданное источником,** определяется соотношением:

$$V = N I,$$

где N – количество переданных сообщений.

Большие объемы информации выражаются с помощью единиц, производных от бита:

1 *Килобит (Кбит)* = 2^{10} = 1024 *бит* (≈10^3 *бит*);

1 *Мегабит (Мбит)* = 2^{20} = 1048576 *бит* (≈10^6 *бит*);

1 *Гигабит (Гбит)* = $2^{30} \approx 10^9$ *бит*;

1 *Терабит (Тбит)* = $2^{40} \approx 10^{12}$ *бит*;

1 *Петабит (Пбит)* = $2^{50} \approx 10^{15}$ *бит*.

Вопросы и упражнения

❶ Как определяется источник информации? Приведите несколько примеров.

❷ Для чего предназначен канал передачи?

❸ Как определяется количество информации в одном сообщении? В N сообщениях?

❹ Какая единица используется для измерения количества информации и в чем ее смысл?

❺ Определите количество информации в отдельном сообщении источников со следующими возможными сообщениями:

а) прописные и строчные буквы латинского алфавита;

б) прописные и строчные буквы греческого алфавита;

в) прописные и строчные буквы румынского алфавита;

г) прописные и строчные буквы русского алфавита;

д) десятичные цифры 0, 1, 2, …, 9;

е) цифры 0, 1, 2, …, 9, знаки +, –, ×, / и скобки ( );

ж) числовые показания в виде *hh:mm* (*hh* – часы, *mm* – минуты) электронных часов;

з) числовые показания в виде *hh:mm:ss* (*ss* – секунды) электронных часов;

и) числовые показания в виде *zz.ll.aa* (*zz* – день, *ll* – месяц, *aa* – год) электронного календаря.

❻ Для каждого из источников, приведенных в упражнении 5, определите количество информации, содержащееся в 1000 сообщений, переданных источником.

❼ Напишите программу, вычисляющую количество информации в N сообщениях, переданных источником с n возможными сообщениями.

2.2. Кодирование и декодирование информации

Назовем **знаком** элемент конечного множества объектов, которые могут различаться. Линейно упорядоченное множество знаков называется **алфавитом.**

Представим далее некоторые из бесчисленного множества используемых людьми алфавитов:

а) алфавит десятичных цифр: 0, 1, 2, …, 9;

б) алфавит прописных латинских букв: *A, B, C, …, Z*;

в) множество знаков зодиака;

г) множество фаз луны.

Особое значение представляют алфавиты, состоящие только из двух знаков. Такие алфавиты называются **двоичными алфавитами**, а их знаки соответственно – **двоичными знаками.**

Приведем несколько примеров двоичных алфавитов:

а) цифры {*0, 1*};

б) пара цветов {*красный, желтый*};

в) пара состояний {*закрыт, открыт*};

г) пара ответов {*да, нет*};

д) пара напряжений {*0 В, 2 В*};

е) пара состояний {*намагничен, ненамагничен*};

ж) пара знаков {+, –} и т.д.

Удобно, чтобы знаки двоичного алфавита были представлены цифрами {*0, 1*}, которые называются **двоичными цифрами.** Последовательность, состоящая из m знаков, некоторые из которых могут повторяться, образует **слово**, а m представляет **длину слова.** Слова, образованные из двоичных знаков, называются **двоичными словами.** Очевидно, что слова могут иметь переменную или постоянную длину. В последнем случае они называются ***m*-позиционными словами.** Приведем далее некоторые множества слов постоянной длины:

1-позиционные: {0, 1};

2-позиционные: {00, 01, 10, 11};

3-позиционные: {000, 001, 010, 011, 100, 101, 110, 111};

4-позиционные: {0000, 0001, ..., 1110, 1111}.

Заметим, что $(m + 1)$-позиционные слова образуются по два из m-позиционных слов добавлением двоичных цифр 0 и 1. Следовательно, множество m-позиционных слов включает 2^m различных слов.

Двоичные слова используются для представления, передачи, хранения и обработки сообщений s_1, s_2, …, s_n источника информации (*рис. 2.2*).

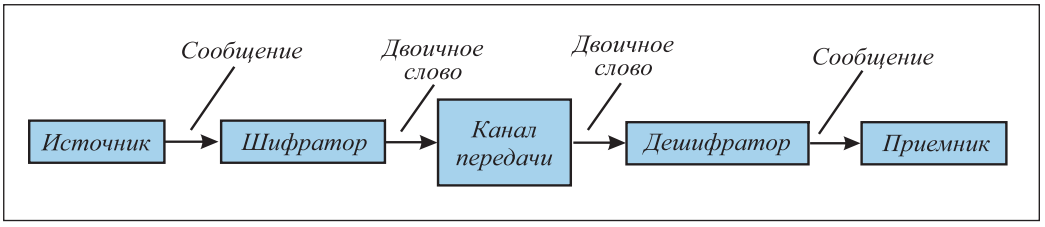


Рис. 2.2. Кодирование и декодирование сообщений в системах передачи информации

Правило преобразования сообщений в слова называется кодом, а соответствующая операция – кодированием. Операция, обратная кодированию, называется декодированием. Технические устройства, которые выполняют соответствующие операции, называются шифратор и дешифратор.

Самым простым является код, в котором возможным сообщениям s_1, s_2, …, s_n соответствуют двоичные слова постоянной длины m. Такой код, называемый ***m*-позиционным кодом**, может быть определен с помощью таблицы, содержащей возможные сообщения и соответствующие им слова. На *рис. 2.3* представлены соответствующие таблицы для источника с n = 2, 3, 4, …, 8 возможными сообщениями.

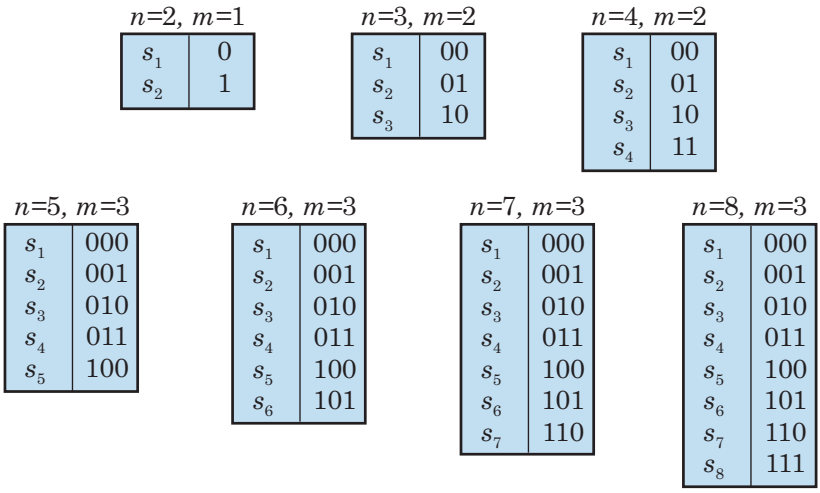


Рис. 2.3. Коды, состоящие из слов постоянной длины (*m*-позиционные коды)

Операции кодирования и декодирования состоят в извлечении необходимых данных из таблицы. Очевидно, что декодирование будет однозначным только тогда, когда двоичные слова, содержащиеся в таблице, различны. Это возможно, если длина m слов кода удовлетворяет неравенству

$$\boldsymbol{2^m \geq n}.$$

После логарифмирования получаем:

$$\boldsymbol{m \geq \log_2 n}.$$

Поскольку выражение $\log_2 n$ представляет количество информации, можно утверждать:

Длина слов любого позиционного кода должна быть больше или равна количеству информации в кодируемом сообщении.

Например, длина слов для кодирования прописных букв латинского алфавита {*A*, *B*, *C*, ...*Z*}, n = 26, определяется соотношением

$$\boldsymbol{m \geq \log_2 26 \approx 4{,}700}.$$

Устанавливая m = 5, можем формировать двоичные 5-позиционные кодовые слова:

A – 00000
B – 00001

C – 00010
D – 00011
E – 00100
...
Z – 11001.

Подобный код был предложен английским философом и государственным деятелем Фрэнсисом Бэконом еще в 1580 году.

Алгоритмы составления **кодов, состоящих из слов с переменной длиной**, значительно сложнее и изучаются в углубленных курсах информатики.

Вопросы и упражнения

❶ Что такое алфавит? Приведите примеры двоичных алфавитов.

❷ Как представляются знаки любого двоичного алфавита?

❸ Объясните, как могут быть получены двоичные (m + 1)-позиционные слова. Чему равно число различных двоичных m-позиционных слов?

❹ Для чего предназначен код? Как определяется m-позиционный код?

❺ Как осуществляются кодирование и декодирование сообщений в случаях, когда код определен с помощью таблицы?

❻ Закодируйте сообщения s_3, s_4 и s_6 источника с семью возможными сообщениями. Используйте 3-позиционный код с *рис. 2.3*.

❼ Декодируйте сообщения 100, 000 и 010, представленные в 3-позиционном коде на *рис. 2.3*, n = 5.

❽ Как определяется количество двоичных знаков, необходимых для формирования слов постоянной длины любого кода?

❾ Используя 3-позиционный код с *рис. 2.3*, n = 6, закодируйте сообщения s_1, s_2, s_6, s_5, s_3, s_6, s_3, s_2, s_1.

❿ Как влияет количество информации некоторого сообщения на длину слова кода?

⓫ Объясните смысл терминов **количество информации** и **информация**.

⓬ Напишите программу, которая кодирует и декодирует буквы латинского алфавита. Используйте код, предложенный Фрэнсисом Бэконом.

⓭ Напишите программу, которая составляет таблицу m-позиционного кода для источника с n возможными сообщениями.

2.3. Часто используемые коды

Любой код, используемый для представления, передачи, хранения и обработки информации, должен быть экономичным и нечувствительным к помехам, а соответствующие устройства кодирования и декодирования – простыми. По мере развития вычислительной техники было разработано много кодов. Эти коды классифицируются на **числовые** и **алфавитно-числовые коды.**

Числовые коды обеспечивают возможность представления цифр {0, 1, 2, ..., 9} с помощью двоичных 4-позиционных слов. Примеры числовых кодов представлены в *таблице 2.2.*

Таблица 2.2

Числовые коды

Цифра	*Название кода*			
	Прямой	*Грея*	*Айкена*	*С избытком 3*
0	0000	0000	0000	0011
1	0001	0001	0001	0100
2	0010	0011	0010	0101
3	0011	0010	0011	0110
4	0100	0110	0100	0111
5	0101	0111	1011	1000
6	0110	0101	1100	1001
7	0111	0100	1101	1010
8	1000	1100	1110	1011
9	1001	1101	1111	1100

Алфавитно-числовые коды представляют с помощью двоичных слов цифры 0, 1, 2, …, 9, строчные и прописные буквы алфавита, знаки препинания, знаки арифметических операций и т.д. В *таблице 2.3* представлен код **ASCII** (*American Standard Code for Information Interchange* – Американский стандартный код для информационного обмена), изобретенный в 1968 году.

Таблица 2.3

Код ASCII

Символ	*Двоичное слово*	*Десятичный эквивалент*	*Символ*	*Двоичное слово*	*Десятичный эквивалент*
Пробел	0100000	32	P	1010000	80
!	0100001	33	Q	1010001	81
"	0100010	34	R	1010010	82
#	0100011	35	S	1010011	83
$	0100100	36	T	1010100	84
%	0100101	37	U	1010101	85
&	0100110	38	V	1010110	86
'	0100111	39	W	1010111	87
(	0101000	40	X	1011000	88
)	0101001	41	Y	1011001	89
*	0101010	42	Z	1011010	90
+	0101011	43	[	1011011	91
,	0101100	44	\	1011100	92
-	0101101	45	]	1011101	93
.	0101110	46	^	1011110	94
/	0101111	47	_	1011111	95

Символ	*Двоичное слово*	*Десятичный эквивалент*	*Символ*	*Двоичное слово*	*Десятичный эквивалент*
0	0110000	48	`	1100000	96
1	0110001	49	a	1100001	97
2	0110010	50	b	1100010	98
3	0110011	51	c	1100011	99
4	0110100	52	d	1100101	100
5	0110101	53	e	1100101	101
6	0110110	54	f	1100110	102
7	0110111	55	g	1100111	103
8	0111000	56	h	1101000	104
9	0111001	57	i	1101001	105
:	0111010	58	j	1101010	106
;	0111011	59	k	1101011	107
<	0111100	60	l	1101100	108
=	0111101	61	m	1101101	109
>	0111110	62	n	1101110	110
?	0111111	63	o	1101111	111
@	1000000	64	p	1110000	112
A	1000001	65	q	1110001	113
B	1000010	66	r	1110010	114
C	1000011	67	s	1110011	115
D	1000100	68	t	1110100	116
E	1000101	69	u	1110101	117
F	1000110	70	v	1110110	118
G	1000111	71	w	1110111	119
H	1001000	72	x	1111000	120
I	1001001	73	y	1111001	121
J	1001010	74	z	1111010	122
K	1001011	75	{	1111011	123
L	1001100	76	|	1111100	124
M	1001101	77	}	1111101	125
N	1001110	78	~	1111110	126
O	1001111	79	Del	1111111	127

Этот код является 7-позиционным и включает $2^7 = 128$ символов. Первые 32 символа (двоичные слова 0000000, 0000001, 0000010, ..., 0011111) определяют технические детали передачи информации и не включены в таблицу. Двоичные слова 0100000, 0100001, 0100010, ..., 1111110 представляют печатные символы текстов на английском языке. Слово 1111111 представляет непечатаемый символ *Delete* (Стирание).

Кодирование сообщений осуществляется путем замены символов соответствующими двоичными словами. Например, слово START представляется в коде ASCII следующей последовательностью двоичных слов:

1010011 1010100 1000001 1010010 1010100.

Очевидно, что декодирование осуществляется в обратном порядке. Например, последовательность двоичных слов

1010011 1010100 1001111 1010000

представляет в коде ASCII слово STOP.

Как правило, языки программирования оперируют не с самими двоичными словами, а с их десятичными эквивалентами. В программах на языке ПАСКАЛЬ десятичные эквиваленты символов могут быть найдены с помощью предопределенной (встроенной) функции `ord`. Например:

`ord('S')=83;` `ord('T')=84;` `ord('A')=65;` `ord('R')=82`

и т.д. Предопределенная функция `chr` возвращает символ, соответствующий указанному десятичному эквиваленту. Так,

`chr(83)='S';` `chr(84)='T';` `chr(65)='A';` `chr(82)='R'.`

Ориентированный на английские тексты, код ASCII не включает буквы с диакритическими знаками и специальные графические символы, встречающиеся в разных европейских языках и в научных работах. Поэтому для современных компьютеров разработаны специальные версии кода ASCII, называемые **расширенными кодами ASCII**. Расширенные коды являются 8-позиционными и включают $2^8 = 256$ символов. Структура соответствующих кодов представлена в *таблице 2.4*.

Часть 1 каждого расширенного кода включает символы от 0 до 127, содержащиеся в обычном коде ASCII. ***Часть 2*** определена для каждой страны в отдельности и включает символы от 128 до 255. Эти символы используются как для представления букв национальных алфавитов, так и для часто используемых научных символов. Для примера в *таблице 2.4* представлены коды букв *Ă, ă, Â, â, Î, î, Ş, ş, Ţ, ţ* из алфавита румынского языка и коды букв *А, Б, В, …* из алфавита русского языка, предложенные в 1992 году фирмой *TISH* (Кишинэу). Очевидно, что использование расширенных кодов обеспечивает обработку информации, представленной на различных языках.

Другим примером алфавитно-числового кода является двоичный 8-позиционный код **EBCDIC** (*Extended Binary Coded Data Interchange Code* – Расширенный двоичный код для обмена данными), который используется для больших электронно-вычислительных машин.

Необходимо подчеркнуть, что расширение области применения 8-позиционных кодов способствует использованию байта и производных от него единиц для измерения количества информации:

1 *байт* = 2^3 = 8 *біюі*;

1 *Килобайт* = $2^{10} \approx 10^3$ *байт*;

1 *Мегабайт* = $2^{20} \approx 10^6$ *байт*;

1 *Гигабайт* = $2^{30} \approx 10^9$ *байт*;

1 *Терабайт* = $2^{40} \approx 10^{12}$ *байт*;

1 *Петабайт* = $2^{50} \approx 10^{15}$ *байт*.

В специальной литературе байт обозначается *В* (*byte*), а соответствующие производные единицы – *КВ*, *МВ*, *ГВ*, *ТВ* и *ПВ*.

Таблица 2.4

Структура расширенных ASCII кодов

Символ	*Двоичное слово*	*Десятичный эквивалент*	*Замечания*
Пробел	00100000	32	Часть 1: – символы кода ASCII
!	00100001	33	
"	00100010	34	
#	00100011	35	
…	…	…	
}	01111101	125	
~	01111110	126	
Del	01111111	127	
А	10000000	128	Часть 2: – специфические символы национальных языков; – псевдографические символы; – научные символы
Б	10000001	129	
В	10000010	130	
…	…	…	
≡	11110000	240	
Ă	11110001	241	
ă	11110010	242	
Â	11110011	243	
â	11110100	244	
Î	11110101	245	
î	11110110	246	
Ş	11110111	247	
ş	11111000	248	
’	11111001	249	
¯	11111010	250	
Ö	11111011	251	
Ţ	11111100	252	
ţ	11111101	253	
□	11111110	254	
	11111111	255	

Вопросы и упражнения

❶ Сколько возможных сообщений может быть закодировано с помощью m-позиционного кода?

❷ Определите коды, которые применяются в операционной системе, с которой вы работаете.

❸ Закодируйте в коде *Грея* следующие последовательности десятичных цифр: 123, 461, 952, 783, 472.

❹ Декодируйте сообщения, представленные в коде *Айкена*:

a) `0011 1111 0100`

b) `1110 0010 1101`

c) `1111 0000 0100`

d) `0010 0001 1011`

e) `0011 1100 1111`

f) `1111 1101 0000`

❺ Закодируйте в ASCII-коде выражения:

a) `A+B`

b) `FOR I=1 TO N`

c) `PRINT A$`

d) `NEXT I`

e) `PAUSE`

f) `PROGRAM`

❻ Декодируйте сообщения, представленные в ASCII-коде:

a) `1000010 1100101 1100111 1101001 1101110;`

b) `1010011 1110100 1101111 1110000;`

c) `1000101 1101110 1100100;`

d) `1101001 0111010 0111101 0110001 0111011.`

❼ Разработайте программу, которая выводит на экран коды следующих символов, введенных с клавиатуры:

а) десятичные цифры `0, 1, 2, ..., 9`;

б) прописные латинские буквы `A, B, C, ..., Z`;

в) строчные латинские буквы `a, b, c, ..., z`;

г) знаки арифметических операций;

д) специальные символы `;, <, =, >, ?, [, ], {, }, /, \`.

2.4. Информация непрерывных сообщений

Источники информации, изученные ранее, определялись с помощью переменной S, которая может принимать значения из конечного множества различных элементов $\{s_1, s_2, ..., s_n\}$, называемого множеством возможных сообщений. Практика показывает, что не все источники информации могут быть определены подобным образом. Для примера приведем ртутные или спиртовые термометры, спидометры автомобилей, микрофоны, видеокамеры и т.д. Такие источники могут быть определены с помощью переменной S (температура, мгновенная скорость, напряжение на выходных клеммах микрофона и т.д.), которая может принимать любые значения из заданного интервала $[s_{min}, s_{max}]$.

Источники информации, которые определены с помощью переменной S, принимающей значения из конечного множества различных элементов, называются **источниками с дискретными сообщениями**. Источники, определяемые с помощью переменной, которая может принимать любые значения из заданного интервала, называются **источниками с непрерывными (аналоговыми) сообщениями.**

Как и дискретные, непрерывные сообщения осуществляются во времени. Следовательно, S – функция времени, $S = S(t)$. С целью оценки количества

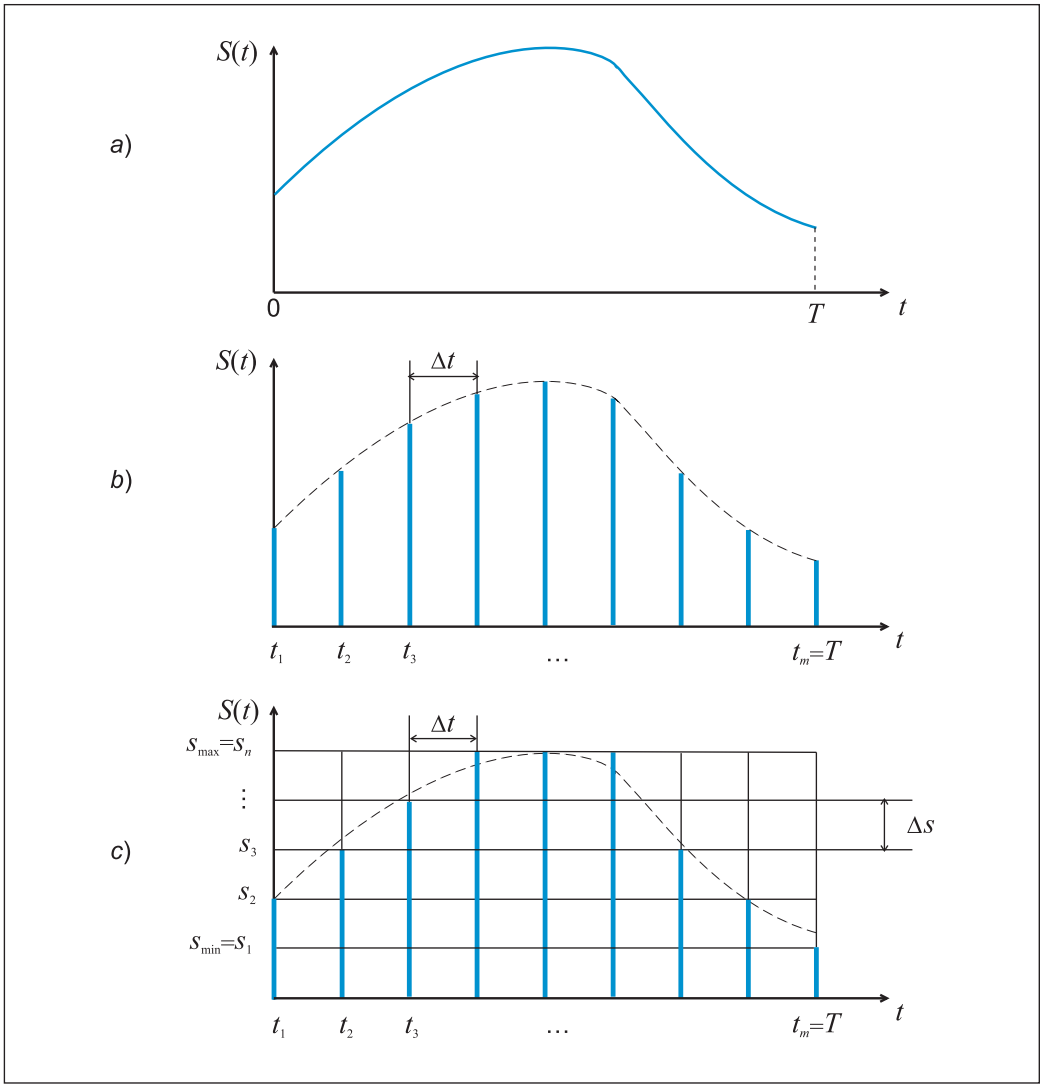


Рис. 2.4. Дискретизация непрерывных сообщений: *a* – непрерывное сообщение; *b* – сообщение, дискетизированное во времени; *c* – сообщение, дискретизированное во времени и по уровню

информации в непрерывных сообщениях будем рассматривать значения функции $S(t)$ только в моменты времени t_1, t_2, …, t_m (*рис. 2.4*). Множество соответствующих значений (отсчетов), обозначаемое $\{S(t_1), S(t_2), \ldots, S(t_m)\}$, называется **выборкой.**

Как правило, моменты времени t_1, t_2, …, t_m определяются в соответствии с соотношением

$$t_i = t_{i-1} + \Delta t.$$

Величина Δt называется **интервалом дискретизации.** Конкретное значение интервала дискретизации Δt выбирается исходя из скорости, с которой $S(t)$ изменется во времени.

Например, в метеорологии изменения температуры происходят на протяжении часов и $\Delta t = 1$ *час*, а в технике обработки звуковых сигналов $\Delta t = 5 \cdot 10^{-5}$ *с*.

Операция преобразования непрерывных сообщений в выборку называется дискретизацией во времени.

Очевидно, что до приема непрерывного сообщения конкретные отсчеты $S(t_1)$, $S(t_2)$, ..., $S(t_m)$ не известны получателю. Известен только интервал $[s_{min}, s_{max}]$, к которому принадлежат рассматриваемые значения. Для оценки количества информации в каждом из отсчетов выборки округлим значения $S(t_i)$, i=1, 2, ..., m до одного из заранее заданных значений s_1, s_2, ..., s_n (*рис. 2.4*).

Заранее выбранные значения s_1, s_2, ..., s_n называются **квантами**, а операция преобразования текущих значений непрерывных сообщений в кванты называется **дискретизацией по уровню** или **квантованием**.

Обычно

$$s_1 = s_{min}, \quad s_2 = s_{min} + \Delta s, \quad ..., \quad s_i = s_{i-1} + \Delta s, \quad ..., \quad s_n = s_{max}.$$

Величина Δs представляет **шаг** или **интервал квантования**. Конкретное значение интервала квантования зависит от физической природы источника информации, точности измерений, разрешающей способности приемника и т.д.

Например, для медицинского термометра $s_{min} = 34^o$, $s_{max} = 42^o$ и $\Delta s = 0{,}1^o$. В метеорологии $s_{min} = -60^o$, $s_{max} = +60^o$, $\Delta s = 1^o$. Для спидометра автомобиля $s_{min} = 0$, $s_{max} = 150$ *км/ч*, $\Delta s = 5$ *км/ч*. Количество отсчетов выборки m и количество квантов n определяются следующими соотношениями (*рис. 2.4, b* и *c*):

$$m = \frac{T}{\Delta t} + 1; \quad n = \frac{|s_{max} - s_{min}|}{\Delta s} + 1,$$

где T – длительность непрерывного сообщения.

Поскольку кванты s_1, s_2, ..., s_n могут рассматриваться как дискретные сообщения, количество информации в одном отсчете:

$$I = \log_2 n = \log_2 \left(\frac{|s_{max} - s_{min}|}{\Delta s} + 1 \right),$$

а количество информации в непрерывном сообщении:

$$V = mI = \left(\frac{T}{\Delta t} + 1 \right) \log_2 \left(\frac{|s_{max} - s_{min}|}{\Delta s} + 1 \right).$$

Например, для медицинского термометра

$$I = \log_2 \left(\frac{|42 - 34|}{0{,}1} + 1 \right) \approx 6{,}34 \text{ } бит,$$

а для спидометра автомобиля

$$I = \log_2 \left(\frac{|150 - 0|}{5} + 1 \right) \approx 4{,}95 \text{ } бит.$$

Количество информации в аудиозаписи для $n = 256$, $\Delta t = 5 \cdot 10^{-5}$ *с* и T = = 45 *мин* составляет

$$V = \frac{45 \cdot 60}{5 \cdot 10^{-5}} \log_2 256 = 4{,}32 \cdot 10^8 \text{ бит} = 432 \text{ Мбит}.$$

Если точность измерения и разрешающая способность приемника возрастают, то интервал дискретизации Δt и шаг квантования Δs должны быть уменьшены. Следовательно, возрастает и количество информации, содержащейся в непрерывном сообщении.

Информация непрерывных сообщений может быть представлена с помощью последовательности двоичных слов. Для этого кванты s_1, s_2, ..., s_n кодируются точно так же, как и любое дискретное сообщение. Например, показания медицинского термометра ($I \approx 6{,}34$ *бит*) могут быть закодированы с помощью 7-позиционного кода. Чаще всего используются прямые числовые коды (*таблица 2.2*), причем кодовое слово представляет количество соответствующих квантов. В некоторых приложениях используется код *Грея*, который нечувствителен к помехам.

Устройство, которое преобразовывает непрерывное сообщение, поданное на его вход, в последовательность кодовых слов, называется **аналого-цифровым преобразователем (АЦП)**. Обратная операция, состоящая в преобразовании кодовых слов в непрерывное сообщение, осуществляется с помощью **цифро-аналогового преобразователя (ЦАП)**. Использование преобразователей необходимо в случаях, когда обрабатываемая информация представлена непрерывными сообщениями: при контроле технологических процессов, в управлении движущимися объектами, в мониторинге физиологических параметров в медицине, для фильтрации и микширования аудиосигналов и т.д.

Вопросы и упражнения

❶ В чем разница между источниками с дискретными и с непрерывными сообщениями?

❷ Приведите несколько примеров источников непрерывных сообщений. Уточните интервал изменения переменной, описывающей источник.

❸ Объясните, как осуществляется операция дискретизации во времени. Как выбирается интервал дискретизации?

❹ Объясните, как осуществляется операция квантования. Как выбирается шаг квантования?

❺ Как влияют интервал дискретизации и шаг квантования на количество информации, извлекаемой из непрерывного сообщения?

❻ Импульсный альтиметр (высотомер) самолета может измерять высоту от 100 *м* до 20 *км*. Погрешность измерения не превышает 1 *м*. Чтобы выполнить одно измерение, нужно 10^{-3} *с*. Определите количество информации, выдаваемой альтиметром за 5 часов полета.

❼ Температура внутри химического реактора записывается на бумажной миллиметровой ленте. По оси абсцисс отображается время (1 *мм* соответствует 1 часу), а по оси ординат – температура (1 *мм* соответствует 10°*C*). Сколько информации содержит запись, осуществленная за 30 дней, если температура может изменяться от 80 до 1000°*C* ?

❽ Для записи звука используется микрофон, напряжение на выходе которого изменяется от 0 до 100 *мкВ*. Устройство записи не различает уровни напряжений, отличающиеся менее чем на 0,1 *мкВ*. Для обеспечения качест-

венного воспроизведения каждую секунду берутся 40000 отсчетов. Сколько информации вырабатывает микрофон на протяжении 3 часов?

❾ Для чего предназначены аналого-цифровые и цифро-аналоговые преобразователи?

❿ Напишите программу, которая вводит с клавиатуры текущие значения отсчетов и выводит на экран коды соответствующих квантов.

⓫ Напишите программу, которая моделирует работу цифро-аналогового преобразователя.

2.5. Квантование изображений

Изображением называется представление некоторого объекта, выполненное на поверхности самим пользователем напрямую или с помощью определенных устройств. Для примера вспомним рисунки, фотографии, изображения, полученные с помощью различных систем – оптических, оптико-механических или оптико-электронных: микроскопа, телескопа, кинопроектора, телевизора и т.д.

Для измерения количества информации изображение делится на **микрозоны**, называемые чаще всего **точками** или **пикселями**. Разложение изображения на точки осуществляется с помощью **растра** (от латинского слова *raster* – «грабли»). Растр представляет плоскую поверхность, в общем случае прямоугольную, на которую нанесены два ряда параллельных линий, перпендикулярных между собой (*рис. 2.5*). Плотность линий и соответственно плотность точек характеризуют разрешающую способность оборудования для воспроизведения или формирования изображений.

Например, для газетных иллюстраций используются растры с разрешением 24–30 *линий/см* (576–900 точек на 1 $см^2$), а для воспроизведения картин – растры с 54–60 *линий/см*. Растр монитора, то есть рисунок, который формирует электронный луч на экране электронно-лучевой трубки (кинескопа), может включать 640 × 480, 800 × 600, 720 × 400, ..., 1024 × 1024 точек.

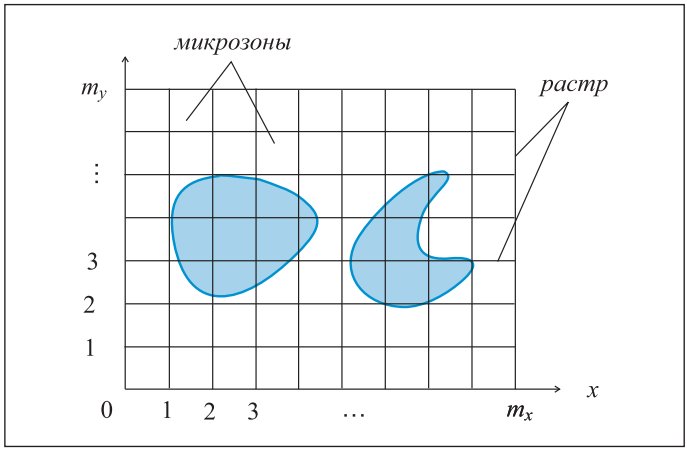


Рис. 2.5. Разложение изображения на микрозоны

Разложение изображения на точки (микрозоны) представляет собой операцию дискретизации в пространстве.

В случае монохромных (черно-белых) изображений каждая микрозона описывается ее **яркостью**, которая в общем случае представляет собой непрерывную величину. Эта величина может быть дискретизована (квантована) по уровню. Количество квантов n будет характеризовать разрешающую способность оборудования для воспроизведения или формирования изображений. Следовательно, количество информации в одном монохромном изображении:

$$I = m_x m_y \log_2 n,$$

где m_x и m_y представляют количество микрозон соответствующего растра по горизонтали и вертикали (*рис. 2.5*).

Поскольку цвета могут быть переданы путем совмещения трех представлений одного и того же изображения в красном, зеленом и синем, то количество информации в цветном изображении определяется соотношением

$$I = 3\, m_x m_y \log_2 n.$$

Изображения движущихся объектов дискретизируются во времени, представляя обычно 24 (в кинематографии) или 25 (в телевидении) кадров в секунду. Следовательно, количество информации в одном фильме продолжительностью T секунд определяется соотношением

$$V = T f I,$$

где f – это частота кадров, а I – количество информации в одном кадре.

Например, в телевидении $m_x \approx m_y = 625$, $n = 32$ и $f = 25$ кадров в секунду. Один кадр будет содержать:

$$I = 3 \cdot 625 \cdot 625 \cdot \log_2 32 \approx 5{,}6 \text{ } \textit{Мбит}.$$

Цветной фильм продолжительностью 1,5 часа будет содержать:

$$V = 1{,}5 \cdot 3\,600 \cdot 25 \cdot I \approx 791 \text{ } \textit{Гбит}.$$

Набор двоичных слов, содержащих информацию о микрозонах, называется цифровым изображением. Операция преобразования изображения в набор двоичных слов называется квантованием (оцифровыванием) изображения.

Изображения, полученные из видеокамер, квантуются с помощью аналого-цифровых преобразователей. Изображения на бумаге могут быть квантованы с помощью специального устройства – **сканера**. Это устройство содержит фоточувствительные ячейки, аналого-цифровые преобразователи и механизм относительного перемещения бумаги и фотоячеек.

Цифровые изображения преобразуются в видимые изображения с помощью цифро-аналоговых преобразователей и устройства формирования растра: электронно-лучевой трубки и системы отклонения монитора, матрицы иголок в механических принтерах и т.д.

Вопросы и упражнения

❶ Назовите операции, необходимые для квантования изображений.

❷ Каково назначение растра? Из каких соображений выбирается плотность линий растра?

❸ Как оценить количество информации, содержащейся в монохромном изображении?

❹ Как можно передать цвета многоцветного изображения? Как измерить количество информации в цветном изображении?

❺ Оцените количество информации в газетной фотографии размерами 10×10 *см*, переданной с помощью растра, содержащего 24 *точек/см*. Каждая точка может иметь следующие оттенки: белый, светло-серый, темно-серый, черный.

❻ Сколько информации содержит цветная фотография размерами 20×20 *см*, воспроизведенная с помощью растра, содержащего 60 *точек/см*? Может быть передано до 256 уровней яркости соответствующих точек.

❼ Растр видеокамеры состоит из 1024×1024 точек. Может быть передано до 64 уровней яркости соответствующих точек. Сколько информации будет содержаться в фильме продолжительностью 3 часа?

❽ Цифровое изображение содержит по одному двоичному слову для каждой точки растра монитора. Сколько уровней яркости можно отобразить на экране, если слова цифрового изображения являются 3-позиционными? 5-позиционными? 8-позиционными?

2.6. Представление и передача информации

Материальный объект, используемый для хранения, передачи или обработки информации, называется **носителем информации**. Различают статические и динамические носители информации.

Статические носители используются для хранения информации или, другими словами, для передачи ее во времени. Информация, записанная на статический носитель, может быть считана с целью последующей обработки или использования. Первыми носителями информации, использованными людьми, были камни, таблички из обожженной глины, папирус. Другим носителем информации является бумага. Сообщения, записанные на бумаге в рукописной форме, в виде рисунков или печатных текстов могут сохраняться очень продолжительное время. В компьютерах в качестве статических носителей информации используются:

– бумага для механических, струйных, лазерных и других принтеров;
– активные слои магнитных лент и дисков;
– отражающие среды оптических дисков и т.п.

Передача информации в пространстве осуществляется с помощью **динамических носителей**. В качестве динамических носителей современная техника использует:

– акустические волны в газах (воздухе) или жидкостях;
– электрические напряжение и ток;
– электромагнитные волны и т.п.

Поскольку передача информации производится в пространстве и времени, как минимум хотя бы одна физическая величина носителя информации должна изменяться.

> *Изменение физической величины, обеспечивающее передачу сообщений, называется сигналом. Характеристика сигнала, используемая для представления (описания) сообщений, называется информационным параметром.*

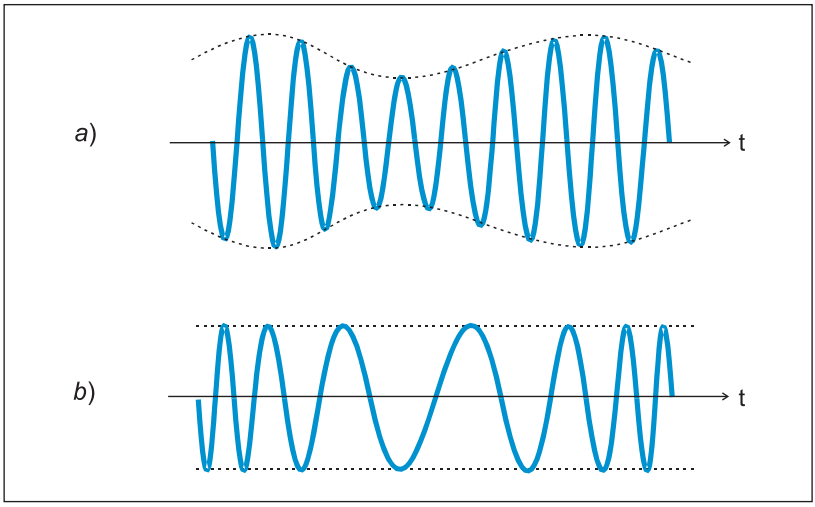


Рис. 2.6. Информационные параметры электромагнитных волн:
a – амплитуда; *b* – частота

Например, в радио- и телевещании в качестве носителя информации используются электромагнитные волны. Амплитуда или частота этих волн может изменяться во времени (*рис. 2.6*). В первом случае информационным параметром является амплитуда колебаний, а во втором – их частота.

В электронных компьютерах в качестве носителя информации обычно используется электрический ток, а напряжение и сила тока являются информационными параметрами сигнала. Форма рассматриваемых сигналов представлена на *рис. 2.7*. В случае уровней напряжения с одним значением напряжения связывается двоичная цифра **0**, а с другим – двоичная цифра **1**. Двоичные цифры **0** и **1** могут также быть связаны соответственно с отсутствием или наличием импульса, с отрицательным или положительным импульсом и др.

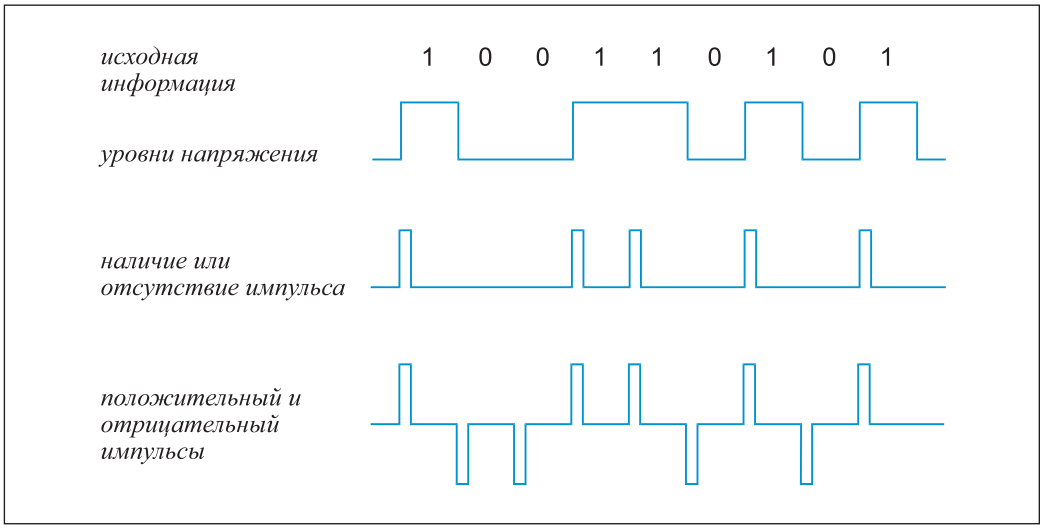


Рис. 2.7. Сигналы, используемые в вычислительной технике

Сигнал называется дискретным, если соответствующий информационный параметр может принимать конечное число значений. Сигнал называется непрерывным, если информационный параметр может принимать произвольные значения из заданного интервала.

Например, сигналы на *рис. 2.6* – непрерывные, а сигналы на *рис. 2.7* – дискретные. Очевидно, что дискретные и непрерывные сигналы являются формой представления соответствующих сообщений.

Любая техническая система использует те сигналы, которые обеспечивают наилучшую реализацию функций, для которых она была спроектирована. Современные компьютеры используют уровни напряжения, телефонные сети – электрический ток, а радио и телевидение – электромагнитные волны и т.д. Практика показывает, что непрерывные сигналы могут быть переданы на значительно большие расстояния, чем дискретные сигналы. Следовательно, для обеспечения связи между компьютерами, находящимися на расстоянии, при передаче дискретных сигналов они должны быть преобразованы в непрерывные. При приеме осуществляется обратное преобразование: непрерывный сигнал преобразуется в дискретный.

Операция, при помощи которой информационный параметр непрерывного сигнала изменяется в зависимости от значений дискретного сигнала, называется модуляцией.

Техническое устройство, которое реализует указанную операцию, называется **модулятором**. Для примера на *рис. 2.8* представлены операции модуляции по амплитуде и частоте.

Операция извлечения дискретного сигнала из непрерывного в соответствии с применяемым способом модуляции называется демодуляцией.

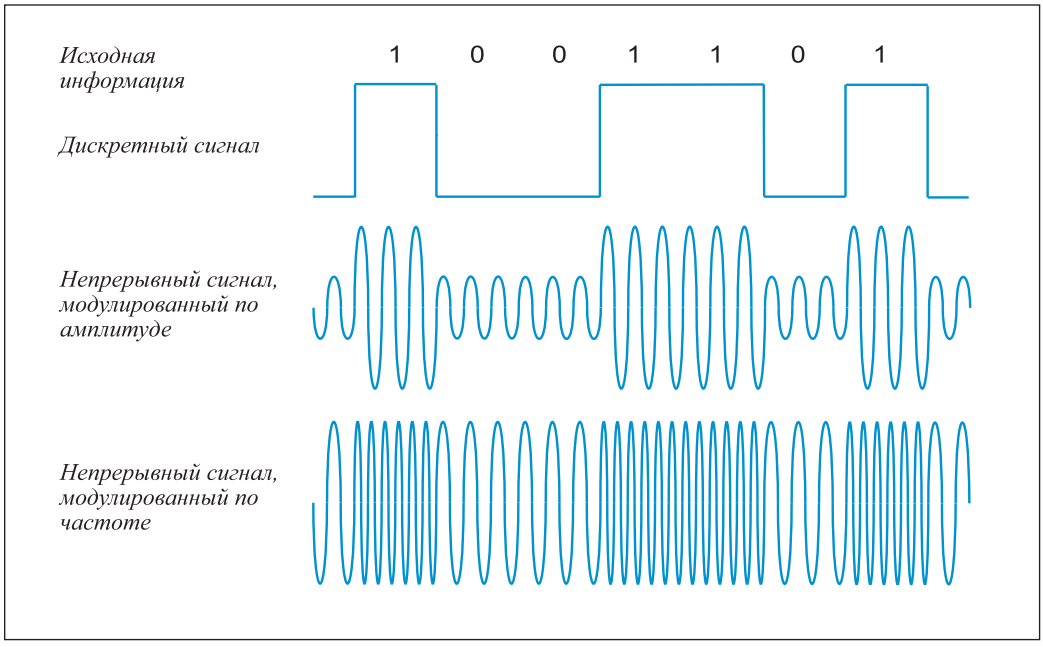


Рис. 2.8. Модуляция непрерывных сигналов

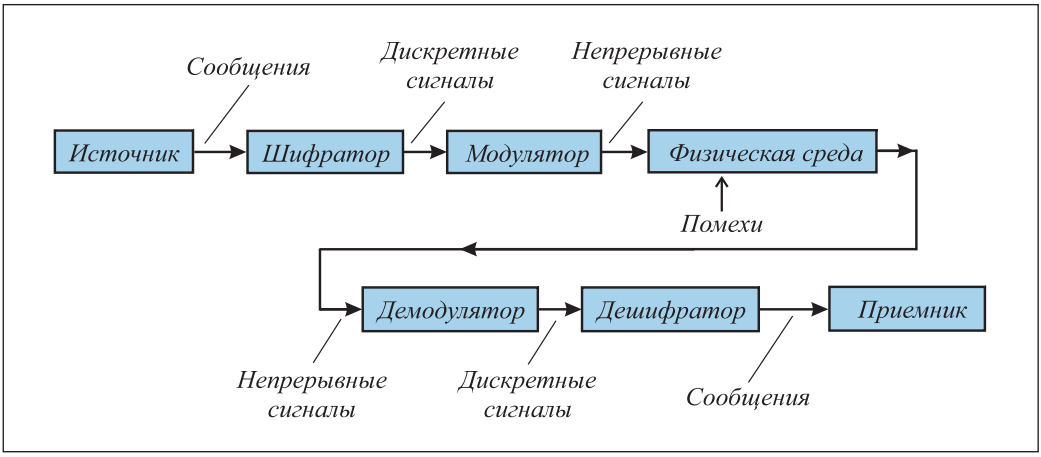


Рис. 2.9. Подробная схема системы передачи информации

Техническое устройство, которое реализует соответствующую операцию, называется **демодулятор.**

Подробная схема системы передачи информации представлена на *рис. 2.9.*

Вспомним назначение компонент рассматриваемой системы:

шифратор – преобразует сообщения, вырабатываемые источником в двоичные слова;

модулятор – преобразует дискретные сигналы, представляющие собой двоичные слова, в непрерывные сигналы;

физическая среда – представляет собой проводники, оптические волокна, эфир и т.п., через которые распространяются непрерывные сигналы;

демодулятор – преобразует непрерывные сигналы в дискретные;

дешифратор – преобразует двоичные слова в сообщения.

Очевидно, что в случае использования кодов для коррекции и обнаружения ошибок система передачи информации будет помехоустойчивой.

Главной характеристикой любой системы передачи информации является ее **пропускная способность**, которая выражается в ***битах в секунду***. Пропускная способность зависит от физических характеристик компонент системы, методов модуляции-демодуляции, статистических характеристик помех. Например, пропускная способность телефонного канала составляет около 34 *Кбит/с*; пропускная способность радиоканала с сантиметровыми волнами – около 1 *Гбит/с*; пропускная способность оптического канала – 1 *Тбит/с*.

Вопросы и упражнения

❶ В чем отличия статических и динамических носителей информации?

❷ Определите тип следующих носителей информации:
а) перфокарты;
б) ультразвуковые волны;
в) перфоленты;
г) акустические волны;
д) фотопленка;
е) гравитационные волны;
ж) фотобумага.

❸ Назовите информационные параметры сигналов от следующих источников:
а) микрофон;
б) радиостанция, волны – длинные, средние или короткие;
в) музыкальный инструмент;
г) радиостанция, ультракороткие волны;
д) видеокамера.

❹ Опишите носители информации и сигналы, используемые в современных компьютерах.

❺ Объясните операции модуляции и демодуляции сигналов.

❻ Каково назначение модулятора? Демодулятора?

❼ На *рис. 2.8* приведены непрерывные сигналы, представляющие двоичное слово 1001101. В коде ASCII (*таблица 2.3*) это слово соответствует символу *M*. Нарисуйте непрерывные сигналы, соответствующие следующим символам:

a)	`>;`	*e)*	`<;`
b)	`r;`	*f)*	`K;`
c)	`W;`	*g)*	`a;`
d)	`9;`	*h)*	`@.`

❽ От чего зависит пропускная способность канала? В каких единицах она измеряется?

❾ Как влияют помехи на пропускную способность канала?

Тест для самопроверки № 2

1. Запишите множество 3-позиционных слов, состоящих из двоичных цифр {0, 1}.

2. В коде, предложенном английским философом Фрэнсисом Бэконом заглавные буквы латинского алфавита представляются следующим образом:

A – 00000, B – 00001, C – 00010, D – 00011, ..., Z – 11001.

Декодируйте, воспользовавшись *кодом Бэкона*, двоичные слова 00010, 00000, 00010, 00001. Используя этот же код, закодируйте слово BAC.

3. Имеется источник информации, множество всевозможных сообщений которого состоит из заглавных и строчных букв латинского алфавита, $S = \{A, B, C, ..., Z, a, b, c, ..., z\}$. Найдите минимальную длину m двоичных слов, которые обеспечивают однозначное кодирование и декодирование сообщений рассматриваемого источника информации.

4. Определите минимальную длину m двоичных слов, которые обеспечивают однозначное кодирование и декодирование числовых сообщений электронного календаря. Соответствующие сообщения имеют вид *дд.мм.гг*, где *дд* представляет день, *мм* – месяц, а *гг* – год (последние две цифры).

5. Какое количество информации содержится в одном символе расширенного кода ASCII?

6. Предполагая, что текст будет представлен в расширенном коде ASCII, определите количество информации, содержащейся в диктанте, написанном в течение 10 минут учеником, который может писать со скоростью около 200 символов в минуту.

7. Напишите на ПАСКАЛЕ программу, которая выводит на экран коды символов, введенных с клавиатуры.

8. Напишите на ПАСКАЛЕ программу, которая выводит на экран символы, соответствующие кодам, введенным с клавиатуры.

9. Вычислите количество информации, содержащейся в цветной фотографии с размерами 10×10 *см*, воспроизведенной с помощью растра 30 *точек*/*см*. Может быть передано до 128 уровней яркости соответствующих точек.

10. Выберите из списка статические носители информации:

a)	перфокарта;	*e)*	электромагнитная волна;
b)	ультразвуковая волна;	*f)*	магнитная карта;
c)	магнитная лента;	*g)*	электрический ток;
d)	фотопленка;	*h)*	бумага для принтера.

11*. Медицинский термометр обеспечивает измерение температуры в диапазоне 34°–42°C с точностью 0,1°. На протяжении одного дня температура пациента была измерена 3 раза. Какое количество информации содержат результаты этих измерений?

12*. Вычислите, какое количество информации содержится в аудиозаписи, сделанной с интервалом дискретизации $\Delta t = 3 \cdot 10^{-5}$ *с*, на протяжении $T = 30$ *мин,* при количестве уровней квантования $n = 128$.

13*. Покажите соответствие между понятиями из левого столбца и определениями из правого столбца:

(1)	код;	(a)	устная или письменная новость, сообщение;
(2)	сигнал;	(b)	множество двоичных чисел;
(3)	информация;	(c)	упорядоченное множество знаков;
(4)	алфавит;	(d)	определенное число битов;
		(e)	правило преобразования сообщений в двоичные слова;
		(f)	изменение физической величины для передачи сообщений.

14*. Укажите соответствие между наименованиями устройств (из левого столбца) системы передачи информации и их назначением (из правого столбца):

(1)	шифратор;	(a)	преобразование непрерывных сигналов в дискретные;
(2)	модулятор;	(b)	преобразование аудиозаписи в двоичные слова;
(3)	демодулятор;	(c)	преобразование сообщений источника в двоичные слова;
(4)	дешифратор;	(d)	преобразование графических изображений в двоичные слова;
		(e)	преобразование двоичных слов в сообщения;;
		(f)	преобразование дискретных сигналов в непрерывные.

* Только для реального профиля.

Глава 3

АРИФМЕТИЧЕСКИЕ ОСНОВЫ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

3.1. Системы счисления

В цифровых компьютерах информация любого вида представляется, хранится и обрабатывается в числовой форме. Числа представляются с помощью элементарных символов, называемых **цифрами.**

> *Совокупность правил представления чисел вместе с множеством используемых цифр носит название системы счисления. Количество цифр определяет основание системы счисления.*

Приведем несколько примеров систем счисления:

– **десятичная система** – это система счисления по основанию 10 с количеством используемых цифр, равным 10, соответственно 0, 1, 2, …, 9;

– **двоичная система** – это система счисления по основанию 2 с количеством используемых цифр равным 2, то есть 0 и 1. Соответствующие цифры называются **двоичными цифрами** или **битами**;

– **троичная система** – это система счисления по основанию 3 с количеством используемых цифр равным 3, соответственно 0, 1 и 2;

– **восьмеричная система** – это система счисления по основанию 8, содержащая 8 цифр: 0,1, 2, ,7;

– **шестнадцатеричная система** – это система счисления по основанию 16, которая содержит 16 цифр: 0, 1, 2, …, 9, *A* (десять), *B* (одиннадцать), *C* (двенадцать), *D* (тринадцать), *E* (четырнадцать), *F* (пятнадцать).

В *таблице 3.1* представлены одни и те же числа в различных системах счисления.

Таблица 3.1.

Представление чисел в различных системах счисления

Десятичная	*Двоичная*	*Восьмеричная*	*Шестнадцатеричная*
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6

Десятичная	*Двоичная*	*Восьмеричная*	*Шестнадцатеричная*
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D

Десятичная	*Двоичная*	*Восьмеричная*	*Шестнадцатеричная*
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
. . .	. . .	. . .	. . .
30	11110	36	1E
. . .	. . .	. . .	. . .
40	101000	50	28
. . .	. . .	. . .	. . .

Десятичная	*Двоичная*	*Восьмеричная*	*Шестнадцатеричная*
50	110010	62	32
. . .	. . .	. . .	. . .
60	111100	74	3C
. . .	. . .	. . .	. . .
70	1000110	106	46
. . .	. . .	. . .	. . .
80	1010000	120	50
. . .	. . .	. . .	. . .
90	1011010	132	5A
. . .	. . .	. . .	. . .
100	1100100	144	64
. . .	. . .	. . .	. . .

Правило представления чисел в десятичной системе видно из следующего примера:

$$(3856{,}43)_{10} = 3\cdot10^3 + 8\cdot10^2 + 5\cdot10^1 + 6\cdot10^0 + 4\cdot10^{-1} + 3\cdot10^{-2}.$$

Заметим, что в этом представлении значение (вес) каждой цифры зависит от положения, которое она занимает в числе. Например, цифра 3 встречается два раза: первый раз она имеет значение 3000, а второй раз – 0,03.

Системы, в которых значение цифр зависит от занимаемого ими положения в представлении (записи) чисел, называются позиционными.

Предположим, что число N имеет целую часть, состоящую из $n + 1$ цифры, а дробную часть – из m цифр:

$$N = c_n c_{n-1} \ldots c_1 c_0, c_{-1} c_{-2} \ldots c_{-m}.$$

Значение этого числа вычисляется в зависимости от основания системы следующим образом:

$$(N)_b = c_n b^n + c_{n-1} b^{n-1} + \ldots + c_1 b^1 + c_0 b^0 + c_{-1} b^{-1} + c_{-2} b^{-2} + \ldots + c_{-m} b^{-m}.$$

Путем соответствующих вычислений выполняется **преобразование** числа $(N)_b$ по основанию b в десятичную систему счисления.

Например,

$(101,1)_{10} = 1\cdot10^2 + 0\cdot10^1 + 1\cdot10^0 + 1\cdot10^{-1} = 101,1;$

$(101,1)_2 = 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 + 1\cdot2^{-1} = 5,5;$

$(101,1)_3 = 1\cdot3^2 + 0\cdot3^1 + 1\cdot3^0 + 1\cdot3^{-1} = 10,333\ldots;$

$(101,1)_8 = 1\cdot8^2 + 0\cdot8^1 + 1\cdot8^0 + 1\cdot8^{-1} = 65,125;$

$(101,1)_{16} = 1\cdot16^2 + 0\cdot16^1 + 1\cdot16^0 + 1\cdot16^{-1} = 257,0625.$

Формально десятичная система не имеет никаких особых преимуществ перед другими системами счисления. Предполагается, что эта система была принята в древние времена благодаря тому факту, что в процессе счета в качестве инструмента использовались пальцы рук.

Компьютер может работать в любой системе счисления. В ходе развития вычислительной техники было установлено, что наиболее удобна двоичная система. Предпочтение ей можно отдать по следующим причинам:

– простота правил для арифметических и логических операций;

– физическое представление цифр с целью хранения и обработки чисел осуществляется гораздо легче для двух символов, чем для десяти: перфорирован – не перфорирован, контакт замкнут – контакт разомкнут, наличие или отсутствие тока и т.д.;

– схемы, которые должны различать только два состояния, более надежны в работе, чем схемы, которые должны различать десять состояний.

Отметим, что в процессе развития цивилизации были созданы и **непозиционные системы счисления**. Ярким примером может служить римская система, использующая цифры *I, V, X, L, C, D, M*. Поскольку правила представления чисел и выполнения арифметических операций в таких системах очень сложны, они имеют ограниченное применение.

Вопросы и упражнения

❶ Как определяется система счисления?

❷ В чем отличие между позиционными и непозиционными системами счисления?

❸ Приведите примеры позиционных систем счисления. Как определяется основание системы счисления?

❹ Переведите в десятичную систему счисления число $(101,1)_b$, записанное в следующих системах счисления:

$b = 4, 5, 6, 7, 9, 11, 12, 13, 14, 15.$

❺ Переведите в десятичную систему счисления следующие числа:

a)	$(328)_9$;	*g)*	$(328)_{10}$;	*m)*	$(328)_{11}$;	*s)*	$(328)_{16}$;
b)	$(516)_7$;	*h)*	$(516)_8$;	*n)*	$(516)_9$;	*t)*	$(516)_{16}$;
c)	$(1010,01)_2$;	*i)*	$(1010,01)_3$;	*o)*	$(1010,01)_8$;	*u)*	$(1010,01)_{16}$;
d)	$(201,12)_3$;	*j)*	$(201,12)_4$;	*p)*	$(201,12)_8$;	*v)*	$(201,12)_{16}$;
e)	$(341,02)_5$;	*k)*	$(341,02)_6$;	*q)*	$(341,02)_8$;	*w)*	$(341,02)_{12}$;
f)	$(FFFF)_{16}$;	*l)*	$(1111)_{16}$;	*r)*	$(ABCD)_{16}$;	*x)*	$(F001)_{16}$.

❻ Какие факторы способствовали использованию в вычислительной технике двоичной системы счисления?

❼ Напишите программу, переводящую числа, записанные в системах счисления по основанию b, $b < 10$, в десятичную систему счисления.

❽ Напишите программу, переводящую числа, записанные в системах счисления по основанию b, $10 < b \leq 36$, в десятичную систему счисления.

3.2. Перевод чисел из одной системы счисления в другую

Перевод числа $(N)_b$ в его десятичный эквивалент осуществляется в соответствии с формулой из предыдущего параграфа:

$$(N)_b = c_n b^n + c_{n-1} b^{n-1} + \ldots + c_1 b^1 + c_0 b^0 + c_{-1} b^{-1} + c_{-2} b^{-2} + \ldots + c_{-m} b^{-m}.$$

Перевод десятичного числа $(N)_{10}$ в его эквивалент по основанию b осуществляется по следующим правилам:

– делением на соответствующее основание целой части и целочисленных частных после каждой операции деления до получения нулевого частного; результат перевода целой части составляется из целочисленных остатков, записанных в порядке, обратном их вычислению;

– умножением на основание дробной части, а затем и дробных частей, полученных в предшествующих умножениях, до тех пор, пока дробная часть очередного произведения не станет равной нулю или до получения требуемого количества цифр дробной части результата; результат преобразования дробной части состоит из целых частей произведений, записанных в порядке их вычисления.

Проанализируем несколько примеров.

1. Перевести десятичное число 53,40625 в его двоичный эквивалент.

$53 : 2 = 26 + {}^{1}/_{2}$;
$26 : 2 = 13 + {}^{0}/_{2}$;
$13 : 2 = 6 + {}^{1}/_{2}$;
$6 : 2 = 3 + {}^{0}/_{2}$;
$3 : 2 = 1 + {}^{1}/_{2}$;
$1 : 2 = 0 + {}^{1}/_{2}$.

Следовательно, целая часть двоичного числа будет 110101.

$0{,}40625 \times 2 = 0{,}8125$;
$0{,}8125 \times 2 = 1{,}625$;
$0{,}625 \times 2 = 1{,}25$;
$0{,}25 \times 2 = 0{,}5$;
$0{,}5 \times 2 = 1{,}0$.

Дробная часть двоичного числа будет 01101. Следовательно,

$$(53{,}40625)_{10} = (110101{,}01101)_2.$$

2. Перевести число 23,7 из десятичной системы в двоичную.

$23 : 2 = 11 + {}^{1}/_{2}$;
$11 : 2 = 5 + {}^{1}/_{2}$;
$5 : 2 = 2 + {}^{1}/_{2}$;
$2 : 2 = 1 + {}^{0}/_{2}$;
$1 : 2 = 0 + {}^{1}/_{2}$;

$0{,}7 \times 2 = 1{,}4$;
$0{,}4 \times 2 = 0{,}8$;
$0{,}8 \times 2 = 1{,}6$;
$0{,}6 \times 2 = 1{,}2$;

$0{,}2 \times 2 = 0{,}4;$
$0{,}4 \times 2 = 0{,}8;$
...

Заметим, что операция может быть продолжена до бесконечности, то есть не существует точного перевода анализируемого числа. Следовательно,

$$(23{,}7)_{10} = (10111{,}101100...)_2.$$

3. Осуществить перевод числа 1996,0625 из десятичной системы в восьмеричную.

$1996 : 8 = 249 + {}^{4}/_{8};$
$249 : 8 = 31 + {}^{1}/_{8};$
$31 : 8 = 3 + {}^{7}/_{8};$
$3 : 8 = 0 + {}^{3}/_{8};$

$0{,}0625 \times 8 = 0{,}5;$
$0{,}5 \times 8 = 4.$

Следовательно,

$$(1996{,}0625)_{10} = (3714{,}04)_8.$$

4. Перевести число 2914,25 из десятичной системы в шестнадцатеричную.

$2914 : 16 = 182 + {}^{2}/_{16};$
$182 : 16 = 11 + {}^{6}/_{16};$
$11 : 16 = 0 + {}^{11}/_{16};$

$0{,}25 \times 16 = 4.$

Следовательно,

$$(2914{,}25)_{10} = (\mathrm{B62{,}4})_{16}.$$

Вопросы и упражнения

❶ Как осуществляется перевод числа из системы по основанию b в десятичную систему?

❷ Переведите в десятичную систему следующие числа:

a) $(100001{,}01111)_2$; *e)* $(\mathrm{AAA{,}BBB})_{16}$; *i)* $(124{,}521)_7$;
b) $(328{,}678)_9$; *f)* $(\mathrm{EE{,}00F})_{16}$; *j)* $(4231{,}124)_5$;
c) $(100{,}100)_{16}$; *g)* $(1221{,}1112)_3$; *k)* $(50{,}505050)_6$;
d) $(10{,}01)_{16}$; *h)* $(1321{,}1312)_4$; *l)* $(7777{,}001)_8$.

❸ Как выполняется перевод десятичного числа в его эквивалент по основанию b?

❹ Переведите в двоичную систему следующие десятичные числа:

a) 13,889; *c)* 53,536; *e)* 93,447;
b) 38,668; *d)* 29,261; *f)* 70,212;

g)	8,347;	*i)*	58,749;	*k)*	3,156;
h)	39,764;	*j)*	4,345;	*l)*	91,428.

Проверьте результаты, осуществляя перевод из двоичной в десятичную систему счисления.

❺ Переведите в восьмеричную систему следующие десятичные числа:

a)	358,932;	*e)*	886,526;	*i)*	795,128;
b)	479,093;	*f)*	971,258;	*j)*	680,895;
c)	591,241;	*g)*	515,914;	*k)*	256,453;
d)	649,113;	*h)*	347,607;	*l)*	838,261.

Проверьте результаты, выполняя перевод из восьмеричной системы в десятичную.

❻ Переведите в шестнадцатеричную систему следующие десятичные числа:

a)	1424,699;	*e)*	5818,961;	*i)*	4985,995;
b)	3517,315;	*f)*	9336,491;	*j)*	9721,678;
c)	9607,201;	*g)*	3442,722;	*k)*	5292,837;
d)	8974,664;	*h)*	4521,449;	*l)*	2734,592.

Проверьте результаты, выполняя перевод из шестнадцатеричной системы в десятичную.

❼ Напишите программу, которая переводит десятичные числа в эквиваленты из соответствующих систем счисления по основанию b, $b < 10$.

❽ Напишите программу, которая переводит десятичные числа в числа систем по основанию b, $10 < b \leq 36$.

3.3. Перевод чисел из двоичной системы счисления в восьмеричную, шестнадцатеричную и обратно

Поскольку $8 = 2^3$, **двоично-восьмеричный перевод** и **восьмерично-двоичный перевод** может быть осуществлен напрямую. Любая восьмеричная цифра представляется тремя двоичными цифрами:

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

Если дано восьмеричное число, то для его перевода в двоичное каждая восьмеричная цифра заменяется тремя двоичными.

Примеры:

$(247{,}315)_8 = (010\ 100\ 111,\ 011\ 001\ 101)_2$;

$(512,07)_8 = (101\ 001\ 010,\ 000\ 111)_2$;

$(3,146)_8 = (011,\ 001\ 100\ 110)_2$.

Если рассматривается двоичное число, то для перевода его в восьмеричное группируем по три двоичные цифры, начиная с позиции запятой влево для целой части и вправо соответственно для дробной части, находя эквиваленты этих троек цифр в восьмеричной системе. При дополнении некоторой группы до трех двоичных цифр, добавление нулей в начале числа для целой части и соответственно в конце числа для дробной части не меняет значение числа.

Примеры:

$(11,011101)_2 = (011,011\ 101)_2 = (3,35)_8$;

$(10,11011)_2 = (010,110\ 110)_2 = (2,66)_8$;

$(1001,01011)_2 = (001\ 001,010\ 110)_2 = (11,26)_8$.

Аналогично поступаем и в случае шестнадцатеричной системы, основание которой $16 = 2^4$. Любая шестнадцатеричная цифра представляется четырьмя двоичными цифрами:

0 = 0000;	8 = 1000;
1 = 0001;	9 = 1001;
2 = 0010;	A = 1010;
3 = 0011;	B = 1011;
4 = 0100;	C = 1100;
5 = 0101;	D = 1101;
6 = 0110;	E = 1110;
7 = 0111;	F = 1111.

Примеры:

$(6AF3,B2)_{16} = (0110\ 1010\ 1111\ 0011,\ 1011\ 0010)_2$;

$(6F1,3CA)_{16} = (0110\ 1111\ 0001,\ 0011\ 1100\ 1010)_2$;

$(11,01101)_2 = (0011,\ 0110\ 1000)_2 = (3,68)_{16}$;

$(10001,01011)_2 = (0001\ 0001,\ 0101\ 1000)_2 = (11,58)_{16}$.

Вопросы и упражнения

❶ Как осуществляется перевод из восьмеричной в двоичную систему счисления и обратно?

❷ Переведите в двоичную систему следующие восьмеричные числа:

a) 15,006; *c)* 43,15; *e)* 21,626;

b) 13,06; *d)* 10,01; *f)* 6,3415;

g)	771,25;	*i)*	42,322;	*k)*	32,271;
h)	12,121;	*j)*	44,523;	*l)*	73,536.

❸ Переведите в восьмеричную систему следующие двоичные числа:

a)	1,1;	*e)*	1101,1;	*i)*	10110,001011;
b)	101,10101;	*f)*	1,000001;	*j)*	11111,0010001;
c)	1111,000101;	*g)*	11110001,101;	*k)*	11001,00101;
d)	10101110,1001;	*h)*	0,00001;	*l)*	1011,0001011.

❹ Напишите программу для перевода чисел из двоичной в восьмеричную систему счисления и из восьмеричной в двоичную систему счисления.

❺ Как осуществляется шестнадцатерично-двоичное и двоично-шестнадцатеричное преобразование?

❻ Переведите в двоичную систему следующие шестнадцатеричные числа:

a)	FFF,AAA;	*e)*	3,1AB;	*i)*	C,DC1;
b)	F,1A;	*f)*	AABB,0000F;	*j)*	942,14A;
c)	1,009;	*g)*	81,91A;	*k)*	CAA,B;
d)	0,00F;	*h)*	10,01;	*l)*	DAD,ABA.

❼ Переведите в шестнадцатеричную систему следующие двоичные числа:

a)	1001011101,01101;	*e)*	1011101,011;	*i)*	1011,0101;
b)	111,01;	*f)*	11,001011101;	*j)*	11001,0110;
c)	1,1;	*g)*	11110,01111;	*k)*	0,00001;
d)	10101110111,00101;	*h)*	111011,0010000;	*l)*	101,01011.

❽ Напишите программу для перевода чисел из двоичной в шестнадцатеричную систему счисления и из шестнадцатеричной в двоичную систему счисления.

❾ Переведите в шестнадцатеричную систему следующие восьмеричные числа:

a)	13,146;	*d)*	104,527;	*g)*	53,627;	*j)*	5,4312;
b)	613,12;	*e)*	451,35;	*h)*	572,004;	*k)*	675,542;
c)	541,723;	*f)*	12,357;	*i)*	644,031;	*l)*	372,271.

❿ Переведите в восьмеричную систему следующие шестнадцатеричные числа:

a)	AA;	*d)*	FFFF;	*g)*	9,AF;	*j)*	AF,31B;
b)	A2B,1F;	*e)*	15F,6A1;	*h)*	3,418F;	*k)*	2C,ACB;
c)	F,3A;	*f)*	3,281;	*i)*	BBB;	*l)*	A1B,39E.

⓫ Напишите программу для восьмерично-шестнадцатеричного и шестнадцатерично-двоичного перевода чисел.

3.4. Арифметические операции в двоичной системе счисления

Арифметические операции над двоичными числами очень просты. Правила действий в двоичной системе представлены в *таблицах 3.2, 3.3* и *3.4*.

Таблица 3.2

Двоичное сложение

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

Таблица 3.3

Двоичное вычитание

0 – 0 = 0
1 – 0 = 1
1 – 1 = 0
10 – 1 = 1

Таблица 3.4

Двоичное умножение

0 × 0 = 0
0 × 1 = 0
1 × 0 = 0
1 × 1 = 1

Примеры:

1. Выполните сложение десятичных чисел 29 и 43 в двоичной системе счисления:

$(29)_{10} = (11101)_2$;
$(43)_{10} = (101011)_2$;

$$\begin{array}{r} 11101 \\ 101011 \\ \hline 1001000 \end{array} \; +$$

Проверка: $(1001000)_2 = (72)_{10}$, результат верен, поскольку $(29)_{10} = (43)_{10} = (72)_{10}$.

2. Выполните вычитание десятичного числа 37 из десятичного числа 46 в двоичной системе счисления:

$(37)_{10} = (100101)_2$;
$(46)_{10} = (101110)_2$;

$$\begin{array}{r} 101110 \\ 100101 \\ \hline 1001 \end{array} \; -$$

Проверка: $(1001)_2 = (9)_{10}$, результат верен, поскольку $(46)_{10} - (37)_{10} = (9)_{10}$.

3. Выполните умножение десятичных чисел 3,25 и 7,125 в двоичной системе счисления:

$(3{,}25)_{10} = (11{,}01)_2$;
$(7{,}125)_{10} = (111{,}001)_2$;

$$\begin{array}{r} 11{,}01 \\ 111{,}001 \\ \hline 1101 \\ 0000 \\ 0000 \\ 1101 \\ 1101 \\ 1101 \\ \hline 10111{,}00101 \end{array} \; \times$$

Проверка: $(10111{,}00101)_2 = (23{,}15625)_{10}$, результат верен, поскольку $(3{,}25)_{10} \times (7{,}125)_{10} = (23{,}15625)_{10}$.

4. Выполните деление десятичного числа 211 на десятичное число 3 в двоичной системе счисления:

$(211)_{10} = (11010011)_2$;
$(3)_{10} = (11)_2$;

```
11010011 | 11
11       |------
-------- | 1000110
 00100
   11
 -----
    11
    11
 -----
    01
```

Следовательно, 11010011 : 11 = 1000110, остаток 1.

Проверка: $(1000110)_2 = (70)_{10}$, результат верен, поскольку $(211)_{10} : (3)_{10} = (70)_{10} + (1)_{10}$.

Заметим, что как при умножении, так и при делении установка запятой, отделяющей целую часть от дробной, осуществляется так же, как и в десятичной системе счисления.

Вопросы и упражнения

❶ Как выполняются арифметические операции в двоичной системе счисления?

❷ Вычислите в двоичной системе:

a)	$34 + 251$;	*h)*	$28{,}5 + 0{,}75$;	*o)*	$32 : 16$;
b)	$68 - 7$;	*i)*	$63{,}125 - 4{,}125$;	*p)*	401×8;
c)	$1512 + 620$;	*j)*	$3{,}0625 \times 2{,}125$;	*q)*	$32 : 8$;
d)	14×8;	*k)*	$0{,}5 \times 0{,}5$;	*r)*	401×4;
e)	$63 : 3$;	*l)*	$1 : 0{,}5$;	*s)*	$32 : 4$;
f)	$1996 - 51$;	*m)*	$40 : 0{,}125$;	*t)*	401×2;
g)	$2015 + 1995$;	*n)*	$32 : 2$;	*u)*	$933 : 3$.

Числа в приведенных выражениях записаны в десятичной системе счисления.

❸ Напишите программу для сложения и вычитания двоичных чисел.

❹ Напишите программу для умножения и деления двоичных чисел.

3.5. Представление натуральных чисел в компьютере

Современные компьютеры используют двоичную систему счисления. Представление натуральных чисел $N = \{0, 1, 2, \ldots\}$ реализуется на фиксированном числе двоичных позиций, как правило, 8, 16, 32 или 64 (*рис. 3.1*).

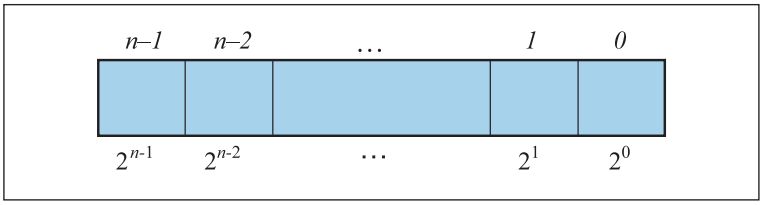


Рис. 3.1. Представление натуральных чисел на *n* двоичных позициях

В позициях 0, 1, …, $n - 1$ записаны двоичные цифры натурального числа, представленного в двоичной системе счисления. Выравнивание двоичных чисел выполняется вправо, возможные незначащие нули размещаются перед двоичным числом.

Для примера на *рис. 3.2* приведено представление натурального числа

$$1039 = (10000001111)_2$$

на 16 двоичных позициях.

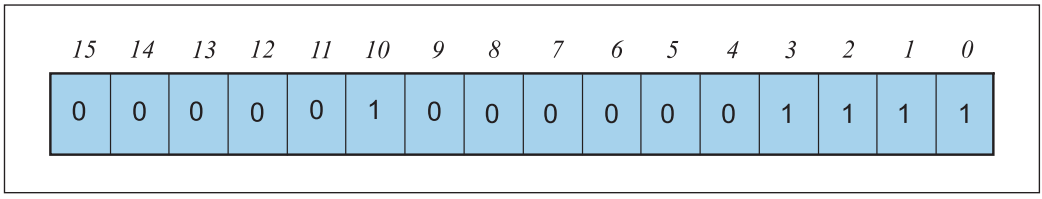


Рис. 3.2. Представление натурального числа 1039 на 16 двоичных позициях

Максимальное число, которое может быть представлено на *n* двоичных позициях (*рис. 3.3*):

$$1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + \ldots + 1 \cdot 2^{n-1} = 2^n - 1.$$

Следовательно, на *n* двоичных позициях могут быть представлены натуральные числа из интервала $[0; 2^n - 1]$.

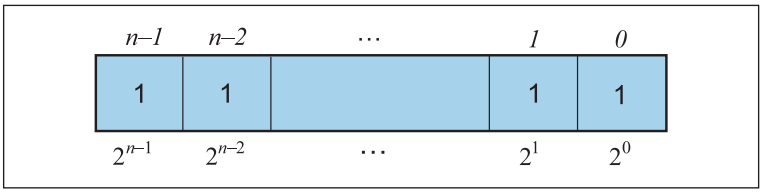


Рис. 3.3. Представление максимального натурального числа на *n* двоичных позициях

Вопросы и упражнения

❶ Как представляются натуральные числа в компьютере?
❷ Представьте натуральные числа 3, 112, 191, 204, 255 на 8 двоичных позициях.
❸ Представьте натуральные числа 3, 255, 1024, 2048, 4096, 65535 на 16 двоичных позициях.
❹ Вычислите максимальные натуральные числа, которые могут быть представлены на 4, 8, 12, 16, 24, 32 и 64 двоичных позициях.

3.6. Представление целых чисел

В компьютере отсутствует возможность прямого представления знаков + и – , принадлежащих положительным и отрицательным числам. По этой причине представление знака числа x осуществляется с помощью особой двоичной цифры, именуемой **цифрой знака (или знаковым битом),** расположенной в позиции $n - 1$ *(рис. 3.4):*

$$S = \begin{cases} 0, & x > 0; \\ 1, & x < 0. \end{cases}$$

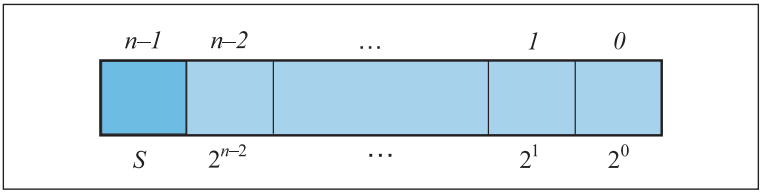


Рис. 3.4. Представление целых чисел на n двоичных позициях

Как правило, двоичные числа со знаком представляются в компьютерах не в исходном виде, а в специальных кодах, дающих определенные преимущества при выполнении арифметических операций. С этой точки зрения известны три способа представления, называемые **двоичными кодами для алгебраических чисел.**

Прямой код (код *величина* и *знак*). Представление любого числа в этом коде очень простое: в цифре знака записывается 0, если число положительное, и 1, если оно отрицательное; в значащей части записывается число (по модулю) в обычной двоичной системе.

Для примера на *рис. 3.5* приведены представления чисел +52 и –52 на 8 двоичных позициях.

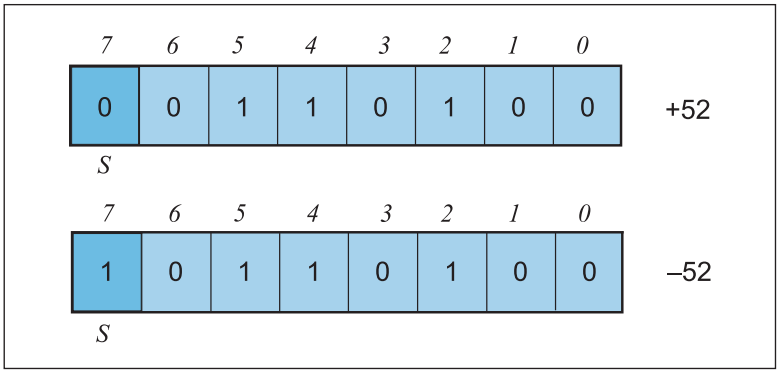


Рис. 3.5. Представление чисел +52 и –52 в прямом коде

В прямом коде на n двоичных позициях могут быть представлены целые положительные и отрицательные числа N, для которых:

$$-(2^{n-1} - 1) \le N \le (2^{n-1} - 1).$$

Заметим, что в прямом коде существуют два двоичных представления для числа 0: 00…0 и 10…0. Прямой код редко используется в компьютерах, поскольку требует сложных алгоритмов выполнения арифметических операций и проверки результатов.

Обратный (инверсный) код. Положительные числа записываются в инверсном коде точно так же, как и в прямом. Если число отрицательное, то сначала оно записывается как положительное, а затем инвертируется каждая двоичная цифра, то есть 1 становится 0 и 0 становится 1. Отсюда и название – обратный код.

На *рис. 3.6* приведены представления чисел +52 и –52 в обратном коде на 8 двоичных позициях.

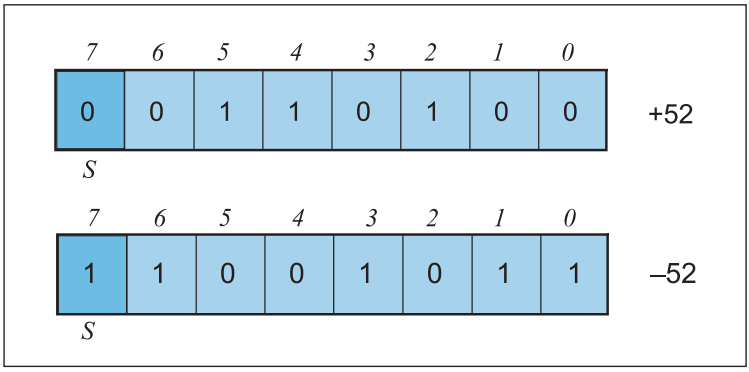


Рис. 3.6. Представление чисел +52 и –52 в обратном коде

Можно легко установить, что интервал целых чисел N , которые могут быть представлены в обратном коде, такой же, как и для прямого кода, а число 0 имеет два двоичных представления: 00…0 и 11…1.

Дополнительный код. В этом коде положительные числа имеют точно такое же представление, как в прямом и инверсном кодах. Если число отрицательное, оно сначала записывается в обратном коде, а затем к младшей значащей цифре (двоичная позиция 0) прибавляется 1.

Для примера на *рис. 3.7* приведены представления чисел +52 и –52 в дополнительном коде. Дополнительный код используется в большинстве компьютеров благодаря простоте выполнения в нем арифметических операций и легкости проверки результатов. В этом коде на n двоичных позициях могут быть представлены целые числа из интервала $[-2^{n-1}, 2^{n-1} - 1]$.

Вопросы и упражнения

❶ Как могут быть представлены в компьютере целые числа? Объясните, как записываются отрицательные числа в прямом, инверсном и дополнительном кодах.

❷ Представьте в прямом коде на 8 двоичных позициях:

a)	+12;	*d)*	–12;	*g)*	+21;
b)	–21;	*e)*	–64;	*h)*	–68;
c)	+68;	*f)*	+105;	*i)*	–112.

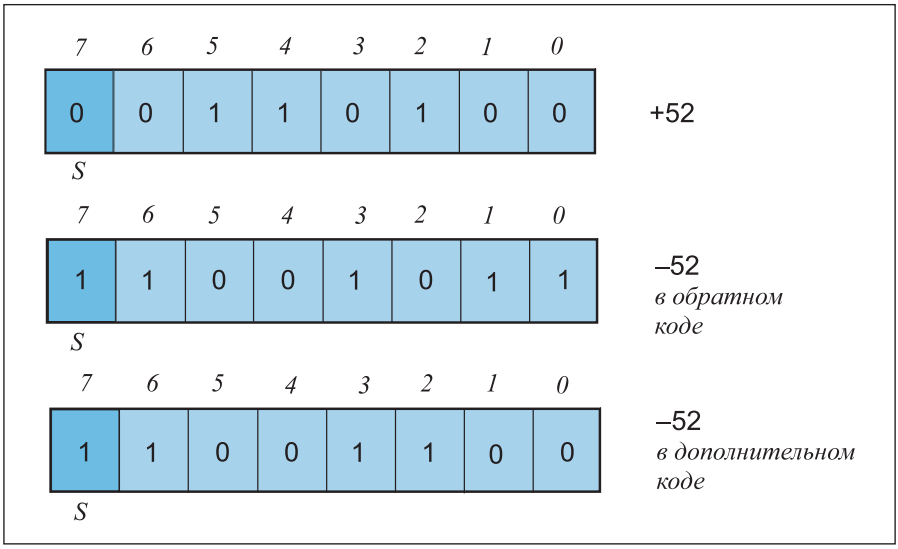


Рис. 3.7. Представление чисел +52 и –52 в дополнительном коде

❸ Представьте в обратном коде на 8 двоичных позициях:

a)	+10;	*d)*	–10;	*g)*	+65;
b)	–65;	*e)*	+101;	*h)*	–101;
c)	+112;	*f)*	–112;	*i)*	–105.

❹ Представьте в дополнительном коде на 8 двоичных позициях:

a)	+40;	*e)*	–40;	*i)*	+16;
b)	+27;	*f)*	–27;	*j)*	+101;
c)	+109;	*g)*	–109;	*k)*	+111;
d)	–16;	*h)*	–101;	*l)*	–111.

❺ Напишите программу, которая считывает с клавиатуры целые числа и выводит на экран их представления в прямом, обратном и дополнительном кодах для n = 8, 16 и 32.

❻ Как представляется число 0 в прямом, обратном и дополнительном кодах? Сколько представлений есть у числа 0 в рассматриваемых кодах?

❼ Определите максимальное число, которое может быть представлено на n двоичных позициях в прямом, обратном и дополнительном кодах.

3.7. Представление вещественных чисел

Вещественные числа представляются в компьютере в формате с фиксированной или плавающей запятой (в экспоненциальной форме).

Представление с фиксированной запятой. В этом представлении считается, что у всех чисел запятая расположена в одной и той же позиции, хотя это

и не соответствует внешней форме представления чисел. Процесс преобразования из внешней формы во внутреннюю и обратно реализуется путем выбора программистом соответствующих масштабных коэффициентов.

Обычно считается, что запятая размещается сразу после цифры знака – случай, когда числа содержат только дробную часть (*рис. 3.8*).

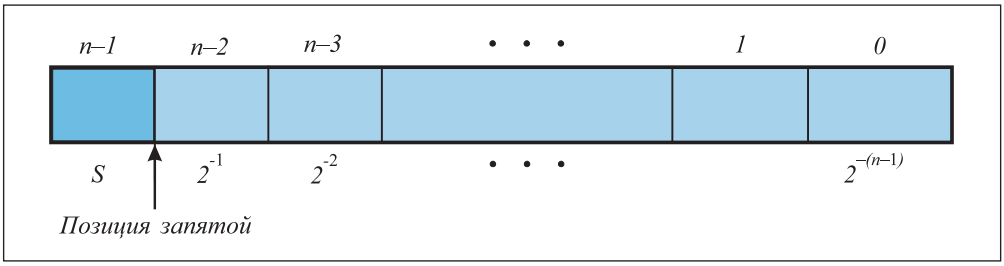


Рис. 3.8. Представление вещественных чисел в формате с фиксированной запятой

Собственно запятая не материализована физически в компьютере. Из *рис. 3.8* следует, что на n двоичных позициях можно представить вещественные числа, абсолютное значение которых

$$0{,}00\ldots0 \le |x| \le 0{,}11\ldots1$$

или в десятичной системе счисления,

$$0 \le |x| \le 1 - 2^{-(n-1)}.$$

Как и в случае целых чисел, вещественные числа меньшие единицы могут быть представлены, с некоторыми изменениями, в прямом, обратном или дополнительном кодах.

Для примера на *рис. 3.9* приведено представление чисел

$$+\,0{,}9375 = +\,15/16 = (0{,}1111)_2,$$
$$-\,0{,}9375 = -\,15/16 = (-0{,}1111)_2$$

в формате с фиксированной запятой на 8 двоичных позициях в прямом коде.

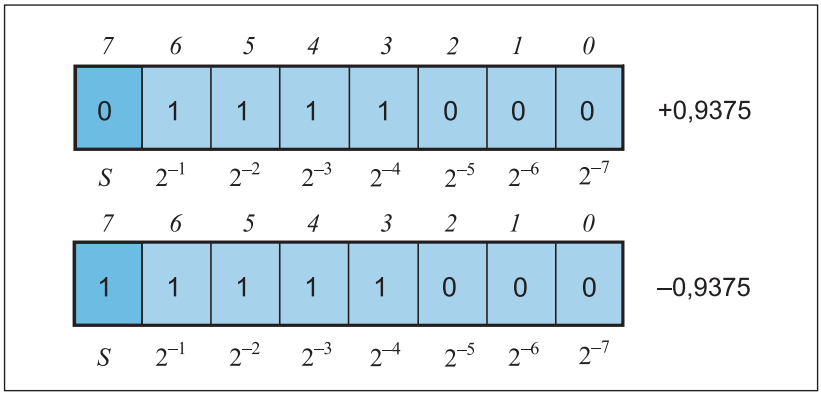


Рис. 3.9. Представление чисел +0,9375 и –0,9375 в формате с фиксированной запятой в прямом коде

Главное преимущество представления с фиксированной запятой состоит в том, что арифметические операции над вещественными числами могут быть выполнены арифметическим устройством, предназначенным для обработки целых чисел.

Представление с плавающей запятой. Операции над числами с фиксированной запятой удобны для однородных потоков данных, когда все вещественные числа по модулю меньше 1. Однако такое представление не эффективно в научных расчетах при одновременной работе с очень большими и очень малыми числами, порядок которых заранее непредсказуем. Для таких задач используется представление с плавающей запятой.

Числа, представляемые в формате с плавающей запятой, могут быть как целыми, так и дробными, а их значение задается соотношением:

$$x = M \times b^E,$$

где b – основание, M – **мантисса,** а E – **экспонента.** В современных компьютерах используется $b = 2$ или 16, а $|M| \le 1$.

Число, представленное в формате с плавающей запятой, является **нормализованным**, если первая после запятой цифра мантиссы отлична от нуля.

Примеры:

1. Число 23 выражается в формате с плавающей запятой так:

$$23 = (10111)_2 = 0{,}10111 \times 2^5,$$

где $M = 0{,}10111$, $b = 2$, $E = 5$.

2. Число 4,9375 записывается в формате с плавающей запятой следующим образом:

$$4{,}9375 = (100{,}1111)_2 = 0{,}1001111 \times 2^3,$$

$M = 0{,}1001111$, $b = 2$, $E = 3$.

3. Число $-0{,}375$ записывается в формате с плавающей запятой так:

$$-0{,}375 = (-0{,}011)_2 = -0{,}11 \times 2^{-1},$$

$M = -0{,}11$, $b = 2$, $E = -1$.

Отметим, что реальная позиция запятой внутри числа зависит от значения экспоненты, то есть запятая меняет свое положение (отсюда и название *плавающая запятая*).

Существует несколько вариантов представления мантиссы и экспоненты на n двоичных позициях. На *рис. 3.10* приведено представление с плавающей запятой **в формате экспонента–мантисса.**

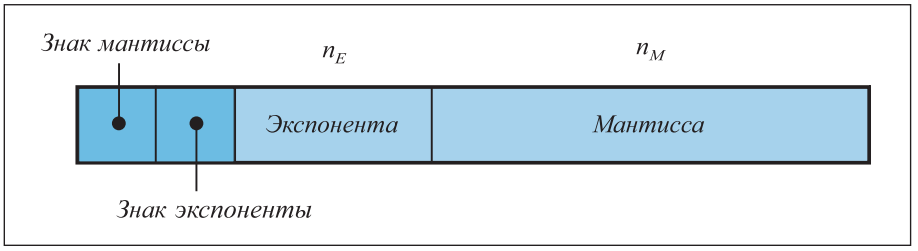


Рис. 3.10. Представление с плавающей запятой в формате *экспонента–мантисса*

Количество двоичных позиций n_E, выделенных экспоненте, определяет область значений чисел, которые могут быть представлены, тогда как количество битов для мантиссы n_M определяет точность представления числа. В современных компьютерах $n_E = 6...15$ и $n_M = 24...64$, что обеспечивает представление чисел в очень широком диапазоне.

На *рис. 3.11* приведено представление чисел в формате **характеристика–мантисса.**

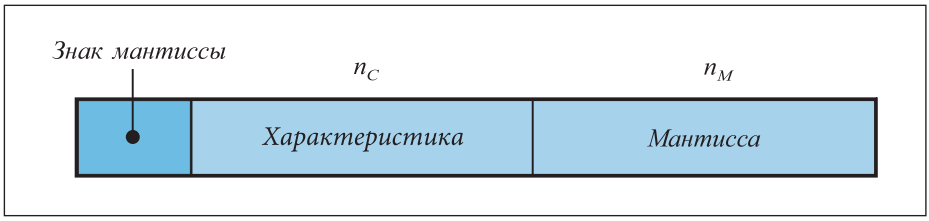


Рис. 3.11. Представление с плавающей запятой в формате *характеристика–мантисса*

Характеристика C – это форма представления экспоненты E. Она определяется соотношением

$$C = E + K.$$

В персональных компьютерах

$$K = 2^{n_c - 1} - 1,$$

где n_C – это количество двоичных позиций, выделенных для характеристики. В этом представлении характеристика в отличие от экспоненты не может принимать отрицательных значений, что упрощает арифметические устройства.

В частности, для $n = 8$ характеристика вычисляется так:

$$C = E + 127,$$

указывая практически знак экспоненты:

– если $C \geq 127$, тогда $E \geq 0$;

– если $C < 127$, тогда $E < 0$.

В *таблице 3.5* представлены форматы *характеристика–мантисса*, используемые в большинстве персональных компьютеров.

Таблица 3.5.

Форматы *характеристика–мантисса*, используемые в персональных компьютерах

Название формата	n	n_C	n_M	*Точность мантиссы, десятичные цифры*	*Область возможных значений чисел*
Обычная точность	32	8	23	6 или 7	$10^{-37}...10^{38}$
Двойная точность	64	11	52	15 или 16	$10^{-307}...10^{308}$
Расширенная точность	80	15	64	19	$10^{-4932}...10^{4932}$

Поскольку в соответствии с условием нормализации первая цифра мантиссы всегда равна 1, эта цифра может быть представлена или не представлена в памяти компьютера. В случае, когда она не представлена, указанная цифра называется **скрытым битом.** Отметим, что техника скрытого бита относится только к представлению чисел в памяти, но не к операциям, выполняемым арифметическим устройством.

Вопросы и упражнения

❶ Какие различия имеют представления с фиксированной и плавающей запятой?

❷ Представьте в формате с фиксированной запятой на 8 двоичных позициях в прямом коде:

a)	+0,875;	*e)*	–0,875;	*i)*	+0,125;
b)	–0,125;	*f)*	+0,3;	*j)*	–0,3;
c)	+0,4;	*g)*	–0,4;	*k)*	+0,15625;
d)	–0,15625;	*h)*	+0,21875;	*l)*	–0,21875.

❸ Укажите преимущества и недостатки у представления с фиксированной запятой.

❹ Как представляются вещественные числа в формате с плавающей запятой? Какие числа, представленные в формате с плавающей запятой, удовлетворяют условию нормализации?

❺ Как выполняется нормализация чисел, представленных в формате с плавающей запятой?

❻ Представьте в формате с плавающей запятой следующие числа:

a)	+1,5;	*e)*	–1,5;	*i)*	–0,0625;
b)	+0,0625;	*f)*	+3,25;	*j)*	+2,7;
c)	–3,25;	*g)*	–32,87;	*k)*	–2,7;
d)	–6,25;	*h)*	+6,25;	*l)*	–0,147.

❼ От чего зависят точность и область значений чисел, представленных в формате с плавающей запятой?

❽ Каковы различия между форматами *экспонента–мантисса* и *характеристика–мантисса*? Как рассчитываются характеристики чисел, представленных в формате с плавающей запятой?

❾ Вычислите характеристики чисел из упражнения 6. Предполагается, что $n_c = 8$.

❿ Объясните термин *скрытый бит*. Каковы преимущества такого представления?

⓫ Как представляются натуральные, целые и вещественные числа в компьютере, за которым вы работаете? Определите количество двоичных позиций, используемых каждым представлением.

Тест для самопроверки № 3

1. Переведите в десятичную систему счисления следующие числа:

a) $(821)_9$; *b)* $(1011{,}121)_3$; *c)* $(341{,}52)_7$.

2. В каких из систем счисления *a)*, *b)*, *c)*, *d)* запись числа 284,6 является неправильной? Объясните ответ.

a) десятичная; *c)* двоичная;

b) восьмеричная; *d)* шестнадцатеричная.

3. Напишите программу на ПАСКАЛЕ, которая выполняет преобразование в десятичную систему чисел, записанных в системе счисления по основанию *b*, $b < 10$. Основание *b* вводится с клавиатуры.

4. Переведите в двоичную систему счисления следующие числа:

a) $(13{,}889)_{10}$; *b)* $(73{,}542)_8$; *c)* $(A8{,}74F)_{16}$.

5. Переведите в восьмеричную систему счисления следующие числа:

a) $(9758{,}93)_{10}$; *b)* $(1011011011{,}01011101)_2$; *c)* $(DA86{,}B1)_{16}$.

6. Переведите в шестнадцатеричную систему счисления следующие числа:

a) $(9471{,}8362)_{10}$; *b)* $(10110111011011{,}0101001101)_2$; *c)* $(425{,}376)_8$.

7. Запишите числа $(1000001111)_2$, $(132)_8$, $(BB)_{16}$, $(222221)_4$ в порядке убывания.

8. Среди представленных ниже чисел встречаются равные друг другу. Запишите соответствующие равенства, например, $(1010)_2 = (12)_8$.

a) $(1010)_8$; *c)* $(723)_8$; *e)* $(21)_{10}$;

b) $(1D3)_{16}$; *d)* $(25)_8$; *f)* $(A)_{16}$.

9. Напишите программу на ПАСКАЛЕ для преобразования чисел из восьмеричной системы счисления в двоичную.

10. Вычислите в двоичной системе счисления:

a) 110001101 + 10111010; *c)* 111011×101;

b) 1101001010 – 101101001; *d)* 101011100 : 110.

11. Представьте (запишите) на 8 двоичных позициях натуральное число 125.

12. Представьте (запишите) на 8 двоичных позициях в дополнительном коде целое число –91.

13. Двоичные слова, приведенные ниже, представляют собой целые числа, записанные в обратном коде на 8 двоичных позициях. Запишите эти числа в десятичной системе счисления.

a) 11111110; *b)* 00111111; *c)* 11011000.

14. Представьте число –0,75 на 8 двоичных позициях в прямом коде с фиксированной запятой.

15. Запишите в формате с плавающей запятой следующие числа:

a) +21,125; *b)* –73,25; *c)* –0,09375.

16. "Черные ящики" выполняют преобразование (перевод) десятичных чисел, подаваемых на входе, в числа, записанные в системе счисления с основанием *b*. Определите это основание и представление соответствующего числа на выходе второго "черного ящика".

17. Двоичные слова, приведенные ниже, представляют собой целые числа, записанные в дополнительном коде на 8 двоичных позициях. Запишите эти числа в десятичной системе счисления.

a) 11100110; *b)* 00101101; *c)* 11111000.

18. Укажите диапазон целых чисел N, которые можно записать в дополнительном коде на n двоичных позициях:

a) $n = 4$; *b)* $n = 8$; *c)* $n = 16$.

19. Двоичные слова, приведенные ниже, представляют собой числа, меньше единицы, записанные в прямом коде на 4 двоичных позициях с фиксированной запятой. Запишите эти числа в десятичной системе счисления.

a) 1011; *b)* 0001; *c)* 1100.

20. Запишите в десятичной системе счисления наименьшее число (A) и наибольшее число (B), которые можно представить в формате с фиксированной запятой в прямом коде на 8 двоичных позициях.

21. Двоичные слова, приведенные ниже, представляют собой действительные числа в прямом коде с плавающей запятой с основанием $b = 2$ в формате экспонента–мантисса. Экспонента занимает $n_E = 2$, а мантисса $n_M = 4$ двоичных позиций. Запишите эти числа в десятичной системе счисления.

a) 11111110; *b)* 00111111; *c)* 11011000.

22. Представьте число –2,75 в прямом коде с плавающей запятой по основанию $b = 2$, в формате экспонента–мантисса. Экспонента занимает $n_E = 2$, а мантисса $n_M = 4$ двоичных позиций.

23. Запишите в десятичной системе счисления наименьшее число (A) и наибольшее число (B), которые могут быть представлены в прямом коде с плавающей запятой по основанию $b = 2$ в формате экспонента–мантисса. Экспонента занимает $n_E = 2$, а мантисса – $n_M = 4$ двоичных позиций.

Глава 4

БУЛЕВА АЛГЕБРА

4.1. Логические переменные и выражения

Булева алгебра или **алгебра логики** является составной частью математики. В ней законы мышления – объект изучения классической логики – изучаются с помощью символических методов. Такое название она получила в честь английского математика Джорджа Буля, который в своей работе *The Laws of Thought* (Законы мышления), опубликованной в 1853 году, заложил основы алгебры логики. Долгие годы булева алгебра считалась просто математическим курьезом, неприменимым к практическим областям науки. Она снова стала актуальной с появлением автоматических телефонных сетей и цифровых компьютеров.

С формальной точки зрения булева алгебра может быть определена множеством элементов {0, 1}, множеством элементарных операций {¯, &, ∨} и определенным набором постулатов. Следовательно, в алгебре логики любая переменная может иметь только одно из двух возможных значений, обозначаемых условно 0 и 1, причем другие значения являются недопустимыми.

Переменные алгебры логики обозначаются через x, y, z, x_1, x_2, ... с индексами или без них, а элементы 0 и 1 называются **логическими константами.**

Элементарные операции алгебры логики имеют следующие названия:

¯ – отрицание (инверсия, логическая операция *НЕ*);

& – конъюнкция (логическое произведение, логическая операция *И*);

∨ – дизъюнкция (логическая сумма, логическая операция *ИЛИ*).

Элементарные операции определяются с помощью специальных таблиц, называемых **таблицами истинности.**

> *Таблица истинности – это таблица, включающая все возможные комбинации значений переменных, для которых определен оператор, и результаты соответствующей операции.*

На *рис. 4.1* представлены таблицы истинности отрицания, конъюнкции и дизъюнкции.

x	$\bar{x}$
0	1
1	0

x	y	$x \,\&\, y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

Рис. 4.1. Таблицы истинности элементарных логических операторов

Отметим, что в булевой алгебре отрицание обозначается горизонтальной линией над соответствующей переменной.

Поскольку переменная x может принимать значения 0 или 1, таблица истинности отрицания содержит только две строки. В случае конъюнкции и дизъюнкции таблицы истинности содержат четыре строки – по одной строке для каждой возможной комбинации 00, 01, 10 и 11 значений переменных x и y.

Логические переменные и константы, объединенные с помощью операторов $\bar{}$, & и $\vee$, образуют **логические выражения**, например:

1) $x \,\&\, y \vee z$;

2) $x \vee y \vee z$;

3) $\bar{x} \,\&\, y \vee z$;

4) $1 \,\&\, x \vee y$;

5) $0 \vee x \vee y$.

6) $1 \vee 0$.

Значения логических выражений могут быть вычислены с помощью таблиц истинности элементарных операторов. Для вычисления выражений устанавливается следующий **приоритет логических операторов:**

1) отрицание;
2) конъюнкция;
3) дизъюнкция.

Например, в выражении

$$x \vee y \,\&\, z$$

сначала выполняется операция &, а после нее – операция $\vee$. В выражении

$$x \,\&\, y \vee \bar{z}$$

выполняется отрицание, затем – конъюнкция и в конце – дизъюнкция.

Порядок выполнения логических операторов может быть изменен с помощью скобок „(” и „)”. Ясно, что в первую очередь выполняются операции, заключенные в скобки.

Например, в случае логического выражения

$$x \vee y \,\&\, \bar{x} \vee z$$

выполняется отрицание, конъюнкция и затем соответствующие операции дизъюнкции. В случае выражения

$$(x \vee y) \,\&\, (\bar{x} \vee z)$$

после отрицания выполняются дизъюнкции и в конце – конъюнкция.

Для систематизации вычисление логических выражений выполняется с помощью специальных таблиц, называемых **таблицами истинности логических выражений**.

> *Таблица истинности логического выражения включает все возможные комбинации значений переменных рассматриваемого выражения и результаты логических операций в порядке их вычисления.*

Для примера на *рис. 4.2* представлена таблица истинности выражения

$$\bar{x} \,\&\, y \vee z,$$

а на *рис. 4.3* – таблица истинности выражения

$$\overline{x \,\&\, y \vee z}.$$

x	y	z	$\bar{x}$	$\bar{x}\,\&\,y$	$\bar{x}\,\&\,y \vee z$
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	0	0	0
1	1	1	0	0	1

Рис. 4.2. Таблица истинности логического выражения $\bar{x}\ \&\ y \vee z$

x	y	z	$x\,\&\,y$	$x\,\&\,y \vee z$	$\overline{x\,\&\,y \vee z}$
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	1	0
1	1	1	1	1	0

Рис. 4.3. Таблица истинности логического выражения $\overline{x\ \&\ y \vee z}$

Отметим, что с целью упрощения записей в логических выражениях разрешается опускать символ оператора &. Например, логические выражения

$$\bar{x}\ \&\ y \vee z,$$
$$(x \vee y)\ \&\ (\bar{x} \vee z)$$

могут быть записаны в виде:

$$\bar{x}y \vee z,$$
$$(x \vee y)(\bar{x} \vee z).$$

Вопросы и упражнения

❶ Какие значения могут принимать логические переменные?

❷ Какие элементарные операторы используются в алгебре логики и как они определяются?

❸ Запомните таблицы истинности отрицания, конъюнкции и дизъюнкции.

❹ Как составляются логические выражения? Укажите приоритет логических операций.

❺ Объясните роль скобок при вычислении логических выражений.

❻ Составьте таблицы истинности следующих логических выражений:

a) $\bar{x}y$; *c)* $\bar{x} \vee y$;

b) $\bar{x}\bar{y}$; *d)* $\overline{x \vee y}$;

e) $\bar{\bar{x}}$;

f) $\bar{x}x$;

g) $\overline{xy}$;

h) $\bar{x} \vee x$;

i) xyz;

j) $x \vee y \vee z$;

k) $xy \vee z$;

l) $x\,(y \vee z)$;

m) $\bar{x} \vee y \vee z$;

n) $\bar{x}\,(y \vee z)$.

❼ Напишите программу, которая считывает с клавиатуры значения логических переменных x, y, z и выводит на экран значения следующих выражений:

a) $\bar{x}$;

b) xy

c) $x \vee y$;

d) $\overline{(x \vee y)}\,z$;

e) $\overline{xy} \vee z$;

f) $x \vee y \vee z$;

g) $xy \vee z$;

h) $\overline{x \vee y \vee z}$;

i) $x \vee xy$;

j) $\bar{x} \vee y \vee z$.

❽ Напишите программу, которая выводит на экран таблицы истинности конъюнкции и дизъюнкции.

❾ Составьте циклический алгоритм, который формирует все возможные комбинации значений переменных x, y и z. Выведите соответствующие комбинации на экран.

❿ Для каждого из приводимых ниже логических выражений напишите программу, которая выводит на экран соответствующую таблицу истинности:

a) $x \vee \bar{x}$;

b) $x\bar{x}$;

c) $\overline{x \vee y}$;

d) $\overline{xy}$;

e) $\bar{x} \vee y$;

f) $\bar{x}y$;

g) $x \vee y \vee z$;

h) xyz;

i) $(x \vee y)(\bar{x} \vee y)$;

j) $(x \vee y)(x \vee \bar{y})$.

4.2. Логические функции

Понятие **логической** или **булевой функции** определяется таким же образом, как и в случае классической алгебры.

Обозначим через x_1, x_2, …, x_n произвольную группу булевых переменных, где n = 1, 2, 3, … .

Поскольку каждая булева переменная может иметь только значения 0 или 1, количество всевозможных комбинаций значений переменных x_1, x_2, …, x_n составляет 2^n.

Очевидно, для n = 1 имеем две комбинации (0 и 1); для n = 2 существуют 2^2 = 4 комбинации (00, 01, 10 и 11); для n = 3 существуют 2^3 = 8 комбинаций (000, 001, 010, 011, 100, 101, 110 и 111) и т.д.

Логическая функция n переменных $y = f(x_1, x_2, ..., x_n)$ есть отображение, которое ставит в соответствие каждой комбинации значений переменных $x_1, x_2, ..., x_n$ значение 0 или 1 переменной y.

Переменные $x_1, x_2, ..., x_n$ называются **независимыми переменными** или **аргументами**, а переменная y – **зависимая переменная** или **функция** аргументов $x_1, x_2, ..., x_n$.

Следовательно, **область определения** функции $y = f(x_1, x_2, ..., x_n)$ – это множество всевозможных комбинаций

$$\begin{array}{c} 0\ 0\ \ldots\ 0 \\ 0\ 0\ \ldots\ 1 \\ \ldots \\ 1\ 1\ \ldots\ 1 \end{array}$$

значений аргументов $x_1, x_2, ..., x_n$ (всего 2^n комбинаций), а **областью значений** логической функции является множество {0, 1}.

Как и в случае классической алгебры, логические функции могут быть определены с помощью **таблиц**, **формул** и **графических методов.**

Таблица истинности логической функции $y = f(x_1, x_2, ..., x_n)$ – это таблица, включающая все возможные комбинации значений аргументов $x_1, x_2, ..., x_n$ и соответствующие им значения зависимой переменной y.

Например, на *рис. 4.4* представлена таблица истинности некоторой логической функции трех переменных.

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Рис. 4.4. Таблица истинности некоторой логической функции трех переменных

Таблица состоит из $2^3 = 8$ строк и 2 столбцов: первого – для всевозможных комбинаций значений аргументов x_1, x_2, x_3 и второго – для соответствующих значений зависимой переменной y. В соответствии с рассматриваемой таблицей комбинации $x_1 = 0, x_2 = 0, x_3 = 0$ соответствует значение $y = f(0,0,0) = 1$; комбинации $x_1 = 0, x_2 = 0, x_3 = 1$ – значение $y = f(0,0,1) = 0$ и т.д.

Определение логических функций с помощью формул осуществляется присваиванием зависимой переменной y значений логического выражения, содержащего аргументы $x_1, x_2, ..., x_n$.

Например,

1) $y = x$;
2) $y = x_1 x_2$;
3) $y = \bar{x}_1 x_2 \vee x_3$;
4) $y = \overline{x_1 x_2 \vee x_3}$.

Очевидно, что, зная формулу логической функции, можно составить ее таблицу истинности. Например, таблица истинности функции

$$y = \overline{x_1 x_2 \vee x_3}$$

будет такой же, как и приведенная на *рис. 4.4*. Чтобы удостовериться в этом, достаточно составить таблицу истинности логического выражения

$$\overline{x_1 x_2 \vee x_3}$$

представленную в других обозначениях на *рис. 4.3* из *параграфа 4.1*.

Существуют также **графические методы определения логических функций.** Эти методы основываются на диаграммах, применяемых в теории множеств, и изучаются в углубленных курсах информатики.

Вопросы и упражнения

❶ Сформулируйте определение понятия *логическая функция*.

❷ Укажите область определения и область значений логической функции.

❸ Какими способами может быть определена логическая функция n переменных?

❹ Как составляется таблица истинности логической функции n переменных? Сколько строк содержит такая таблица?

❺ Как можно составить таблицу истинности логической функции, если известна ее формула?

❻ Логическая функция трех переменных $y = f(x_1, x_2, x_3)$ определена с помощью таблицы истинности, приведенной на *рис. 4.4*. Укажите комбинации значений аргументов x_1, x_2, x_3, для которых рассматриваемая функция имеет значение $y = 1$. Укажите соответствующие комбинации для значения функции $y = 0$.

❼ Составьте таблицы истинности следующих функций:

a) $y = x$;
b) $y = \bar{x}$;
c) $y = x_1 x_2$;
d) $y = x_1 \vee x_2$;
e) $y = \overline{x_1 x_2}$;
f) $y = \overline{x_1 \vee x_2}$;
g) $y = \overline{x_1 x_2 x_3}$;
h) $y = x_1(x_2 \vee \bar{x}_3)$;
i) $y = \bar{x}_1 \vee x_2 x_3$;
j) $y = x_1 \vee \overline{x_2 x_3}$.

❽ Напишите программы, которые считывают с клавиатуры значения логических переменных x_1, x_2, x_3, x_4 и выводят на экран значения следующих функций:

a) $y = x_1 x_2 \vee x_3 x_4$;
b) $y = (x_1 \vee x_2)(x_3 \vee x_4)$;
c) $y = \overline{x_1 x_2 x_3 x_4}$;
d) $y = \overline{x_1 \vee x_2 \vee x_3 \vee x_4}$;
e) $y = \overline{x_1 x_2} \vee \overline{x_3 x_4}$;
f) $y = (\overline{x_1 \vee x_2})(\overline{x_3 \vee x_4})$;
g) $y = x_1 \bar{x}_2 \vee x_3 \bar{x}_4$;
h) $y = \bar{x}_1 x_2 \vee \bar{x}_3 x_4$;
i) $y = \bar{x}_1 \bar{x}_2 \vee \bar{x}_3 \bar{x}_4$;
j) $y = x_1 x_2 x_3 x_4 \vee \bar{x}_2 x_3 x_4$;
k) $y = x_1 \vee x_2 x_3 \vee x_4$;
l) $y = x_1 \vee x_2 \vee x_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4$.

❾ Напишите программы, которые составляют таблицы истинности следующих функций:

a) $y = x$;

b) $y = \bar{x}$;

c) $y = x_1 x_2$;

d) $y = x_1 \vee x_2$;

e) $y = \overline{x_1 x_2}$;

f) $y = \bar{x}_1 \vee \bar{x}_2$;

g) $y = \overline{x_1 \vee x_2}$;

h) $y = \bar{x}_1 \bar{x}_2$;

i) $y = \overline{x_1 x_2 x_3}$;

j) $y = \bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3$;

k) $y = \overline{x_1 \vee x_2 \vee x_3}$;

l) $y = \bar{x}_1 \bar{x}_2 \bar{x}_3$;

m) $y = x_1 \vee x_2 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4$;

n) $y = x_1(x_2 \vee \bar{x}_3 \vee \bar{x}_4)$.

4.3. Часто используемые логические функции

Рассмотрим n независимых переменных x_1, x_2, ..., x_n. Возникает вопрос, сколько логических функций n переменных существует в булевой алгебре? **Количество возможных логических функций** можно получить путем следующих рассуждений.

Поскольку любая логическая функция может быть определена с помощью таблицы истинности, то число возможных функций n переменных совпадает с количеством различных таблиц истинности.

Для того чтобы определить логическую функцию, в столбце y таблицы истинности указываются значения функции – 0 или 1 для каждой из всевозможных 2^n комбинаций значений аргументов. Поскольку в таблице истинности 2^n строк, существует

$$m = 2^{2^n}$$

логических функций n переменных. Соответствующие функции обозначаются y_j, $j = 0, 1, \ldots, m-1$.

Например, в случае, когда $n = 1$, существует $m = 2^{2^1} = 2^2 = 4$ логических функций, представленных на *рис. 4.5*.

x	y_0	y_1	y_2	y_3
0	0	1	0	1
1	0	0	1	1

Рис. 4.5. Логические функции одной переменной

Очевидно,

$$y_0 = f(x) = 0;$$
$$y_1 = f(x) = \bar{x};$$
$$y_2 = f(x) = x;$$
$$y_3 = f(x) = 1.$$

Функции y_0 и y_3 называются **функция-константа 0** и **функция-константа 1** соответственно. Функция y_1 – это **логическая функция *НЕ*** или **отрицание**, а функция y_2 называется **функцией повторения**.

Для $n = 2$ существует

$$m = 2^{2^2} = 2^4 = 16$$

логических функций, представленных на *рис. 4.6*.

x_1	x_2	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Рис. 4.6. Логические функции двух переменных

Функции y_0 и y_{15} нам уже знакомы, константа 0 и константа 1 соответственно:

$$y_0 = f(x_1, x_2) = 0;$$
$$y_{15} = f(x_1, x_2) = 1.$$

Функция y_8 может быть записана в виде

$$y_8 = f(x_1, x_2) = x_1 x_2.$$

Очевидно, что y_8 имеет название **логическая функция *И*** или **конъюнкция.**
Функция y_{14} может быть записана в виде

$$y_{14} = f(x_1, x_2) = x_1 \vee x_2.$$

Следовательно, функция y_{14} называется **логическая функция *ИЛИ*** или **дизъюнкция.**

Логические функции НЕ, И, ИЛИ, введенные с помощью элементарных операторов $\bar{}$, &, $\vee$ соответственно называются элементарными логическими функциями.

Из *рис. 4.6* следует, что

$$y_1 = \overline{x_1 \vee x_2}.$$

Рассматриваемая функция называется **логической функцией *ИЛИ-НЕ.***
Аналогичным образом, функция y_7 может быть представлена в виде

$$y_7 = f(x_1, x_2) = \overline{x_1 x_2}.$$

Рассматриваемая функция называется **логической функцией *И-НЕ.***
Функция

$$y_9 = f(x_1, x_2) = \bar{x}_1 \bar{x}_2 \vee x_1 x_2$$

принимает значение 1 только тогда, когда $x_1 = x_2 = 0$ или $x_1 = x_2 = 1$. Эта функция называется **логической функцией *СОВПАДЕНИЕ*** или ***ЭКВИВАЛЕНТНОСТЬ*** и обозначается с помощью символа „≡".

Анализируя таблицу на *рис. 4.6*, также заметим, что

$y_3 = f(x_1, x_2) = \bar{x}_1$ (отрицание переменной x_1);
$y_{12} = f(x_1, x_2) = x_1$ (повторение переменной x_1);
$y_5 = f(x_1, x_2) = \bar{x}_2$ (отрицание переменной x_2);
$y_{10} = f(x_1, x_2) = x_2$ (повторение переменной x_2).

Для лучшего запоминания на *рис. 4.7* представлены таблицы истинности логических функций *НЕ, И, ИЛИ, И-НЕ, ИЛИ-НЕ* и *СОВПАДЕНИЕ*.

НЕ

x	$\bar{x}$
0	1
1	0

И

x_1	x_2	x_1x_2
0	0	0
0	1	0
1	0	0
1	1	1

ИЛИ

x_1	x_2	$x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

И-НЕ

x_1	x_2	$\overline{x_1x_2}$
0	0	1
0	1	1
1	0	1
1	1	0

ИЛИ-НЕ

x_1	x_2	$\overline{x_1 \vee x_2}$
0	0	1
0	1	0
1	0	0
1	1	0

СОВПАДЕНИЕ

x_1	x_2	$x_1 \equiv x_2$
0	0	1
0	1	0
1	0	0
1	1	1

Рис. 3.7. Часто используемые логические функции

Аналогично можно рассматривать логические функции трех переменных, число которых составляет $m = 2^{2^3} = 2^8 = 256$; логические функции четырех переменных, число которых $m = 2^{2^4} = 2^{16} = 65536$ и т.д. Заметим, что, будучи конечным, число булевых функций все равно огромно. Однако доказано, что **любая логическая функция *n* переменных, $n \geq 2$, может быть выражена с помощью формулы, содержащей только элементарные операторы** $\bar{}$, &, $\vee$. Указанное свойство упрощает техническую реализацию устройств, предназначенных для вычисления логических функций произвольного числа аргументов.

Вопросы и упражнения

❶ Определите число логических функций пяти и шести переменных.

❷ Назовите элементарные логические функции и составьте соответствующие им таблицы истинности.

❸ Запомните таблицы истинности часто используемых логических функций *НЕ*, *И*, *ИЛИ*, *И-НЕ*, *ИЛИ-НЕ* и *СОВПАДЕНИЕ*.

❹ Напишите программу, которая выводит на экран таблицу истинности одной из логических функций y_0, y_1, y_2, y_3 двух переменных.

❺ Напишите программу, которая выводит на экран таблицу истинности одной из логических функций y_j от n переменных.

Тест для самопроверки № 4

1. Составьте таблицу истинности логического выражения $x \vee y\bar{z}$.

2. Какие из приведенных ниже логических выражений равны друг другу? Напоминаем, что два логических выражения равны, если их значения совпадают для всевозможных комбинаций значений соответствующих переменных.

a) $(\bar{x} \vee \bar{y})(x \vee y)$; *c)* $(x \vee y) \vee \overline{xy}$;

b) $xy \vee \overline{xy}$; *d)* $x\bar{y} \vee \bar{x}y$.

3. Запишите все комбинации значений переменных *x*, *y* и *z*, при которых значение выражения $xy \vee \bar{z}$ равно 1.

4. Напишите программу на ПАСКАЛЕ, которая вводит с клавиатуры значения логических переменных *x, y, z* и выводит на экран значение выражения $x\bar{y} \vee z$.

5. Напишите на ПАСКАЛЕ программу, которая выводит на экран таблицу истинности выражения $\bar{x}\bar{y} \vee z$.

6. Укажите область определения и множество значений логической функции $y = x_1(x_2 \vee \bar{x}_3)$.

7. Составьте таблицу истинности логической функции $y = x_1(\bar{x}_2 \vee x_3)$.

8. Напишите программу на ПАСКАЛЕ, которая вводит с клавиатуры значения логических переменных x_1, x_2, x_3 и выводит на экран значение функции $y = x_1(\bar{x}_2 \vee \bar{x}_3)$.

9. Напишите программу на ПАСКАЛЕ, которая выводит на экран таблицу истинности логической функции $y = \bar{x}_1(x_2 \vee x_3)$.

10. Определите число логических функций от 5 переменных.

11. Составьте таблицы истинности логических функций *И-НЕ* и *ИЛИ-НЕ*.

12. Какие из приведенных ниже логических функций равны друг другу? Напоминаем, что две логические функции равны, если их значения совпадают на всевозможных комбинациях значений независимых переменных.

a) $y = \overline{x_1 \vee x_2}$; *c)* $y = \bar{x}_1\bar{x}_2$;

b) $y = \overline{x_1 x_2}$; *d)* $y = \bar{x}_1 \vee \bar{x}_2$.

13. Логическая функция от трех переменных $y = f(x_1, x_2, x_3)$ определена при помощи следующей таблицы истинности:

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Выберите из приведенных ниже выражений то, которое определяет рассматриваемую функцию.

a) $y = x_1 \vee x_2 \vee x_3$; *c)* $y = \overline{x_1 x_2 x_3}$;

b) $y = \overline{x_1 \vee x_2 \vee x_3}$; *d)* $y = \bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3$.

Глава 5

ЛОГИЧЕСКИЕ СХЕМЫ

5.1. Логические элементы

Логическая схема – это устройство, предназначенное для вычисления логических функций. Для реализации логических схем необходимо, чтобы двоичные значения **0, 1** аргументов и соответствующих логических функций были представлены значениями определенных физических величин, например, давления, температуры, напряжения или силы электрического тока, освещенности и т.п. В зависимости от используемых величин различаем следующие логические устройства: механические, пневматические, гидравлические, электромеханические, электронные, оптические и т.п. В гидравлических и пневматических устройствах логические значения **0** или **1** могут представляться малыми и соответственно большими значениями давления жидкости или газа, в электромеханических и электронных устройствах – присутствием или отсутствием электрического тока, с помощью уровней напряжения и т.д.

Для более ясного понимания принципов работы логических устройств рассмотрим сначала контактные схемы. Базовыми элементами этих схем являются **элементы коммутации** – нормально разомкнутые и нормально замкнутые электрические контакты.

В случае **нормально разомкнутых контактов** электрическая цепь разомкнута, если контакты выключены, и замкнута, если они включены. В случае **нормально замкнутых контактов** электрическая цепь замкнута, если контакты выключены, и разомкнута в противном случае (*рис. 5.1*).

	Выключены	*Включены*
Нормально разомкнутые контакты		
Нормально замкнутые контакты		

Рис. 5.1. Нормально разомкнутые и нормально замкнутые контакты

Например, контакты дверной кнопки электрического звонка являются нормально разомкнутыми, а контакты кнопки *Пауза* обычного магнитофона – нормально замкнутыми.

В контактных схемах логические значения аргументов представлены состояниями соответствующих электрических контактов. Логическому значению **1** соответствует состояние *включен*, а логическому значению **0** соответствует состояние *выключен*.

Электрическая схема, реализующая **логическую функцию *НЕ*** и используемое обозначение, представлена на *рис. 5.2*.

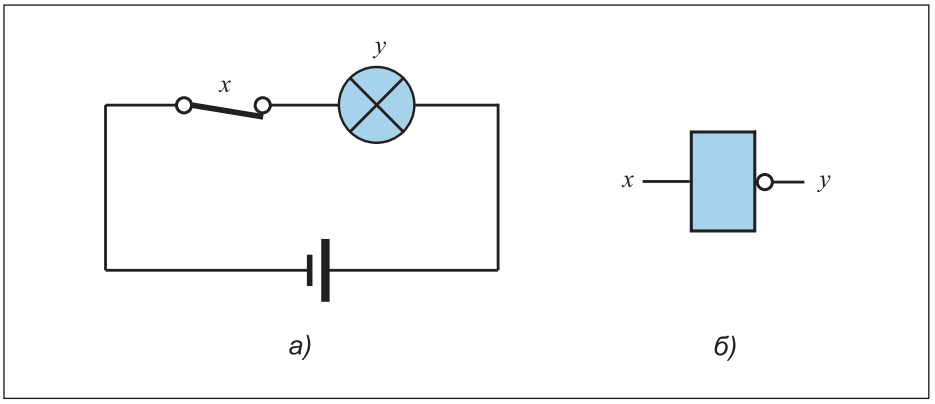


Рис. 5.2. Схема с контактами для реализации логической функции *НЕ (а)* и используемое обозначение *(б)*

Аргумент x представлен с помощью нормально замкнутого контакта, а значения зависимой переменной y – с помощью состояний электрической лампы: *погашена* (логическое значение **0**) или *горит* (логическое значение **1**). Заметим, что лампа будет гореть ($y = 1$), если нормально замкнутый контакт не включен ($x = 0$).

Логическая функция *И* реализуется последовательным соединением электрических контактов. Электрическая схема, реализующая функцию *И* двух переменных, и используемое обозначение представлены на *рис. 5.3*.

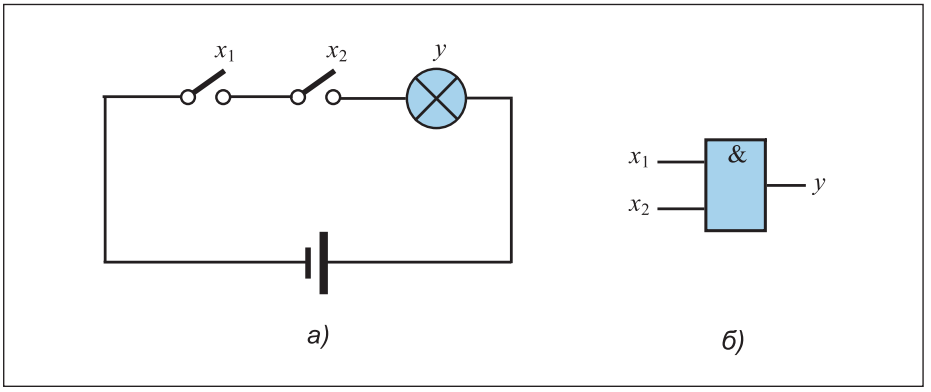


Рис. 5.3. Схема с контактами для реализации логической функции *И (а)* и используемое обозначение *(б)*

Переменные x_1 и x_2 представлены с помощью двух нормально разомкнутых контактов, а значение переменной y – с помощью лампы. Заметим, что лампа будет гореть ($y = 1$) только тогда, когда оба нормально разомкнутых контакта одновременно включены ($x_1 = 1$ и $x_2 = 1$).

Логическая функция *ИЛИ* реализуется параллельным соединением электрических контактов. Соответствующая схема представлена на *рис. 5.4*.

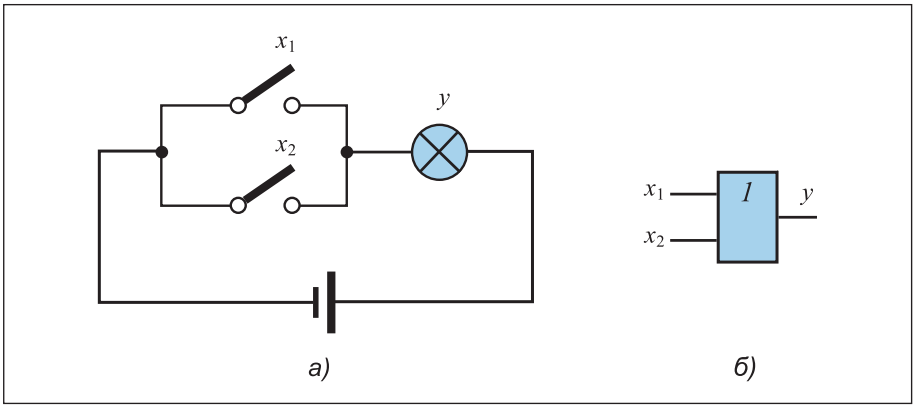


Рис. 5.4. Схема с контактами для реализации логической функции *ИЛИ (а)* и используемое обозначение *(б)*

Отметим, что лампа будет гореть ($y = 1$), если хотя бы один из нормально разомкнутых контактов будет включен ($x_1 = 1$ или $x_2 = 1$).

Поскольку скорость замыкания-размыкания электрических контактов очень мала, то в современных компьютерах значения **0, 1** представлены уровнями напряжения, а в качестве коммутирующего элемента используется транзистор.

Транзистор – это электронный прибор, изготовленный внутри или на поверхности полупроводникового кристалла. Современные технологии позволяют получать $10^6 - 10^7$ транзисторов на 1 $см^2$ площади кристалла.

В режиме коммутации транзистор можно рассматривать как обычный выключатель, который в одном состоянии проводит электрический ток (контакты замкнуты), а в другом – нет (контакты разомкнуты). Однако, в отличие от обычных выключателей, открытие и закрытие транзистора осуществляется с помощью электрического тока.

Существуют различные типы транзисторов. На *рис. 5.5* представлен транзистор *n-p-n* (сокращение относится к внутренней структуре транзистора) и эквивалентные схемы, иллюстрирующие его работу в режиме коммутации.

Транзистор *n-p-n* имеет три вывода: эмиттер *Э*, база *Б* и коллектор *К*. В режиме коммутации эмиттер и коллектор могут рассматриваться как контакты, замыкающиеся и размыкающиеся с помощью напряжения, поданного на базу. Отметим, что современные транзисторы позволяют осуществлять

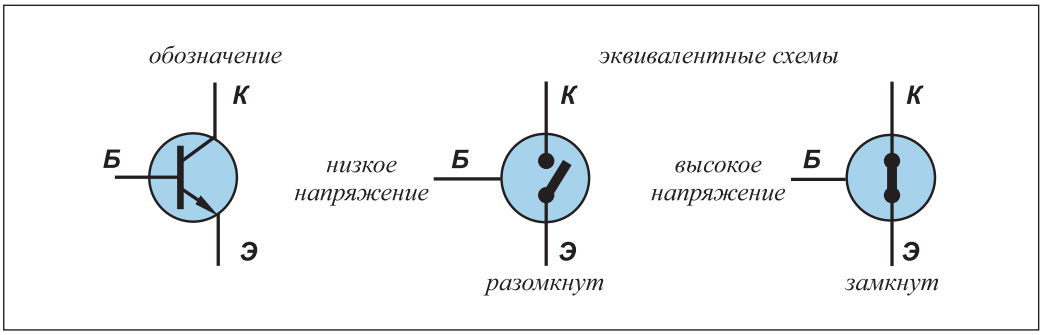


Рис. 5.5. Транзистор *n-p-n*

до 10^6 – 10^9 переключений в секунду. Как и в случае электрических контактов, рассмотренных выше, использование различных типов транзисторов и их последовательное или параллельное соединение позволяет реализовать логические функции *НЕ*, *И*, *ИЛИ*.

> *Логические схемы, предназначенные для вычисления часто используемых логических функций, называются элементарными логическими схемами или логическими элементами.*

Обозначения логических элементов приведены на *рис. 5.6*.

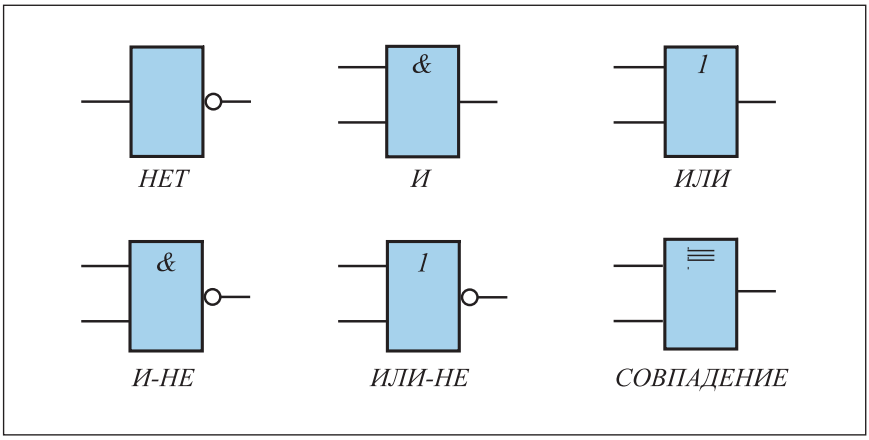


Рис. 5.6. Обозначения логических элементов

Известно, что любая логическая функция может быть выражена с помощью формулы, включающей только элементарные операторы $\bar{}$, &, $\vee$. Следовательно, **любая логическая функция произвольного количества аргументов может быть реализована путем соединения логических элементов *НЕ, И, ИЛИ*.**

Например, функция

$$y = x_1 x_2 \vee \bar{x}_2 x_3$$

может быть реализована с помощью следующих логических элементов:

– элемента *НЕ* для вычисления $\bar{x}_2$;

– двух логических элементов *И* для вычисления конъюнкций $x_1 x_2$ и $\bar{x}_2 x_3$;

– элемента *ИЛИ* для вычисления дизъюнкции $x_1 x_2 \vee \bar{x}_2 x_3$;

Схема логической цепи для вычисления рассматриваемой функции представлена на *рис. 5.7*.

Вопросы и упражнения

❶ Как можно представить двоичные значения **0** и **1**?

❷ Как работают нормально разомкнутые и нормально замкнутые контакты?

❸ Как представляются двоичные значения **0** и **1** в контактных схемах?

❹ Используя оборудование из физической лаборатории, соберите схемы, изображенные на *рис. 5.2, 5.3* и *5.4*. Проверьте таблицы истинности функций, реализованных данными схемами.

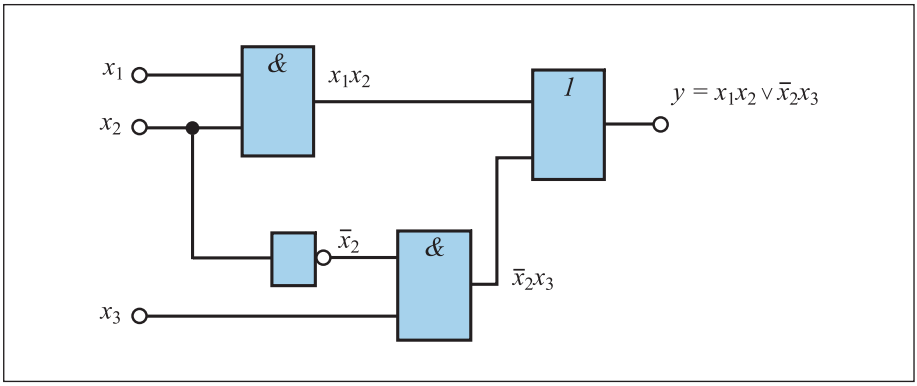


Рис. 5.7. Логическая схема для реализации функции $y = x_1x_2 \vee \bar{x}_2x_3$

❺ Как с помощью контактных схем реализуются функции *НЕ*, *И*, *ИЛИ*?

❻ Какова роль элементов коммутации при реализации логических функций?

❼ Какова роль транзистора в современных компьютерах?

❽ Запомните обозначения логических элементов. Объясните, как используются логические элементы для реализации произвольных логических функций.

❾ Используя элементы *НЕ*, *И*, *ИЛИ*, составьте логические схемы для вычисления следующих функций:

a) $y = x_1x_2x_3$;

b) $y = x_1 \vee x_2 \vee x_3$;

c) $y = \bar{x}_1x_2x_3$;

d) $y = \bar{x}_1 \vee x_2 \vee x_3$;

e) $y = x_1x_2 \vee x_3x_4$;

f) $y = (x_1 \vee x_2)(x_3 \vee x_4)$;

g) $y = \overline{x_1x_2}$;

h) $y = \overline{x_1 \vee x_2}$;

i) $y = x_1x_2 \vee \overline{x_2x_3}$;

j) $y = (x_1 \vee x_2)(\overline{x_2 \vee x_3})$;

k) $y = x_1x_2 \vee \bar{x}_1x_3 \vee x_3x_4$;

l) $y = \bar{x}_1x_2 \vee \bar{x}_1x_3 \vee x_2x_3$;

m) $y = \bar{x}_1x_2 \vee x_1\bar{x}_2$;

n) $y = x_1x_2 \vee \bar{x}_1\bar{x}_2$;

o) $y = x_1(x_2 \vee x_3 \vee x_4)$;

p) $y = x_1 \vee \overline{x_2x_3x_4}$.

❿ Электромагнитное реле – это устройство, с помощью которого осуществляется замыкание и размыкание электрических контактов. Соответствующие контакты включаются электромагнитом. Как можно использовать реле для реализации логических функций *НЕ*, *И*, *ИЛИ*?
Из деталей школьного физического набора соберите электромагнитное реле и проверьте работу схем, предложенных вами.

⓫ Представляя двоичные значения переменных на выходе наличием (значение **1**) или отсутствием (значение **0**) жидкости, разработайте гидравлическую схему для реализации логических функций *НЕ*, *И*, *ИЛИ*. Соберите соответствующие схемы, используя краны и принадлежности школьного химического набора. Проверьте таблицы истинности реализованных логических функций.

5.2. Классификация логических схем

Логические схемы классифицируются на комбинационные и последовательностные.

В **комбинационной схеме** значения переменных на выходе определяются только текущими значениями переменных на входе в соответствии с логическими функциями схемы.

В **последовательностной схеме** значения переменных на выходе зависят не только от значений входных переменных в текущий момент времени, но и от последовательности их подачи.

Другими словами, комбинационные схемы представляют собой логические устройства без элементов памяти, в то время как последовательностные схемы содержат элементы двоичной памяти. Следовательно, комбинационная схема выполняет числовую обработку информации, которую в общем случае можно выразить с помощью логических функций, не содержащих временных параметров.

Обозначение комбинационных схем приведено на *рис. 5.8*.

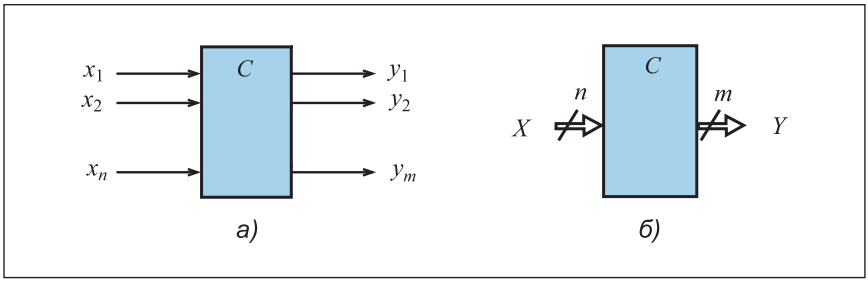


Рис. 5.8. Обозначение комбинационных схем: *а* – детальное; *б* – обобщенное

Комбинационная схема имеет n входных переменных $X = <x_1, x_2, \ldots, x_n>$, и m выходных переменных: $Y = <y_1, y_2, \ldots, y_m>$. Соединения, осуществляющие передачу (на вход или с выхода) значений группы переменных, обозначаются с помощью двойной линии. В случае необходимости количество переменных указывается возле наклонной черты, пересекающей двойную линию группы переменных. Например, на *рис. 5.8, б* показано, что группа X содержит n переменных, а группа Y – m переменных.

С точки зрения теории информации комбинационная схема представляет собой преобразователь кодов: на вход подаются двоичные слова исходного кода, а на выходе появляются соответствующие двоичные слова из кода, в который осуществляется преобразование. Например, исходным кодом может быть *EBCDIC*, а результирующим кодом – *ASCII*.

5.3. Сумматор

При обработке информации одна из главных задач любого компьютера состоит в выполнении арифметических операций и в особенности сложения и вычитания. Устройства, осуществляющие указанные операции, основываются на схемах, выполняющих сложение и вычитание двух двоичных цифр.

Полусумматор – это комбинационная схема, предназначенная для сложения двух двоичных цифр. Таблица истинности, поясняющая работу полусумматора, основывается на правиле сложения двух двоичных цифр и представлена на *рис. 5.9*. Здесь a и b представляют две суммируемые двоичные цифры, s – цифру суммы соответствующего разряда, а t – цифру переноса в следующий разряд.

a	b	s	t
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Рис. 5.9. Таблица истинности для сложения двух двоичных цифр

Для разработки одной из возможных схем полусумматора выразим функции выходных переменных s и t:

$$s = \bar{a}b \vee a\bar{b};$$

$$t = ab.$$

Схема, реализующая функции s, t, и используемое обозначение представлены на *рис. 5.10*.

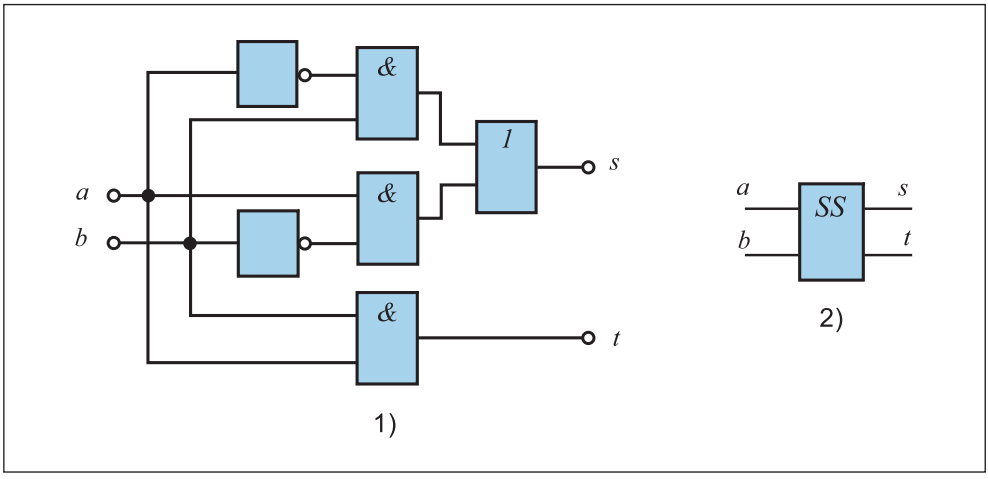


Рис. 5.10. Схема полусумматора (1) и используемое обозначение (2)

Рассмотрим два двоичных числа

$$A = a_{n-1}\ a_{n-2}\ \ldots\ a_j\ \ldots\ a_0$$

и

$$B = b_{n-1}\ b_{n-2}\ \ldots\ b_j\ \ldots\ b_0,$$

где a_j и b_j представляют двоичные цифры разряда j. При суммировании цифр a_j и b_j разряда j нужно учитывать и цифру переноса t_{j-1} из разряда $j - 1$:

$$
\begin{array}{l}
\qquad\qquad\quad t_{j-1} \\
\quad a_{n-1}\ a_{n-2}\ \ldots\ a_j\ \ldots\ a_0 \\
+ \\
\quad b_{n-1}\ b_{n-2}\ \ldots\ b_j\ \ldots\ b_0.
\end{array}
$$

Таким образом, получаем комбинационную схему, вычисляющую сумму $t_{j-1} + a_j + b_j$, называемую **элементарным сумматором.**

Элементарный сумматор может быть реализован путем каскадного соединения двух полусумматоров SS_1 и SS_2 (*рис. 5.11*).

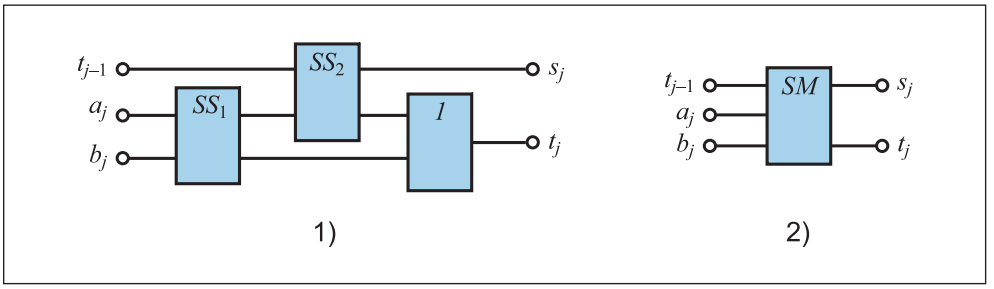


Рис. 5.11. Схема элементарного сумматора (1) и используемое обозначение (2)

Полусумматор SS_1 вычисляет сумму $(a_j + b_j)$, а полусумматор SS_2 суммирует перенос t_{j-1} с суммой $(a_j + b_j)$, вычисленной первым полусумматором. Перенос t_j в старший разряд $j + 1$ вычисляется логическим элементом *ИЛИ*, который объединяет промежуточные переносы полусумматоров SS_1 и SS_2.

Сумма двоичных чисел A и B вычисляется с помощью комбинационной схемы, называемой **сумматором**. Сумматор может быть реализован путем каскадного соединения n элементарных сумматоров (*рис. 5.12*).

Элементарный сумматор SM_0, соответствующий самой младшей значащей цифре, может быть заменен полусумматором, поскольку для нулевого разряда перенос из предыдущего разряда отсутствует. Перенос, образующийся на выходе элементарного сумматора SM_{n-1} самого старшего значащего разряда, используется для указания переполнения емкости сумматора на n бит.

Из анализа *рис. 5.10, 5.11* и *5.12* следует, что сложное устройство – сумматор на n бит – получен соединением более простых устройств, то есть n элементарных сумматоров. Каждый элементарный сумматор в свою очередь получен с использованием двух полусумматоров и логического элемента *ИЛИ*.

Метод разработки сложных устройств (например, сумматора) путем соединения необходимого числа более простых одинаковых устройств (например, элементарных сумматоров) называется **методом иерархического проектирования.** В соответствии с данным методом все компоненты компьютера принадлежат определенным **уровням иерархии**, например:

уровень 1 – транзисторы;
уровень 2 – логические элементы;
уровень 3 – полусумматоры, элементарные сумматоры и т.д.;
уровень 4 – сумматоры, вычитатели и т.д.;
уровень 5 – арифметические устройства, устройства управления и т.д.

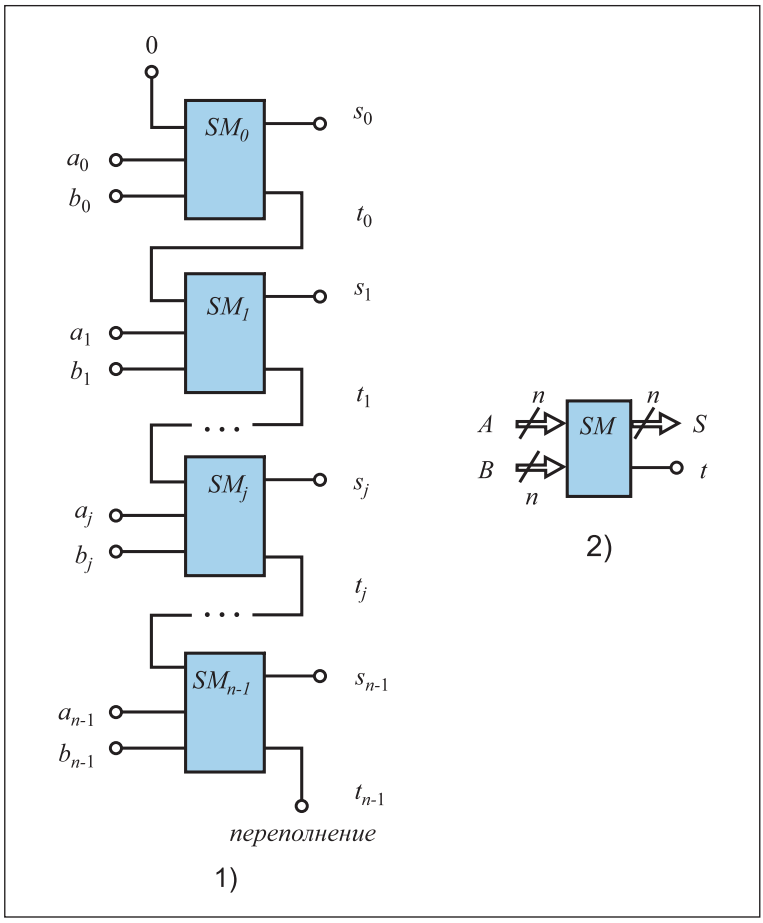


Рис. 5.12. Схема сумматора (1) и используемое обозначение (2)

Компоненты нижнего иерархического уровня используются в качестве простейших «кубиков» для построения компонентов верхнего уровня.

Теоретически устройства любого цифрового компьютера могут быть разработаны и без иерархического метода проектирования. Например, в случае элементарного сумматора использование полусумматоров не обязательно. Достаточно составить таблицу истинности элементарного сумматора, представить выходные функции при помощи формул и использовать соответствующие логические элементы.

Для компонентов более высокого уровня попытка обойтись без метода иерархического проектирования делает невозможной разработку сложных устройств. Например, в случае n-битового сумматора соответствующая таблица истинности состоит из 2^{2n} строк. Для $n = 16$ получаем $2^{2 \cdot 16} = 2^{32} \approx 10^9$ строк. Очевидно, что формулы выходных функций сумматора на 16 битов практически не могут быть записаны. Следовательно, мы вынуждены использовать метод иерархического проектирования и реализовать сумматор путем соединения n элементарных сумматоров.

Аналогичным образом, используя метод иерархического проектирования, можно разработать комбинационные схемы, предназначенные для вычитания двоичных чисел: **полувычитатель**, **элементарный вычитатель** и **вычитатель**.

Вопросы и упражнения

❶ Укажите назначение полусумматора; элементарного сумматора; сумматора на n битов.

❷ Сотавьте таблицу истинности элементарного сумматора. Таблица должна содержать пять столбцов: три для входов a_j, b_j, t_{j-1} и два для выходов s_j, t_j.

❸ Напишите на ПАСКАЛЕ программу, которая моделирует работу элементарного сумматора. Двоичные цифры a_j, b_j и цифра переноса t_{j-1} от младшего разряда вводятся с клавиатуры, а цифра суммы s_j и цифра переноса к старшему разряду – t_j должны быть выведены на экран.

❹ Объясните суть метода иерархического проектирования. Обязательно ли применение данного метода? Аргументируйте ваш ответ.

❺ Сколько логических элементов *НЕ*, *И*, *ИЛИ* содержит сумматор на 16 битов? А на 32 бита?

❻ **Полувычитатель** – это комбинационная схема, предназначенная для вычитания двух двоичных цифр. Соответствующая схема имеет входы a, b и выходы d, i. С помощью d обозначена разность $a - b$, а с помощью i – заем из старшего ближайшего разряда. Составьте таблицу истинности и разработайте схему полувычитателя.

❼ **Элементарный вычитатель** – это комбинационная схема, способная вычислять разность d_j и заем i_j из ближайшего старшего разряда, если на вход подаются уменьшаемое a_j, вычитаемое b_j и заем i_{j-1} из предыдущего разряда. Используя метод иерархического проектирования, разработайте схему элементарного вычитателя.

❽ Используя метод иерархического проектирования, разработайте схему **вычитателя** на n битов.

❾ Сколько логических элементов *НЕ*, *И*, *ИЛИ* содержит вычитатель на 16 битов? А на 32 бита?

❿ Обязательно ли применение метода иерархического проектирования при разработке вычитателя на n битов? Аргументируйте ваш ответ.

5.4. Часто используемые комбинационные схемы

Часто используемые комбинационные схемы представлены на *рис. 5.13*.

Сумматор – это комбинационная схема, предназначенная для сложения двух двоичных чисел. Таблица истинности и схема сумматора были рассмотрены в предыдущем параграфе.

Компаратор – это комбинационная схема, которая сравнивает два двоичных числа A и B, указывая на трех выходах одну из возможных ситуаций: $A < B$, $A > B$ и $A = B$.

Шифратор – это комбинационная схема, которая выполняет преобразование сообщений s_1, s_2, …, s_n в двоичные слова в соответствующем коде. Каждое сообщение s_i представлено значениями $x_1 = 0$, …, $x_i = 1$, …, $x_n = 0$, поданными на вход шифратора, а закодированное (зашифрованное) слово – с помощью выходных переменных y_1, y_2, …, y_m.

Например, переменные x_1, x_2, x_3, … могут представлять состояния клавиш <*A*>, <*B*>, <*C*>, … клавиатуры. Соответствующий шифратор обеспечит на выходе слово в коде ASCII для нажатой клавиши.

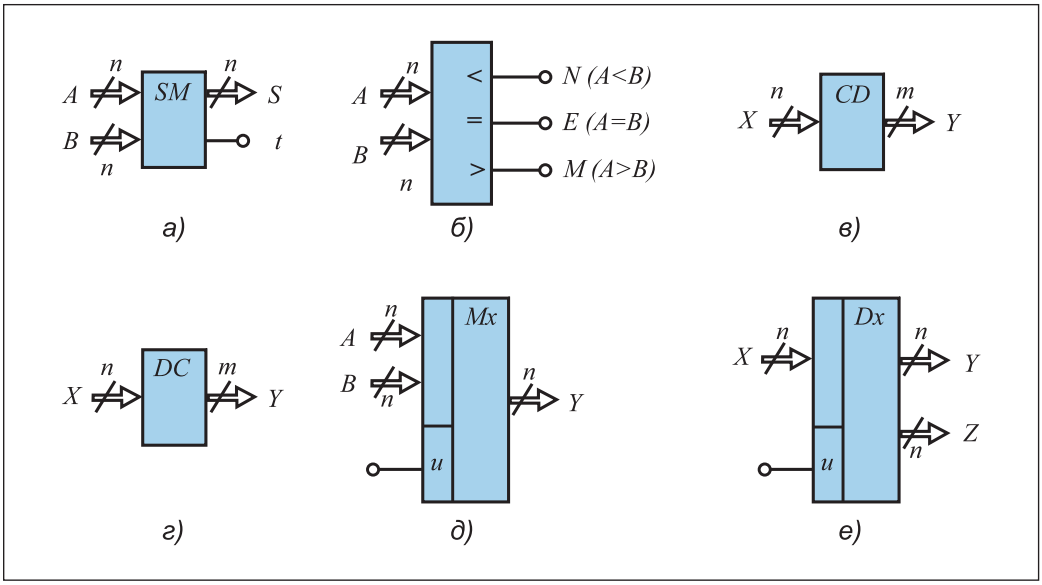


Рис. 5.13. Часто используемые комбинационные схемы:
а – сумматор; *б* – компаратор; *в* – шифратор; *г* – дешифратор;
д – мультиплексор; *е* – демультиплексор

Дешифратор – это комбинационная схема, которая генерирует логический сигнал 1 на выходе, различном для каждой комбинации входных переменных. Другими словами, дешифратор осуществляет операцию, обратную той, которую выполняет шифратор.

Дешифраторы используются для определения операций, которые должен выполнять процессор, для определения состояния устройств ввода-вывода, для синтеза символов и т.п.

Мультиплексор – это комбинационная схема, предназначенная для выбора потока данных. На рис. *5.13, д* представлен мультиплексор, который передает на выход биты двоичного числа A ($u = 0$) или B ($u = 1$). В современных компьютерах мультиплексоры используются для передачи данных от нескольких источников к одному приемнику.

Демультиплексор распределяет поток данных с входа X на один из выходов Y ($u = 0$) или Z ($u = 1$). В качестве примера вспомним передачу информации от одного источника к нескольким приемникам.

Вопросы и упражнения

❶ Объясните назначение часто используемых комбинационных схем: сумматора, компаратора, шифратора, дешифратора, мультиплексора и демультиплексора.

❷ Составьте таблицу истинности компаратора на 2 бита.

❸ Сколько входов и сколько выходов может быть у шифратора? Сколько входов и сколько выходов может быть у дешифратора?

❹ На панели управления принтера установлены кнопки *ON LINE* (работа под управлением центрального устройства), *OFF LINE* (автономная рабо-

та), *LINE FEED* (продвинуть на строку) и *FORM FEED* (продвинуть на страницу). Составьте таблицу истинности шифратора, который вырабатывает на выходе следующие двоичные комбинации:

00 – *ON LINE;*

01 – *OFF LINE;*

10 – *LINE FEED;*

11 – *FORM FEED.*

❺ На панели управления принтера установлены светодиодные индикаторы: *READY* (готов), *PAPER* (отсутствие бумаги), *TEST* (режим тестирования) и *LOAD* (режим загрузки информации). Составьте таблицу истинности дешифратора, который управляет светодиодами. Соответствующие режимы закодированы с помощью следующих двоичных комбинаций:

00 – *READY;*

01 – *PAPER;*

10 – *TEST;*

11 – *LOAD.*

❻ Клавиатура компьютера содержит функциональные клавиши <*F1*>, <*F2*>, ..., <*F12*>. Составьте таблицу истинности шифратора, который выдает на выходе двоичное число, соответствующее нажатой клавише.

❼ Устройства ввода-вывода учебного компьютера имеют следующие адреса:

0000 – клавиатура;

0001 – монитор;

0010 – механический принтер;

0011 – струйный принтер;

0100 – лазерный принтер;

0101 – дисковод для гибких дисков *A*;

0110 – дисковод для гибких дисков *B*;

0111 – привод жесткого диска *C*;

1000 – привод жесткого диска *D*.

Составьте таблицу истинности дешифратора, который выбирает устройство, указываемое соответствующим адресом.

❽ Арифметические операции учебного компьютера закодированы следующим образом:

сложение – 000;

вычитание – 001;

умножение	– 010;
деление	– 011;
сравнение	– 100.

Составьте таблицу истинности дешифратора арифметических операций рассматриваемого компьютера.

❾ Составьте таблицу истинности мультиплексора с двумя входными линиями.

5.5. *RS*-триггер

Известно, что в последовательностных схемах значения выходных переменных зависят не только от комбинаций входных переменных, но и от **последовательности их подачи**. Другими словами, последовательностная схема запоминает информацию о двоичных комбинациях, поданных на входы схемы в предыдущие моменты времени. Такое возможно благодаря тому, что последовательностные схемы состоят из комбинационных схем и элементов **двоичной памяти.**

> *Элемент двоичной памяти – это схема с двумя различными состояниями, предназначенная для хранения одного бита информации. Соответствующая схема называется триггером.*

На *рис. 5.14* представлена схема простейшего триггера, выполненного на логических элементах *ИЛИ-НЕ*, называемого **асинхронным *RS-триггером***. У схемы есть два входа, обозначаемые R и S, и два выхода, обозначаемые Q и $\overline{Q}$. Заметим, что выходные сигналы Q и $\overline{Q}$ подаются на вторые входы элементов *ИЛИ-НЕ*. Соответствующие соединения называются **обратными связями.** Именно благодаря этим соединениям данная схема обладает двумя различными состояниями и, следовательно, обеспечивает запоминание одного бита информации.

В самом деле, пусть входные сигналы $R = S = 0$, а выходы $Q = 1$, $\overline{Q} = 0$. Из-за обратной связи значение $Q = 1$ заставляет другой выход принять значение $\overline{Q} = 0$. В свою очередь, благодаря обратной связи, значение $\overline{Q} = 0$ подтвержда-

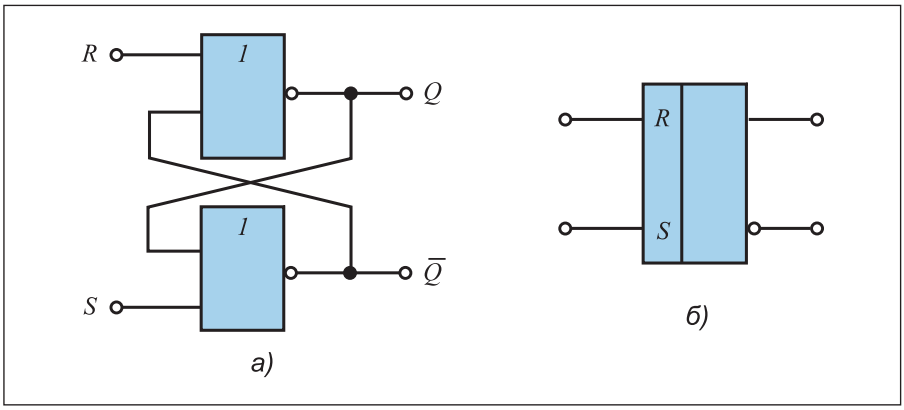


Рис. 5.14. Схема асинхронного *RS*-триггера (*а*) и используемое обозначение (*б*)

ет выходной сигнал $Q = 1$. Точно так же в случае, когда выходы $Q = 0$, $\bar{Q} = 1$, из-за обратной связи значение $\bar{Q} = 1$ удерживает другой выход в состоянии $Q = 0$. Следовательно, входные значения $R = S = 0$ не изменяют состояние триггера, обеспечивая хранение ранее записанной двоичной цифры.

Удобно, чтобы состоянию $Q = 1$, $\bar{Q} = 0$ соответствовала двоичная цифра 1, а состоянию $Q = 0$, $\bar{Q} = 1$ – двоичная цифра 0. Выход Q называется **прямым** или **истинным выходом**, а выход $\bar{Q}$ – **инверсным выходом** или **выходом отрицания**. Состояние триггера указывается значением прямого выхода Q.

Рассмотрим случай, когда $R = 0$, $S = 1$. Предположим, что триггер находится в состоянии 0, то есть $Q = 0$ и $\bar{Q} = 1$. В этом случае значение $S = 1$ переведет $\bar{Q} = 0$, который в свою очередь, через обратную связь, обеспечит $Q = 1$. Следовательно, триггер переходит в состояние 1 ($Q = 1$, $\bar{Q} = 0$). Это состояние, как установлено ранее, сохранится и после того, как сигнал S изменит свое значение с 1 на 0. Вход S, который обеспечивает установку триггера в состояние 1, называется **входом установки**.

Аналогично можно установить, что в случае, когда триггер находится в состоянии 1 ($Q = 1$, $\bar{Q} = 0$), а $S = 0$, и $R = 1$, триггер перейдет в состояние 0 ($Q = 0$, $\bar{Q} = 1$). Это состояние сохранится и после перехода сигнала R от значения 1 к значению 0. Вход R, который обеспечивает установку триггера в состояние 0, носит название **входа сброса**.

Входная комбинация $R = 1$ и $S = 1$ заставляет выходы перейти в состояния $Q = 0$ и $\bar{Q} = 0$. Значения $Q = \bar{Q} = 0$ не соответствуют условию противоположности сигналов на выходах. Следовательно, для триггера RS входная комбинация $R = S = 1$ является **запрещенной**.

Режимы работы рассматриваемого триггера приведены в *таблице 5.1*.

Таблица 5.1.

Режимы работы асинхронного *RS*-триггера

Входы		***Режим работы***	***Выход Q***
R	***S***		
0	0	хранение	хранимый бит
0	1	установка	1
1	0	сброс	0
1	1	запрещено	–

Прилагательное *асинхронный* в названии *RS*-триггера указывает на характер влияния управляющих сигналов R и S на состояние триггера. Из рассмотрения схемы, приведенной на *рис. 5.14*, следует, что управляющие сигналы, поданные на входы R и S, могут изменять состояние триггера в произвольные моменты времени.

> *Последовательностные схемы, состояние которых может быть изменено управляющими сигналами в произвольные моменты времени, называются асинхронными схемами.*

Современный компьютер содержит десятки тысяч триггеров. Изменение их состояний в произвольные моменты времени трудно контролировать, и это может привести к ошибкам в работе. Для исключения ошибок необходи-

мо, чтобы поведение последовательностных схем контролировалось значениями управляющих сигналов, подаваемых на входы в точно определенные дискретные моменты времени. Эти моменты времени задаются с помощью специальных импульсов, называемых **сигналами синхронизации.**

> *Последовательностные схемы, состояние которых может быть изменено управляющими сигналами только в моменты времени, определяемые сигналами синхронизации, называются синхронными схемами.*

Обычно сигнал синхронизации обозначается с помощью буквы *C* (от английского *clock* – „часы") и вырабатывается специальным устройством, называемым **системными часами.**

На *рис. 5.15* представлена схема **синхронного *RS*-триггера.** Данная схема состоит из асинхронного *RS*-триггера (*рис. 5.14*) и двух логических элементов *И*, которые разрешают подачу управляющих сигналов на входы асинхронного триггера только тогда, когда сигнал синхронизации *C* принимает значение 1.

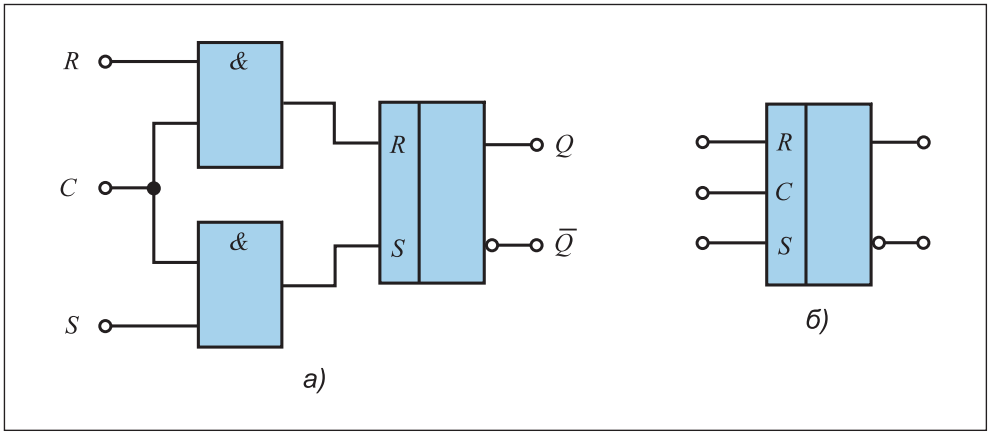


Рис. 5.15. Схема синхронного *RS*-триггера (*а*) и его обозначение (*б*)

Вопросы и упражнения

❶ Чем отличаются комбинационные и последовательностные схемы?

❷ В чем назначение триггера?

❸ Как работает триггер на основе логических элементов *ИЛИ-НЕ*? Для чего предназначены обратные связи?

❹ Объясните режимы работы асинхронного *RS*-триггера. Почему комбинация $R = S = 1$ не может быть подана на входы рассматриваемого триггера?

❺ Чем отличаются асинхронные и синхронные последовательностные схемы?

❻ Объясните, как работает синхронный *RS*-триггер. Каково назначение логических элементов *И*, входящих в состав данного триггера?

❼ На *рис. 5.16* представлена схема простейшего триггера, реализованного на основе логических элементов *И-НЕ*, называемого **асинхронным $\overline{R}\overline{S}$–триггером.** Схема имеет следующие режимы функционирования:
– хранение ($\overline{R} = 1$, $\overline{S} = 1$);

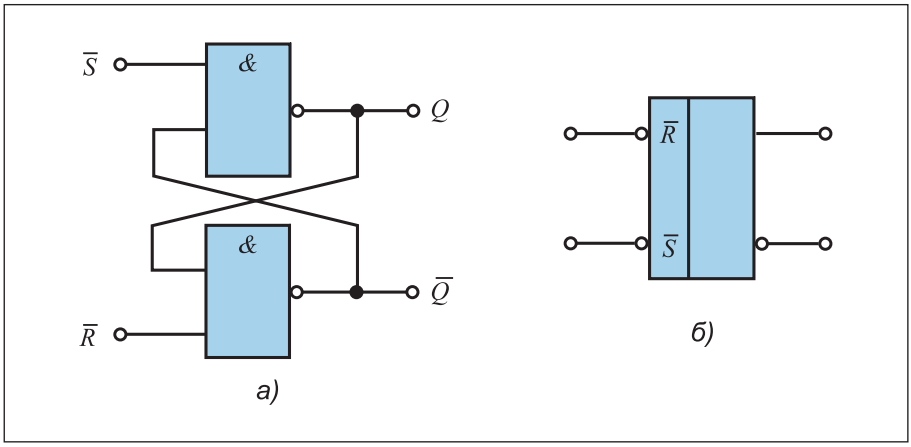


Рис. 5.16. Схема асинхронного $\overline{R}\overline{S}$-триггера (*а*) и используемое обозначение (*б*)

– установка ($\overline{R} = 1$, $\overline{S} = 0$);
– сброс ($\overline{R} = 0$, $\overline{S} = 1$).

Объясните, как работает рассматриваемый триггер. Почему входная комбинация $\overline{R} = 0$, $\overline{S} = 0$ является запрещенной?

❽ Используя асинхронный–триггер $\overline{R}\overline{S}$, разработайте схему **синхронного $\overline{R}\overline{S}$–триггера.**

❾ Используя схемы *рис. 5.14, а* и *5.15, а*, нарисуйте подробную схему (на уровне логических элементов *И*, *ИЛИ-НЕ*) синхронного *RS*-триггера.

❿ Нарисуйте подробную схему (на уровне логических элементов *И*, *И-НЕ*) синхронного–триггера $\overline{R}\overline{S}$.

5.6. Часто используемые последовательностные схемы

Регистр (*рис. 5.17, а*) – это цифровое устройство, предназначенное для временного хранения произвольного двоичного числа,

$$D = d_{n-1} \ldots d_1 d_0.$$

Регистр состоит из триггеров и комбинационных схем, разрешающих запись, чтение или перенос (сдвиг) информации. Запись информации в регистр осуществляется путем подачи на вход W (*Write* – „писать“) соответствующего импульса. Так как каждый триггер может запоминать только один бит, то **емкость** регистра определяется числом используемых триггеров.

В некоторых приложениях, например, при умножении и делении двоичных чисел, при записи и чтении данных на диске, при передаче данных по телефонным линиям и т.п., появляется необходимость сдвига влево или вправо информации, содержащейся в некотором регистре. Для этого используются **сдвигающие регистры** (*рис. 5.17, б, в*).

Последовательность состояний **регистра со сдвигом влево** приведена на *рис. 5.18*.

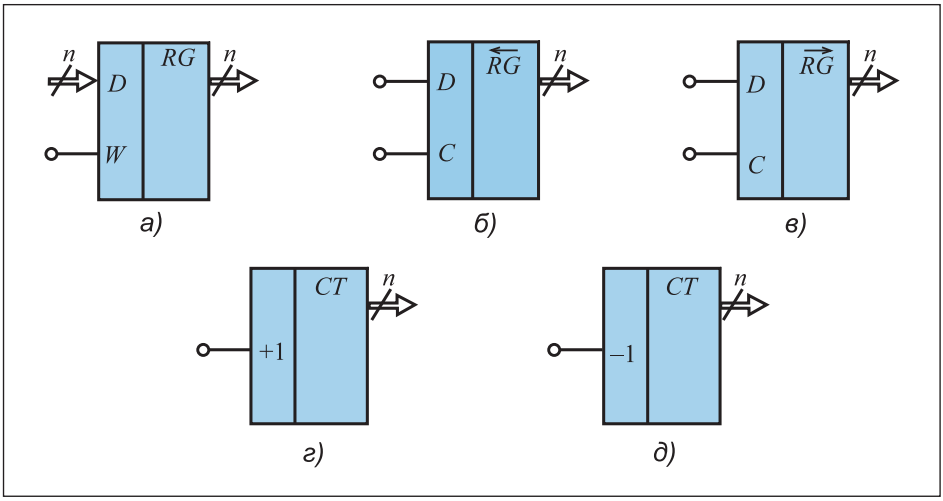


Рис. 5.17. Часто используемые последовательностные схемы:
а – регистр; *б* – регистр сдвига влево; *в* – регистр сдвига вправо; *г* – прямой счетчик; *д* – обратный счетчик

Предполагается, что начальным состоянием регистра является 0101, а импульсы подаются на вход C в момент времени t_1, t_2, t_3 и т.д. Очевидно, что рассматриваемый регистр вычисляет произведение $D \times 2$.

Аналогичным образом, **регистр со сдвигом вправо** осуществляет деление $D : 2$.

Счетчик – это последовательностная схема, которая подсчитывает количество импульсов, поступивших на ее вход. Счетчики классифицируются по следующим критериям:

– способ кодирования информации на выходе (двоичные, двоично-десятичные и т.д.);

– порядок изменения состояний счетчика (прямые, инверсные и реверсивные счетчики).

В общем случае, счетчики создаются путем соединения триггерных схем с комбинационными схемами, определяющими режим изменения состояния счетчиков при поступлении на вход каждого нового импульса.

Двоичный счетчик подсчитывает в двоичной системе счисления количество импульсов, поступивших на его вход. **Емкость** двоичного счетчика зависит

Моменты времени	d_3	d_2	d_1	d_0
начальный	0	1	0	1
t_1	1	0	1	0
t_2	0	1	0	0
t_3	1	0	0	0
t_4	0	0	0	0
t_5	0	0	0	0
...	...			

Рис. 5.18. Последовательность состояний регистра со сдвигом влево

от количества входящих в него триггеров. Считая, что двоичным числам соответствуют различные выходные состояния счетчика, получаем в результате диапазон счета от 0 до $2^n - 1$, где n – это количество триггеров.

На *рис. 5.17, г* представлен **прямой двоичный счетчик,** а на *рис. 5.19* – таблица последовательности состояний 3-битового прямого двоичного счетчика.

Моменты времени	d_3	d_2	d_1	d_0
начальный	0	0	0	
t_1	0	0	1	
t_2	0	1	0	
t_3	0	1	1	
t_4	1	0	0	
t_5	1	0	1	
t_6	1	1	0	
t_7	1	1	1	
t_8	0	0	0	
t_9	0	0	1	
…	…			

Рис. 5.19. Последовательность состояний прямого двоичного счетчика на 3 бита

Счетчики, изменяющие свое состояние в соответствии с таблицей на *рис. 5.19*, называются **прямыми,** поскольку содержимое счетчика увеличивается на единицу с поступлением на вход „+1” каждого нового импульса. Если же в счетчик предварительно запишем некоторое число, а каждый новый импульс, поступивший на вход „–1”, уменьшает его содержимое на единицу, то получим **инверсный счетчик** (*рис. 5.17, д*).

Вопросы и упражнения

❶ В чем назначение регистра? От чего зависит емкость регистра?

❷ В регистр со сдвигом влево *(рис. 5.17, б)* загружено двоичное число 1001. Каким станет содержимое регистра после подачи на вход *C* одного импульса? Двух импульсов?

❸ В регистр со сдвигом вправо загружено одно из следующих двоичных слов:

a) 00000; *f)* 00001;

b) 10000; *g)* 10001;

c) 01000; *h)* 01010;

d) 00100; *i)* 01100;

e) 00010; *j)* 00110.

Каким станет содержимое регистра после поступления на вход *C* двух последовательных импульсов?

❹ Напишите на ПАСКАЛЕ программу, которая моделирует работу регистра сдвига слева направо на n битов.

❺ Для чего предназначены счетчики? Как меняются состояния прямого двоичного счетчика? А состояния инверсного счетчика?

❻ Прямой счетчик на 4 бита находится в одном из следующих исходных состояний:

a)	0000;	*f)*	1010;
b)	0010;	*g)*	1100;
c)	0100;	*h)*	1111;
d)	1000;	*i)*	0101;
e)	1001;	*j)*	0110.

Каким станет состояние счетчика после поступления на вход 5-ти импульсов? А 8-ми импульсов?

❼ Инверсный счетчик на 4 бита находится в исходном состоянии 1001. Каким станет состояние счетчика после поступления m входных импульсов? Число m может принимать значения 1, 4, 5, 8, 11, 17.

❽ Напишите на ПАСКАЛЕ программу, которая моделирует работу прямого двоичного счетчика на n битов.

5.7. Генераторы импульсов

Импульсы используются в цифровых устройствах для того, чтобы обеспечить их последовательную работу во времени. Как правило, генераторы импульсов выполняются на основе логических элементов и элементов задержки.

Элемент задержки представляет собой электронную схему, реализующую логическую функцию повторения $y = x$, причем выходной сигнал y повторяет входной сигнал x с задержкой (опозданием) на Δ единиц времени.

Из курса физики известно, что скорость распространения сигналов конечна. Следовательно, любой проводник может рассматриваться как элемент задержки. Для того чтобы увеличить "инерционность" электронных схем и достичь значительных задержек, в их состав включаются конденсаторы и резисторы. В этом случае задержка Δ определяется емкостью и сопротивлением соответствующих компонентов.

На *рис. 5.20* представлены обозначение и временные диаграммы элемента задержки.

Простейшая схема **генератора периодических импульсов**, реализованного на основе элемента задержки и логического элемента *И-НЕ*, представлена на *рис. 5.21*.

В исходном состоянии $x = 0$ и $y = 1$, а на выходе элемента задержки поддерживается значение логической 1. Когда на вход подается сигнал запуска $x = 1$, сигнал на выходе принимает значение 0 (*рис. 5.22*). Затем логическое значение 0 с задержкой Δ подается на второй вход логического элемента *И-НЕ*. Следовательно, выход принимает значение 1. Уровень логи-

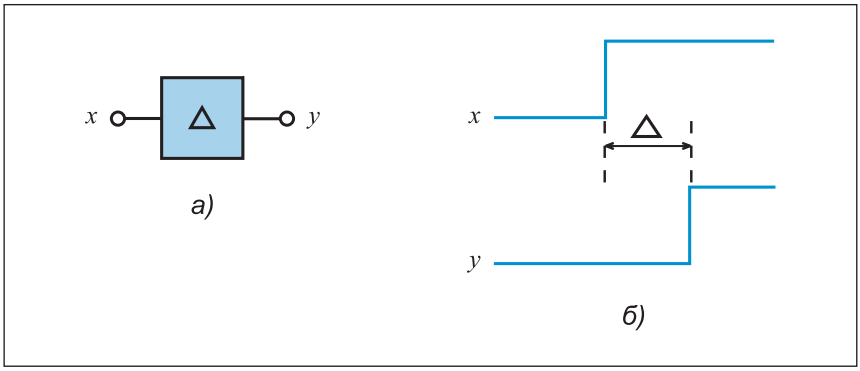


Рис. 5.20. Обозначение (*а*) и временные диаграммы (*б*) элемента задержки

ческой 1 после задержки Δ вновь будет подан на вход логического элемента *И-НЕ*, заставляя тем самым появиться на выходе значение $y = 0$ и т.д.

Следовательно, на выходе y генератора вырабатывается последовательность импульсов длительности Δ. Процесс генерации может быть остановлен путем подачи на управляющий вход сигнала $x = 0$.

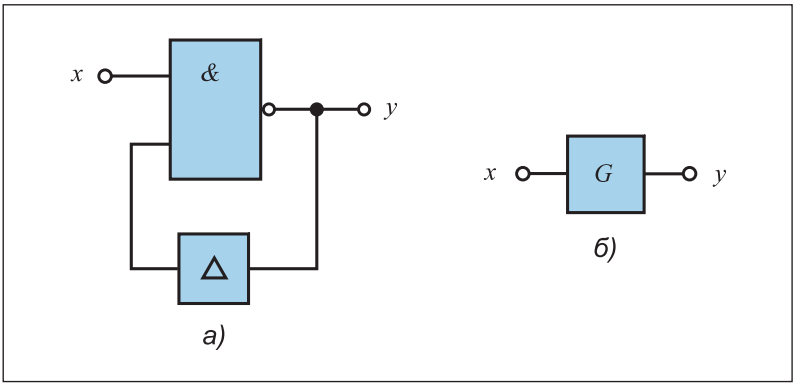


Рис. 5.21. Схема (*а*) и обозначение генератора периодических импульсов (*б*)

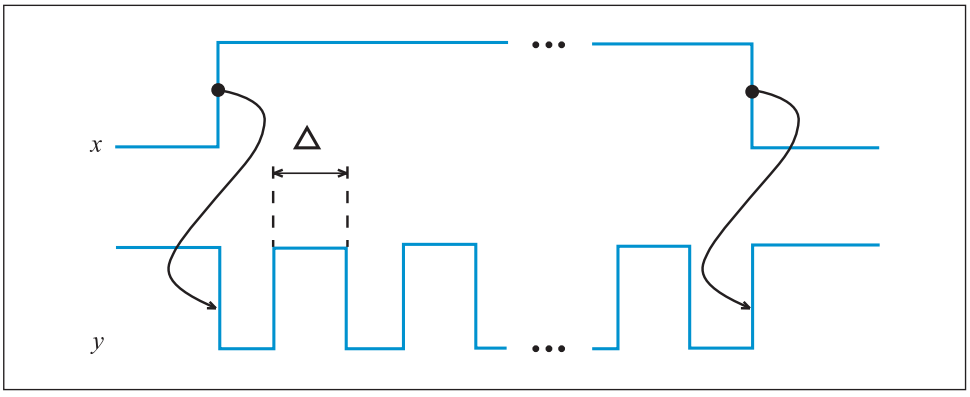


Рис. 5.22. Временные диаграммы генератора периодических импульсов

Вопросы и упражнения

❶ Для чего предназначен элемент задержки? Нарисуйте временные диаграммы рассматриваемого элемента.

❷ Объясните, как работает генератор периодических импульсов. От чего зависит длительность импульсов?

❸ Замените логический элемент *И-НЕ*, входящий в состав генератора периодических импульсов, представленного на *рис. 5.21*, логическим элементом *ИЛИ-НЕ*. Объясните, как будет работать данная схема. Нарисуйте временные диаграммы полученного генератора.

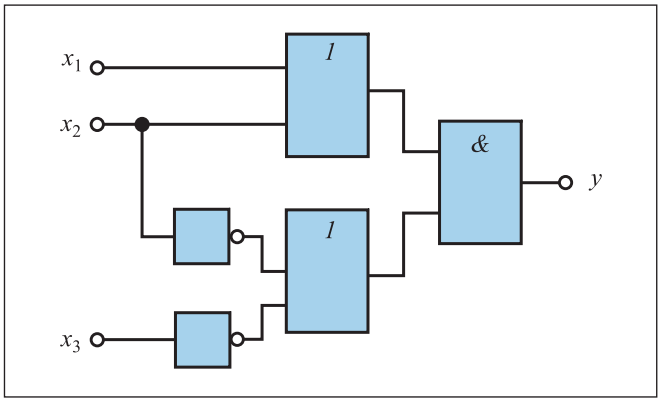
❹ Известно, что изменение физических параметров не может осуществляться мгновенно. Следовательно, любой логический элемент обладает задержкой δ, называемой **паразитной задержкой**, значение которой зависит от особенностей соответствующей схемы.
Исключите из схемы, представленной на *рис. 5.21*, элемент задержки, подавая выходной сигнал логического элемента *И-НЕ* прямо на его вход. Объясните, как будет работать полученная схема. От чего зависит длительность импульсов на выходе логического элемента? Нарисуйте соответствующие временные диаграммы.

❺ Последовательностная схема состоит из логического элемента *НЕ*, сигнал с выхода которого подается прямо на его вход. Как будет работать данная схема?

Тест для самопроверки № 5

1. Нарисуйте схему логической цепи для вычисления функции $y = \bar{x}_1 x_2 \vee x_1 x_3$.

2. На входы приведенной ниже логической схемы поданы следующие логические сигналы: $x_1 = 1$, $x_2 = 0$ и $x_3 = 1$. Перерисуйте схему в тетрадь и укажите на рисунке значения сигналов на входах и выходах каждого логического элемента.



3. Запишите логическую функцию, реализуемую схемой, приведенной в пункте 2.

4. Разработайте на ПАСКАЛЕ программу, которая моделирует работу полусумматора. Двоичные цифры *a* и *b* вводятся с клавиатуры, а цифра суммы *s* и цифра переноса *t* выводятся на экран.

5. Сколько логических элементов *НЕ*, *И*, *ИЛИ* содержит сумматор на 8 битов? Обоснуйте ваш ответ.

6. Напишите на ПАСКАЛЕ программу, которая моделирует работу сумматора на 8 битов. Двоичные числа *A* и *B* считываются с клавиатуры, а сумма *S* и цифра переполнения *t* выводятся на экран.

7. Укажите соответствие между названиями часто используемых комбинационных схем (из левого столбца) и их назначением (из правого столбца):

		(a)	преобразование сообщений в двоичные слова;
(1)	сумматор;	(b)	преобразование сигналов;
(2)	компаратор;	(c)	распределение потоков данных;
(3)	шифратор;	(d)	преобразование двоичных слова в сообщения;
(4)	дешифратор;	(e)	вычисление суммы двух двоичных чисел;
(5)	мультиплексор;	(f)	выбор элементов изображений;
(6)	демультиплексор.	(g)	сравнивнение двух двоичных числа;
		(h)	выбор потоков данных.

8. На панели управления некоторой электронной игры установлены светодиоды для отображения состояний ВВЕРХ, ВНИЗ, НАЛЕВО и НАПРАВО. Составьте таблицу истинности дешифратора, который управляет светодиодами. Перечисленным состояниям соответствуют следующие двоичные комбинации:

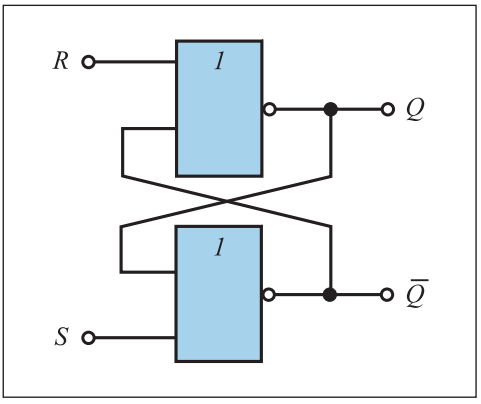
00 – ВВЕРХ;

01 – ВНИЗ;

10 – НАЛЕВО;

11 – НАПРАВО.

9. Зная значения входных сигналов $R = 0$ и $S = 1$, определите значения функций Q и $\bar{Q}$ на выходе приведенного ниже триггера.



10. Нарисуйте подробную схему (на уровне логических элементов *И, И-НЕ*) синхронного триггера $\bar{R}\bar{S}$.

11. Укажите соответствие между названиями часто используемых последовательностных схем (столбец справа) и их назначением (столбец слева):

(1)	триггер;	(a)	сдвиг двоичного слова слева направо;
(2)	регистр;	(b)	преобразование двоичных слов в символы;
(3)	регистр со сдвигом влево;	(c)	прибавление единицы к текущему значению;
(4)	регистр со сдвигом вправо;	(d)	хранение двоичной цифры;
(5)	прямой счетчик;	(e)	сложение двоичных слов, представленных в обратном коде;
(6)	обратный счетчик.	(f)	вычитание единицы из текущего значения;
		(g)	сдвиг двоичного слова справа налево;
		(h)	вычитание двоичных слов, представленных в прямом коде;
		(i)	хранение двоичного слова.

12. В регистр сдвига вправо загружено двоичное слово 10011011. Каким будет содержимое регистра после поступления на вход *C* трех последовательных импульсов?

13. В регистр сдвига влево загружено двоичное слово 11011101. Каким будет содержимое регистра после поступления на вход *C* четырех последовательных импульсов?

14. Напишите на ПАСКАЛЕ программу, которая моделирует работу регистра сдвигом влево. Исходное содержимое *R* регистра и число импульсов *m* вводятся с клавиатуры, а конечное содержимое регистра выводится на экран.

15. Прямой 8-разрядный счетчик находится в начальном состоянии 10001101. Каким будет состояние счетчика после поступления на его вход 6-ти импульсов?

16. Обратный 8-разрядный счетчик находится в начальном состоянии 10101110. Каким будет состояние счетчика после поступления на его вход 5-ти импульсов?

17. Напишите на ПАСКАЛЕ программу, которая моделирует работу прямого счетчика. Начальное состояние *A* счетчика и количество поступивших на вход импульсов *m* вводятся с клавиатуры, а конечное содержимое счетчика выводится на экран.

Глава 6

УСТРОЙСТВО И РАБОТА КОМПЬЮТЕРА

6.1. Функциональная схема компьютера

Функциональная схема цифрового компьютера представлена на *рис. 6.1*.

В соответствии с данной схемой цифровой компьютер содержит следующие **функциональные блоки:**

– блок памяти, предназначенный для хранения исходных, промежуточных и конечных данных задачи вместе с командами, указывающими последовательность вычислений;

– арифметико-логическое устройство, необходимое для выполнения элементарных арифметических и логических операций;

– одно или более устройство ввода и соответственно вывода, необходимых для связи компьютера с внешней средой;

– центральное устройство управления, которое генерирует последовательности управляющих сигналов, необходимых для последовательного выполнения команд.

Арифметико-логическое устройство (АЛУ) и центральное устройство управления (ЦУУ) образуют **центральное устройство обработки информации** или, короче, **процессор.**

Память современных компьютеров организована на двух уровнях: блок **внутренней памяти** с высокой скоростью работы и один или более блоков **внешней памяти** с низкой скоростью, но с емкостью во много раз большей, чем у внутренней памяти.

Внутренняя память (называемая иногда главной, центральной или оперативной памятью) хранит программу и используемые ею данные во время выполнения этой программы. Ее наличие является одним из важнейших условий работы компьютера.

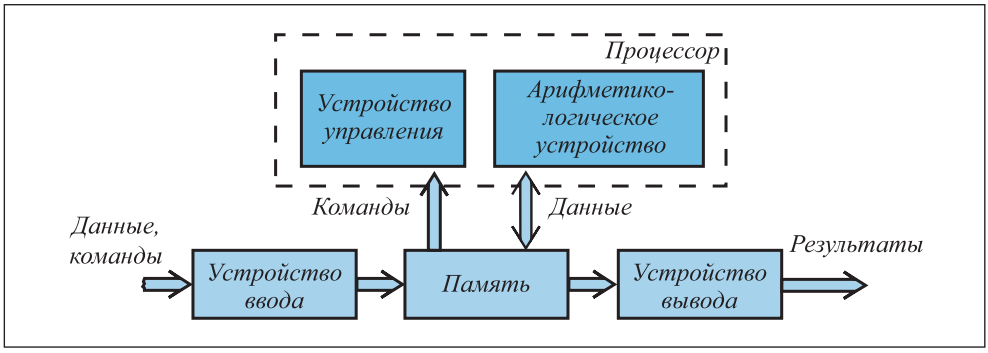


Рис. 6.1. Функциональная схема компьютера

Внешняя память играет роль хранилища больших объемов информации и часто используемых программ с возможностью их загрузки во внутреннюю память за короткий интервал времени. В настоящее время в качестве внешней памяти используются устройства на магнитных дисках или лентах, устройства на оптических дисках и т.п.

Блоки внешней памяти и устройства ввода-вывода называются **периферийным оборудованием.**

Для обеспечения эффективного взаимодействия процессора, внутренней памяти и периферийного оборудования, в случае персональных компьютеров их функциональная схема реализуется физически в соответствии с блок-схемой, представленной на *рис. 6.2*.

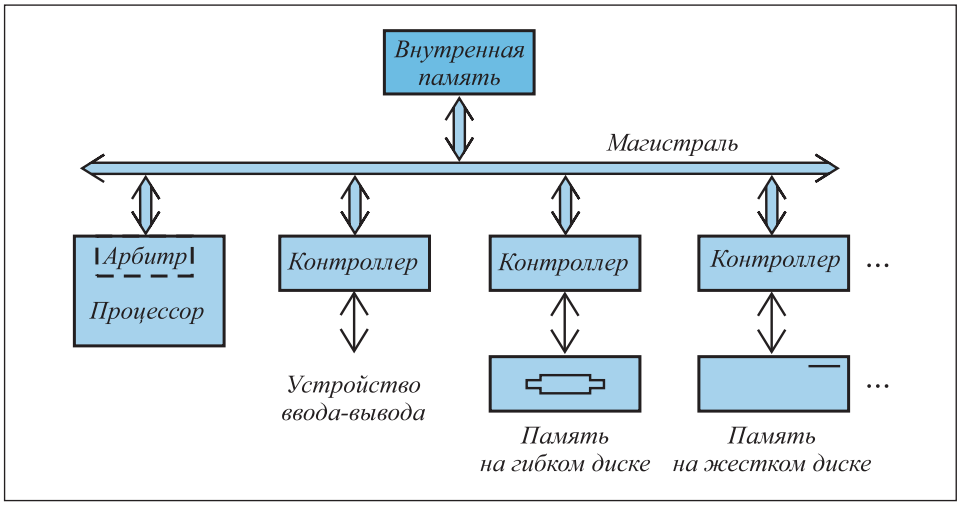


Рис. 6.2. Блок-схема персонального компьютера

Из *рис. 6.2* видно, что все современные компьютеры имеют **модульную конфигурацию**. Каждый модуль (контроллер, принтеры, устройства на магнитных дисках и т.д.) функционирует независимо и, следовательно, может включаться или исключаться из состава компьютера независимо от остальных. Таким образом, конфигурация компьютера может быть изменена в зависимости от области применения вычислительной системы.

Например, типовая издательская система содержит несколько видов принтеров: механический – для черновых текстов, лазерный или цветной – для макетируемых страниц, рисунков и фотографий и т.п. Система оперативного управления большим объемом данных должна содержать магнитные диски большой емкости, а компьютер, предназначенный для видеомонтажа, должен иметь в комплекте видеокамеры, мониторы с соответствующим разрешением, клавиатуры, аналогичные режиссерскому пульту, и т.п.

Вопросы и упражнения

❶ Назовите функциональные блоки компьютера и объясните их назначение.

❷ В чем заключается роль внутренней памяти? Как реализуется внешняя память современных компьютеров?

❸ Назовите известные вам периферийные устройства.

❹ Назовите компоненты персонального компьютера и объясните их назначение.

❺ Расскажите о структуре и взаимодействии компонентов компьютера.

❻ Как может быть изменена конфигурация вычислительной системы? Какие преимущества имеет модульная конфигурация компьютера?

❼ Нарисуйте блок-схему компьютера, на котором вы работаете. Какие компоненты являются обязательными, а какие – нет для функционирования компьютера?

❽ Как можно подключить к компьютеру дополнительное устройство на магнитных дисках? Лазерный принтер? Считыватель документов (сканер)? Видеокамеру?

6.2. Форматы команд

Для решения любой задачи компьютер должен знать в каждый момент времени как операцию, которая должна быть выполнена, так и данные, которые в ней участвуют. Эти операции сообщаются компьютеру с помощью команд.

> *Команда представляет собой последовательность двоичных цифр, с помощью которой процессору указывается операция, которую необходимо выполнить, и расположение операндов.*

Соответствующая двоичная последовательность, называемая иногда **словом команды**, разделена на поля, каждое из которых имеет строго определенное назначение. Количество и назначение полей называется **форматом команды**. На *рис. 6.3* представлены форматы, используемые в современных компьютерах.

В общем случае для выполнения заданной операции необходимо, чтобы ее команда содержала три адреса (*рис. 6.3, а*). Первые два адреса используются для получения операндов, над которыми будет осуществлена операция, указанная в поле *Код операции*. Результат операции будет помещен по адресу, указанному в поле *Адрес результата*.

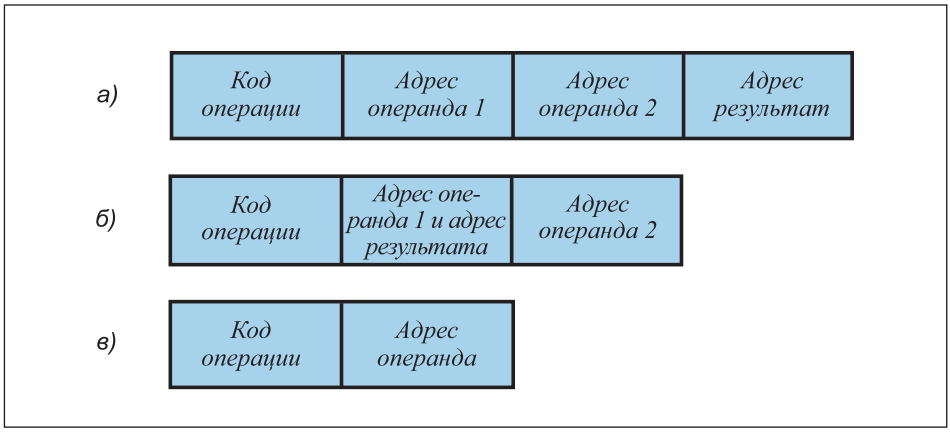


Рис. 6.3. Формат команды с тремя (*а*), двумя (*б*) и одним адресом (*в*)

Рассмотрим пример. Предположим, что арифметические и логические операции кодированы так, как указано ниже (для простоты будем использовать десятичные эквиваленты соответствующих двоичных полей):

01 – сложение;
02 – вычитание;
03 – логическая операция *И*;
04 – логическая операция *ИЛИ*.

Команда

```
01 100 110 215
```

сообщает процессору, что необходимо сложить числа из ячеек 100 и 110 и разместить полученную сумму в ячейку с адресом 215.

Команда

```
02 100 110 215
```

сообщает процессору, что из числа, записанного в ячейку 100, необходимо вычесть число, записанное в ячейку 110. Полученный результат будет помещен в ячейку 215.

Аналогичным образом, команда

```
03 200 300 100
```

задает логическую операцию *И* над битами слов из ячеек 200 и 300. Результат будет размещен в ячейке 100.

Отметим, что в команде указываются не **значения операндов**, а **адреса ячеек**, в которых находятся соответствующие данные. Указанное обстоятельство делает возможным использование одних и тех же программ для обработки любых исходных данных. Тот факт, что команды работают с адресами, содержимое которых должно быть обработано, а не с самим содержимым, составляет основной принцип работы цифровых компьютеров. Рассматриваемый принцип позволяет разрабатывать и вводить в компьютер программы независимо от конкретных данных, которые будут этой программой обрабатываться.

В формате с тремя адресами (*рис. 6.3, а*) адреса задаются **явно**. Для более компактного представления команд используется **неявное задание** некоторых адресов. В этом случае слово команды не содержит полей, предназначенных для косвенных адресов.

В частности, если результат, полученный после выполнения любой операции, помещается по адресу одного из операндов, то в соответствующий формат будут входить только два адреса (*рис. 6.3, б*). Следовательно, адрес результата задается неявно. Например, команда

```
01 100 110
```

сообщает процессору, что необходимо сложить числа из ячеек 100 и 110 и поместить полученную сумму в ячейку 100. Очевидно, что после записи суммы исходное число, находившееся в ячейке 100, будет утеряно.

Установлено, что формат с двумя адресами (самый распространенный в настоящее время) обеспечивает написание программ с количеством команд, сравнимым с количеством команд при использовании большего числа адресов.

Формат с одним адресом (*рис. 6.3, в*) применяется в компьютерах, процессор которых содержит специальный регистр, называемый **аккумулятором**. В аккумуляторе хранится первый операнд, и в него же помещается результат выполнения соответствующей операции. Следовательно, адрес первого операнда и адрес результата задаются неявно. Например, одноадресная команда

```
01 100
```

сложит число из аккумулятора с числом, хранящимся в ячейке 100, а полученная сумма будет помещена в аккумулятор. Естественно, исходное число из аккумулятора будет утеряно.

Команды с одним адресом эффективны с точки зрения длины слова и скорости работы компьютера. Однако программа, написанная с использованием одноадресных команд, будет длиннее, чем программа, написанная с использованием двух- или трехадресных команд.

Отметим, что современные компьютеры имеют команды различных форматов. Формат каждой команды указывается в поле *Код операции*.

Вопросы и упражнения

❶ Перечислите форматы команд, применяемых в современных компьютерах. Объясните способ неявного задания адресов операндов.

❷ Объясните назначение полей команд с тремя и двумя адресами.

❸ Как определяются адреса операндов и адрес результата в случае одноадресных команд?

❹ Объясните, как будут выполняться следующие трехадресные команды:

a) `01 200 201 202;` *c)* `03 100 150 250;`

b) `04 202 201 200;` *d)* `02 250 300 310.`

Коды арифметических и логических команд приведены в предыдущем параграфе.

❺ Каким станет содержимое ячейки 100 после выполнения команды

```
01 200 300 100,
```

если в ячейки 200 и 300 записаны числа 17 и 31 соответственно?

❻ Объясните, как будут выполнены следующие двухадресные команды:

a) `01 200 201;` *c)* `03 100 150;`

b) `04 202 201;` *d)* `02 250 300.`

❼ Каким станет содержимое ячейки 200 после выполнения команды

```
01 200 100,
```

если в ячейки 100 и 200 записаны числа 18 и 32 соответственно?

❽ Объясните, как будут выполнены следующие одноадресные команды:

a) `01 100;` *c)* `02 400;`

b) `03 200;` *d)* `04 150.`

❾ Каким станет содержимое аккумулятора после выполнения команды

```
01 100,
```

если перед выполнением команды в ячейке 100 было записано число 12, а в аккумуляторе – число 26?

❿ Назовите преимущества и недостатки форматов команд с тремя, двумя или с одним адресом.

6.3. Типы команд

Команды любого компьютера делятся на четыре группы:

– команды для обработки данных, которые осуществляют арифметические и логические операции над данными, определенными с помощью операндов;

– команды для передачи данных, которые пересылают информацию между регистрами и/или ячейками без ее изменения;

– команды перехода, которые в зависимости от результатов проверки некоторого условия изменяют последовательность выполнения команд программы;

– команды ввода-вывода, которые обеспечивают связь компьютера с внешней средой.

Команды для обработки данных обрабатывают данные, хранящиеся во внутренней памяти и регистрах процессора. Наиболее известными среди команд данной группы являются те, которые выполняют арифметические операции: *сложение, вычитание, умножение* и *деление.*

Логические команды типа *И*, *ИЛИ*, *НЕ* также являются командами для обработки данных и выполняются над отдельными битами двоичной информации. В класс команд для обработки данных включаются также и команды типа: *стирание* содержимого некоторой ячейки или определенного регистра, *дополнение* содержимого некоторой ячейки, *увеличение* на единицу (инкрементация) содержимого некоторого регистра и т.п. Наконец, к классу команд для обработки данных относятся команды *сдвига* двоичных слов, в которых часть адреса команды содержит целое число, определяющее количество позиций, на которое осуществляется сдвиг.

Команды для передачи данных пересылают информацию между ячейками внутренней памяти, между регистрами и между ячейками и регистрами без изменения передаваемой информации. В команде должен быть определен явным или косвенным образом адрес источника и адрес приемника передаваемой информации. Во время и после переноса информация источника остается неизменной. Наиболее часто употребляемые команды этой группы – те, с помощью которых содержимое произвольных ячеек памяти переносится в определенный регистр или аккумулятор, а также команда обратного переноса: из регистра – в ячейку внутренней памяти.

Команды перехода используются для изменения порядка выполнения команд. В обычном режиме команды любой программы анализируются и выполняются последовательно – точно в том же порядке, в котором они размещены в памяти. Этот порядок может быть изменен с помощью команд условного и безусловного перехода.

Команды условного перехода обеспечивают выбор определенной ветви, по которой будет продолжено выполнение программы в зависимости от проверяемого условия. Использование команд условного перехода дает программисту возможность принимать логические решения в ходе выполнения программы.

Команда безусловного перехода содержит адрес команды, на которую будет осуществлен переход и которая тем самым будет выполнена следующей.

Команды ввода-вывода обеспечивают связь компьютера с периферийным оборудованием. Устройства, с которыми будет осуществлена операция ввода-вывода, задаются в адресной части соответствующей команды. Как правило, команды этого вида содержат не только информацию о характере обмена (ввод или вывод), но и дополнительные параметры, необходимые для правильной работы периферийных устройств. В этих же командах определяются регистры или ячейки, в которые будут помещены или из которых будут взяты соответствующие данные.

Вопросы и упражнения

❶ Как классифицируются команды компьютера? Укажите назначение команд каждой группы.

❷ Приведите несколько примеров команд для обработки данных. Дайте оценку числа всевозможных команд для обработки данных.

❸ В чем назначение команд передачи данных? Оцените число всевозможных команд передачи данных.

❹ Когда и как используются команды перехода? Какие условия проверки могут анализироваться этими командами?

❺ Укажите назначение команд ввода-вывода. Какую информацию содержат эти команды?

6.4. Машинный язык и язык ассемблера

Для решения любой задачи в память компьютера необходимо загрузить соответствующую программу и данные, предназначенные для обработки. Команды программы и обрабатываемые данные записываются во внутреннюю память в виде последовательностей двоичных цифр, которые центральное устройство управления может извлекать и исполнять.

> *Программы, представленные в виде двоичных последовательностей, напрямую исполняемых компьютером, называются программами на языке машинных кодов (машинном языке).*

Для пользователя программа в машинных кодах может быть представлена в виде последовательностей двоичных цифр или, более компактно, – восьмеричных, десятичных или шестнадцатеричных чисел, записываемых в соответствующих ячейках памяти.

Разработка программ на языке машинных кодов является утомительной и неэффективной работой. Для упрощения процесса разработки программ договорились записывать команды на некотором символическом языке, называемом **языком ассемблера**. В данном языке коды операций представляются

группами символов так, чтобы они как можно лучше подсказывали смысл операции. Эта группа символов, длиной, как правило, три, известна под названием **мнемоники команды.**

Например, коды команд учебного компьютера из предыдущего параграфа могут быть обозначены символически в соответствии с *таблицей 6.1*.

Таблица 6.1.

Мнемоника команд

Код	*Мнемоника*	*Значение команды*
01	ЗАГ	Загрузить аккумулятор
02	ЗАП	Запомнить аккумулятор
03	СЛЖ	Сложение
04	ВЫЧ	Вычитание
05	БП	Безусловный переход
06	УП	Условный переход
07	СТОП	Стоп

Адреса ячеек внутренней памяти могут быть заданы с помощью символических обозначений, выбранных пользователем. Названия должны напоминать (подсказывать) назначение (смысл) содержимого соответствующих ячеек.

Например, ячейка 185, в которой хранится число x, может быть обозначена с помощью X; ячейку 213 для числа y обозначим через Y; ячейку 200, в которую будет помещена сумма $x + y$, обозначим через S. На языке ассемблера фрагмент программы для сложения чисел x и y примет форму:

```
ЗАГ X
СЛЖ Y
ЗАП S.
```

В общем случае существует прямое соответствие между записью команд на языке ассемблера и записью этих же команд на языке машинных кодов, которое облегчает трансляцию (перевод) программ на языке ассемблера в программы на языке машинных кодов.

> *Трансляция состоит в замене мнемонических команд и символических адресов на соответствующие двоичные последовательности. Такая замена осуществляется специальной программой, которую называют программой ассемблера или просто ассемблером.*

Язык машинных кодов и язык ассемблера относят к **машинно-зависимым языкам**. Такая зависимость состоит в том, что форматы, коды и мнемоника команд выражают (передают) внутреннюю структуру компьютера. Программы, написанные на этих языках, являются самыми короткими и быстрыми, но сам процесс программирования требует большого объема работы. Упрощение процесса программирования обеспечивается использованием **машинно-независимых языков** (*FORTRAN*, *BASIC*, *PASCAL*, *C* и т.д.), в которых операции обработки и типы данных не связаны с внутренним устройством конкретной модели компьютера. Однако, к сожалению, отрыв пользователя от внутренней структуры компьютера снижает эффективность соответствующих программ.

Вопросы и упражнения

❶ В чем разница между машинным языком и языком ассемблера?

❷ Как представляются коды команд и адреса ячеек на языке ассемблера?

❸ В чем назначение трансляции и как она реализуется для программ, написанных на языке ассемблера?

❹ Пусть символ *X* обозначает ячейку 100, символ *Y* – ячейку 101, а символ *S* – ячейку 102. Представьте на языке ассемблера (см. *табл. 6.1*) следующие программы:

a)
```
01 100
04 101
02 102
02 100
```

b)
```
01 100
02 100
03 100
```

c)
```
01 101
04 100
02 102
```

d)
```
01 101
03 101
03 101
03 101
02 102
```

e)
```
01 100
02 101
03 101
02 100
```

f)
```
01 100
02 102
01 101
02 100
```

Каким будет содержимое ячеек 100, 101 и 102 до и после выполнения каждой программы?

❺ Пусть символические обозначения *X*, *Y* и *S* определяют соответственно ячейки 100, 200 и 300. Транслируйте (вручную) следующие программы, написанные на языке ассемблера:

a)
```
ЗАГ X
ВЫЧ Y
ЗАП S
ЗАГ Y
```

b)
```
ЗАГ X
ЗАГ X
УП S
СТОП
```

c)
```
ЗАГ Y
СЛЖ X
ЗАП S
СТОП
```

d)
```
ЗАГ Y
СЛЖ Y
СЛЖ Y
СЛЖ Y
ЗАП S
```

e)
```
ЗАГ X
ЗАП S
ЗАГ Y
ЗАП X
ЗАГ X
```

f)
```
ЗАГ X
ЗАП S
ЗАГ Y
ЗАП X
ЗАГ S
```

Объясните, как будет выполнена каждая из программ.

❻ В чем разница между машинно-зависимыми и машинно-независимыми языками? Перечислите преимущества и недостатки каждого из названных типов языка.

6.5. Аппаратные и программные ресурсы компьютера

Общее число команд любого компьютера зависит, в первую очередь, от его мощности. В случае большой вычислительной системы это число может превысить 1000, в то время как для очень маленьких компьютеров оно не больше 100. Некоторая операция может быть выполнена на одних компью-

терах с помощью единственной команды, в то время как в других компьютерах, для которых нет такой команды, эта же операция выполняется с помощью некоторой последовательности имеющихся в наборе команд.

Операции, осуществляемые при помощи электронных компонентов компьютера, известны как **аппаратно-реализуемые операции**, в то время как операции, выполняемые с помощью некоторой последовательности команд, известны как **программно-реализуемые операции**. Например, операция извлечения квадратного корня в одном типе компьютеров может быть выполнена при помощи электронных компонентов, а в другом типе компьютеров – с помощью подпрограмм. Разделение на аппаратную и программную реализацию зависит от типа компьютера. На *рис. 6.4* представлено такое разделение для компьютера средней мощности.

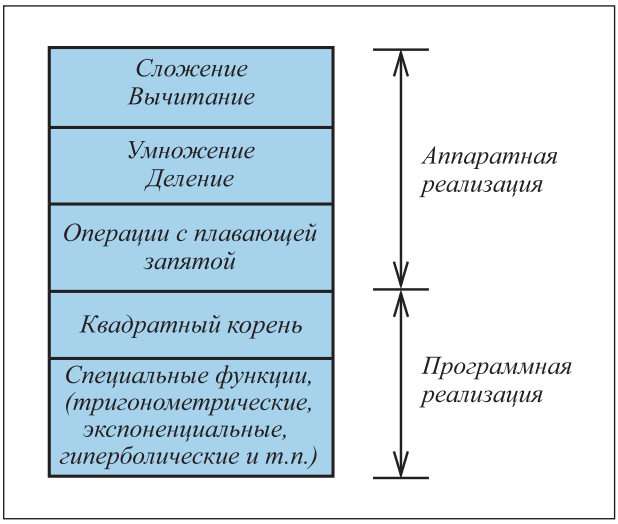


Рис. 6.4. Разделение между аппаратной и программной реализацией

Аппаратно-реализуемые операции выполняются при помощи одной команды, в то время как программно-реализуемые операции требуют большего числа команд. Следовательно, аппаратно-реализуемые операции выполняются быстрее, хотя соответствующий компьютер сложнее и, следовательно, дороже. Напротив, программно-реализуемые операции выполняются медленно, хотя соответствующие компьютеры проще и, очевидно, дешевле.

Из анализа принципов работы процессора следует, что все устройства любого компьютера становятся бесполезными в отсутствие программ, которые управляют ходом выполнения операций, необходимых для решения поставленных задач. Точно так же программы становятся бесполезными при отсутствии соответствующих цифровых устройств. Следовательно, использование вычислительной техники возможно только при наличии как оборудования, называемого **техническим обеспечением**, так и соответствующих программ, называемых **программным обеспечением**.

Техническое обеспечение любой современной вычислительной системы включает процессор, внутреннюю память, устройства внешней памяти, устройства ввода-вывода и т.п. **Программное обеспечение** включает подпрограммы

для выполнения программно-реализуемых операций, программы для доступа к устройствам ввода-вывода, ассемблеры, редакторы текста, компиляторы алгоритмических языков и, естественно, программы, разрабатываемые каждым пользователем.

Отметим, что в специализированной литературе техническое обеспечение иногда называют английским термином *hardware* (“металлические изделия”), а программное обеспечение – термином *software* (“мягкие изделия”). Соответственно реализация с помощью аппаратных средств называется реализацией с помощью *hardware*, а программная реализация – реализацией с помощью *software*.

Вопросы и упражнения

❶ От чего зависит общее количество команд произвольного компьютера?

❷ Как выполняются операции по обработке данных в случае их аппаратной и программной реализации?

❸ Изучите набор команд компьютера, на котором вы работаете, и определите метод реализации следующих операций:
- умножение и деление двоичных чисел;
- сложение и вычитание чисел с плавающей запятой;
- умножение и деление чисел с плавающей запятой;
- извлечение квадратного корня;
- вычисление тригонометрических функций.

❹ В чем преимущества и недостатки аппаратной реализации? А программной реализации?

❺ Из чего состоит техническое и программное обеспечение вычислительной системы? Какие ресурсы имеет компьютер, на котором вы работаете?

6.6. Внешняя память на магнитных лентах и дисках

Принцип работы рассматриваемых видов памяти состоит в регистрации (записи) информации на магнитном слое, находящемся в движении. Магнитный слой нанесен на нейтральный носитель, как правило, на ленту из гибкого материала или алюминиевый диск. В качестве магнитного слоя чаще всего применяется окись железа или тончайшие металлические пленки из кобальт-никелевого сплава, нанесенные (напыленные) в вакууме.

Запись и чтение информации осуществляются с помощью магнитной головки, изображенной на *рис. 6.5*.

Головка состоит из сердечника, собранного, как правило, из очень тонких пластин (0,05 мм) из пермаллоя, и обмотки.

В ненамагниченном слое магнитные поля частичек окиси железа ориентированы хаотично и взаимно компенсируют друг друга. Чтобы **записать** двоичную цифру 0 или 1, через обмотку магнитной головки пропускается соответствующий импульс тока. Импульс, проходящий по обмотке, создает в зазоре сильное магнитное поле, которое намагничивает слой, находящийся в данный момент времени под головкой. Направление намагничивания, а значит, и записанная двоичная информация зависят от направления тока в

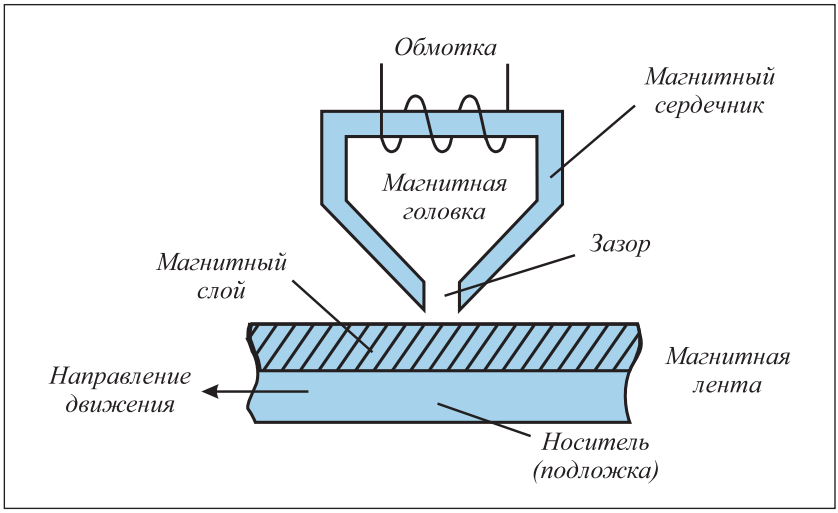


Рис. 6.5. Магнитная головка для записи и чтения информации

обмотке магнитной головки. На *рис. 6.6* приведен пример записи двоичной последовательности 101101 на подвижный магнитный слой.

Расстояние b определяет длину участка, необходимого для записи одной двоичной цифры. Его величина зависит от скорости движения носителя, физических свойств магнитного слоя, от конструкции магнитной головки и т.д.

> *Количество элементов двоичной памяти на единицу длины носителя называется плотностью записи информации.*

В случае магнитных носителей плотность записи задается величиной $1/b$. Конкретные значения плотности записи меняются в зависимости от устройств записи и фирмы производителя, находясь в пределах сотен и тысяч битов на миллиметр длины носителя.

Во время операции **чтения** магнитное поле частичек окиси железа, проходя мимо зазора магнитной головки (*рис. 6.5*), индуцирует в обмотке сигнал порядка 10^{-3} вольт. Этот сигнал усиливается и преобразовывается в стандартный сигнал, который представляет соответствующую двоичную цифру 0 или 1.

В большинстве случаев **внешняя память на магнитной ленте** представляет собой автономное периферийное устройство, которое передает информацию из/во внутреннюю память компьютера после приема соответствующих команд от процессора. Устройство памяти на магнитной ленте (*рис. 6.7*) состоит из механизма протяжки ленты, устройства записи-чтения и соответствующих схем управления.

Операция чтения или записи осуществляется во время перемещения ленты. Между двумя последовательными операциями записи/чтения лента останавливается. Очевидно, что записанная информация может быть прочитана только в порядке ее физического размещения на ленте. Вследствие этого устройства на магнитных лентах называются устройствами внешней памяти с **последовательным доступом.**

> *Время, необходимое для выбора требуемой информации из множества данных, записанных на некотором носителе, называется временем доступа.*

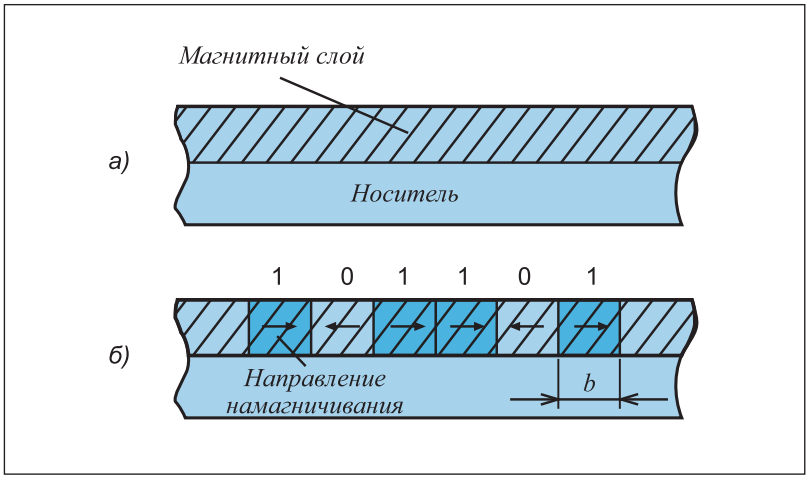


Рис. 6.6. Состояние магнитного слоя до (*а*) и после записи (*б*)

Время доступа устройства на магнитной ленте зависит от скорости движения ленты и места размещения читаемой информации: в начале, в конце или в середине ленты. В случае информации, записанной в конце ленты, время доступа может достигать нескольких минут.

Емкость памяти произвольной магнитной ленты зависит от плотности записи, количества дорожек, длины ленты и составляет порядка 10^8 байтов.

Из-за большого времени доступа и относительно малой емкости магнитные ленты используются в общем случае только для архивирования информации.

Устройство на магнитных дисках в настоящее время является самым распространенным типом внешней памяти цифровых компьютеров. Носитель информации составлен из набора дисков, которые могут быть фиксированными или сменными. Диски вращаются со скоростью порядка тысячи оборотов в минуту. Каждый диск покрыт слоем ферромагнитного материала. Для

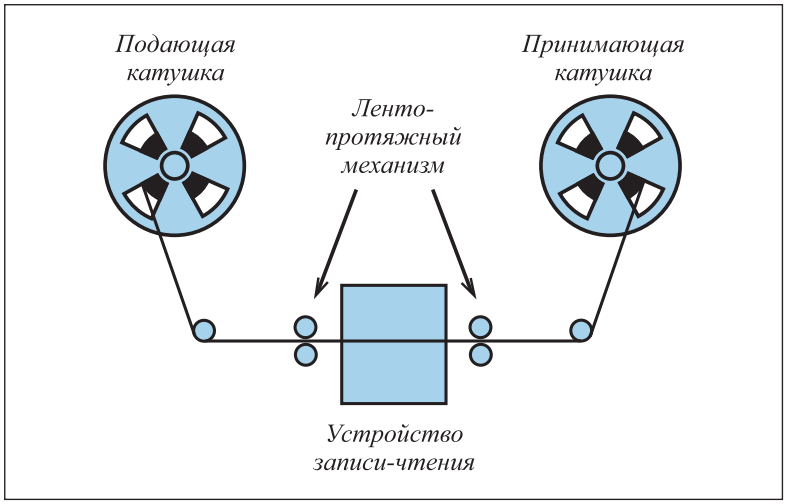


Рис. 6.7. Устройство памяти на магнитной ленте

записи и чтения информации над каждой поверхностью диска установлена отдельная магнитная головка (*рис. 6.8*).

Головки смонтированы на подвижном держателе, управляемом механизмом для точной установки на нужную дорожку. Все головки дискового блока памяти позиционируются одновременно.

Время доступа дисковых устройств складывается из времени, необходимого для перемещения набора магнитных головок от текущего к требуемому цилиндру (*рис. 6.8*), и из времени, необходимого для того, чтобы соответствующий сектор диска переместился прямо под магнитную головку. На практике пользуются **средним временем доступа**, которое для современных дисковых устройств составляет порядка 10^{-3} секунд.

Отметим, что в мощных компьютерах используются дисковые устройства с неподвижными магнитными головками – по одной головке на каждую дорожку. Указанные устройства обеспечивают время доступа порядка 10^{-4} секунды, однако они являются очень дорогими.

Для обмена информацией между компьютерами, используются одиночные диски из гибкого материала, которые называются **гибкими дисками** или **дискетами**. Дискета помещается в кассету из пластмассы или в особый конверт. Физическая организация данных на дискетах такая же, как и для пакетов из дисков, однако соответствующие устройства значительно проще, а значит, и дешевле. Для того чтобы отличать их от гибких дисков, внутренние диски персональных компьютеров называются **жесткими дисками**, ***hard*-дисками** или **винчестерами (*winchester*)**.

Емкость памяти пакета из магнитных дисков зависит от количества дисков, количества цилиндров и плотности записи. В настоящее время достигнута емкость порядка 10^{12} байтов для одного диска.

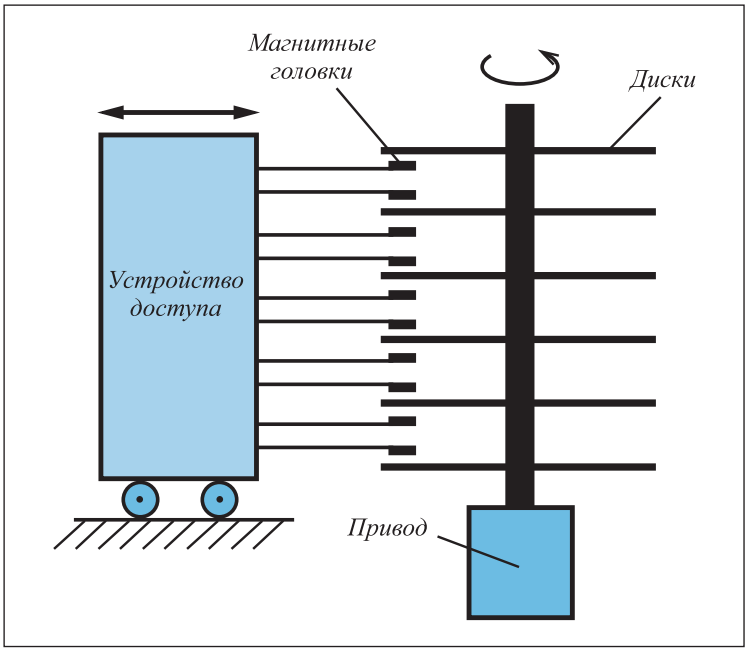


Рис. 6.8. Дисковое устройство с подвижными головками

Вопросы и упражнения

❶ Как представляются двоичные цифры 0 и 1 при магнитной записи?

❷ Для чего предназначена магнитная головка?

❸ От чего зависит плотность магнитной записи информации?

❹ Как считывается информация, записанная на магнитном слое?

❺ Объясните, как работает устройство памяти на магнитной ленте, изображенное на *рис. 6.7*.

❻ От чего зависит емкость памяти магнитной ленты?

❼ Магнитная лента имеет длину 750 *м*. Запись информации осуществляется на 8 дорожках плюс одна дорожка для бита четности. Объем записанной информации составляет 47 *МБ*. Определите плотность записи информации на магнитной ленте.

❽ Скорость магнитной ленты равна 2 *м/с*. На подающей катушке (*рис. 6.7*) имеется 750 *м* ленты. Определите время доступа к данным, которые находятся в середине ленты.

❾ Как работает устройство на магнитных дисках с подвижными головками?

❿ Как организована информация на пакете магнитных дисков?

⓫ В чем разница между устройствами внешней памяти с прямым и последовательным доступом?

⓬ От чего зависит время доступа устройств на магнитных дисках?

⓭ Определите емкость дискеты, с которой вы работаете.

⓮ Для жесткого диска, с которым вы работаете, определите:
– емкость диска;
– среднее время доступа.

6.7. Внешняя память на оптических дисках

Принцип работы устройств **памяти на оптических дисках** состоит в хранении информации на движущемся отражающем слое. Отражающий слой из алюминия, золота или серебра нанесен на прозрачную основу из пластмассы.

В зависимости от режима записи и чтения информации различаем:

1) Оптические диски **только для чтения**. Информация на такие диски записывается производителем и не может быть изменена пользователем. Для таких дисков используется английское обозначение *CD-ROM* (***C**ompact **D**isc – **R**ead **O**nly **M**emory*).

2) **Записываемые** оптические диски. Информация на такие диски записывается самим пользователем, но только один раз. В дальнейшем такой диск может использоваться многократно только для чтения. Английское обозначение таких дисков *CD-R* (***C**ompact **D**isc – **R**ecordable*).

3) **Перезаписываемые** оптические диски. Рассматриваемые диски допускают многократные циклы записи-чтения информации и обозначаются как *CD-RW* (***C**ompact **D**isc – **R**e**W**ritable*).

Для обеспечения совместимости устройств записи/чтения формат данных и размеры оптических дисков стандартизованы. На *рис. 6.9* представлена

структура оптического диска *CD-ROM*, предназначенного для широкого круга пользователей.

На таких дисках записанные двоичные цифры представляют собой последовательность углублений (на английском *pit*) на одной из поверхностей диска. Углубления размещены на поверхности с небольшими перерывами, а их последовательность образует дорожку спиральной формы.

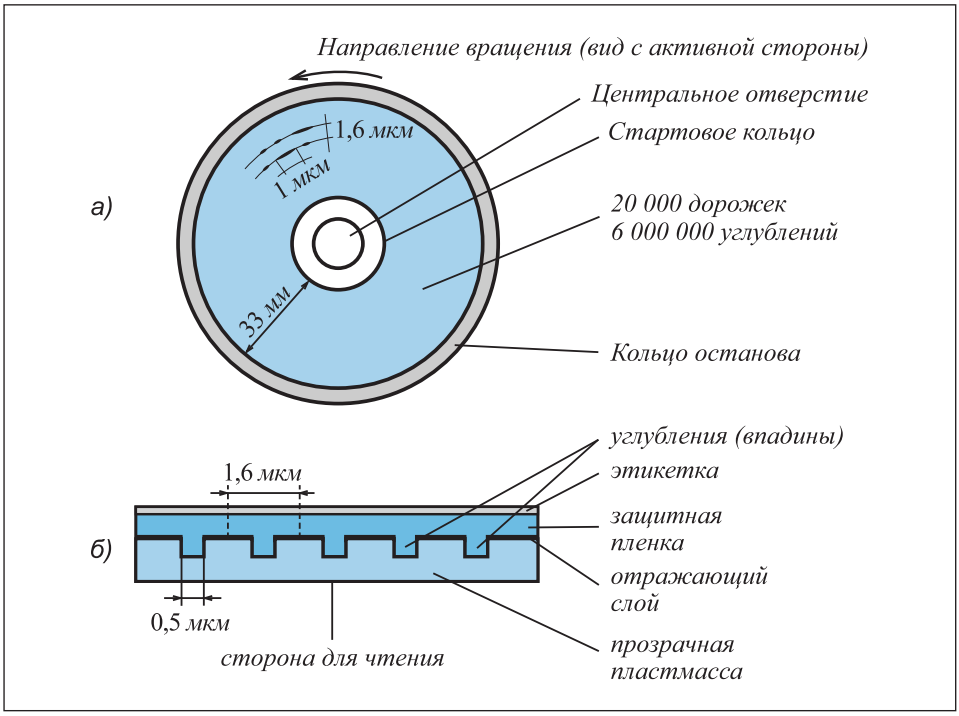


Рис. 6.9. Структура оптического диска *CD-ROM*: *а* – расположение дорожек на диске; *б* – вид на диск в разрезе (перпендикулярно дорожкам)

Размеры углублений имеют порядок одного микрона (1 *мкм* = 10^{-3} *мм*), расстояние между витками спирали 1,6 *мкм*, длина спирали 5 300 *м*. Диск содержит 20 000 дорожек (витков), на которых находится около $6 \cdot 10^9$ углублений. Емкость памяти диска равна 640 *Мегабайтам*.

Чтение оптического диска осуществляется с помощью лазерного луча, который, отразившись от активной поверхности, попадает на фоточувствительную ячейку (*рис. 6.10*).

Проходя вдоль соответствующих дорожек, лазерный луч отражается, когда свет попадает в **точку фокусировки** и рассеивается в противном случае. Другими словами, углубления на рабочей поверхности оптического диска изменяют (модулируют) интенсивность отраженного пучка. Вследствие этого на выходе фотоячейки формируется сигнал, который воспроизводит последовательности двоичных цифр 0, 1, записанных на диск на этапе его создания.

В современных устройствах памяти на оптических дисках источник света – лазер и фоточувствительная ячейка – фотодиод выполняются в едином блоке, именуемом **оптической головкой считывания**. Угловая скорость диска

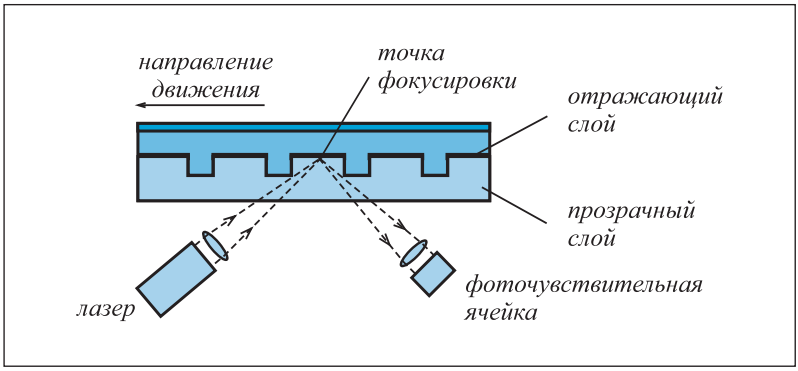


Рис. 6.10. Чтение оптических дисков

составляет 200 – 600 оборотов в минуту, а линейная скорость – 1,4 *м/с*. Несмотря на очень большие скорости, оптический диск практически не изнашивается, так как между оптической головкой и диском отсутствует прямой механический контакт. Вследствие этого длительность эксплуатации диска *CD-ROM* определяется качеством отражающего и защитного слоев. При использовании алюминия из-за окисления отражающий слой темнеет, ограничивая срок эксплуатации такого диска 10 – 15 годами. При использовании отражающего слоя из золота только преднамеренные механические воздействия, разрушающие защитный слой, могут повлиять на качество оптического диска.

Как правило, оптические диски *CD-ROM* используются для тиражирования операционных систем, трансляторов, энциклопедий, электронных игр, а также другой информации, предназначенной для очень большого числа пользователей.

Структура *CD-R* и *CD-RW* **оптических дисков** приведена на *рис. 6.11*.

На *рис. 6.11* видим, что рассматриваемые диски содержат особый слой, называемый **слоем для записи**. В случае ***CD-R* дисков** данный слой состоит из специального органического материала – *цианина* или *фталоцианина*, который темнеет при нагреве. Запись информации осуществляется с помощью мощного лазерного луча, вызывающего потемнение соответствующих участков слоя для записи.

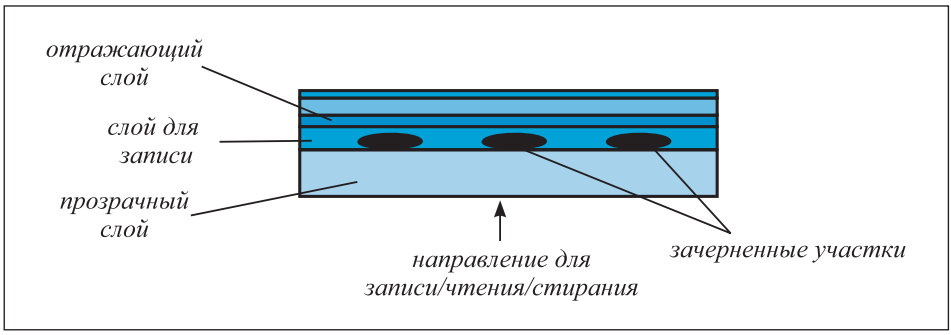


Рис. 6.11. Структура *CD-R* и *CD-RW* оптических дисков

При чтении затемненные области блокируют прохождение лазерного луча к отражающему слою, изменяя, таким образом, интенсивность света, попадающего на фоточувствительную ячейку (*рис. 6.10*). Поскольку лазерный луч, используемый для чтения, является слабым, ранее записанный диск можно читать многократно без разрушения хранимой на нем информации.

В случае ***CD-RW* дисков** материал слоя для записи подбирается так, чтобы он вновь становился прозрачным при нагреве до особой температуры, называемой **критической температурой**. Очевидно, что после стирания на перезаписываемый диск можно записывать новые двоичные данные. Современные *CD-RW* диски выдерживают до 10000 циклов записи/стирания.

Общим недостатком записываемых и перезаписываемых оптических дисков является их повышенная чувствительность к температуре. Продолжительность их эксплуатации меньше, чем *CD-ROM* дисков. Такие диски дороже, поскольку их отражающий слой выполнен из серебра или золота.

Обычно *CD-R* и *CD-RW* диски используются для распространения оперативной информации, предназначенной для определенного круга пользователей – баз данных, специализированных пакетов программ, отчетов особо важных симпозиумов, картин со знаменитых выставок, мультимедийных документов и т.п.

Отметим, что в результате непрерывного развития компьютерных технологий, в последние годы широкое распространение получили *DVD* оптические диски (Digital Versatile Disc – Цифровой Многоцелевой Диск). Емкость DVD диска примерно в семь раз больше, чем емкость CD диска.

Вопросы и упражнения

❶ Как представляются двоичные цифры 0, 1 при оптической записи?

❷ Укажите емкость памяти оптического диска.

❸ Как записывается информация на оптический диск *CD-ROM*?

❹ От чего зависит емкость памяти оптического диска?

❺ Известно, что длина спирали, вдоль которой записывается информация оптического диска, равна 5300 *м*. Емкость памяти диска составляет 640 *Мбайт*. Определите плотность записи информации на оптическом диске.

❻ В каких областях применяются *CD-ROM* диски?

❼ От чего зависит время доступа к информации на оптическом диске?

❽ Линейная скорость оптического диска составляет 1,4 *м/с*. Зная плотность записи информации d = 128 *Кбайтов/м*, определите скорость передачи данных от дискового устройства к центральному устройству. Напомним, что скорость передачи данных измеряется в *битах*, *Кбитах* или *Мбитах* в *секунду*.

❾ На 20000 дорожках (витках) оптического диска записано около 640 *Мбайтов* информации. Сколько информации содержит одна дорожка оптического диска?

❿ На одном *CD-ROM* диске записано в двоичном коде около 74 *минут* музыки. Определите, сколько информации содержит одна песня длительностью 4 *минуты* 30 *секунд*. Сколько дорожек занимает соответствующая песня?

⓫ Как считывается информация с оптического диска? В чем назначение оптической головки считывания?

⓬ Объясните назначение слоев записываемого оптического диска. Как осуществляется запись информации на рассматриваемый диск?

⓭ От чего зависит длительность эксплуатации *CD-ROM* диска? Дисков *CD-R* и *CD-RW*?

⓮ Как записывается и стирается информация на перезаписываемом оптическом диске?

⓯ В каких областях применяются *CD-R* и *CD-RW* диски?

⓰ Найдите технические параметры дискового оптического устройства, установленного на вашем компьютере. Выучите правила эксплуатации дисков и устройств внешней памяти на оптических дисках.

6.8. Видеомонитор и клавиатура

Видеомонитор – это устройство вывода, с помощью которого информация отображается на экране электронно-лучевой трубки. Функциональная схема видеомонитора представлена на *рис. 6.12*.

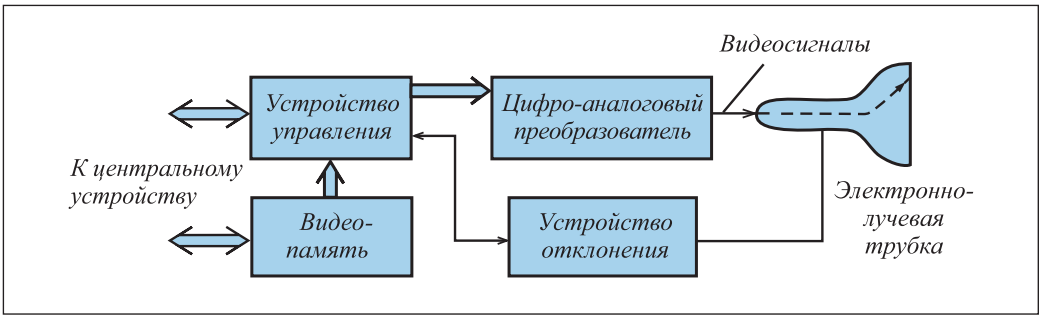


Рис. 6.12. Функциональная схема видеомонитора

Как и в случае обычных телевизоров, изображение на экране электронно-лучевой трубки формируется из точек. Точки светятся под воздействием электронного луча. Цвет и яркость каждой точки задаются видеосигналами, поданными на соответствующие входы электронно-лучевой трубки. Обход точек в заранее установленном порядке, как правило, по строкам слева направо и сверху вниз, осуществляется при помощи устройства отклонения электронного луча.

Каждой точке на экране соответствует одна ячейка в памяти видеомонитора. Ячейки видеопамяти содержат информацию относительно цвета и яркости соответствующих точек на экране электронно-лучевой трубки. Управляющее устройство читает ячейки видеопамяти в порядке обхода точек на экране. Содержимое каждой ячейки интерпретируется как команда, задающая цвет и яркость соответствующей точки. Видеосигналы, необходимые для управления электронным лучом, формируются цифро-аналоговыми преобразователями.

Данные в видеопамяти монитора записываются центральным устройством компьютера. В любой момент времени компьютер может изменить содержание видеопамяти. Как следствие, изменяется и изображение на экране видеомонитора. Изображения можно сделать движущимися путем изменения содержимого видеопамяти с частотой, используемой в кинематографии.

В общем случае видеомонитор может функционировать в одном из двух режимов: текстовом или графическом.

В текстовом режиме экран делится на условные зоны, каждая из которых называется **знакоместом**. Как правило, эти зоны образуют 25 строк по 80 символов в каждой. На каждом знакоместе может быть отображен любой символ из общего набора в 256 символов. Набор символов состоит из прописных и строчных букв латинского алфавита, десятичных цифр, математических символов, знаков пунктуации, букв национальных алфавитов и некоторых псевдографических символов, используемых для вывода на экран таблиц, диаграмм, рамок и т.п. Каждое знакоместо может иметь отдельные цвета для символа и фона, на котором он изображен. Это позволяет выводить на экран текст с разноцветными буквами.

В графическом режиме пользователь может управлять выводом на экран каждой точки в отдельности. Количество точек по горизонтали и вертикали определяет разрешение монитора. Например, выражение "разрешение 640×200" означает, что видеомонитор отображает 640 точек по горизонтали и 200 – по вертикали.

Существует множество международных стандартов, которые регламентируют разрешение и количество цветов видеомониторов. Эти характеристики являются общепринятыми и соблюдаются фирмами-производителями.

Например, в случае персональных компьютеров самые распространенные стандарты называются *EGA*, *VGA* и *SVGA*. Стандарт *EGA* устанавливает, что видеомонитор имеет разрешение 640×350 точек, допуская использование 64 цветов.

Стандарт *VGA*, сохраняя совместимость с *EGA*, предлагает как дополнительную характеристику разрешение 640×480 точек при 256 различных цветах.

Для улучшения качества изображения стандарт *SVGA* дополнен разрешением 1024×768 точек.

Отметим, что технические достижения последних лет сделали возможной замену электронно-лучевых трубок на плоские экраны. Это привело к существенному уменьшению размеров мониторов и значительному улучшению качества изображения. Такие мониторы характеризуются уменьшенным энергопотреблением и разрешением до 1920×1200 точек.

Клавиатура – это устройство ввода, которое преобразует нажатие клавиш в двоичные слова, воспринимаемые компьютером.

Электронная часть любой клавиатуры состоит из шифратора. На входы шифратора подаются логические сигналы, формируемые при нажатии клавиш. На выходе появляется слово из определенного двоичного кода, стандартного для каждого семейства компьютеров (*ISO*, *ACSII* и т.п.). Некоторые виды клавиатур снабжаются аудио-генератором, который при нажатии клавиш издает специфический звук.

У нас, как и в англоязычных странах, самой распространенной является клавиатура типа (с раскладкой) *QWERTY*, название которой происходит от расположения символов *Q*, *W*, *E*, *R*, *T* и *Y* в верхнем ряду алфавитно-цифровых клавиш. Во франкоговорящих странах используется клавиатура *AZERTY*, в Германии – *QWERTZ* и т.п.

Несмотря на то, что раскладка клавиш и их количество могут отличаться, назначение основных клавиш на всех клавиатурах одинаково.

Клавиши делятся на следующие группы: алфавитно-цифровые, функциональные и специальные. Группа алфавитно-цифровых клавиш включает клавиши десятичных цифр, клавиши символов английского и русского алфавитов, клавиши математических символов и знаков пунктуации. Группа функ-

циональных клавиш включает клавиши <*F1*>, <*F2*>, ... ,<*F12*>. Данные клавиши не имеют предопределенного назначения и их использование зависит от программы, выполняемой на компьютере. Специальные клавиши применяются для управления положением курсора, для ввода в компьютер двоичных слов, не имеющих отдельных клавиш, и т.п.

В мощных компьютерах видеомонитор и клавиатура могут образовывать единое устройство, именуемое **консолью**. Консоль, применяемая для управления вычислительной системой, называется **монитором**.

Вопросы и упражнения

❶ Объясните, как работает видеомонитор. Назовите назначение основных частей видеомонитора.

❷ Как можно изменить изображение, выводимое на экран видеомонитора? Как “оживить” изображение на экране?

❸ Чем отличаются режимы работы видеомонитора?

❹ Перечислите основные показатели качества видеомонитора.

❺ Определите тип видеомонитора, на котором вы работаете. Определите разрешение экрана и количество доступных цветов.

❻ Укажите составные части клавиатуры. Как определить тип клавиатуры?

❼ Назовите группы клавиш и их назначение.

6.9. Принтеры

Принтеры – это устройства вывода, которые предоставляют результаты в виде отпечатанного документа. В зависимости от используемого **принципа печати** принтеры классифицируются на:

– механические принтеры, в которых печать осуществляется с помощью молоточков или иголок;

– лазерные принтеры, в которых печать осуществляется с использованием электро-статических методов, как в копировальных аппаратах;

– струйные принтеры;

– термопринтеры, работа которых основывается на использовании специальной бумаги, меняющей цвет при нагреве.

Принцип работы **матричного игольчатого принтера** представлен на *рис. 6.13*.

Печатающая головка содержит группу тонких металлических иголок, которые в нужный момент ударяют через красящую ленту по бумаге. Конфигурация ударяющих игл в каждый из моментов времени и продвижение головки вдоль линии печати определяет печатаемые изображения.

Матричные принтеры могут работать как в графическом, так и в алфавитно-цифровом (текстовом) режиме.

В **графическом режиме** компьютер управляет печатью каждой точки в отдельности. Ясно, что из точек могут быть сформированы любые изображения: графики, эскизы, а также символы, разработанные пользователем. Печать информации в графическом режиме является очень медленной из-за

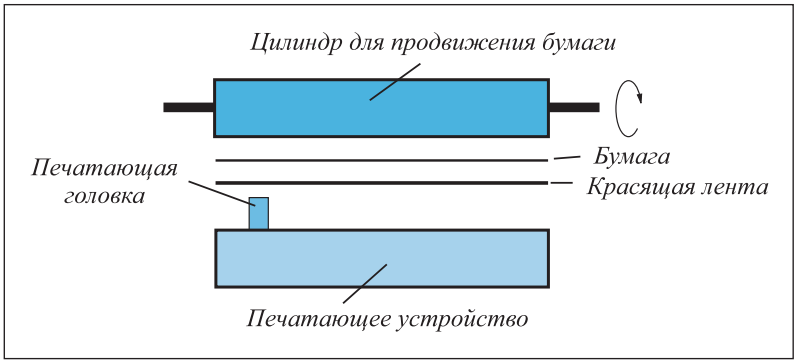


Рис. 6.13. Принцип работы матричного игольчатого принтера

большого количества перемещений печатающей головки и пошаговой подачи бумаги.

В **алфавитно-цифровом режиме** компьютер посылает печатающему устройству только коды печатаемых символов. Каждому коду соответствует изображение из точек, хранящееся в специальной памяти принтера. Соответствующие изображения печатаются за один или, самое большее, два-три прохода печатающей головки.

Печать в алфавитно-цифровом режиме выполняется быстрее, однако можно печатать только те символы, изображения которых записаны в памяти принтера. У простых матричных принтеров есть несколько стандартных наборов символов, записанных в постоянную память. Более производительные матричные принтеры обеспечивают возможность программной загрузки множества наборов символов, конфигурация которых задана пользователем.

Отметим, что чем больше количество игл в печатающей головке, тем выше качество печати. В настоящее время используются принтеры с 9 или с 24 иглами. Качество печати можно улучшить с помощью повторной печати (2 – 4 раза) одного и того же символа на том же месте. Скорость печати механических игольчатых принтеров составляет 150 – 500 символов в минуту.

Принцип работы **лазерных принтеров** представлен на *рис. 6.14*.

Главным элементом такого принтера является барабан, покрытый полупроводящим слоем, который изменяет свои электрические свойства под воздействием света.

При печати текущей страницы сперва электризуется поверхность барабана. Далее с помощью лазерного луча на заряженную поверхность барабана проецируются точки печатаемого изображения. Поскольку освещенные участки изменяют свою электрическую проводимость, соответствующие заряды нейтрализуются. Следовательно, на поверхности барабана формируется невидимое электрическое изображение. Проявление изображения осуществляется с помощью очень мелких частиц красящего порошка, притягиваемых заряженными участками барабана. Изображение с барабана переносится на бумагу и закрепляется путем нагрева.

Далее все электрические заряды с поверхности барабана нейтрализуются, а остатки порошка удаляются.

Лазерные принтеры обладают самыми лучшими характеристиками среди современных принтеров. Тексты и графика, отпечатанные с помощью таких

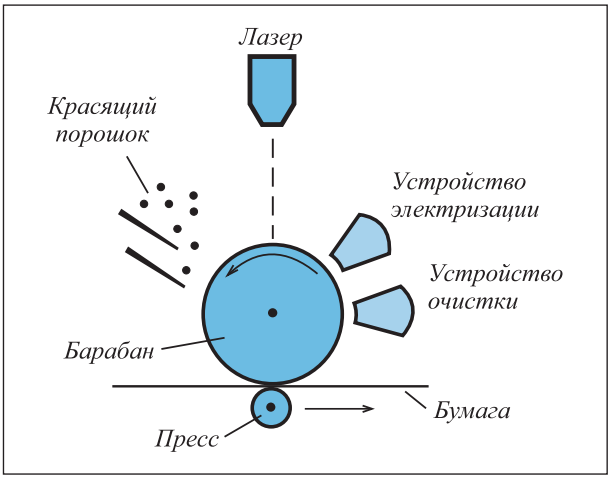


Рис. 6.14. Принцип действия лазерного принтера

принтеров, не отличаются от типографских. Скорость печати составляет 5 – 15 страниц в минуту.

Струйные чернильные принтеры формируют точки печатаемого изображения из микроскопических капель, наносимых на бумагу через специальные сопла. Они обеспечивают очень хорошее качество и применяются для цветной печати. Данные принтеры дороже, чем механические и требуют особого технического ухода.

Термопринтеры используют матрицу из игл, селективный кратковременный нагрев которых осуществляется в зависимости от того, какой символ должен быть напечатан. Скорость печати у них относительно низка, около 300 строк в минуту, и их основное преимущество состоит в малых размерах.

Вопросы и упражнения

❶ Как классифицируются принтеры в зависимости от принципа печати?

❷ Какие узлы входят в состав любого принтера?

❸ В чем принцип действия матричного игольчатого принтера? Как работает такой принтер?

❹ Объясните, как работает лазерный принтер. В чем главное преимущество лазерного принтера?

❺ Определите тип принтера, которым пользуетесь вы. Найдите технические параметры принтера: набор символов, режимы функционирования, емкость буферной памяти, скорость печати.

6.10. Классификация компьютеров

Характеристика любого компьютера включает следующие показатели:
– скорость выполнения операций;
– емкость внутренней памяти;

– состав, емкость и время доступа устройств внешней памяти;

– состав и соответствующие технические параметры периферийного оборудования;

– масса и габариты;

– стоимость.

В зависимости от этих параметров современные компьютеры делятся на 4 категории:

– суперкомпьютеры;

– большие компьютеры;

– миникомпьютеры;

– микрокомпьютеры (персональные компьютеры).

Суперкомпьютеры могут выполнять около 10^{15} (1000 триллионов) операций в секунду, а их цена превышает 20 миллионов долларов. Исследовательские и конструкторские разработки в области суперкомпьютеров осуществляются в США и Японии фирмами *IBM, Gray Research*, *Fujitsu ETA Systems*, *Sutherland* etc. Суперкомпьютеры используются в чрезвычайно сложных системах обработки данных: в аэронавтике, ядерной физике, астронавтике, сейсмологии, при прогнозах погоды и т.п.

Большие компьютеры (также мэйнфрейм, от английского mainframe – „основной шкаф") могут выполнять сотни миллионов операций в секунду, их цена лежит в пределах от 20 тысяч до нескольких миллионов долларов. Как правило, большие компьютеры включают в свой состав десятки устройств на магнитных дисках, десятки принтеров, сотни консолей, находящихся на различных расстояниях от центрального устройства. Данные компьютеры используются в больших вычислительных центрах и работают в круглосуточном режиме. Главные фирмы-производители больших компьютеров – *IBM*, *Hitachi, Amdahl, Fujitsu* и др.

Миникомпьютеры выполняли десятки и сотни триллионов операций в секунду, а их цена не превышало 200 – 300 тысяч долларов. Периферийное оборудование одного миникомпьютера включало в себя несколько магнитных дисков, один или два принтера, множество консолей. Миникомпьютеры были более простыми в обращении, чем большие компьютеры. Они применялись в системах автоматизированного проектирования, в промышленной автоматике, для обработки данных в научных экспериментах и т.п. Среди фирм, которые производили миникомпьютеры, отметим такие, как *IBM*, *Wang*, *Texas Instruments*, *Data General*, *DEC, Hewlett-Packard* и т.п. В настоящее время миникомпьютеры полностью вытеснены персональными компьютерами.

Микрокомпьютеры, называемые также и **персональными компьютерами**, продаются по относительно низким ценам – между 100 и 15000 долларов и обеспечивают скорость вычислений порядка миллиарда операций в секунду. Блок-схема персонального компьютера представлена на *рис. 6.2*.

Как правило, периферийное оборудование персонального компьютера включает в себя устройство на жестком диске, устройство на гибких дисках, устройство на оптических дисках, принтер и консоль. Модульная конструкция и группирование всего оборудования вокруг одной магистрали для передачи данных обеспечивает возможность реконфигурации микрокомпьютера в зависимости от индивидуальных потребностей каждого пользователя.

Корпорации, производящие микрокомпьютеры, находятся в очень многих странах. Современные всемирно признанные лидеры – это фирмы *IBM*, *Apple*, *Hewlet Packard*, *Dell*, *Asus*, *Acer* и т.п.

Вопросы и упражнения

❶ Назовите основные параметры, характеризующие современный компьютер. Определите технические и экономические параметры компьютера, на котором вы работаете.

❷ Как классифицируются компьютеры в зависимости от их технических и экономических параметров?

❸ Дайте краткую характеристику каждой категории компьютеров: суперкомпьютеров, больших компьютеров, мини- и микрокомпьютеров.

❹ Используя один из поисковых серверов, найдите в Интернете подробную информацию об основных производителях персональных компьютеров.

6.11. Микропроцессор

Микропроцессор – это интегральная схема, которая выполняет функции центрального устройства обработки информации – выборки и исполнения команд.

Как правило, микропроцессор содержит арифметическое и управляющее устройства, группу регистров, предназначенных для временного хранения часто используемых данных, магистрали и соответствующие схемы управления (*рис. 6.15*).

Микропроцессор взаимодействует с устройствами памяти и периферийными устройствами с помощью трех магистралей: *Данные*, *Адреса* и *Команды*. Поток

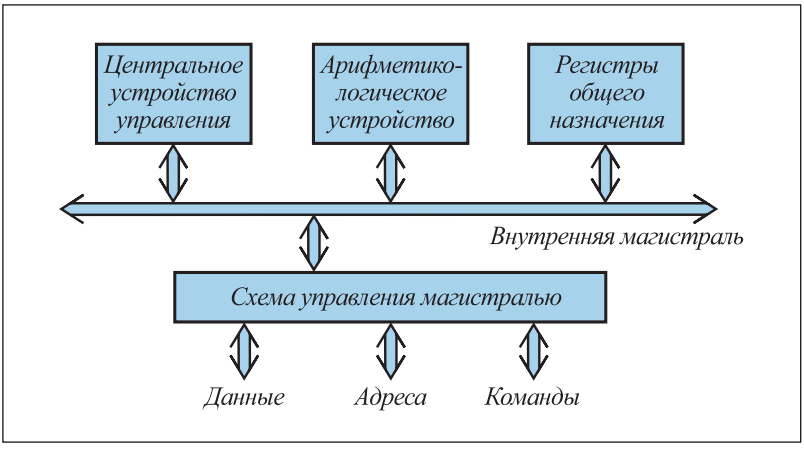


Рис. 6.15. Функциональная схема микропроцессора

информации через магистрали контролируется *Схемой управления магистралью*.

Выборка и исполнение команд происходит под управлением **центрального устройства управления**. Для этого с помощью магистрали адреса указывается адрес ячейки внутренней памяти, а с помощью магистрали команд передаются сигналы для записи или чтения. Данные для чтения или записи передаются по магистрали данных.

Микропроцессоры характеризуются следующими параметрами:

– длина слова;

– частота системных часов;
– емкость магистралей.

Длина слова представляет количество битов двоичной последовательности, которая может быть записана в регистрах и обработана арифметическим устройством микропроцессора. Большинство микропроцессоров имеют длину слова 32, 64 и более бита.

Частота системных часов представляет собой количество импульсов в секунду, вырабатываемых генератором тактовых импульсов, входящего в состав центрального устройства управления. Частота системных часов измеряется в **Мегагерцах** (1 *МГц* = 10^6 *Гц*). Поскольку тактовые импульсы синхронизируют выполнение микроопераций во всех устройствах микропроцессора, частота системных часов характеризует быстроту микропроцессора.

Емкость магистрали представляет собой количество разрядов двоичных слов, передаваемых по магистрали. Желательно, чтобы емкость магистрали данных была равна или больше длины слова микропроцессора. В противном случае для того, чтобы передать одно слово, необходимо несколько циклов магистрали данных.

Емкость магистрали адреса определяет пространство адресов, которое напрямую доступно микропроцессору. Так, микропроцессор с емкостью магистрали адреса в 16 *бит* имеет доступ к 2^{16} ячейкам внутренней памяти, а микропроцессор с емкостью магистрали адресов в 32 *бит* может адресовать напрямую 2^{32} ячеек.

Длина слова, частота системных часов и емкость магистралей определяют **производительность микропроцессора**, которая измеряется в ***Mips*** – миллионов команд в секунду. Для примера в *таблице 6.2* представлены главные характеристики микропроцессоров из семейства *Intel*.

Таблица 6.2.

Главные характеристики микропроцессоров из семейства Intel

Микропроцессор	Длина слова, *бит*	Частота, *МГц*	Производительность, *Mips*
Pentium I	32	200	75
Pentium II	32	300	200
Pentium III	32	1400	400
Pentium 4	32	3800	1200
Pentium D	64	3400	2500

Тип микропроцессора и частоту системных часов персонального компьютера с которым вы работаете можно узнать с помощью контекстного меню пиктограммы „My computer". Напоминаем, что контекстное меню выводится на экран с помощью щелчка правой кнопкой мыши по соответствующей пиктограмме.

Вопросы и упражнения

❶ Объясните назначение устройств, входящих в состав микропроцессора *(рис. 6.15)*.

❷ Назовите главные параметры микропроцессора. Объясните смысл каждого параметра.

❸ Определите тип и главные параметры микропроцессора из состава компьютера, на котором вы работаете.

Тест для самопроверки № 6

1. Укажите соответствие между названиями функциональных блоков компьютера (из левого столбца) и описанием их назначения (из правого столбца):

	(a) выполнение элементарных арифметических и логических операций;
	(b) раскраска изображений с последующей записью на оптических дисках;
	(c) извлечение данных из компьютера;
	(d) выдача сигналов управления, необходимых для последовательного выполнения команд;
(1) устройство ввода;	(e) ввод документов в память компьютера и исправление грамматических ошибок;
(2) память;	
(3) арифметико-логическое устройство;	(f) хранение как исходных, промежуточных и конечных данных задачи, так и команд, определяющих порядок вычислений;
(4) устройство вывода;	
(5) устройство управления;	(g) автоматическая обработка информации в соответствии с программой, хранящейся в памяти;
(6) процессор;	
	(h) выполнение арифметических вычислений и отображение полученных результатов на экране;
	(i) ввод данных из внешней среды в компьютер.

2. Назовите пять периферийных устройств.

3. Какие из следующих утверждений являются истинными?

a) компьютер может быть создан и без использования устройства внешней памяти;

b) внутренняя память является периферийным устройством;

c) внешняя память является более медленной, чем внутренняя память;

d) устройство на магнитных дисках является внутренней памятью;

e) компьютер может быть создан и без использования устройства внутренней памяти;

f) внешняя память является периферийным устройством.

4*. Расскажите о назначении магистрали персонального компьютера.

5*. Нарисуйте блок-схему персонального компьютера. Какие компоненты являются обязательными, а какие не являются обязательными для работы компьютера?

---

* Только для реального профиля.

6*. Известно, что трехадресные команды содержат следующие поля: *Код операции, Адрес операнда 1, Адрес операнда 2* и *Адрес результата*. Объясните назначение каждого из этих полей.

7*. Объясните, как выполнятся следующие трехадресные команды:

a) 01 101 153 342;

b) 04 508 391 216;

c) 03 751 852 031;

d) 02 450 709 011.

Арифметические и логические операции закодированы следующим образом: 01 – сложение; 02 – вычитание; 03 – умножение; 04 – деление.

8*. Определите тип (обработки, передачи, перехода, ввода-вывода) следующих команд:

a) сложение двух чисел;

b) запись последовательности байтов на магнитный диск;

c) сдвиг двоичного слова справа налево;

d) считывание последовательности байтов с оптического диска;

e) сравнение двух чисел и передача управления в зависимости от результата сравнения;

f) умножение двух чисел;

g) загрузка в регистр процессора определенного числа из внутренней памяти.

9*. Пусть символьное имя *X* обозначает ячейку 205, имя *Y* – ячейку 421, а имя *S* – ячейку 783. Представьте на языке ассемблера (см. *таблицу 6.1*) следующую программу:

```
01 205
02 783
04 421
03 205.
```

10*. Пусть символьные имена *X*, *Y* и *S* определяют, соответственно, ячейки 971, 583 и 461. Оттранслируйте следующую программу, записанную на языке ассемблера (см. *таблицу 6.1*):

```
ЗАГ Y
ЗАП S
ЗАГ X
ЗАП Y
ЗАГ S
```

11. Укажите тип (технический или программный) следующих вычислительных ресурсов:

a) процессор;

b) редактор текстов;

c) приложение для табличных вычислений;

d) принтер;

e) приложение электронной почты;

f) клавиатура;

g) операционная система;

h) монитор.

12. Чем отличаются устройства внешней памяти с произвольным и устройства внешней памяти с последовательным доступом?

13. Какие из следующих утверждений истинны?

a) устройство памяти на магнитной ленте обеспечивает прямой доступ к записям;

b) время доступа устройства на магнитной ленте больше, чем время доступа устройства на магнитных дисках;

c) устройство памяти на магнитных дисках обеспечивает последовательный доступ к записям;

d) на магнитной ленте записи сгруппированы по цилиндрам;

e) время доступа устройства на жестком диске меньше, чем время доступа устройства на гибких дисках.

14. Какие из следующих утверждений истинны?

a) Все оптические диски имеют следующий недостаток – однажды записанную на них информацию невозможно перезаписать.

b) В общем случае, емкость жестких магнитных дисков больше, чем емкость оптических дисков.

c) Емкость оптического диска зависит от скорости его вращения.

d) Емкость гибкого магнитного диска больше, чем емкость оптического диска.

e) Число циклов записи/стирания перезаписываемых оптических дисков ограничено.

15*. Линейная скорость оптического диска составляет 5,6 *м/с*, а плотность записи информации составляет 512 *Кбайт/м*. Определите скорость передачи данных от устройства на оптических дисках к центральному устройству.

16. Выберите из приведенного ниже списка параметры, которые характеризуют монитор:

a) разрешение;

b) количество страниц, которое может отобразить на экране за одну секунду;

c) диагональ экрана;

d) число цветов, которое можно отобразить на экране;

e) скорость печати.

17. Выберите из приведенного ниже списка параметры, характеризующие принтеры:

a) разрешение;

b) количество цветов, используемых при печати;

c) диагональ принтера;

d) количество команд в секунду;

e) скорость печати.

18. Перечислите основные параметры, характеризующие компьютер.

19*. Перечислите основные параметры микропроцессора.

Глава 7

КОМПЬЮТЕРНЫЕ СЕТИ

7.1. Введение в компьютерные сети

Одновременно с расширением области использования компьютеров выросло и количество пользователей, желающих иметь доступ к средствам для эффективной обработки и хранения совместно используемой информации.

Например, при проектировании нового здания большое число специалистов – архитектор, инженер, пожарный и др. хотят одновременно получать доступ и вносить, если необходимо, изменения в строительные чертежи, находящиеся в процессе разработки. Авиакомпании могут продавать билеты на один и тот же рейс в агентствах, находящихся в разных городах.

Простейшее решение данной задачи состоит в подключении к центральному компьютеру большой мощности множества терминалов (*рис. 7.1*).

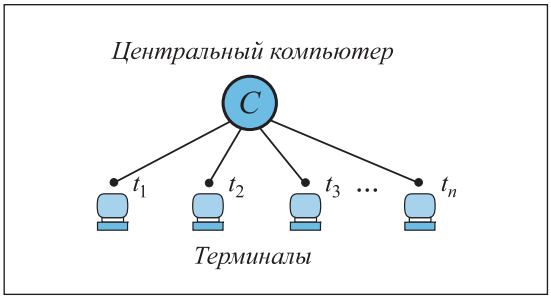


Рис. 7.1. Централизованная вычислительная система

Как правило, терминал состоит из монитора, клавиатуры и, если необходимо, принтера. Главным недостатком централизованной системы обработки данных является низкая надежность и неэффективное использование вычислительных ресурсов.

Со временем появилась тенденция к переходу от централизованных систем к размещению компьютеров у каждого пользователя и обеспечению их эффективного взаимодействия с помощью специальных соединений (*рис. 7.2*).

> *Компьютерной сетью называется множество компьютеров, которые могут обмениваться информацией с помощью системы связи.*

Компьютеры любой сети подключаются к системе связи с помощью специально предназначенных блоков ввода-вывода, называемых **сетевыми адаптерами**. Естественно, что в пределах одной определенной сети каждый компьютер, точнее каждый сетевой адаптер, имеет уникальный адрес, называемый **сетевым адресом**.

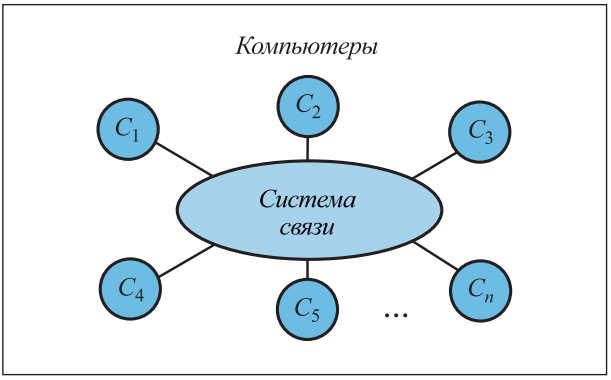


Рис. 7.2. Компьютерная сеть

Например, компьютерная сеть может быть построена с использованием в качестве **системы связи** существующей телефонной сети. В таком случае сетевой адаптер включает в себя модулятор для преобразования цифровых сигналов компьютера в телефонные сигналы и демодулятор для обратного преобразования. Соответствующее устройство ввода-вывода носит название **модем (модулятор-демодулятор)**. Сетевой адрес задается номером телефона, к которому подключен модем.

В общем случае система связи состоит из **линий передачи** сигналов. Этими линиями могут быть:

– кабели на основе витых проводов (“витая пара”);
– коаксиальные кабели;
– оптические кабели;
– микроволновые радиоканалы (наземные или спутниковые).

Кабели на основе витых пар аналогичны телефонным и обеспечивают скорость передачи до 1 *Мбит/с*. **Коаксиальные кабели** похожи на те, которые применяются в телевизионных кабельных сетях. Они обеспечивают скорость передачи до 1 *Гбит/с*. **Оптический кабель** состоит из стеклянного или прозрачного пластмассового волокна, покрытого защитной оболочкой. Оптический сигнал от лазера-источника распространяется по волокну и принимается фоточувствительной ячейкой. Скорость передачи информации по оптическому кабелю может достигать 1 *Тбит/с*.

Микроволновые радиоканалы состоят из ретрансляционных станций, которые обеспечивают прием и передачу сигналов на сантиметровых волнах. На Земле приемо-передающие станции расположены в радиусе прямой видимости антенн, на расстоянии 40–50 *км* друг от друга. В случае космических линий связи соответствующие станции размещаются на спутниках. Скорость передачи микроволновых радиоканалов составляет около 10 *Гбит/с*.

В зависимости от **региона**, занимаемого компьютерами сети, существуют следующие **типы сетей**:

– локальные сети;
– региональные сети;
– глобальные сети.

В **локальных сетях** компьютеры занимают малую зону (диаметром до 2 *км*) и обслуживают одну организацию. Как правило, локальные сети состоят из компьютеров, находящихся в пределах одного или нескольких зданий. Обычно

в качестве линий передачи применяются кабели на основе витой пары или коаксиальные кабели.

Региональные сети покрывают площадь одного города или района. Линии связи реализуются на основе коаксиальных кабелей или маленьких приемопередающих станций, называемых **радиомодемами**.

Глобальные сети охватывают площадь одной страны, одного или нескольких континентов. В качестве линий передачи применяются оптические или микроволновые радиоканалы (наземные или спутниковые).

Главное преимущество сетей состоит в **разделении** или, другими словами, **совместном использовании** данных, программ и компьютеров каждой сети.

Например, в случае локальной сети могут совместно использоваться файлы, диски большой емкости, принтеры, сканеры и другая периферия. Очевидно, будучи доступным одновременно нескольким пользователям, соответствующее периферийное оборудование используется более эффективно. Таким образом, специалисты одной организации могут работать совместно над общими проектами: годовым бюджетом, планом продаж, обновлением баз данных и т.п.

В случае глобальных сетей коллективы исследователей могут осуществлять сложные расчеты на уникальном суперкомпьютере или совместно анализировать результаты особо дорогостоящих научных экспериментов. На основе рассматриваемых сетей создаются различные службы: передача файлов, электронная почта, передача новостей, группы общения по интересам, электронные игры, реклама, перевод денег и т.п.

Вопросы и упражнения

❶ Назовите причины, которые привели к появлению компьютерных сетей.

❷ Укажите недостатки централизованных вычислительных сетей.

❸ Назовите главные компоненты компьютерной сети.

❹ Объясните назначение системы связи из состава компьютерной сети.

❺ Какие функции выполняет сетевой адаптер? Как идентифицируются компьютеры, входящие в состав сети? Определите тип сетевого адаптера, с которым вы работаете.

❻ Из чего состоит система связи?

❼ Для чего предназначен модем? Для чего предназначен радиомодем?

❽ Назовите скорость передачи данных по следующим линиям связи:
- кабель на основе витой пары;
- коаксиальный кабель;
- оптический кабель;
- микроволновой радиоканал.

❾ Оцените время передачи одного видеофильма (≈800 *Гбит*) по известным вам линиям связи.

❿ Определите тип линий связи компьютерной сети, в которой вы работаете.

⓫ Как классифицируются компьютерные сети в зависимости от площади охвата?

⓬ Определите тип компьютерной сети (локальная, региональная или глобальная), в которой вы работаете.

⓭ В чем преимущества компьютерных сетей? Какие услуги предлагает компьютерная сеть?

7.2. Технологии взаимодействия в компьютерной сети

Ресурсами компьютерной сети являются: периферийное оборудование, линии связи, сами компьютеры, файлы, базы данных, исполняемые программы и т.п. Эффективное использование указанных ресурсов предполагает совместную работу или, другими словами, кооперацию компьютеров и программ, которые выполняются на них.

> *Технологией взаимодействия называется способ совместной работы компьютеров и программ в сети.*

Наиболее часто в компьютерных сетях используется технология клиент-сервер и равный-с-равным.

В **технологии клиент-сервер** общий ресурс, например, цветной принтер или диск большой емкости, управляется специально выделенным компьютером, называемым **сервером**. Компьютер, который желает получить доступ к данным ресурсам, называется **клиентом**. Для использования соответствующего ресурса клиент посылает серверу **запрос**. Сервер анализирует принятые запросы и, в

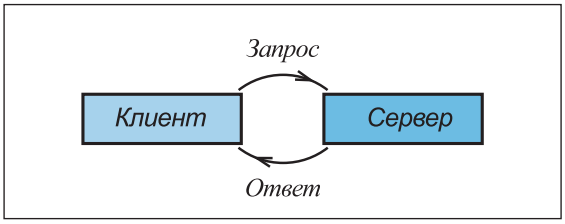


Рис. 7.3. Модель клиент-сервер

зависимости от статуса каждого клиента, принимает или отвергает их (*рис. 7.3*).

Ясно, что компьютер, который управляет общими файлами, будет называться **сервером файлов** или **файл-сервером**, а компьютер, управляющий принтерами, – **сервером печати**.

Компьютер, который управляет линиями связи, другими общими ресурсами и, возможно, обеспечивает доступ к внешним сетям, называется **сервером сети**. Как правило, именно на этом компьютере работает сетевая операционная система. Остальные компьютеры сети имеют более скромные параметры и называются **рабочими станциями**.

Главными преимуществами технологии клиент-сервер являются:

– эффективное использование дорогого оборудования;

– рациональное распределение работы (заданий) между компьютерами в зависимости от их мощности;

– надежная защита особо важных данных, которые хранятся только на сервере.

К сожалению, технология клиент-сервер сложна и требует мощных компьютеров.

В **технологии равный-с-равным** функции всех компьютеров в сети идентичны. Каждый компьютер в сети работает одновременно и как сервер, и как рабочая станция, предоставляя в общее распоряжение ресурсы, которыми располагает: часть файлов на жестком диске, устройство на оптическом диске, принтер и т.п. Как более простая, данная технология приме-

няется в малых локальных сетях. Для больших сетей технология равный-с-равным не обеспечивает надежной защиты данных.

Аналогичным образом технология клиент-сервер применяется и для организации совместной работы двух и более программ.

> *Программа, предоставляющая во время своего выполнения определенные услуги, называется сервером, а программы, которые обращаются к этим услугам, называются клиентами.*

Например, программа-сервер, которая управляет базой данных, выполняет следующие функции:

– обеспечивает защиту и безопасность данных;

– принимает и, если клиент получил соответствующую авторизацию, выполняет запросы на модификацию (изменение) данных;

– принимает запросы на чтение данных и в зависимости от статуса клиента разрешает или запрещает доступ к соответствующим данным;

– ведет журнал, в который заносит все операции, осуществляемые над базой данных.

Программа-клиент обеспечивает взаимодействие пользователя с базой данных и выполняет следующие функции:

– предлагает пользователю простой и удобный интерфейс;

– проверяет и редактирует данные, вводимые пользователем;

– направляет запросы программе-серверу;

– выводит информацию, извлеченную из базы данных.

Программы-серверы и программы-клиенты могут выполняться на одном и том же или на разных компьютерах. В последнем случае обработка данных является **распределенной**. Передача данных между программой-клиентом и программой-сервером осуществляется с помощью системы связи (*рис. 7.2*). Компьютер, на котором выполняется программа-сервер, называется **хостом** (с английского *host*).

Обычно в локальных сетях программа-клиент выполняется на рабочих станциях, а программа-сервер – на сетевом сервере. В случае региональных или глобальных сетей на каждом мощном компьютере выполняется несколько программ-серверов, которые предлагают разнообразные услуги программам-клиентам, выполняющимся на других компьютерах.

Вопросы и упражнения

❶ Объясните термин *технологии взаимодействия в сети*. Какие технологии взаимодействия в сети вы знаете?

❷ Как организована работа компьютеров в сети в случае технологии клиент-сервер?

❸ Какие преимущества и недостатки имеет технология клиент-сервер?

❹ Объясните назначение сетевого сервера, файл-сервера, сервера печати и рабочей станции.

❺ Как организована работа компьютеров в сети в случае технологии равный-с-равным? Какие преимущества и недостатки имеет данная технология?

❻ Определите технологию взаимодействия, реализованную в вашей сети. Есть ли в вашей сети файловый сервер и/или сервер печати?

❼ Существует ли в сети, с которой вы работаете, сетевой сервер? Обоснуйте ваш ответ.

❽ Реализована ли в вашей сети технология равный-с-равным? Обоснуйте ваш ответ.

❾ Как взаимодействуют программы в случае технологии клиент-сервер?

❿ Назовите функции программы-сервера и программы-клиента.

⓫ Объясните термин *хост*.

⓬ Компания открыла в разных городах страны агентства по продаже авиабилетов. Данные обо всех авиарейсах и о наличии свободных мест хранятся в компьютере, расположенном в центральном офисе компании.
Какие сетевые технологии могут обеспечить эффективную работу агентств по продаже авиабилетов? Необходимы ли в данном случае программы-серверы и программы-клиенты? Какие функции будут выполнять данные программы?

⓭ Коллективные электронные игры возможны при совместной работе 5 – 10 компьютеров, объединенных в сеть. В такой игре каждый играет против всех. Соответствующие компьютеры предлагают для общего использования раздел на жестком диске.
Какая сетевая технология обеспечит совместную работу компьютеров?

⓮ Разработайте технологию сетевого взаимодействия для компьютеров:

а) складов фирмы;
б) выставочных залов музея;
в) читальных залов библиотеки;
г) торговых залов магазинов, принимающих кредитные карты;
д) системы банкоматов;
е) системы оперативной проверки регистрационных номеров автомобилей (каждый полицейский экипаж снабжен микрокомпьютером с радиомодемом);
ж) лаборатории (класса) по информатике.

7.3. Топология и архитектура компьютерных сетей

Система связи любой компьютерной сети (*рис. 7.2*) обеспечивает передачу данных между компьютерами. Обычно передаваемые данные группируются в пакеты.

Каждый **пакет данных** содержит следующую информацию:
– адрес получателя;
– собственно данные;
– управляющую информацию;
– адрес отправителя.

Заметим, что пакет данных можно представить себе как обычный конверт, который пересылается с помощью традиционной почтовой службы. Путь, который проходит пакет, зависит от топологии сети.

> *Геометрическая конфигурация связей между компьютерами называется топологией сети.*

Конкретные топологии современных сетей создаются с применением основных структур: звезда, кольцо, магистраль, распределенная топология и т.п. (*рис. 7.4*).

В случае **топологии типа звезда** связь между двумя компьютерами C_i, C_j обеспечивается через центральный компьютер C_1. По этой причине компью-

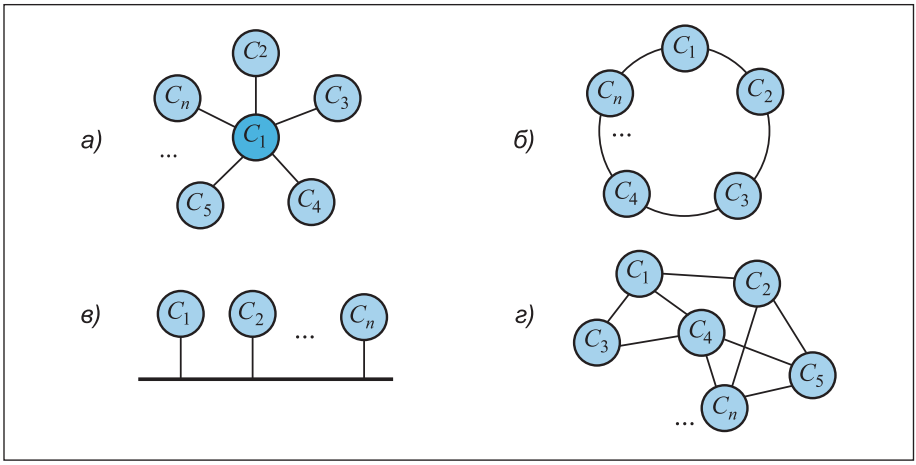


Рис. 7.4. Топологии сетей:
а – звезда; *б* – кольцо; *в* – магистраль; *г* – распределенная

тер C_1, называемый **главным компьютером**, играет важную роль в работе сети, осуществляя диспетчеризацию пакетов данных. Ясно, что выход из строя главного компьютера прерывает работу всей сети. Следовательно, главный компьютер должен быть очень надежным.

В **топологии типа кольцо** соединения между компьютерами образуют замкнутую цепочку. Пакет, посланный компьютером C_i, принимается компьютером C_{i+1}, который в свою очередь отсылает его компьютеру C_{i+2} и т.п., пока пакет не достигнет компьютера-приемника C_j. Поскольку выход из строя любого компьютера прерывает работу всей сети, все компьютеры C_1, C_2, ..., C_n должны быть очень надежными.

В **топологии типа магистраль** присутствует один канал, к которому подключены все компьютеры. Каждый компьютер "подслушивает" магистраль и принимает пакеты, адресованные только ему. Любой компьютер может отослать пакет только тогда, когда магистраль свободна.

Сети, основанные на топологии типа магистраль, очень надежны, поскольку связь между компьютерами C_i, C_j сохраняется даже в том случае, когда все другие компьютеры не работают.

В **распределенных топологиях** между каждой парой компьютеров существует несколько путей передачи данных. Например, один пакет данных, посланный компьютером C_1 компьютеру C_n (*рис. 7.4, г*), может достичь адресата по маршруту $C_1 - C_2 - C_5 - C_n$, а другой пакет – по маршруту $C_1 - C_4 - C_n$. Очевидно, сеть будет функционировать даже в том случае, если один или несколько компьютеров и линий связи выйдут из строя.

Обычно топологии типа звезда, кольцо и магистраль используются для локальных сетей. Региональные и глобальные сети имеют распределенную топологию. Объединение локальных сетей в региональные и глобальные сети осуществляется в соответствии с основными топологиями на *рис. 7.4*, где каждый узел C_1, C_2, ..., C_n представляет собой подсеть. Следовательно, современные сети имеют иерархическую структуру.

> *Набор правил для управления процессами обмена данными в сети называется протоколом обмена или просто протоколом.*

Любой **протокол** определяет режимы адресации компьютеров, длину и состав пакетов данных, алгоритмы обнаружения и исправления ошибок, режимы физического подключения сетевых адаптеров и кабелей и т.п. Одновременно с появлением первых компьютерных сетей каждый производитель вычислительной техники создавал свои собственные протоколы связи, что делало невозможным взаимное соединение компьютеров разного происхождения. Этот недостаток был устранен путем стандартизации протоколов. Напоминаем, что стандарт представляет собой документ, в котором регламентируется качество, характеристики, форма и т.п. определенного изделия. Международные **стандарты** разрабатываются Международной организацией по стандартизации (*ISO – International Standards Organisation*). Другой организацией, играющей важную роль в стандартизации изделий электроники и вычислительной техники, является Институт инженеров-электриков и электронщиков (*IEEE – Institute of Electrical and Electronics Engineers*).

> *Архитектурой сети называется набор ее основных характеристик: топология, протоколы связи, технология взаимодействия.*

Далее рассматриваются наиболее распространенные архитектуры компьютерных сетей.

Ethernet (сеть в эфире) – локальные сети, реализованные в соответствии со стандартом *IEEE* 802.3. Используют магистраль из кабеля с витыми парами, из коаксиального или оптоволоконного кабеля. Скорость передачи данных достигает 100 *Мбит/с*. Данная архитектура была разработана фирмами *XEROX*, *Intel* и *DEC*.

Token–Ring (кольцо с жетоном) – локальные сети, реализованные в соответствии со стандартом *IEEE* 802.5. Используют кольцо из кабеля с витыми парами или из коаксиального кабеля. Скорость передачи данных достигает 16 *Мбит/с*. Данная архитектура разработана фирмой *IBM*.

DATAKIT – локальные, региональные или глобальные сети, разработанные фирмой *Bell Laboratories*. С точки зрения топологии эта сеть состоит из множества соединенных друг с другом звезд. Для оптоволоконного кабеля достигнута скорость передачи до 1,5 *Гбит/с*.

SNA (*Sistem Network Arhitecture*) – архитектура, разработанная фирмой *IBM* для локальных, региональных и глобальных сетей. Протоколы данной архитектуры основаны на стандартах *ISO*. Сперва топология имела тип звезды, а сейчас изменилась на распределенную топологию, поддерживая и локальные сети.

ARPANET – архитектура, спроектированная несколькими университетами и корпорациями под эгидой Министерства обороны США (*Advanced Research Projects Agency*). Данная архитектура базируется на распределенной топологии и использует различные линии связи – от телефонных линий до спутниковых микроволновых каналов. Соответствующие линии связи соединяют отдельные суперкомпьютеры и разнообразные локальные и региональные сети, занимающие около половины земной поверхности. Архитектура *ARPANET* включает в себя следующие протоколы:

– протокол ***IP*** (*Internet Protocol*), предназначенный для соединения друг с другом локальных, региональных и глобальных сетей;

– протокол сервисов, основанных на соединениях, именуемый ***TCP*** (*Transmission Control Protocol*);

– протокол передачи файлов, называемый ***FTP*** (*File Transfer Protocol*);

– протокол электронной почты – ***SMTP*** (*Simple Mail Transfer Protocol*);

– протокол для удаленного подключения к компьютеру, именуемый ***TELNET***.

На основе архитектуры *ARPANET* была разработана глобальная компьютерная сеть *Internet*.

Вопросы и упражнения

❶ Объясните термин *пакет данных*. Какую информацию содержит каждый пакет данных?

❷ Объясните термин *топология сети*.

❸ Нарисуйте основные типы топологии сетей. Назовите преимущества и недостатки каждой из топологий.

❹ Объясните, как передаются пакеты данных в следующих сетях: звезда, кольцо, магистраль, распределенная топология.

❺ Уточните путь, по которому пойдет пакет, посланный компьютером C_2 компьютеру C_4 (*рис. 7.4, а*).

❻ Найдите путь, по которому пойдут пакеты, посланные компьютером C_2 компьютеру C_4 (*рис. 7.4, б*).

❼ Сколько путей передачи пакетов существует между компьютерами C_1, C_n на *рис. 7.4, г*? Какой путь самый короткий?

❽ Что произойдет с сетями на *рис. 7.4*, если выйдет из строя компьютер C_1? А если выйдет из строя компьютер C_2?

❾ Определите топологию локальной сети, с которой вы работаете. Перечислите преимущества и недостатки данной топологии.

❿ Для чего предназначен протокол? Какие нормы содержит протокол?

⓫ Аргументируйте необходимость стандартизации протоколов. Кто разрабатывает соответствующие стандарты?

⓬ Назовите протоколы, используемые в локальной сети, с которой вы работаете.

⓭ Объясните, как объединяются локальные сети в региональные и глобальные сети. Нарисуйте топологию региональных или глобальных сетей, к которым вы имеете доступ.

⓮ Объясните термин *архитектура сети*.

⓯ Приведите примеры архитектур локальных, региональных и глобальных сетей.

⓰ Охарактеризуйте архитектуру сети, с которой вы работаете.

⓱ Перечислите протоколы, используемые в архитектуре *ARPANET*. Применяются ли эти протоколы в сети, с которой вы работаете?

7.4. Глобальная сеть *ИНТЕРНЕТ*

Глобальная сеть ИНТЕРНЕТ основывается на распределенной топологии и состоит из отдельных компьютеров, локальных, региональных и глобальных подсетей (*рис. 7.5*).

Соединение сетей друг с другом осуществляется с помощью специальных сетевых устройств, называемых шлюзами и маршрутизаторами. **Шлюз** (*gate-*

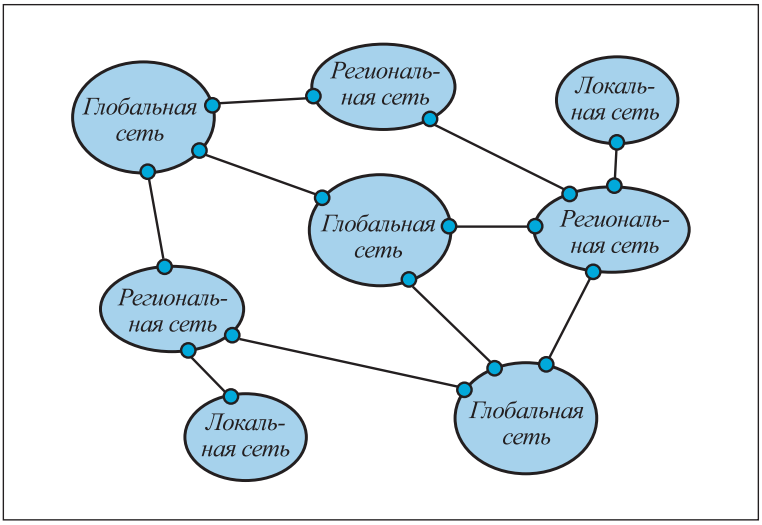


Рис. 7.5. Топология ИНТЕРНЕТа

way) – это специальный компьютер, предназначенный для соединения двух сетей с различными протоколами. **Маршрутизатор** (*router*) – это выделенный компьютер, с помощью которого соединяются сети, использующие одинаковые протоколы, и который используется для определения наилучшего пути передачи пакетов. Компьютеры, подключенные к ИНТЕРНЕТу, называются **хостами** (*host*). Работу сети регламентируют около 100 протоколов.

Идентификация компьютеров в пределах сети производится с помощью **ИНТЕРНЕТ *адресов***. Они могут быть двух типов: цифровые и символические.

Цифровой адрес состоит из 32 двоичных цифр (4 *байтов*) и имеет структуру, представленную на *рис. 7.6*.

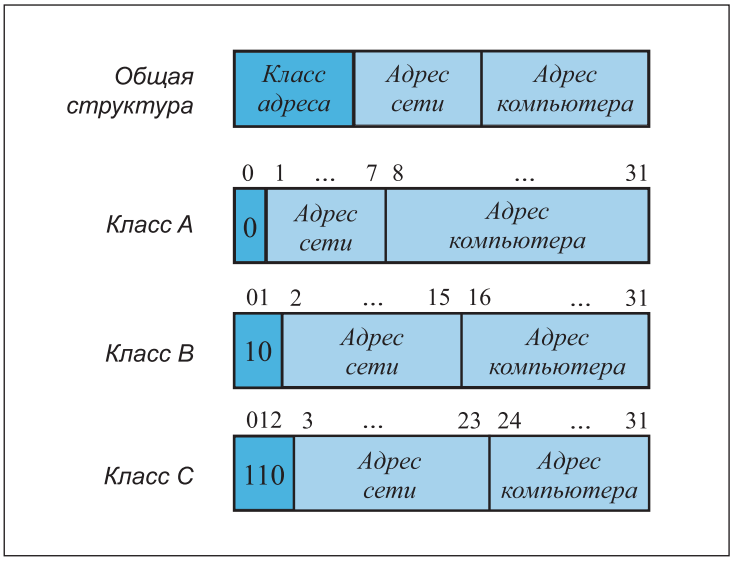


Рис. 7.6. Структура цифровых адресов

Поскольку ИНТЕРНЕТ – это “сеть сетей”, цифровой адрес содержит адрес подсети (поле *Адрес сети*) и адрес компьютера внутри самой подсети (поле *Адрес компьютера*). В зависимости от максимального количества компьютеров, которые можно идентифицировать внутри подсети, адреса делятся на классы *A*, *B* и *C*. Адреса, принадлежащие классам *D* и *E*, имеют специальное назначение. Характеристика ИНТЕРНЕТ адресов представлена в *таблице 7.1*.

Таблица 7.1

Характеристика цифровых адресов

Класс	Количество доступных адресов	
	сетей	компьютеров
A	2^7=128	2^{24} = 16 777 216
B	2^{14}=16 384	2^{16} = 65 536
C	2^{21}=2 097 152	2^8 = 256

Адреса класса *A* выделены для больших сетей, как правило, для глобальных. Сеть такого рода может содержать около 16 миллионов компьютеров. Адреса класса *B* предназначены для средних сетей, как правило, для региональных. Сеть такого рода может содержать около 65 тысяч компьютеров. Адреса класса *C* зарезервированы за относительно малыми сетями, включающими до 256 компьютеров. Каждый ИНТЕРНЕТ адрес уникален. Адреса присваиваются компьютерам **Информационным центром сети** (*Network Information Center*).

Например, двоичное число

10010010 00110011 00001001 11110111

является адресом класса *B* и идентифицирует компьютер номер

00001001 11110111

который входит в состав сети с номером

010010 00110011.

Для удобства двоичные адреса представляются в десятичной форме (каждый байт в отдельности). Десятичные числа, соответствующие каждому байту, отделяются друг от друга точками.

В случае рассмотренного выше примера получаем:

$(10010010)_2=(146)_{10}$;

$(00110011)_2=(51)_{10}$;

$(00001001)_2=(9)_{10}$;

$(11110111)_2=(247)_{10}$.

Следовательно, в десятичной форме рассматриваемый адрес будет 146.51.9.247.

Адреса в десятичной и тем более в двоичной форме неудобны для большинства пользователей. По этой причине чаще используются символические адреса.

Символический адрес состоит из имени компьютера-хоста и имен доменов, отделенных друг от друга точками. **Домен** представляет собой группу компьютеров, образованную по тематическим или географическим признакам. Любой домен может быть разделен на субдомены, приобретая таким образом иерархическую структуру. Имена доменов указываются в порядке роста области охвата.

Например, символические адреса

```
c1.lme.ch.md
```

```
c5.lme.ch.md
```

идентифицируют компьютеры c1 и c5 из домена lme (Лицей "Михай Еминеску").

Символические адреса

```
c1.lic.ch.md
```

```
c9.lic.ch.md
```

идентифицируют компьютеры c1 и c9 из домена lic (Лицей "Ион Крянгэ").

Домены lme и lic – это субдомены домена ch (Кишинэу). В свою очередь, ch – это субдомен домена md (Республика Молдова).

Аналогичным образом, символические адреса

```
rector.ase.men.ro
```

```
decan.ase.men.ro
```

определяют компьютеры rector и decan домена ase (Академия экономических знаний). Домен ase – это субдомен домена men (Министерство народного образования), а men – это субдомен домена ro (Румыния).

Обычно домен самого верхнего уровня является страной (md, ro, us и т.п.) или видом организации (com – коммерческая, mil – военная, edu – образовательная и т.п.).

Отношения включения между доменами можно отобразить при помощи диаграмм *Эйлера*, часто используемых в теории множеств. Для примера, на *рис. 7.7* представлена такая диаграмма для некоторых символических адресов домена md.

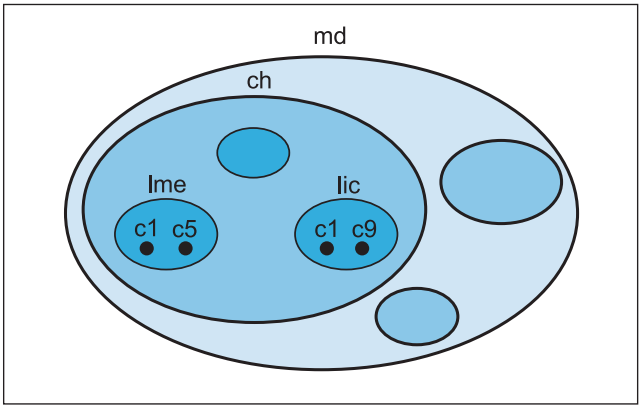


Рис. 7.7. Диаграмма *Эйлера* для символических адресов

Для **преобразования** символических адресов в цифровые и обратно в каждом домене имеется **сервер имен** (*name server*). Эта программа управляет соответствующим доменом без вмешательства серверов верхнего уровня иерархии. Следовательно, в ИНТЕРНЕТе не существует центрального компьютера, отвечающего за приблизительно 10^{10} адресов сети.

В случае примера, приведенного ранее (*рис. 7.7*), адреса компьютеров c1 и c5 обрабатываются на сервере lme, а адреса компьютеров c1 и c9 – на сервере lic. Сервер ch будет работать только с адресами серверов lic и lme, не вмешиваясь в обработку адресов из доменов lic и lme. Аналогично, сервер men не обрабатывает адреса компьютеров rector и decan, предоставляя эту работу серверу ase.

Вопросы и упражнения

❶ Нарисуйте топологию ИНТЕРНЕТа. Как соединяются друг с другом подсети внутри ИНТЕРНЕТа?

❷ Для чего предназначены шлюзы и маршрутизаторы?

❸ Как идентифицируются компьютеры в ИНТЕРНЕТе? В чем преимущества и недостатки цифровых адресов? А символических адресов?

❹ На какие классы подразделяются цифровые адреса? Для чего предназначена такая классификация?

❺ Объясните назначение полей *Адрес сети* и *Адрес компьютера*, входящие в состав цифрового адреса.

❻ Кто выделяет адреса для компьютеров из ИНТЕРНЕТа?

❼ Определите классы следующих адресов. Уточните адрес подсети и адрес компьютера в подсети.

a)	45.201.19.63;	*d)*	192.109.58.170;
b)	201.165.213.91;	*e)*	15.21.207.250;
c)	154.36.79.200;	*f)*	217.15.69.113.

❽ По каким критериям компьютеры объединяются в домены? Приведите примеры нескольких доменов верхнего уровня.

❾ Даны следующие символические адреса:

a)	c1.lme.ch.md;	*f)*	c4.lme.ch.md;
b)	c3.lme.ch.md;	*g)*	c5.lme.ch.md;
c)	c1.lic.ch.md;	*h)*	c9.lic.ch.md;
d)	director.lic.ch.md;	*i)*	prof.lic.ch.md;
e)	elev1.lic.ch.md;	*j)*	elev4.lic.ch.md.

Уточните домены каждого компьютера и отношения включения между доменами. Нарисуйте диаграммы *Эйлера* для рассматриваемых адресов.

❿ Нарисуйте диаграммы *Эйлера* для приведенных ниже символические адресов. Уточните домены компьютеров и соответствующие отношения включения.

a) rector.ase.men.ro;

b) decan.ase.men.ro;

c) student.info.ase.men.ro;

d) rector.ase.met.md;

e) decan.ase.met.md;

f) student.cib.met.md.

⓫ В чем назначение сервера имен? Какие адреса обрабатываются такими серверами?

⓬ Уточните серверы имен для адресов из упражнений 9 и 10. Укажите адреса, обрабатываемые каждым сервером.

⓭ Определите цифровой и символический адреса компьютера, на котором вы работаете. Уточните класс адресов, адрес подсети и адрес компьютера в рамках подсети. Нарисуйте диаграмму *Эйлера*, представляющую отношения включения между доменами, к которым принадлежит ваш компьютер.

7.5. Сервисы ИНТЕРНЕТа

ИНТЕРНЕТ предлагает следующую гамму услуг:
– удаленный доступ к компьютерам;
– передача файлов;
– электронная почта;
– новости и конференции (дискуссии);
– распространение и поиск информации и т.п.

Взаимодействие компьютеров и программ, которые представляют данные услуги, основывается на модели клиент-сервер. Обычно на компьютере потребителя выполняется программа-клиент, а на компьютере, предоставляющем сервис (оказывающем услугу), выполняется программа-сервер.

Сервис Telnet позволяет пользователю получить доступ к другому компьютеру, находящемуся на произвольном расстоянии (так называемый удаленный доступ). После установления соединения компьютер пользователя используется как простой терминал, находящийся на большом расстоянии от центрального компьютера. Пользователь может запускать на удаленном компьютере различные программы, просматривать файлы, изменять текущую директорию и т.п. Защита компьютеров и соответствующих данных обеспечивается применением паролей. Сервис Telnet применяется для совместного использования дорогостоящих ресурсов, например суперкомпьютеров.

Сервис передачи файлов или, короче, **сервис FTP** (*File Transfer Protocol*) позволяет пользователю копировать файлы, находящиеся на компьютерах, размещенных в различных географических точках. Данный сервис предлагает два режима передачи файлов:

– двоичный режим, при котором сохраняется последовательность битов файла, так что оригинал и копия идентичны с точностью до бита;

– текстовой режим, в котором передаются наборы символов в коде ASCII.

В общем случае, для получения доступа к серверу FTP клиент должен ввести пароль. Однако существуют публичные серверы (*FTP anonymous*), которые разрешают доступ к файлам без необходимости ввода специального пароля.

Сервис электронной почты (*electronic mail* или, сокращенно, *e-mail*) имитирует режимы работы обычной почты.

Электронное письмо, называемое **сообщением** (*message*), содержит:
– адрес получателя;
– тему сообщения, выраженную в нескольких словах;
– адрес отправителя;
– текст письма;
– файлы, которые могут быть присоединены к сообщению.

Присоединенные файлы могут быть любого типа: текстовые, звук, изображения, программы и т.п.

Письма хранятся в специальных файлах, называемых **почтовыми ящиками** (*mail box*). Адрес любого почтового ящика имеет форму:

<Имя ящика>@<Адрес компьютера>,

где:

<Имя ящика> – это название почтового ящика (обычно фамилия пользователя или аббревиатура);

@ – символ коммерческого “at”;

<Адрес компьютера> – символический адрес компьютера – клиента, на котором создан почтовый ящик.

Примеры:

1) petrescu@c1.lme.ch.md

2) florea@director.lic.ch.md

3) ionescu@c1.lme.ch.md

4) barbu@director.lic.ch.md

Читатели могут отправлять письма авторам данного учебника по адресу:

Anatol_Gremalschi@yahoo.com

Сообщения пересылаются через сеть почтовых серверов, которые выполняют роль обычных почтовых отделений.

Сервис электронной почты очень популярен благодаря его неоспоримым преимуществам – скорости, возможности присоединять к письмам файлы любого типа и развитым возможностям редактирования писем.

Самым современным сервисом распространения и поиска информации в ИНТЕРНЕТе является **сервис WWW** (***W****orld* ***W****ide* ***W****eb* – Всемирная Паутина). В данном сервисе информация представляется в форме *Web*-страниц.

Web*-страница** – это файл, написанный на языке *HTML* (H****yper****t****ext* ***M****arkup* ***L****anguage* – Язык разметки гипертекста), который содержит, кроме самой информации, ссылки на другие страницы *Web*. Адресуемые ссылками страницы могут размещаться на этом же компьютере или на компьютерах, расположенных в различных географических точках (*рис. 7.8*).

Взаимодействие программ в рамках сервиса *WWW* выполняется по типу клиент-сервер. Пользователь, который хочет предложить широкой публике определенную информацию, устанавливает на своем компьютере программу *WWW*-сервер и разрабатывает одну или несколько *Web*-страниц. Сервер принимает запросы, поступающие от других компьютеров, и предоставляет доступ к соответствующим страницам. Компьютер, на котором

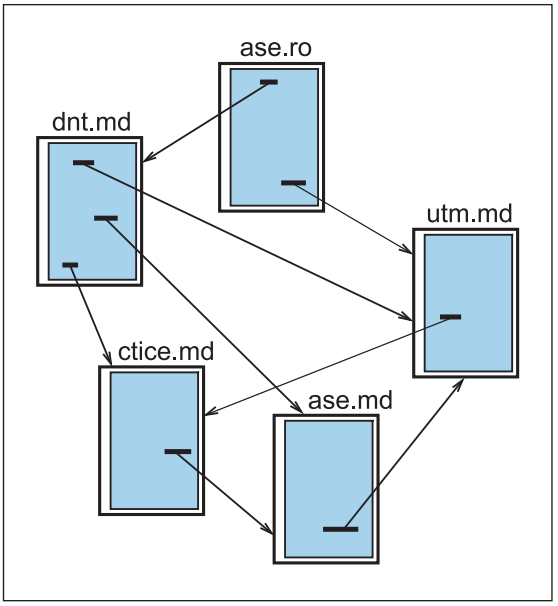


Рис. 7.8. Web-страницы

размещены *Web*-страницы и *WWW*-сервер, носит название **сайт** (*site* – местонахождение“).

Программа-клиент обеспечивает прием и отображение на экране *Web*-страниц, которые считываются с различных компьютеров (серверов) в ИНТЕРНЕТе. Сразу после запуска эта программа отображает установленную по умолчанию домашнюю страницу (*home page*) и ожидает указаний пользователя. Когда пользователь активизирует ссылку, программа-клиент устанавливает соединение с *Web*-сервером и копирует с него страницу, определенную ссылкой. Скопированная страница выводится на экран.

Далее, когда пользователь активизирует другую ссылку, программа-клиент вновь устанавливает соединение с одним из компьютеров сети, вновь читает *Web*-страницу и т.п. Другими словами, пользователь “перелистывает” *Web*-страницы, находящиеся на различных компьютерах, независимо от их географического расположения. По этой причине программы-клиенты называются **программами перелистывания** или **программами навигации** (на английском *browser* или *explorer*).

В рамках сервиса *WWW* ресурсы сети определяются с помощью специальных адресов, называмых адресами *URL* (*Uniform Resource Locator* – Универсальный указатель ресурсов). Данные адреса имеют форму:

<протокол>: // <Символический адрес>[:<порт>]/<путь>/<файл>,

где:

<протокол> – определяет название протокола для передачи данных по сети;

<Символический адрес> – адрес компьютера, содержащего соответствующий файл;

<порт> – порт доступа (необязателен);

<путь>/<файл> – спецификация файла.

Для примера приведем несколько адресов *URL*, содержащих интересную информацию:

http://www.edu.md – сайт Министерства Просвещения и молодежи, Молдова;

http://www.ctice.edu.md – сайт Центра Информационных и Коммуникационных Технологий в Образовании;

http://www.dnt.md – сайт Ассоциации *Dynamic Network Tehnologies*, Молдова;

http://www.itc.ro/museum/museum.html – сайт Национального музея искусств, Румыния;

http://www.nmsi.ac.uk – сайт Музея науки и техники, Великобритания;

http://www.nasa.gov – сайт агентства *NASA*, США.

Напоминаем, что обозначение http определяет протокол передачи гипертекста (*Hypertext Transfer Protocol*).

В настоящее время количество файлов в ИНТЕРНЕТе исчисляется миллиардами. Естественно, не может быть и речи о поиске необходимой информации путем чтения каждого файла в отдельности. Для упрощения поиска информации в рамках ИНТЕРНЕТа были созданы специальные поисковые серверы (*search engine*).

Поисковый сервер – это компьютер, который непрерывно исследует сеть и читает *Web*-страницы или другие ресурсы, выставленные на общее обозрение. Они классифицируются в зависимости от их содержания, а их адреса запоминаются в базе данных на поисковом сервере.

Программа-клиент посылает поисковому серверу запрос, в котором указывает, признаки необходимой информации. Сервер опрашивает базу данных и передает клиенту список адресов, по которым может быть найдена запрошенная информация.

Для примера приведем адреса наиболее популярных серверов поиска:

http://www.yahoo.com – сервер *YAHOO (Yet Another Hierarhicaly Organizated Oracle* – еще один иерархически организованный оракул) компании Yahoo! Inc;

http://www.google.com – сервер *Google* корпорации Google Inc;

http://www.bing.com – сервер *Bing* корпорации Microsoft Inc;

http://yandex.ru – сервер *Yandex* компании *„Яндекс”*;

Доступ к данным серверам бесплатный.

Вопросы и упражнения

❶ Назовите гамму услуг, предлагаемых в ИНТЕРНЕТе. Как взаимодействуют компьютеры сети в процессе оказания некоторой услуги?

❷ Для чего предназначен сервис *FTP*? Какие файлы могут передаваться с помощью данного вида сервиса?

❸ Изучите программу *FTP*, установленную на вашем компьютере. Скопируйте несколько файлов с публичных серверов *FTP* или с других серверов, к которым вы имеете доступ.

❹ От чего зависит скорость передачи файлов? Определите скорость передачи файлов с нескольких серверов *FTP*, расположенных в Республике Молдова, Румынии, США и России.

❺ Как задаются адреса в сервисе электронной почты?

❻ Объясните, как взаимодействуют компьютеры клиент и сервер почты в рамках соответствующей службы.

❼ Из каких частей состоит электронное письмо? Какие из них необязательны?

❽ Объясните, как формируется электронный адрес. Узнайте почтовые адреса ваших друзей.

❾ Изучите программу электронной почты, установленную на компьютере, с которым вы работаете. Проверьте, имеет ли данная программа следующие возможности:
- использование нескольких почтовых ящиков;
- сортировку и хранение писем в папках;
- написание писем по шаблонам;
- шифрование и дешифрование корреспонденции;
- оповещение в момент поступления новой корреспонденции;
- проверка того факта, что посланное письмо дошло до адресата;
- использование электронных подписей.

❿ В чем преимущества электронной почты? Может ли она заменить традиционную почту?

⓫ Создайте группы по четыре пользователя электронной почты и экспериментально определите скорость передачи корреспонденции.

⓬ Какую информацию содержит *Web*-страница? Как эти страницы образуют всемирную паутину”?

⓭ Для чего предназначен *Web*-сервер? А *Web*-клиент? Как взаимодействуют эти программы?

⓮ Объясните, как работает *Web*-клиент. Как данная программа находит *Web*-страницы, размещенные на разных компьютерах?

⓯ Объясните способ указания ресурсов в ИНТЕРНЕТе с помощью адресов *URL*. Для чего предназначены различные поля таких адресов?

⓰ Изучите программу просмотра ИНТЕРНЕТа, установленную на вашем компьютере. Каковы возможности данной программы? Прочитайте *Web*-страницы, адреса которых приведены в данном параграфе.

⓱ Для чего предназначен поисковый сервер? Какие услуги предлагает такой сервер?

⓲ Найдите с помощью любого поискового сервера поставщиков услуг (провайдеров) ИНТЕРНЕТ в Республике Молдова.

⓳ Кроме сервисов, изученных в данном параграфе, сеть ИНТЕРНЕТ предлагает и другие сервисы, такие как *Archie*, *Gopher*, *WAIS*, подписки новостей, конференции и т.п. Используя поисковый сервер, найдите информацию об этих службах.

⓴ Создайте *Web*-страницы вашего класса и лицея.

Тест для самопроверки № 7

1. Определите продолжительность передачи видеофильма в архивированном виде (≈750 *Мбит*) по следующим линиям связи:

a) по телефонной линии 36 *Кбит/с*;

b) по витому кабелю 1 *Мбит/с*;

c) по коаксиальному кабелю 1 *Гбит/с*;
d) по оптическому кабелю 1 *Тбит/с*.

2. Какие из следующих утверждений являются истинными?
a) Локальные сети покрывают только площадь определенных населенных пунктов.
b) Глобальные сети покрывают поверхность стран, одного или нескольких континентов.
c) В общем случае, региональные сети соединяют компьтеры, принадлежащие только одной компании.
d) Как правило, локальные сети соединяют компьтеры только одной организации.

3. Назовите главные компоненты компьютерной сети.

4. Какие из следующих утверждений являются истинными?
a) В технологии *равный с равным* функции всех компьютеров сети идентичны.
b) Сервер сети выполняет те же функции, что и любой другой компьютер сети.
c) В технологии клиент-сервер общий ресурс управляется специально назначенным компьютером.
d) Любой компьютер-клиент имеет доступ к данным любого другого компьютера из сети.

5* . Какие из следующих утверждений являются истинными?
a) В топологии звезда связь между двумя компьютерами сети осуществляется через центральный компьютер.
b) В распределенной топологии связь между двумя компьютерами сети осуществляется с помощью магистрали.
c) В топологии магистраль связь между двумя компьютерами сети осуществляется при помощи центрального компьютера.
d) В топологии кольцо все компьютеры имеют одинаковые функции.

6*. Назовите назначение сетевого протокола.

7*. Объясните назначение термина *архитектура сети*.

8. Нарисуйте топологию сети ИНТЕРНЕТ.

9. Какие из следующих утверждений истинны?
a) ИНТЕРНЕТ представляет собой локальную сеть, которая имеет доступ к другим локальным, региональным или глобальным сетям.
b) ИНТЕРНЕТ представляет собой совокупность сетей страны.
c) ИНТЕРНЕТ представляет собой сеть очень большой международной компании.
d) ИНТЕРНЕТ представляет собой распределенную сеть, образованную из глобальных, региональных и локальных сетей, соединенных между собой.
e) ИНТЕРНЕТ состоит из всех компьютеров мира, соединенных между собой.

10. Как идентифицируются компьютеры в ИНТЕРНЕТе?

11*. Определите класс числового ИНТЕРНЕТ адреса 214.121.216.109. Уточните адрес подсети и адрес компьютера в сети.

12. Нарисуйте диаграмму *Эйлера* для символических адресов, приведенных ниже. Уточните компьютерные домены и соответствующие отношения принадлежности.

---

* Только для реального профиля.

a) directie.orhei.md;

b) contabilitate.orhei.md;

c) presedinte.orhei.md;

d) directie.cahul.md;

e) contabilitate.cahul.md;

f) presedinte.calarasi.md.

13. Укажите соответствие между названиями служб ИНТЕРНЕТа (из левого столбца) и их назначением (из правого столбца):

(1) Telnet;

(2) FTP;

(3) E-mail;

(4) WWW;

(a) перенос файлов с удаленного (находящегося на удалении) компьютера на компьютер пользователя;

(b) поиск информации на магнитном диске вашего компютера и ее отображение на экране;

(c) редактирование графических изображений и их передача через ИНТЕРНЕТ;

(d) передача электронных писем;

(e) обработка данных с помощью табличного процессора и передача результатов обработки с помощью электронной почты;

(f) представление, передача и поиск информации в ИНТЕРНЕТе;

(g) доступ к другим компьютерам, компьютер пользователя при этом становится терминалом, находящимся на расстоянии компьютера.

14. Укажите технологию взаимодействия в сети ИНТЕРНЕТ:
a) от равного к равному;
b) клиент-сервер;
c) от равного к равному и клиент-сервер.

15. Какие из следующих утверждений являются истинными:
a) При создании Web-страниц используется язык ПАСКАЛЬ.
b) Для отправки электронного письма необходимо иметь подключение к ИНТЕРНЕТу.
c) Web-страница может содержать только текст и изображения.
d) При создании Web-страниц используется язык HTML.
e) Web-страница может содержать текст, изображения, звук и видео.
f) Электронные письма не могут быть прочитаны за границей страны.
g) Поисковый сервер может считывать информацию с любого компьютера, подключенного к ИНТЕРНЕТу.

16. Выберите из приведенного ниже списка адреса электронной почты:

a) www.directie.orhei.md;

b) apetrescu@contabilitate.orhei.md;

c) ivulpe@presedinte.orhei.md;

d) bjosu@directie.cahul.md;

e) www.contabilitate.cahul.md;

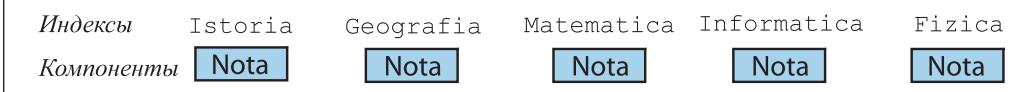
f) munteanu@presedinte.calarasi.md.

ОТВЕТЫ К ТЕСТАМ ДЛЯ САМОПРОВЕРКИ

Тест № 1

1.

```
type SituatiaScolara = array [Obiect] of Nota
```



2. Множество значений типа данных `OrarulLectiilor` состоит из двумерных массивов. Строки определяются индексами типа `ZiDeScoala`, а столбцы – индексами типа `Lectie`.

3.

a) `x[1]+ x[2]+ x[3]+ x[4];`

b) `y[7]+ y[8]+ y[9]+ y[10];`

c) `abs(x[3]);`

d) `abs(y[6]);`

e) `x[1]+ y[10].`

4.

```
Program RTA9;
{ Ответ на Тест 1, задание 4 }
type Numere = array [1..50] of integer;
var A : Numere;
    n, i : integer;
begin
  write('Введите n='); readln(n);
  { Считываем числа с клавиатуры }
  for i:=1 to n do
    begin
      write('A[', i, ']='); readln(A[i]);
    end;
  { Выводим числа на экран }
  writeln('Числа в обратном порядке:');
```

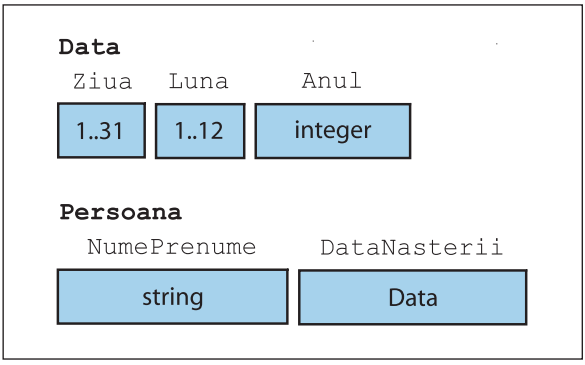
```
  for i:=n downto 1 do
    writeln('A[', i, ']=', A[i]);
  readln;
end.
```

5. Над строками типа `string` можно выполнять операцию конкатенации (сцепления, слияния), обозначаемую как „+”. Текущая длина любой величины `v` типа `string` может быть найдена с помощью предопределенной функции `length(v)`, возвращающей значение типа `integer`. Над строками типа `string` разрешены операции отношения <, <=, =, >=, >, <>. Результат этих операций имеет тип `boolean`.

6.

```
Program RTA10;
{ Ответ на Тест 1, задание 6 }
var S : string;
    i : integer;
begin
  writeln('Введите строку:');
  readln(S);
  writeln('Строка в обратном порядке:');
  for i:=length(S) downto 1 do
    write(S[i]);
    writeln;
  readln;
end.
```

7.



8.

```
Program RTA11;
{ Ответ на Тест 1, задание 8 }
type Angajat = record
                 NumePrenume : string;
                 Salariu : real;
               end;
     ListaDePlata = array [1..100] of Angajat;
var L : ListaDePlata;
```

```
    s : real;
    i, n : integer;
begin
  write('Введите n='); readln(n);
  { Ввод данных о каждом из сотрудников }
  for i:=1 to n do
    begin
      writeln('Введите данные о сотруднике ', i);
      write('Фамилия и имя: ');
      readln(L[i].NumePrenume);
      write('Заработная плата: ');
      readln(L[i].Salariu);
    end;
  { Поиск максимальной зарплаты }
  s:=0;
  for i:=1 to n do
    if L[i].Salariu > s then s:=L[i].Salariu;
  writeln('Максимальная зарплата = ', s:10:2);
  { Вывод данных }
  writeln('Сотрудники с максимальной зарплатой:');
  for i:=1 to n do
    if L[i].Salariu = s then writeln(L[i].NumePrenume);
  readln;
end.
```

9. Оператор `with` используется для уменьшения объема текста программ на ПАСКАЛЕ путем исключения повторений имен переменных типа **`record`** (запись).

10. Значения переменной `V`:

[ ], ['X'], ['Y'], ['Z'], ['X', 'Y'], ['X', 'Z'], ['Y', 'Z'], ['X', 'Y', 'Z'].

Значения переменной `I`:

[ ], [8], [9], [8, 9].

11.

```
Program RTA12;
{ Ответ на тест 1, Задание 11 }
var S : string;
    i, n : integer;
begin
  write('Введите строку символов, состоящую из ');
  writeln('заглавных букв латинского алфавита:');
  readln(S);
  { Подсчитываем гласные в строке }
  n:=0;
  for i:=1 to length(S) do
    if S[i] in ['A', 'E', 'I', 'O', 'U'] then n:=n+1;
  writeln('Количество гласных = ', n);
readln;
end.
```

12. Данные файлов ПАСКАЛЯ хранятся на носителях информации периферийных устройств: на магнитных дисках и лентах, на оптических дисках, на бумаге принтера, документосчитывающих устройствах и др. Процедура `assign` ассоциирует (связывает) переменные типа *файл* из программы на ПАСКАЛЕ с файлами на носителях информации периферийных устройств.

13. В зависимости от **типа разрешенных операций** над компонентами, файлы классифицируются как:

– входные файлы (разрешено только чтение);

– выходные файлы (разрешена только запись);

– обновляемые файлы (разрешены как чтение, так и запись).

В зависимости от **способа доступа** к компонентам, файлы классифицируются как:

– файлы с последовательным доступом или последовательные файлы (доступ к компоненте *i* возможен только после того, как считана/записана компонента *i* – 1);

– файлы с произвольным или прямым доступом (к любой компоненте можно обратиться непосредственно по ее порядковому номеру *i* в файле).

14.

```
Program RTA13;
{ Ответ на Тест 1, задание 14 }
type Angajat = record
                 NumePrenume : string;
                 Salariu : real;
               end;
     FisierAngajati = file of Angajat;
var A : Angajat;
    F : FisierAngajati;
    i, n : integer;
begin
  { Создание файла SALARII.DAT }
  assign(F, 'SALARII.DAT');
  rewrite(F);
  write('Введите n='); readln(n);
  for i:= 1 to n do
    begin
      writeln('Введите данные о сотруднике ', i);
      write('Фамилия, имя: '); readln(A.NumePrenume);
      write('Заработная плата: '); readln(A.Salariu);
      { Запись данных о сотруднике в файл F }
      write(F, A);
    end;
  close(F);
end.
```

15.

```
Program RTA14;
{ Raspuns la Testul 1, Itemul 15 }
type Angajat = record
                 NumePrenume : string;
```

```
                Salariu : real;
              end;
     FisierAngajati = file of Angajat;
var A : Angajat;
    F : FisierAngajati;
begin
  assign(F, 'SALARII.DAT');
  reset(F);
  writeln('Данные, считанные из файла SALARII.DAT:');
  while not eof(F) do
    begin
      read(F, A);
      writeln(A.NumePrenume, ' ', A.Salariu:10:2);
    end;
  readln;
  close(F);
end.
```

16.

```
Program RTA15;
{ Ответ на Тест 1, задание 16 }
var a, b : real;
    c : string;
    Intrare, Iesire : text;
begin
  assign(Intrare, 'REZULTAT.TXT');
  reset(Intrare);
  assign(Iesire, 'MEDII.TXT');
  rewrite(Iesire);
  while not eof(Intrare) do
    begin
      readln(Intrare, a, b, c);
      writeln(Iesire, (a+b)/2, ' ', c);
      writeln((a+b)/2, ' ', c);
    end;
  readln;
  close(Intrare);
  close(Iesire);
end.
```

Тест № 2

1. 000, 001, 010, 011, 100, 101, 110, 111.

2. Результат декодирования: CACB. Результат кодирования: 00001 00000 00010.

3. Латинский алфавит состоит из 26 букв. Следовательно, количество всевозможных сообщений источника информации $n = 26 + 26 = 52$. Количество информации, содержащейся в одном сообщении:

$$I = \log_2 52 \approx 5{,}7 \text{ бит}.$$

Из неравенства $m \geq I$, получаем $m = 6$.

4. Число всевозможных сообщений электронного календаря $n = 31 \times 12 \times 100 = 37200$. Количество информации, содержащейся в одном сообщении:

$$I = \log_2 37\,200 \approx 15{,}2 \text{ бит}.$$

Из неравенства $m \geq I$, получаем $m = 16$.

5. $I = 8$ *бит* (1 *байт*).

6. $V = 10$ мин $\times$ 200 символов/минут $\times$ 1 байт = 2000 *байт*.

7.

```
Program RTA16;
 { Ответ на Тест 2, задание 7 }
var c : char;
begin
  while not eof do
    begin
      readln(c);
      writeln(c, ' - ', ord(c));
    end;
end.
```

8.

```
Program RTA17;
 { Ответ на Тест 2, задание 8 }
var i : integer;
begin
  while not eof do
    begin
      readln(i);
      writeln(i, ' - ', char(i));
    end;
end.
```

9. $I = (10 \times 10) \times 30 \times \log_2 128 = 21\,000$ *бит* = 2625 *байт*

10. *a)*, *c)*, *d)*, *f)*, *h)*.

11. $V = 3 \cdot I = 3 \cdot \log_2 \left(\frac{|42-34|}{0{,}1} + 1 \right) \approx 19{,}02 \text{ бит}.$

12. $V = \frac{30 \cdot 60}{3 \cdot 10^{-5}} \log_2 128 = 4{,}2 \cdot 10^8 \text{ бит} \approx 400{,}5 \text{ Мбит}.$

13. (1) – (e); (2) – (f); (3) – (a); (4) – (c).

14. (1) – (c); (2) – (f); (3) – (a); (4) – (e).

Тест № 3

1. *a)* 667; *b)* ≈31,5926; *c)* ≈176,7551.

2. *b)* 8 не является восьмеричной цифрой; *c)* 2, 8, 4 и 6 не являются двоичными цифрами.

3.

```
Program RTA18;
{ Ответ на Тест 3, задание 3 }
var b : integer;  { основание }
    x : string;   { число, вводимое с клавиатуры }
    y : integer;  { число, преобразованное по основанию 10 }
    i : integer;
    bi : integer; { основание в степени i }
begin
  write('Введите основание b='); readln(b);
  write('Введите число, записанное по основанию ', b, ': ');
  readln(x);
  y:=0;
  bi:=1; { основание в степени 0 }
  for i:=length(x) downto 1 do
    begin
      y:=y+(ord(x[i])-48)*bi;
      bi:=bi*b;
    end;
  writeln('Число, записанное по основанию 10: ', y);
  readln;
end.
```

4. *a)* ≈1101,1110001; *b)* 111011,10110001; *c)* 10101000,011101001111.

5. *a)* ≈23036,7341; *b)* 1333,272; *c)* 155206,542.

6. *a)* ≈24FF,D611; *b)* 2DDB,534; *c)* 115,7F.

7. $(222221)_4$, $(1000001111)_2$, $(BB)_{16}$, $(132)_8$.

8. $(1D3)_{16} = (723)_8$; $(25)_8 = (21)_{10}$.

9.

```
Program RTA19;
{ Ответ на Тест 3, задание 9 }
var x : string;  { восьмеричное число }
    y : string;  { двоичное число }
    i : integer;
    T : array ['0'..'7'] of string;
    { T[i] – двоичное представление восьмеричной цифры i }
begin
  { Инициализация массива }
  T['0']:='000'; T['1']:='001';
  T['2']:='010'; T['3']:='011';
```

```
  T['4']:='100'; T['5']:='101';
  T['6']:='110'; T['7']:='111';
  write('Введите восьмеричное число x='); readln(x);
  y:='';
  for i:=1 to length(x) do
    y:=y+T[x[i]];
  writeln('Двоичное представление: ', y);
  readln;
end.
```

10. *a)* 1001000111; *b)* 111100001; *c)* 100100111; *d)* 111010.

11.

0	1	1	1	1	1	0	1

12.

1	0	1	0	0	1	0	1

13. *a)* –1; *b)* +64; *c)* –39.

14.

1	1	1	0	0	0	0	0

15. *a)* $0{,}10101001\times2^{5}$; *b)* $-0{,}100100101\times2^{7}$; *c)* $-0{,}11\times2^{-3}$.

16. $b = 8$; $x = 25$.

17. *a)* –26; *b)* 45; *c)* –8.

18. *a)* [–8, 7]; *b)* [–128, 127]; *c)* [–32768, 32767].

19. *a)* –0,375; *b)* 0,125; *c)* –0,5.

20. $A = -0{,}9921875$; $B = 0{,}9921875$.

21. *a)* –0,109375; *b)* 7,5; *c)* –0,25.

22.

1	0	1	0	1	0	1	1

23. $A = -7{,}5$; $B = 7{,}5$.

Тест № 4

1.

x	y	z	$\bar{z}$	$y\bar{z}$	$x \vee y\bar{z}$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	1	0	1

x	y	z	$\bar{z}$	$y\bar{z}$	$x \vee y\bar{z}$
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	0	0	1

2. *a)* и *d)*.

3. {000, 010, 100, 110, 111}.

4.

```
Program RTA20;
{ Ответ на Тест 4, задание 4 }
var x, y, z, f : boolean;
    xx, yy, zz, ff : 0..1;
begin
  write('Введите x='); readln(xx);
  if xx=0 then x:=false else x:=true;
  write('Введите y='); readln(yy);
  if yy=0 then y:=false else y:=true;
  write('Введите z='); readln(zz);
  if zz=0 then z:=false else z:=true;
  f:=x and (not y) or z;
  if f=false then ff:=0 else ff:=1;
  writeln('Значение логического выражения: ', ff);
  readln;
end.
```

5.

```
Program RTA21;
{ Ответ на Тест 4, задание 5 }
var x, y, z, f : boolean;
    xx, yy, zz, ff : 0..1;
begin
  writeln('x y z  f');
  writeln('———');
  for x:=false to true do
    for y:=false to true do
      for z:=false to true do
        begin
          f:=(not x) and (not y) or z;
          if x=false then xx:=0 else xx:=1;
          if y=false then yy:=0 else yy:=1;
          if z=false then zz:=0 else zz:=1;
          if f=false then ff:=0 else ff:=1;
          writeln(xx, ' ', yy, ' ', zz, '  ', ff);
        end;
  readln;
end.
```

6. Область определения : {000, 001, 010, 011, 100, 101, 110, 111}. Область значений: {0, 1}.

7.

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

8.

```
Program RTA22;
{ Ответ на Тест 4, задание 8 }
var x1, x2,x3, y : boolean;
    xx1, xx2, xx3, yy : 0..1;
begin
  write('Введите x1='); readln(xx1);
  if xx1=0 then x1:=false else x1:=true;
  write('Введите x2='); readln(xx2);
  if xx2=0 then x2:=false else x2:=true;
  write('Введите x3='); readln(xx3);
  if xx3=0 then x3:=false else x3:=true;
  y:=x1 and ((not x2) or (not x3));
  if y=false then yy:=0 else yy:=1;
  writeln('Значение логической функции y=', yy);
  readln;
end.
```

9.

```
Program RTA21;
{ Ответ на Тест 4, задание 9 }
var x1,x2, x3, y : boolean;
    xx1, xx2, xx3, yy : 0..1;
begin
  writeln('x1 x2 x3  y');
  writeln('———————');
  for x1:=false to true do
    for x2:=false to true do
      for x3:=false to true do
        begin
          y:=(not x1) and (x2 or x3);
          if x1=false then xx1:=0 else xx1:=1;
          if x2=false then xx2:=0 else xx2:=1;
          if x3=false then xx3:=0 else xx3:=1;
```

```
            if y=false then yy:=0 else yy:=1;
            writeln(' ', xx1, '  ', xx2, '  ', xx3, '  ', yy);
          end;
  readln;
end.
```

10. $2^{2^5} = 4294967296$.

11.

И-НЕ

x_1	x_2	$\overline{x_1 x_2}$
0	0	1
0	1	1
1	0	1
1	1	0

ИЛИ-НЕ

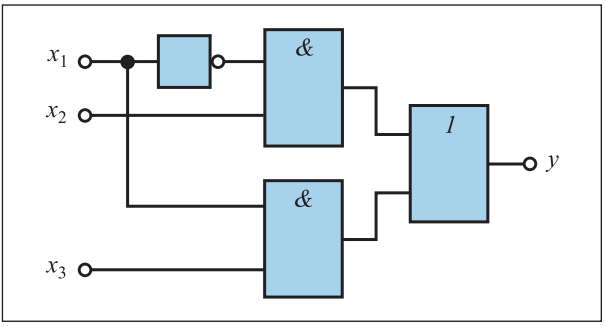
x_1	x_2	$\overline{x_1 \vee x_2}$
0	0	1
0	1	0
1	0	0
1	1	0

12. *a)* и *c)*; *b)* и *d)*.

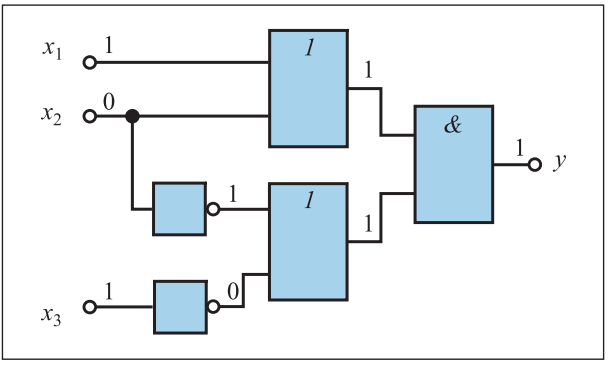
13. *b)*.

Тест № 5

1.



2.



3. $y = (x_1 \vee x_2)\,(\bar{x}_2 \vee \bar{x}_3)$.

4.

```
Program RTA24;
{ Ответ на Тест 5, задание 4 }
var a, b, s, t : char;
begin
  write('Введите a='); readln(a);
  write('Введите b='); readln(b);
  if (a='0') and (b='0') then begin s:='0'; t:='0'; end;
  if (a='0') and (b='1') then begin s:='1'; t:='0'; end;
  if (a='1') and (b='0') then begin s:='1'; t:='0'; end;
  if (a='1') and (b='1') then begin s:='1'; t:='1'; end;
  writeln('s=', s, '  ', 't=', t);
  readln;
end.
```

5. Полусумматор содержит 2 логических элемента *НЕ*, 3 логических элемента *И* и один логический элемент *ИЛИ* (*рис. 5.10*). Элементарный сумматор состоит из двух полусумматоров и логического элемента *ИЛИ* (*рис. 5.11*). Следовательно, элементарный сумматор содержит 4 логических элемента *НЕ*, 6 логических элементов *И* и 3 логических элемента *ИЛИ*. Очевидно, что сумматор на *n* битов состоит из 4*n* логических элементов *НЕ* , 6*n* логических элементов *И* и 3*n* логических элементов *ИЛИ* (*рис. 5.12*). Для сумматора на *n* = 8 битов, получим: 32 логических элемента *НЕ*; 48 логических элементов *И*; 24 логических элемента *ИЛИ*.

6.

```
Program RTA25;
{ Ответ на Тест 5, задание 6 }
label 1;
var A, B, S : string;
    t : char;   { перенос }
    i : integer;
begin
  writeln('Введите два двоичных числа A и B. Каждое число');
  writeln('должно состоять ровно из 8 двоичных цифр.');
  write('A='); readln(A);
  write('B='); readln(B);
  t:='0';        { начальный перенос t=0 }
  S:='xxxxxxxx'; { изначально сумма S неизвестна }
  for i:=8 downto 1 do
    begin
       { сложение двоичных цифр A[i], B[i], T[i-1] }
       if (A[i]='0') and (B[i]='0') and (t='0') then
         begin S[i]:='0'; t:='0'; goto 1; end;
       if (A[i]='0') and (B[i]='0') and (t='1') then
         begin S[i]:='1'; t:='0'; goto 1; end;
       if (A[i]='0') and (B[i]='1') and (t='0') then
         begin S[i]:='1'; t:='0'; goto 1; end;
       if (A[i]='0') and (B[i]='1') and (t='1') then
         begin S[i]:='0'; t:='1';  goto 1; end;
```

```
        if (A[i]='1') and (B[i]='0') and (t='0') then
          begin S[i]:='1'; t:='0'; end;
        if (A[i]='1') and (B[i]='0') and (t='1') then
          begin S[i]:='0'; t:='1'; goto 1; end;
        if (A[i]='1') and (B[i]='1') and (t='0') then
          begin S[i]:='0'; t:='1'; goto 1; end;
        if (A[i]='1') and (B[i]='1') and (t='1') then
          begin S[i]:='1'; t:='1'; goto 1; end;
  1: end;
  writeln('S=', S);
  writeln('Цифра переполнения t=', t);
  readln;
end.
```

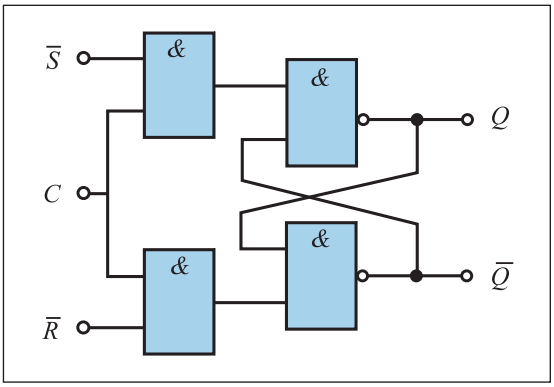
7. (1) – (e); (2) – (g); (3) – (a); (4) – (d); (5) – (h); (6) – (c).

8. y_1 – ВВЕРХ; y_2 – ВНИЗ; y_3 – НАЛЕВО; y_4 – НАПРАВО.

x_1	x_2	y_1	y_2	y_3	y_4
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

9. $Q = 1$ и $\bar{Q} = 0$.

10.



11. (1) – (d); (2) – (i); (3) – (g); (4) – (a); (5) – (c); (6) – (f).

12. 00010011.

13. 11010000.

14.

```
Program RTA26;
{ Ответ на Тест 5, задание 14 }
var R : string;
    n, m, i, j : integer;
```

```
begin
  writeln('Введите начальное содержимое регистра');
  write('R='); readln(R);
  n:=length(R); { емкость регистра }
  write('Введите m='); readln(m);
  for j:=1 to m do
    begin
      { сдвиг на одну позицию }
      for i:=1 to n-1 do R[i]:=R[i+1];
      R[n]:='0';
    end;
  writeln('Содержимое регистра после сдвига');
  writeln('R=', R);
  readln;
end.
```

15. 10010011.

16. 10101001.

17.

```
Program RTA27;
{ Ответ на Тест 5, задание 17 }
label 1;
var A : string;
    n, m, i, j : integer;
    t : char; { перенос }
begin
  writeln('Введите начальное состояние счетчика');
  write('A='); readln(A);
  n:=length(A); { емкость счетчика }
  write('Введите m='); readln(m);
  for j:=1 to m do
    begin
      { прибавление единицы к содержимому счетчика }
      t:='1';
      for i:=n downto 1 do
        begin
          { прибавление переноса к цифре A[i] }
          if (A[i]='0') and (t='0') then
            begin A[i]:='0'; t:='0'; goto 1; end;
          if (A[i]='0') and (t='1') then
            begin A[i]:='1'; t:='0'; goto 1; end;
          if (A[i]='1') and (t='0') then
            begin A[i]:='1'; t:='0'; goto 1; end;
          if (A[i]='1') and (t='1') then
            begin A[i]:='0'; t:='1'; goto 1; end;
     1: end;
    end;
  writeln('B=', A);
  readln;
end.
```

Тест № 6

1. (1) – (i); (2) – (f); (3) – (a); (4) – (c); (5) – (d); (6) – (g).

2. Клавиатура, мышь, монитор, принтер, устройство на магнитном диске, устройство на оптическом диске, считыватель документов (сканер).

3. *a)*, *c)*, *f)*.

4. Магистраль обеспечивает связь процессора с внутренней памятью и периферийными устройствами.

5. См. *рис. 6.2*. Обязательные компоненты: процессор, магистраль, внутренняя память, контроллер, устройство ввода-вывода.

6. *Код команды* – в этом поле указывается операция, которая должна быть выполнена. *Адрес операнда 1* и *Адрес операнда 2* – в этих полях указываются адреса ячеек внутренней памяти, содержащих, соответственно, первый и второй операнды. *Адрес результата* – указывает адрес ячейки внутренней памяти, в которую будет помещен результат операции.

7. *a)* число из ячейки 101 складывается с числом из ячейки 153, а полученный результат помещается в ячейку 342;
b) число из ячейки 508 делится на число из ячейки 391, а полученный результат заносится в ячейку 216;
c) число из ячейки 751 умножается на число из ячейки 852, а полученный результат заносится в ячейку 031;
d) от числа из ячейки 450 вычитается число из ячейки 709, а полученный результат заносится в ячейку 011.

8. *a)*, *c)*, *f)* – команды обработки; *g)* – команда передачи; *e)* – команда перехода; *b)*, *d)* – команды ввода-вывода.

9.

```
INC X
MEM S
SCD Y
ADU X
```

10.

```
01 583
02 461
01 971
02 583
01 461
```

11. *a)*, *d)*, *f)*, *h)* – технические ресурсы; *b)*, *c)*, *e)*, *g)* – программные ресурсы.

12. В случае внешней памяти с последовательным доступом запись/чтение очередной записи возможны только после завершения операций записи/чтения всех предыдущих записей. А в случае внешней памяти с прямым (произвольным) доступом любая запись может быть записана/считана немедленно – без необходимости записи/считывания предыдущих записей.

13. *b)*, *c)*, *e)*.

14. *b)*, *e)*.

15. 2867,2 *Кбайт/с*.

16. *a)*, *c)*, *d)*.

17. *a)*, *b)*, *e)*.

18. Основными параметрами любого компьютера являются:
– скорость выполнения операций;
– емкость внутренней памяти;
– состав, емкость и время доступа устройств внешней памяти;
– состав и соответствующие технические параметры периферийного оборудования;
– масса и габариты;
– стоимость.

19. Основные параметры микропроцессора:
– длина (разрядность) слова;
– частота системных часов (тактового генератора);
– емкость (разрядность) магистралей;
– производительность.

Тест № 7

1. *a)* ≈6 *часов*; *b)* 12 мин. 30 с; *c)* ≈0,73 с; *d)* ≈0,00072 с.

2. *b)*, *d)*.

3. Компьютеры, сетевые адаптеры, коммуникационная структура.

4. *a)*, *c)*.

5. *a)*, *d)*.

6. Сетевой протокол определяет способ адресации компьютеров, размер и состав пакетов данных, алгоритм обнаружения и исправления ошибок, способы физического соединения сетевых адаптеров и сетевых кабелей.

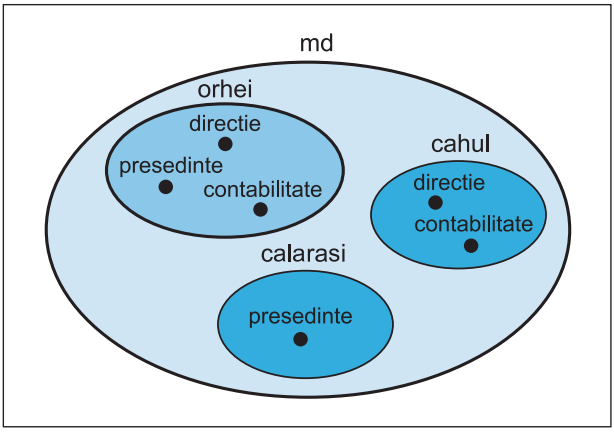
7. Архитектура сети представляет собой набор основных характеристик сети: топология, протоколы обмена информации, технологии взаимодействия в сети.

8. См. *рис. 7.5*.

9. *d)*.

10. В ИНТЕРНЕТе компьютеры идентифицируются при помощи адресов. Адреса могут быть двух типов: числовые и символические.

11. Поскольку $(214)_{10} = (11010110)_2$, а биты 022 имеют значение 110, то данный ИНТЕРНЕТ адрес принадлежит классу *C*. В случае класса *C* адрес подсети указывается в битах 3–32 числового адреса. Из первого байта числового адреса выбираем биты 3–7 и преобразуем полученное двоичное число в десятичную систему: $(10110)_2 = (22)_{10}$. Следовательно, адресом подсети является 22.121.216. Значение 109 последнего байта числового адреса представляет адрес компьютера в подсети.



12. md – домен самого верхнего уровня. Этот домен включает в себя субдомены orhei, cahul и calarasi. Субдомен orhei включает в себя компьютеры directie, contabilitate и presedinte. Субдомен cahul включает в себя компьютеры directie и contabilitate. Субдомен calarasi включает в себя компьютер presedinte.

13. (1) – (g); (2) – (a); (3) – (d); (4) – (f).

14. *b)*.

15. *b)*, *d)*, *e)*.

16. *b)*, *c)*, *d)*, *f)*.

Библиография

1. Bolun Ion. *Inițiere în rețele. INTERNET*. Chișinău, Editura ASEM, 1997.

2. Ceapâru Mihai. *Comunicația prin intermediul rețelelor de calculatoare*. Bucureşti, Editura Tehnică, 1996.

3. Cerchez Emanuela. *Internet. Manual pentru liceu. Filiera teoretică*. Iaşi, Editura Polirom, 2000.

4. Cercez Emanuela, Şerban Marinel. *Informatica pentru gimnaziu*. Iaşi, Editura Polirom, 2002.

5. Gremalschi Anatol, Mocanu Iurie, Spinei Ion. *Informatica. Limbajul PASCAL*. Chişinău, Editura Ştiința, 2003.

6. Gremalschi Ludmila, Mocanu Iurie. *Structura şi funcționarea calculatorului*. *Material didactic pentru licee şi colegii*. Chişinău, Editura Lyceum, 1996.

7. Gremalschi Anatol, Bejan Viorel, Gremalschi Ludmila. *Structura calculatoarelor numerice*. *Material didactic*. Chişinău, U.T.M., 1996.

8. Levine John R. *Internet pentru toți*. Bucureşti, Editura Teora, 1996.

9. Lowe Doug. *Rețele pentru toți*. Bucureşti, Editura Teora, 1995.

10. Mârşanu Radu. *Sisteme de calcul. Manual pentru licee de informatică, clasa a IX-a*. R.A. Bucureşti, Editura Didactică şi Pedagogică, 1995.

11. Mârşanu Radu, Velicanu Manole. *Tehnică de calcul. Manual pentru clasa a XII-a*. Bucureşti, Editura ALL, 1999.

12. Mihoc Dan, Iliescu Sergiu Stelian. *Elemente de informatică. Mecanizarea şi automatizarea producției. Manual pentru licee industriale, clasa a XII-a*. R.A. Bucureşti, Editura Didactică şi Pedagogică, 1995.

13. Mucenic Băşoiu, Mihai Băşoiu, Eugen Ştefan. *Compact disc*. Bucureşti, Editura Teora, 1995.

14. Pfaffenberger Bryan, Petersen Judy. *Dicționar explicativ de calculatoare*. Bucureşti, Editura Teora, 1996.

15. Petrescu Adrian, Iacob Francisc, Racoviță Zoe. *Inițiere în structura calculatoarelor electronice*. Bucureşti, Editura Teora, 1996.

16. Petrescu Silviu. *Informatica aplicată: manual de informatică pentru clasa a XII-a*. Bucureşti, Editura Teora, 1998.

17. Secrieru Nicolae, Gremalschi Anatol, Cornea Ion. *Arhitectura şi organizarea microprocesoarelor*. Chişinău, Editura Universitas, 1995.

18. Velicanu Manole, Vasilescu Adrian. *Bazele informaticii. Manual pentru clasa a XII-a*. Bucureşti, Editura ALL, 1999.

19. Залогова Л.А., Плаксин М.А., Русаков С.В., Русакова О.Л. и др. *Информатика. Задачник-практикум в 2 т.* / Под ред. Семакина И.Г., Хеннера Е.К.: Том 1. – М.: Лаборатория Базовых Знаний, 1999.

20. Семакин И.Г., Залогова Л.А., Русаков С.В., Шестакова Л.В. *Информатика. Базовый курс для 7-9 классов*. – М.: Лаборатория Базовых Знаний, 1999.

Manualul acesta este proprietatea Ministerului Educației al Republicii Moldova.

Liceul/gimnaziul _____

Manualul nr. _____

Anul de folosire	Numele şi prenumele elevului	Anul şcolar	Aspectul manualului	
			la primire	la restituire
1.				
2.				
3.				
4.				
5.				

• Dirigintele trebuie să controleze dacă numele elevului este scris corect.

• Elevul nu trebuie să facă nici un fel de însemnări pe pagini.

• Aspectul manualului (la primire şi la restituire) se va aprecia folosind termenii: *nou, bun, satisfăcător, nesatisfăcător*.

Imprimare la Tipografia „SEREBIA” SRL,
str. Calea Ieşilor, 10;
MD-2069, Chişinău, Republica Moldova
Comanda nr. 110