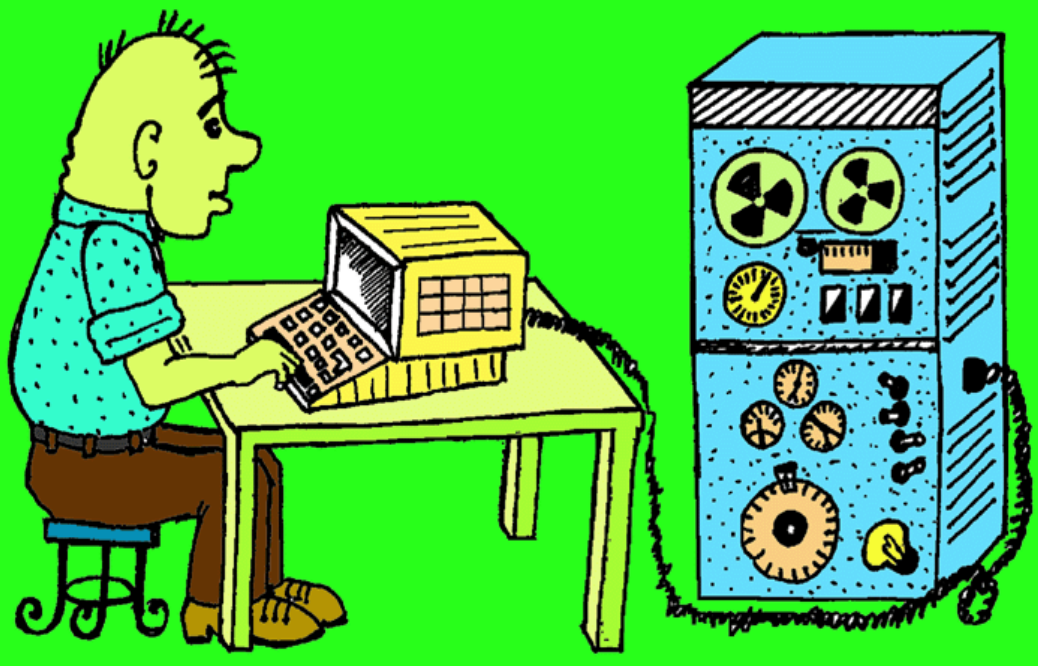
МАЙЕР Р.В.

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ



ЗАДАЧИ И ПРОГРАММЫ

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ГОУ ВПО "Глазовский государственный педагогический институт имени В. Г. Короленко"

Майер Р.В.

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ

задачи и программы на языке Pascal

Глазов 2011

УДК 37.022
ББК 32.81
М14

Печатается по решению кафедры информатики Глазовского государственного педагогического института имени В. Г. Короленко от 15.01.2011 (протокол N 6).

Автор: Майер Роберт Валерьевич.

Рецензенты:

Р. Р. Камалов, доктор педагогических наук, доцент, проректор по науке ГОУ ВПО "Глазовский государственный педагогический институт имени В. Г. Короленко".

А. Г. Русских, кандидат технических наук, доцент, заведующий кафедрой автоматизированных систем управления Глазовского инженерно–экономического института (филиала ИжГТУ).

Майер Р.В. Теоретические основы информатики. Задачи и программы на языке Pascal: Учебн. пособ. для студ. высш. учеб. заведений [Электронный ресурс]. — Глазов: ГГПИ, 2011. — 73 с.

Учебное пособие включает в себя систему учебных задач по следующим темам курса "Теоретические основы информатики": формула Шеннона, кодирование и декодирование, передача информации по каналу связи, автоматы и их модели, машина Поста, машина Тьюринга, алгорифмы Маркова, нейросети и персептроны, основы теории игр. Отличие от других пособий состоит в том, что большая часть рассматриваемых задач предполагает написание компьютерных программ на языке Pascal.

Для преподавателей информатики и студентов вузов, изучающих информатику в качестве профильной дисциплины.

Посети Web–сайт
"Информационные технологии
и физическое образование"
http://maier-rv.glazov.net
http://komp-model.narod.ru

ПРЕДИСЛОВИЕ

Традиционная методика проведения занятий по дисциплине "Теоретические основы информатики" предполагает решение задач на доске с их последующим переписыванием в рабочие тетради. С целью оценки знаний студентов организуются контрольные работы по наиболее важным темам: основы теории кодирования, теория автоматов, абстрактные машины Поста и Тьюринга, нейросети и т.д. Подобный подход имеет недостатки: некоторым студентам неинтересно заниматься абстрактными рассуждениями, им не понятно, как все это использовать на практике. Занятия по информатике становится чем–то похожим на занятия по дискретной математике или логике. Студенты с высокой креативностью не имеют возможности проявить себя в полной мере.

Идея развивающего обучения находит свою реализацию в подходе, предполагающем решение системы специально подобранных учебных задач на компьютере. Имеет смысл часть практических занятий проводить в компьютерном классе, предоставив студентам возможность писать программы и проверять их работу на ПЭВМ. Вместо задания: "закодируйте это сообщение на листочке бумаги" преподаватель предлагает написать программу, кодирующую данное сообщение, что гораздо интереснее. Студент уже не может ограничиться переписыванием решения задачи с доски, — он вынужден самостоятельно или с помощью преподавателя вникать в программный код, вносить в него изменения и т.д. Все это создает новые возможности для активизации мыслительной деятельности обучаемых, делает их рассуждения более конкретными и связанными с практическими задачами, открывает определенные возможности для проявления творческих способностей. По крайней мере для студентов педагогического вуза весьма полезно освежить свои знания по алгоритмизации и программированию, — это пригодится будущим учителям информатики.

Пособие содержит тексты задач, часть из которых решается без компьютера, а часть требует написания программы. Для некоторых задач приведены решения, содержащие логические рассуждения, вычисления и программы на языке Pascal (Borland Pascal 7.0, Free Pascal). Автор рекомендует давать студентам программные коды наиболее важных, ключевых задач, предоставляя им возможность самостоятельно решать простые задачи.

Майер Р.В.

1.1. Вам известно, что Вася бросил кубик. Вася послал сообщение: "Кубик упал на четную грань". Сколько информации в сообщении?

1.2. Вам известно, что Вася бросил кубик. Вася послал сообщение: "Кубик упал на грань 5". Сколько информации в сообщении?

1.3. Маша загадала число x от 1 до 256. Сколько информации содержится в сообщении: 1) $x < 9$; 2) $x > 224$; 3) $x > 97$; 4) $x < 129$?

1.4. Вам известно, что поезд придет завтра где–то между 10.00 и 21.00. Вася послал сообщение: "Поезд придет между 12.00 и 14.00. Сколько информации в сообщении?

1.5. Маша загадала целое число x от 1 до 256. Вася задает вопросы Маше: $x > 100$? $x > 150$? и т.д. Какие вопросы должен задавать Вася, чтобы быстрее отгадать число x? Сколько таких вопросов он должен задать?

1.6. Алфавит содержит 4 буквы, используемые с вероятностями 0,2, 0,3, 0,4 и 0,1. Сообщение состоит из 1 символа. Напишите программу, вычисляющую количество информации в сообщении по формуле Шеннона.

Количество информации в одном символе вычисляется так:

$$I_1 = \sum_{i=1}^{N} -p_i \log_2 p_i = -p_1 \log_2 p_1 - \ldots - p_4 \log_2 p_4.$$

Для нахождения двоичного логарифма в среде Pascal используют формулу: $\log_2 x = \ln x / \ln 2$. *При* $x = 0$ *функция* $\ln x$ *не определена, поэтому вместо нее следует использовать* $\ln(x + 0,0001)$.

1.7. Используя решение предыдущей задачи, определите количество информации I_1 в одном символе, если: 1) $p_1 = p_2 = p_3 = p_4 = 0,25$; 2) $p_1 = p_2 = p_3 = 0,01$, $p_4 = 0,97$. Сравните полученные результаты.

1.8. Проводится опыт с двумя исходами, вероятности которых p_1 и p_2. На компьютере постройте график зависимости энтропии H опыта от вероятности одного из исходов p_1. Когда энтропия H максимальна?

Необходимо построить график функции

$$H(p_1) = \sum_{i=1}^{2} -p_i \log_2 p_i = -p_1 \log_2 p_1 - (1 - p_1) \log_2 (1 - p_1).$$

в интервале [0,01; 0,99]. Следует использовать: $\log_2 x = \ln x / \ln 2$.

1.9. Задано сообщение из N символов на алфавите из 5 букв: a, b, c, d, e. Напишите программу, которая определяет частоту (вероятность) использования каждого символа и по формуле Шеннона вычисляет информацию I_1, приходящуюся на 1 символ, и общее количество информации I в сообщении.

```
uses crt;                                              { ПР - 1.1 }
var n,i,a,q,w,s,e:integer;    r,s1:string;
  h1, pi, pa,pq,pw,ps,pe, h:real;
begin clrscr;
r:='aqwseaeswaqeawseqweasesaeseeewawwqwwesassea';
a:=0; q:=0; w:=0; s:=0; e:=0; N:=length(r);
for i:=1 to length(r) do begin
if r[i]='a' then a:=a+1;
if r[i]='q' then q:=q+1;
if r[i]='w' then w:=w+1;
if r[i]='s' then s:=s+1;
if r[i]='e' then e:=e+1;
end;
Writeln('буква a встречается ',a,' раз');
Writeln('буква q встречается ',q,' раз');
Writeln('буква w встречается ',w,' раз');
Writeln('буква s встречается ',s,' раз');
Writeln('буква e встречается ',e,' раз');
pa:=a/N; pq:=q/N; pw:=w/n; ps:=s/n; pe:=e/n;
Writeln('Pa=',pa); Writeln('Pq=',pq); Writeln('Pw=',pw);
Writeln('Ps=',ps); Writeln('Pe=',pe);
h1:=-N*(((pa*(ln(pa+0.001)/ln(2)))+
(pq*(ln(pq+0.001)/ln(2)))));
h:=h1-N*((pw*(ln(pw+0.001)/ln(2)))+
(ps*(ln(ps+0.001)/ln(2)))+
(pe*(ln(pe+0.001)/ln(2))));
writeln('H=',h); readkey;
end.
```

1.10. С помощью предыдущей программы убедитесь в том, что информативность сообщения I максимальна тогда, когда все символы используются с равными вероятностями. Если в сообщении повторяется один символ, то информативность сообщения равна 0.

1.11. Разрешение монитора 800×600, число цветов — 16. Какой объем видеопамяти нужен для хранения 4 страниц изображения?

1.12. Какой объем имеет аудиофайл, время звучания которого 2 минуты при частоте дискретизации 44,1 кГц и разрешении 16 бит?

2. КОДИРОВАНИЕ И ДЕКОДИРОВАНИЕ

2.1. Закодируйте десятичное число двоичным кодом: а) 743_{10}; б) 334_{10}; в) $61,375_{10}$; г) $160,25_{10}$; д) $131,82_{10}$.

2.2. Переведите двоичное число в десятичную систему счисления а) 10110101_2; б) 11100110_2; в) 01011011_2; г) 11101101_2; д) 01110101_2.

2.3. Закодируйте десятичное число шестнадцатиричным кодом: а) 445_{10}; б) 334_{10}; в) $261,375_{10}$; г) $160,25_{10}$; д) $131,82_{10}$.

2.4. Переведите десятичное число в двоично–десятичную систему счисления: а) 2576_{10}; б) 3137_{10}; в) 1263_{10}; г) 6361_{10}; д) 8431_{10}.

2.5. Напишите программу, переводящую двоичные 16–ти разрядные числа в шестнадцатиричную систему счисления.

2.6. Заданное число переведите в код типа Double: а) -587,375; б) -348,635; в) 273,178; г) -768,156; д) -849,367.

При записи числа в формате с плавающей запятой в ОЗУ число нормализуют, то есть представляют в виде $M \cdot 2^{K_2}$, после чего отдельно записывают порядок K и мантиссу M. Для хранения чисел типа Double используют 8 байт (64 бита), порядок занимает 11 бит и имеет диапазон от -1023 до 1023, мантисса — 52 бита, знак числа — 1 бит. Для упрощения вычислений к значению порядка перед записью в ОЗУ прибавляется смещение $1023_{10} = 1111111111_2$. Самый старший бит указывает знак числа: 0 — "плюс", 1 — "минус". Первая цифра мантиссы всегда 1, она не кодируется. Например, закодируем число $-315,8125_{10}$. Представим в нормализованном виде: $-315,8125_{10} = 100111011,1101_2 = 1,001110111101_2 \cdot 2^8$. Порядок числа $8_{10} = 1000_2$, смещенный порядок $1000_2 + 1111111111_2 = 10000000111_2$. Получаем:

$$1|10000000111|00111011110100000000...00000000000000000000$$

В шестнадцатиричном коде: $C073BD0000000000_{16}$.

2.7. Исходя из кода типа Double восстановите число в десятичном коде: а) $407E530000000000_{16}$; б) $15A623000000000_{16}$; в) $F23D41E000000000_{16}$; г) $209B572000000000_{16}$; д) $83F179A000000000_{16}$.

2.8. Напишите программу, переводящую трехзначное десятичное число в: а) двоичное; б) шестнадцатиричное; в) двоично–десятичное.

2.9. Напишите программу, переводящую десятичное число в шестнадцатиричную систему счисления.

Используется программа ПР – 2.1. Десятичное число, которое необходимо перевести в шестнадцатиричную систему счисления, при-

сваивается переменной a. В начале цикла значение переменной a присваивается переменной b, а переменной a присваивается целая часть от деления a на 16. После этого определяется разность $b - 16a$, соответствующая цифра записывается в младший разряд. После этого все повторяется снова.

```
uses crt;                                              { ПР - 2.1 }
var a,b,c,d:integer;
label m;
BEGIN clrscr; write('введите десятичное число ');
  read(a); c:=20;
  Repeat
    c:=c-1; b:=a; a:=a div 16;
    d:=b-16*a; gotoxy(c,5);
    If d=10 then begin write('A'); goto m; end;
    If d=11 then begin write('B'); goto m; end;
    If d=12 then begin write('C'); goto m; end;
    If d=13 then begin write('D'); goto m; end;
    If d=14 then begin write('E'); goto m; end;
    If d=15 then begin write('F'); goto m; end;
    write(d);
    m:
  until a=0;
  ReadKey;
END.
```

2.9. Напишите программу, восстанавливающую число, записанное в формате с плавающей запятой, и выводящее его на экран в двоичной и десятичной системе счисления.

2.10. Напишите программу, которая переводит двоичное число вида 10..1,11..1 в формат с плавающей запятой (отдельно кодируются мантисса и порядок). На мантиссу отводится 3 байта, на порядок — 1 байт.

2.11. Сколько символов алфавита могут быть закодированы двоичным кодом длиной: а) 2 знака; б) 4 знака; в) 8 знаков.

2.12. Сколько символов алфавита могут быть закодированы восьмиричным кодом длиной: а) 2 знака; б) 4 знака; в) 8 знаков.

2.13. Сколько символов алфавита могут быть закодированы шестнадцатиричным кодом длиной: а) 2 знака; б) 4 знака; в) 8 знаков.

2.14. Напишите программу, кодирующую сообщения кодом ASCII.

2.15. Составьте равномерный код русского алфавита, в котором все кодовые комбинации содержат по три знака 0, 1, 2, 3. Напишите

программу, кодирующую и декодирующую сообщение из 25 букв.

2.16. Напишите программу, считывающую сообщение из файла 1.txt, кодирующую его путем замены букв на другие символы (цифры), и записывающую его в файл 2.txt.

2.17. Напишите программу, считывающую закодированное сообщение из файла 2.txt, декодирующую его, и записывающую его в 3.txt.

2.18. Напишите программу, генерирующую случайное сообщение в алфавите $A = a, b, c, d$. Вероятности используемых букв соответственно равны: 0,32, 0,23, 0,30, 0,15.

2.19. Напишите программу, кодирующую двоичным кодом сообщение из 20 букв на алфавите из 8 букв. На каждую букву приходится по 3 двоичных разряда: 000, 001, 010, ... , 111.

```
uses crt;                                              { ПР - 2.2 }
var  f,s,a,b:string; i,k:integer; x:real;
begin  clrscr;
  s:='aaabbcbddeeebbcadeaabbddeeffgh';
  writeln(s);  b:='';
  for i:=1 to length(s) do  begin
    a:=s[i];
    if a='a' then b:=b+'000';  if a='b' then b:=b+'001';
    if a='c' then b:=b+'010';  if a='d' then b:=b+'011';
    if a='e' then b:=b+'100';  if a='f' then b:=b+'101';
    if a='g' then b:=b+'110';  if a='h' then b:=b+'111';
    end; write(b);  writeln;  randomize;  s:='';
  for i:=1 to length(b) do  begin   f:=copy(b,3*i-2,3);
    if f='000' then s:=s+'a'; if f='001' then s:=s+'b';
    if f='010' then s:=s+'c';
    if f='011' then s:=s+'d';
    if f='100' then s:=s+'e';
    if f='101' then s:=s+'f';
    . . . . . .
    end; write(s);  readln;
end.
```

2.20. Создайте программу, декодирующую сообщения, закодированные предыдущим способом.

2.21. Создайте программу, шифрующую сообщение на английском языка путем их перемешивания (первый с третьим, четвертый с шестым и т.д.) и добавления новых символов.

2.22. Напишите программу, которая дешифрует зашифрованное в предыдущей задаче сообщение.

3. ПЕРЕДАЧА ИНФОРМАЦИИ ПО КАНАЛУ СВЯЗИ

3.1. Имеется сообщение из 30 букв на алфавите из 8 букв. Напишите программу, которая кодирует каждую букву тремя битами, случайно с вероятностью $p = 0,1$ вносит ошибки (инвертирует биты) и декодирует сообщение. Результат каждого действия должен выводиться на экран.

3.2. Имеется сообщение 01101 ... 01 из 30 бит. Напишите программу, кодирующую его помехоустойчивым кодом, в котором каждый бит утраивается, затем вносит ошибки с заданной вероятностью $p = 0,1$ и декодирует сообщение.

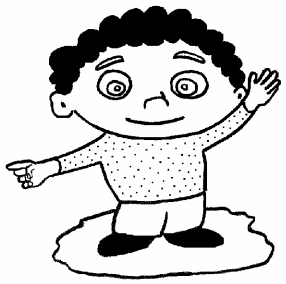
```
uses crt;                                              { ПР - 3.1 }
var aa1,aa,y,z,zz,x1,x : string;
  i : byte;  s,k,p : real;
BEGIN clrscr;
  writeln('Исходное сообщение: ');
  aa1:='10101010101010101010101010101010101010101010101';
  writeln(aa1);  writeln('Задайте вероятность ошибки: ');
  readln(p);
  for i:=1 to length(aa1) do begin x:=copy(aa1,i,1);
   if x='0' then y:='000';
   if x='1' then y:='111'; z:=z+y;
   end; writeln;
  writeln('Закодированное сообщение: '); writeln(z);
  randomize;   zz:='';
  for i:=1 to length(z) do begin
      s:=random(1000)/1000;
      if (s<p)and(z[i]='0') then zz:=zz+'1';
      if (s<p)and(z[i]='1') then zz:=zz+'0';
      if s>=p then zz:=zz+z[i];
    end; writeln;
  writeln('Сообщение с ошибкой: '); writeln(zz);
  aa:='';  i:=1;
  while i < length(zz) do begin
      x:=copy(zz,i,3);
      if x='000' then y:='0'; if x='001' then y:='0';
      if x='010' then y:='0'; if x='011' then y:='1';
      if x='100' then y:='0'; if x='101' then y:='1';
      if x='110' then y:='1'; if x='111' then y:='1';
      aa:=aa+y; i:=i+3;
  end; writeln;
```

```
  writeln('После декодера: ');  writeln(aa);
  for i:=1 to length(aa) do begin
      x1:=copy(aa1,i,1); x:=copy(aa,i,1);
      if x1<>x then k:=k+1;
    end; k:=k/length(aa);
  writeln('Отн. кол-во ошибок: ',k); readln;
END.
```

3.3. Используя предыдущую программу, исследуйте зависимость относительного числа ошибок $k = n_{\text{ош}}/N$ декодирования от вероятности p, с которой инвертируются биты при передаче сообщения по каналу связи. Величину k определите, сравнивая исходное и конечное сообщения. Постройте график зависимости $k = k(p)$.

3.4. Имеется сообщение 01101 ... 01. Напишите программу, которая разбивает его на кадры по 7 бит и добавляет восьмой бит четности так, чтобы количество единиц в байте было бы четным.

```
uses crt,dos;                                        { ПР - 3.2 }
var  s,s1,a,a1,b,b1,p,f:string;
  i,i1,j,j1,k,k1,q,q1,n,x1,t,x:integer; w:real;
BEGIN
 clrscr; n:=49; s1:=''; randomize;
 For i:=1 to 70 do begin
   if random(1000)/1000<0.5 then s:=s+'1' else s:=s+'0'
 end;
 writeln('Исходное сообщение '); writeln(s);
 writeln('Сообщение с битом четности');
 for i:=1 to round(length(s)/7) do begin
   a:=copy(s,7*i-6,7); x:=0;
   for j:=1 to 7 do begin
     b:=copy(a,j,1);
     If b='1' then x:=x+1; end;
   if (x mod 2)=0 then a:=a+'0' else a:=a+'1'; writeln(a);
   f:=f+a; end; writeln;
 for i:=1 to length(f) do begin
   t:=t+1; w:=random(100)/100;
   if (w<0.1) and (f[i]='1') then f[i]:='0';
   if (w<0.1) and (f[i]='0') then f[i]:='1';
 end; writeln('Сообщение с ошибкой');
 writeln(f); readln;
END.
```

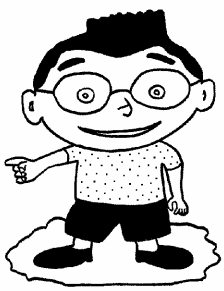


3.5. Дополните предыдущую программу так, чтобы она вносила бы ошибки в сообщение, получающееся после добавления битов четности, а затем выявляла бы их.

3.6. Напишите программу, которая кодирует сообщение "1101011.." используя (5,2)–код, а затем осуществляет декодирование. При этом блок из 2 символов кодируется блоком из 5 символов по правилу: $a_1 = x_1$, $a_2 = x_1$, $a_3 = x_1 \oplus x_2$, $a_4 = x_2$, $a_5 = x_2$. Декодирующая функция: $b' = b_1 \oplus b_3 \oplus b_5, y_1 = b_1 \oplus (b_1 \oplus b_2)b', y_2 = b_5 \oplus (b_5 \oplus b_4)b'$.

3.7. Дополните программу, написанную при решении предыдущей задачи, так, чтобы она моделировала передачу сообщения по каналу с шумом, в котором с заданной вероятностью q инвертируются биты.

```
uses crt,dos;                                          { ПР - 3.3 }
var s,t,u,x1,x2,a1,a2,a3,a4,a5,o,o1,o2:string;
y1,y2,b1,b2,b3,b4,b5,bb,d1,d2,i,code:integer; q,z:real;
BEGIN clrscr; s:='10101101010100011000101110101101';
  Randomize; writeln('Исходное сообщение ',s); t:='';
  for i:=1 to round(length(s)/2) do begin
  x1:=copy(s,2*i-1,1); x2:=copy(s,2*i,1);
  a1:=x1; a2:=x1; val(x1,d1,code); val(x2,d2,code);
  str((d1+d2)mod 2,a3); a4:=x2; a5:=x2;
  u:=u+a1+a2+a3+a4+a5+' ';
  end; writeln('После кодера ',u); t:=''; q:=0.05;
  for i:=1 to length(u) do begin z:=random(100)/100;
  if (copy(u,i,1)=' ') then t:=t+' ';
  if (z>=q)and(copy(u,i,1)<>' ') then t:=t+copy(u,i,1);
  if (z<q)and(copy(u,i,1)<>' ') then begin
  if (copy(u,i,1)='1') then t:=t+'0' else t:=t+'1'; end;
  end; writeln('После канала ',t);
  for i:=1 to round(length(t)/6) do begin
  val(copy(t,6*i-5,1),b1,code);
  val(copy(t,6*i-4,1),b2,code);
  val(copy(t,6*i-3,1),b3,code);
  val(copy(t,6*i-2,1),b4,code);
  val(copy(t,6*i-1,1),b5,code); bb:=(b1+b3+b5)mod 2;
  y1:=(b1+((b1+b2)mod 2)*bb)mod 2; str(y1,o1);
  y2:=(b5+((b5+b4)mod 2)*bb)mod 2; str(y2,o2);
  writeln(b1,b2,b3,b4,b5,' ',y1,' ',y2);
  o:=o+o1+o2;
  end; writeln('Исходное       ',s);
  writeln('После декодера ',o); Readln;
END.
```



3.8. Напишите программу, кодирующую четыре бита семью битами по следующему правилу: к четырем информационным битам i_1, i_2, i_3, i_4 добавляются три бита четности p_1, p_2, p_3 так, что $p_1 = i_1 \oplus i_2 \oplus i_3$, $p_2 = i_2 \oplus i_3 \oplus i_4, p_3 = i_3 \oplus i_2 \oplus i_4$.

3.9. Дополните программу из предыдущей задачи так, чтобы она случайно вносила ошибки (инвертировала биты), декодировала сообщение, выявляла кадры с ошибками и исправляла их.

3.10. Источник вырабатывает сообщение 10110010, кодер разбивает его на блоки длиной $N-1$ и добавляет 1 бит четности так, что получаются кадры длиной N. Они поступают в канал связи, в котором с вероятностью p вносятся ошибки (инвертируются биты), и, пройдя через него, попадают в декодер. На передачу 1 бита затрачивается 1 такт машинного времени (допустим, 1 мс). Реализуется система с переспросом: декодер выявляет кадры с ошибками и по каналу переспроса посылает сигнал о повторе передачи соответствующего кадра. На его повторную передачу снова затрачивается N тактов. Сигнал по каналу переспроса не вносит задержки. Промоделируйте этот процесс на ПЭВМ, определите скорость передачи.

Для компьютерного моделирования может быть использована программа ПР - 3.4. Длина кадра N равна величине константы $Dlina\_k$, при этом число информационных бит составляет $Dlina\_k - 1$, число кадров равно $Chislo\_k = 500$. Скорость передачи определяется так: $skorost = Chislo\_k(Dlina\_k - 1)/t$, где $(Dlina\_k - 1)$ — число информационных бит, t — число затраченных тактов. Скорость передачи зависит от длины кадров и вероятности ошибки.

```
uses crt, dos;                                   { ПР - 3.4 }
const kadr=5; p=0.07;
var i,Ch_kadrov :integer; t :longint; q,skorost: real;
BEGIN  Ch_kadrov:=2000;
  For i:=1 to Ch_kadrov do
   begin t:=t+kadr;  writeln('KADR ', i);
    If random(1000)/1000<q*kadr then begin t:=t+1;
       writeln('Oshibka v kadre ',i,' ',t);
       i:=i-1; end
       else write('Pravilno ',t,' ');
   end; skorost:=Ch_kadrov/t*(kadr-1);
 writeln(t,' ',skorost); Readkey;
END.
```

---

11. Используя решение предыдущей задачи, изучите зависимость скорости передачи информации по каналу связи от вероятности ошибки p, постройте графики. Длина кадра равна: 1) 4 бит; 2) 8 бит.

При увеличении вероятности ошибки растет частота переспросов, скорость передачи информации уменьшается. Поэтому получается убывающая кривая, которая с увеличением p приближается к 0.

3.12. Постройте график зависимости скорости передачи информации от длины кадра, если вероятность ошибки равна 0,02; 0,1; 0,3.

Если $p = 0,1$, то при длине кадра 2 или 3 бита скорость передачи невелика за счет большого числа проверочных битов четности. С ростом длины кадра она уменьшается из-за увеличения вероятности ошибки в кадре. Существует некоторая оптимальная длина кадра, при которой скорость передачи информации максимальна.

3.13. Считая, что емкость канала связи C равна максимальной скорости передачи информации, изучите зависимость емкости канала от вероятности ошибки, постройте график. Сравните полученные результаты с расчетными значениями для двоичного симметричного канала с шумом.

Емкость (пропускная способность) двоичного симметричного канала связи с вероятностями ошибки p и правильной передачи $(1 - p)$ равна $C = 1 + p \log_2 p + (1 - p) \log_2 (1 - p)$. При $p = 0$ (ошибок нет) емкость канала максимальна и равна 1; когда $p = 0,5$, емкость канала $C = 0$. Для нахождения емкости C моделируемого канала связи с переспросом зададим вероятность ошибки $0,05$ и найдем скорости передачи при различных длинах кадра: 2, 3, 4, ..., 64 бит. Обнаружим, что при $N = 5$ скорость передачи максимальна и равна 0,60, – это и есть емкость канала связи C. Повторим эту процедуру при других p, каждый раз определяя максимальную скорость передачи.

Построим график зависимости емкости моделируемого канала связи от вероятности ошибки. Видно, что при увеличении p от 0 до 0,5 она уменьшается от 1 до 0. Эта зависимость похожа на расчетную кривую для двоичного симметричного канала связи, но точного совпадения нет.

3.14. Имеется сообщение 01101...01 из 30 бит. Напишите программу, которая кодирует его помехоустойчивым кодом, утраивающим каждый бит, затем с заданной вероятностью $p = 0,1$ инвертирует биты и декодирует сообщение, исправляя внесенные ошибки.

3.15. Имеется сообщение 01101...01. Напишите программу, которая разбивает его на кадры по 7 бит и добавляет восьмой бит четности, чтобы количество единиц в байте было бы четным. Дополните ее так, чтобы она вносила бы ошибки в сообщение, получающееся после добавления битов четности, а затем выявляла бы кадры с ошибками.

4.1. Имеется программа, моделирующая функционирование детерминированного автомата. Изучите ее работу, постройте диаграмму Мура, таблицу переходов, таблицу выходов и систему команд.

Компьютерная программа ПР - 4.1 моделирует работу автомата с четырьмя внутренними состояниями $q = 1, 2, 3, 4$, на вход которого поступает последовательность символов S:="baba ... acbb". При ее работе на экране появляется номер состояния автомата и символ на его выходе в результате первого, второго, третьего и последующих шагов.

```
uses crt;                                              { ПР - 4.1 }
var i: integer; x,y,S,a,b,c: string;
    q: integer; label m;
BEGIN S:='babacaabcbabcbcbaaccbcbabcbccaacbb';
writeln('S=',S);q:=1;
For i:=1 to length(S) do begin x:=copy(S,i,1);
If (x='a')and(q=1) then begin q:=1; y:='B'; goto m; end;
If (x='c')and(q=1) then begin q:=2; y:='A'; goto m; end;
If (x='b')and(q=1) then begin q:=4; y:='C'; goto m; end;
If (x='b')and(q=2) then begin q:=2; y:='D'; goto m; end;
If (x='a')and(q=2) then begin q:=3; y:='B'; goto m; end;
If (x='c')and(q=2) then begin q:=4; y:='A'; goto m; end;
If (x='a')and(q=3) then begin q:=4; y:='C'; goto m; end;
If (x='c')and(q=3) then begin q:=3; y:='E'; goto m; end;
If (x='b')and(q=3) then begin q:=2; y:='C'; goto m; end;
If (x='a')and(q=4) then begin q:=4; y:='E'; goto m; end;
If (x='b')and(q=4) then
        begin q:=3; y:='F'; goto m; end;
If (x='c')and(q=4) then
        begin q:=1; y:='D'; goto m; end;
m: write(q,' ',y,' | ');
end; ReadKey;
END.
```

4.2. Нарисуйте диаграмму Мура для детерминированного автомата с тремя внутренними состояниями, входным алфавитом $X = a, b, c$ и выходным алфавитом $Y = A, B, C, D$. Автомат должен реагировать на каждый входной символ. Напишите компьютерную программу, моделирующую его работу.

4.3. Промоделируйте вероятностный автомат с тремя внутренними состояниями, диаграмма Мура которого представлена на рис. 4.1.1. Запишите для него систему команд, указав вероятности.

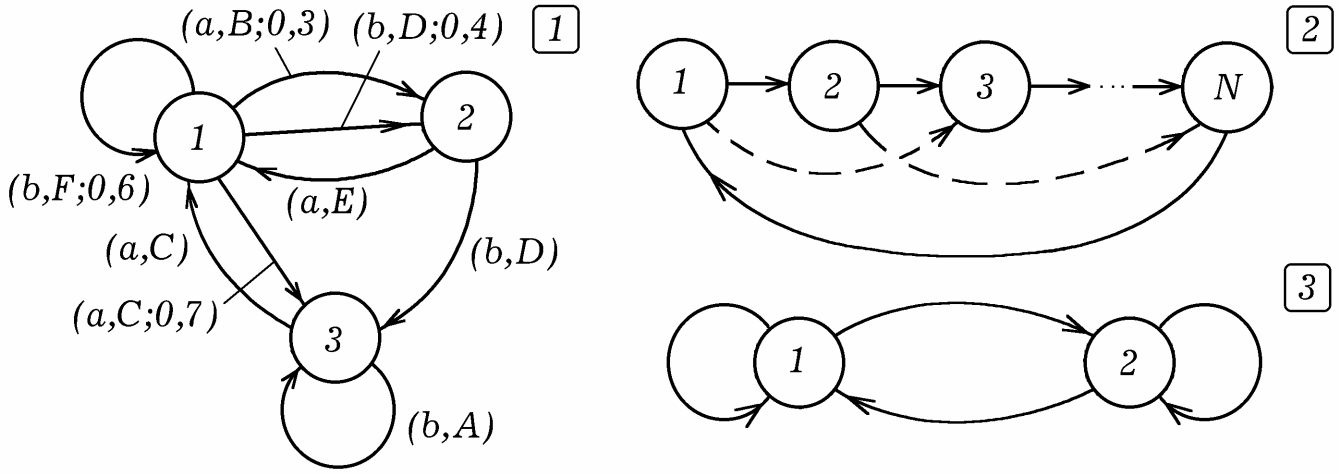
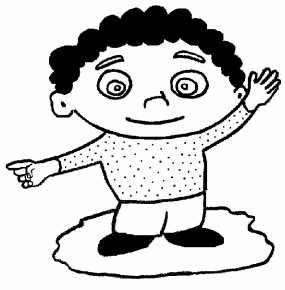


Рис. 4.1. Диаграммы Мура абстрактных автоматов.

Система команд: $1,a \to \{(2,B;0,3),(3,C;0,7)\}$, $1,b \to \{(1,F;0,6),\ (2,D;0,4)\}$, $2,a \to (1,E)$; $2,b \to (3,D)$; $3,a \to (1,C)$; $3,b \to (3,A)$. *Программа ПР–4.2, моделирующая этот вероятностный автомат, представлена ниже.*

```
uses crt;                                              { ПР - 4.2 }
var i: integer; x,y,S: string;
    r: real; q: integer; label m;
BEGIN Randomize; S:='babaaabbabbabbababbaa'; q:=1;
  For i:=1 to length(S) do begin
  x:=copy(S,i,1); r:=random(100)/100;
  If (q=1)and(x='a')and(r<0.3) then
                              begin q:=2; y:='B'; goto m; end;
  If (q=1)and(x='a')and(r>0.3) then
                              begin q:=3; y:='C'; goto m; end;
  If (q=1)and(x='b')and(r<0.4) then
                              begin q:=2; y:='D'; goto m; end;
  If (q=1)and(x='b')and(r>0.4) then
                              begin q:=1; y:='F'; goto m; end;
  If (q=2)and(x='a') then begin q:=1; y:='E'; goto m; end;
  If (q=2)and(x='b') then begin q:=3; y:='D'; goto m; end;
  If (q=3)and(x='a') then begin
     q:=1; y:='C'; goto m; end;
  If (q=3)and(x='b') then begin
     q:=3; y:='A'; goto m; end;
  m: writeln(x,' ',q,' ',y,' | ');
  end; ReadKey;
END.
```



4.4. Создайте компьютерную модель своего вероятностного автомата с тремя внутренними состояниями. Постройте для него соответствующую диаграмму Мура.

4.5. Создайте компьютерную модель ученика, представляющего собой вероятностный автомат, который до обучения выполняет случайную последовательность действий, а после обучения — правильную последовательность действий. Учтите забывание, утрачивание навыка с течением времени.

Ученик обучен, если выполняет правильную последовательность действий: $1 \to 2 \to 3 \to 4 \to ... \to N \to 1 \to 2...$ *(рис. 4.1.2), не совершая при этом неправильных переходов* $1 \to 3$, $2 \to N$ *и т.д. Переходя в другое состояние, ученик с вероятностью* p *выполняет правильное действие, либо с вероятностью* $(1-p)$ *неправильное действие. Поэтому можно ограничиться рассмотрением работы вероятностного автомата с двумя состояниями 1 или 2 (рис. 4.1.3). До обучения автомат случайно переходит из одного состояния в другое:* $1 \to 1 \to 2 \to 2 \to 2 \to 1 \to 2 \to 2 \to 1...$, *а после — выполняет правильную последовательность действий:* $1 \to 2 \to 1 \to 2 \to 1 \to 2 \to 1 \to ...$

Изначально автомат необучен, пусть вероятность правильного действия $p = 0.01$. *Программа ПР - 4.3, моделирующая ученика, содержит цикл, в котором выбор каждой операции осуществляется с помощью генератора случайных чисел. Если случайное число* x *из интервала* $[0;1]$, *меньше* p, *то ученик совершает правильное действие, если нет, — делает ошибку. Процесс обучения приводит к изменению матрицы вероятностей: вероятность правильного выбора* p *увеличивается на* αq, *где* α *— коэффициент научения, а вероятность ошибки* q *уменьшается на ту же величину. Степень сформированности навыка характеризуется вероятностью* p *правильного перехода. Чтобы учесть забывание необходимо на каждом временном шаге уменьшать* p *на* gp *(*g *— коэффициент забывания) и на такую же величину увеличивать* $q : p := p - g * p;\ q := q + g * p;$. *При этом* $p + q = 1$.

```
uses dos, crt, graph;                                  { ПР - 4.3 }
var t,Gd,Gm : integer; x,p,q,a,g : real;
BEGIN
  Gd:=Detect; InitGraph(Gd, Gm, 'c:\bp\bgi');
  Randomize;
  line(0,450,640,450); line(10,0,10,480);
  p:=0.01; q:=1-p; a:=0.003; g:=0.0004;
  Repeat
   inc(t); x:=random(1000)/1000;
   If (x>p)and(t<4000) then
```

```
   begin  p:=p+a*q; q:=q-a*q; end;
   p:=p-g*p; q:=q+g*p; {забывание}
   circle(10+round(t/15),450-round(400*p),2);
  until (t>10000)or(KeyPressed);
  Repeat until KeyPressed; CloseGraph;
END.
```

Программа ПР–4.3 позволяет промоделировать ситуации:

1. Обучение с поощрением: при выполнении правильного действия ученика "поощряют", пересчитывая вероятности p и q. Так как сначала ученик ошибается гораздо чаще (q превосходит p), то обучение происходит медленно (рис. 4.2.1). Зато по мере увеличения "знаний" вероятность совершения правильного действия растет. Акты обучения происходят все чаще, вероятность p увеличивается до 1. Программа содержит строку: $If\ (x < p)and(t < 4000)\ then\ begin\ p := p + a * q;\ q := q - a * q;\ end;$

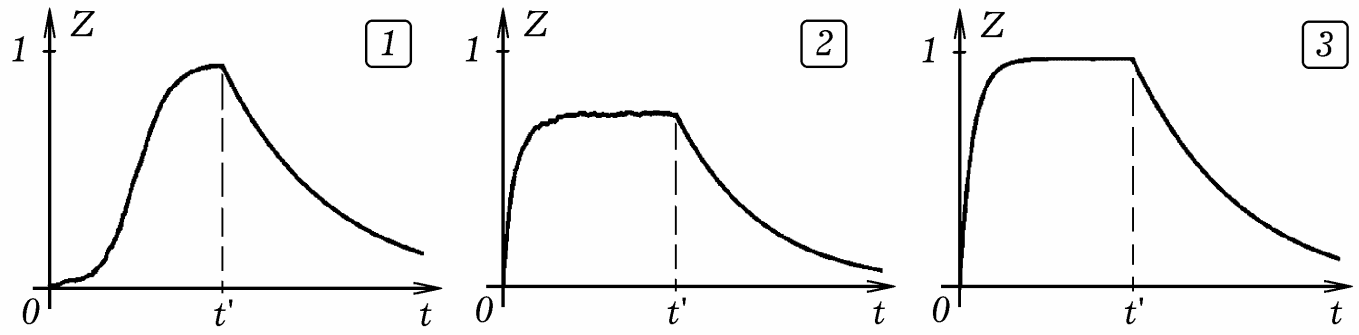


Рис. 4.2. Кривые научения при различных способах обучения.

2. Обучение с наказанием: в случае ошибки ученика "наказывают", подсказывая ему правильный ответ, что приводит к росту p и уменьшению q. Сначала ученик ошибается часто, поэтому уровень его "знаний" быстро растет, вероятность ошибки q падает (рис. 4.2.2). Акты обучения происходят реже, вероятность правильного действия не достигает 1 (за счет забывания). Программа содержит строку: $If\ (x > p)and(t < 4000)\ then\ begin\ p := p + a * q;\ q := q - a * q;\ end;$

3. Обучение с поощрением и наказанием: при правильном ответе ученика поощряют, а при неправильном наказывают, подсказывая правильный ответ. В обоих случаях вероятность правильного действия p растет, а вероятность ошибки q снижается. За счет того, что при любом действии учащегося его учат, уровень знаний быстро растет и достигает 1 (рис. 4.2.3). Программа содержит строку: $If\ t < 4000\ then\ begin\ p := p + a * q;\ q := q - a * q;\ end;$

4.6. Промоделируйте работу исполнителя, перемещающегося по вертикальной поверхности в соответствии с заданной программой.

Используется программа ПР–4.4. Последовательность команд исполнителя записана в виде: $a :=' drd...luld'$, где u – вверх, d – вниз, r – вправо, l – влево. Программа содержит цикл, в котором вырезается по одному символу из последовательности $a :=' drd...luld'$, в результате чего светящаяся точка на экране монитора смещается с некоторым шагом вверх, вниз, вправо или влево.

```
uses crt, graph;                                    { ПР - 4.4 }
Var k4,k1,k,k2,k3,i,Gd,Gm,p: integer; a,x: string;
BEGIN  Gd:=Detect; InitGraph(Gd, Gm, 'c:\BP\BGI');
 moveto(320,240); a:='drdrdruuulululululululddddddd'; k:=0;
 For i:=1 to length(a) do
   begin x:=copy(a,i,1);
   If x='r' then begin k1:=k+30; linerel(k1,0) end;
   If x='u' then begin k2:=k+30; linerel(0,k2) end;
   If x='l' then begin
      k3:=k-30; linerel(k3,0) end;
   If x='d' then begin
      k4:=k-30; linerel(0,k4) end;
   delay(5000);
   end; Readkey; CloseGraph;
END.
```

4.7. Дополните программу ПР–4.4 так, чтобы исполнитель перемещался не только вверх, вниз, вправо и влево, но и по диагонали (на 45^o): вправо–вверх, влево–вниз и т.д.

4.8. Напишите программу, моделирующую движение черепашки, описываемое алгоритмом, составленным из трех команд: Повторить(число раз), Поворот(угол), Вперед(смещение).

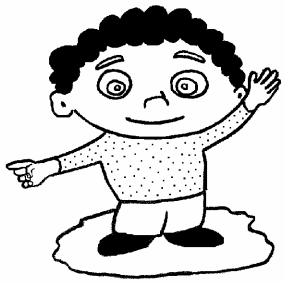
Рассмотрим алгоритм:

```
Повторить_40_раз{
    Поворот(-20); Вперед(4*t);
    Поворот(60); Вперед(5*t);
    Поворот(-100); Вперед(10);   }
```

Для его реализации используется программа ПР–4.5.

```
uses crt, graph;                                    { ПР - 4.5 }
var ugol,dlina,Gd,Gm,t: integer; alfa,x,y,x1,y1: real;
Procedure Napravo(ugol: integer);
begin alfa:=alfa+ugol*3.14/180; end;
Procedure Shag(dlina: integer);
```

```
begin circle(round(x),round(y),2);
x1:=x+dlina*cos(alfa); y1:=y+dlina*sin(alfa);
line(round(x),round(y),round(x1),round(y1)); x:=x1; y:=y1;
end;
BEGIN x:=320; y:=240;
  Gd:=Detect; InitGraph(Gd, Gm, 'c:\bp\bgi');
Repeat inc(t); Napravo(-10); Shag(4*t);
  Napravo(60);   Shag(5*t);
  Napravo(-100); Shag(10);
until t>40; Readkey; CloseGraph;
END.
```



4.9. Напишите тестирующую программу, проверяющую умение решать 10 примеров по арифметике и ставящую оценку. Числа в примерах должны выбираться случайно.

Используемая программа ПР–4.6 содержит цикл, в котором задаются случайные значения переменных a и b и генерируется пример типа "$a \cdot b = ?$". Тестируемый вводит ответ x, компьютер сравнивает его с правильным ответом $c = ab$. Определяется число правильных ответов, ставится оценка.

```
uses crt;                                          { ПР - 4.6 }
var s,i,j,a,b,c,n:integer;
BEGIN
clrscr; randomize; s:=0;
For j:=1 to 5 do
  begin
   a:=random(9); b:=random(9); c:=a*b; writeln;
   writeln('Сколько будет     ',a,'*',b );
   writeln('ответ'); readln(n);
   If n=c then begin write('ВЕРНО'); s:=s+1;
   If n<>c then write('НЕТ'); end;
   writeln('Число правильных ответов=',s);
  end;
  If s=5 then begin write ('оценка 5'); end;
  If s=4 then begin write ('оценка 4'); end;
  If s=3 then begin write ('оценка 3'); end;
  If s=2 then begin write ('оценка 2'); end;
  If (s=1) or (s=0) then
      begin writeln('попробуйте заново'); end;
  ReadKey;
END.
```

4.10. Промоделируйте работу гомеостата Эшби — адаптирующегося устройства с тремя степенями свободы x_1, x_2, x_3, которое при выводе из положения равновесия (0,0,0) самостоятельно в него возвращается.

Гомеостат описывается тремя дифференциальными уравнениями:

$$dx_1/dt = a_{11}x_1 + a_{12}x_2 + a_{13}x_3,$$
$$dx_2/dt = a_{21}x_1 + a_{22}x_2 + a_{23}x_3,$$
$$dx_3/dt = a_{31}x_1 + a_{32}x_2 + a_{33}x_3.$$

Программа ПР–4.7 для нахождения коэффициентов a_{ij}, определяющих работы гомеостата, работает так. Значения a_{ij} и x_j задаются случайным образом. Вычисляются скорости $v_1 = dx_1/dt$, $v_2 = dx_2/dt$, $v_3 = dx_3/dt$, и координаты x_1, x_2, x_3 в последовательные моменты времени. Если модуль хотя бы одной из координат превысил 1, то коэффициенты a_{ij} изменяются случайным образом. Так продолжается до тех пор, пока система не вернется в положение равновесия (0,0,0). Найденные значения коэффициентов a_{ij} печатаются в файл.

```
uses dos, crt, graph;                              { ПР - 4.7 }
const n=3; dt=0.01;
var x1,x2,x3,v1,v2,v3 : real; Gd, Gm,i,j,k : integer;
    a: array[1..n,1..n]of real; F: text;
BEGIN
  Assign(F,'111.bak');Append(F);
  Gd:=Detect; InitGraph(Gd, Gm, 'c:\bp\bgi');
  Randomize; x1:=0.1;
  For i:=1 to n do
           For j:=1 to n do a[i,j]:=0.1-random(200)/1000;
  line(10,100,640,100);line(100,10,100,480);
  line(400,10,400,480);
  Repeat
    v1:=a[1,1]*x1+a[1,2]*x2+a[1,3]*x3;
    v2:=a[2,1]*x1+a[2,2]*x2+a[2,3]*x3;
    v3:=a[3,1]*x1+a[3,2]*x2+a[3,3]*x3;
    x1:=x1+v1*dt; x2:=x2+v2*dt; x3:=x3+v3*dt;
    If abs(x1)>1 then
           For j:=1 to n do a[1,j]:=0.1-random(200)/1000;
    If abs(x2)>1 then
           For j:=1 to n do a[2,j]:=0.1-random(200)/1000;
    If abs(x3)>1 then
           For j:=1 to n do a[3,j]:=0.1-random(200)/1000;
    If x1>1 then x1:=0.99; if x1<-1 then x1:=-0.99;
    If x2>1 then x2:=0.99; if x2<-1 then x2:=-0.99;
```

```
    If x3>1 then x3:=0.99; if x3<-1 then x3:=-0.99;
    circle(100+round(x1*100),100-round(x2*100),1);
    circle(400+round(x2*100),100-round(x3*100),1);
    k:=k+1;
  until (k>40000)or(keypressed);
  For i:=1 to 3 do writeln(F,a[i,1],
        ' ',a[i,2],' ',a[i,3],' ');
  Readkey; Close(F);
END.
```

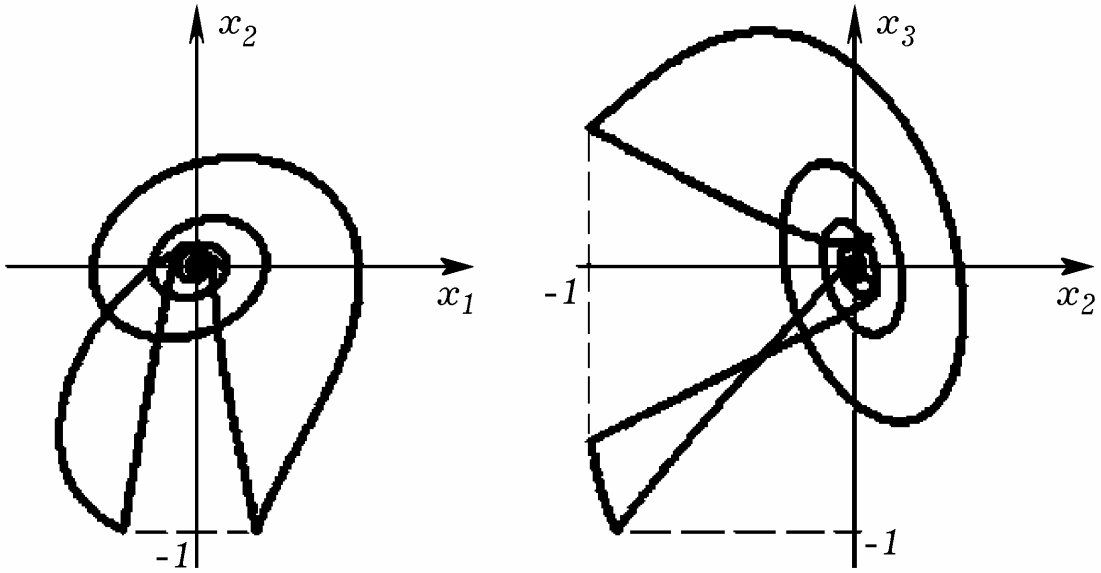


Рис. 4.3. Подбор параметров гомеостата.

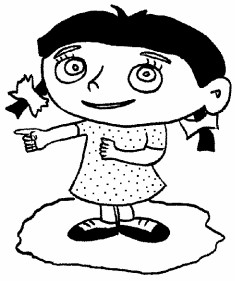
В результате работы программы ПР–4.7 рассчитываются коэффициенты a_{ij}, при которых гомеостат находится в положении равновесия (0,0,0). Эта задача имеет несколько решений. Допустим, получились следующие значения:

$$A = \begin{pmatrix} -0.089 & -0.001 & 0.07 \\ 0.082 & -0.043 & -0.028 \\ -0.072 & -0.045 & 0.027 \end{pmatrix}.$$

После нахождения оптимальных параметров гомеостата, можно исследовать, как он ведет себя при выходе из положения равновесия. В программе ПР – 4.8 случайным образом задаются начальные координаты x_1, x_2, x_3 из интервала [-0,9; 0,9] и рассчитывается "поведение" гомеостата в последующие моменты времени (рис. 4.4). Во всех случаях гомеостат возвращается в положение равновесия.

```
uses dos, crt, graph;                              { ПР - 4.8 }
const n=3; dt=0.01;
var x1,x2,x3,v1,v2,v3,k : real; Gd, Gm,i,j : integer;
    a: array[1..n,1..n]of real; F: text;
```

```
BEGIN Gd:=Detect; InitGraph(Gd, Gm, 'c:\bp\bgi');
Randomize; x1:=0.1;
a[1,1]:=-0.089; a[1,2]:=-0.001; a[1,3]:=0.07;
a[2,1]:=0.082; a[2,2]:=-0.043; a[2,3]:=-0.028;
a[3,1]:=-0.072; a[3,2]:=-0.045; a[3,3]:=0.027;
Repeat k:=k+0.1;
  v1:=a[1,1]*x1+a[1,2]*x2+a[1,3]*x3;
  v2:=a[2,1]*x1+a[2,2]*x2+a[2,3]*x3;
  v3:=a[3,1]*x1+a[3,2]*x2+a[3,3]*x3;
  x1:=x1+v1*dt; x2:=x2+v2*dt; x3:=x3+v3*dt;
  If k>5000 then begin x1:=0.9-random(1800)/1000;
  x2:=0.9-random(1800)/1000; x3:=0.9-random(1800)/1000;
  k:=0; delay(10000);
  cleardevice; line(0,240,640,240);
  line(150,10,150,480); line(450,10,450,480);
  end;
circle(150+round(x1*150),240-round(x2*150),1);
circle(450+round(x2*150),240-round(x3*150),1);
until keypressed;
Repeat until keypressed; CloseGraph;
END.
```



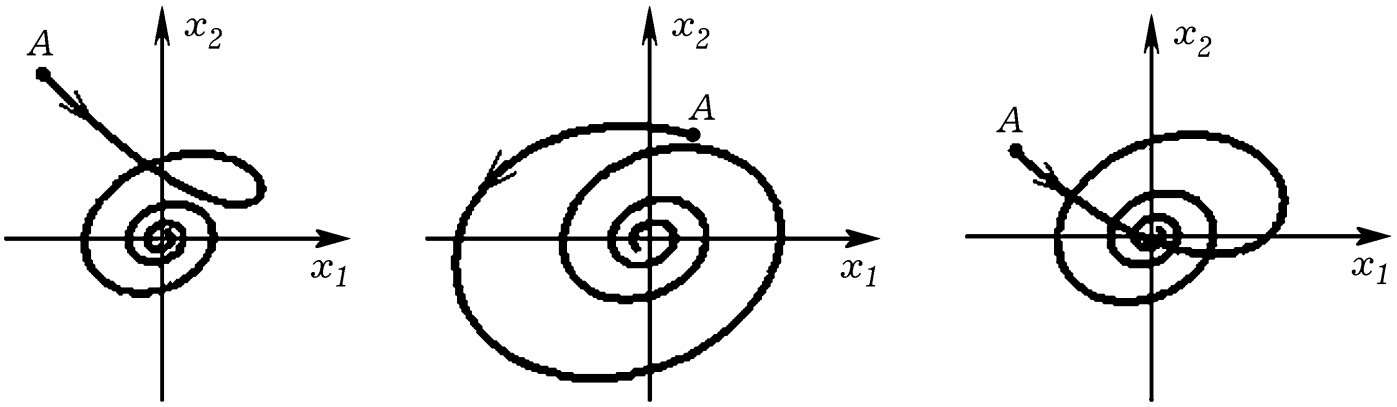


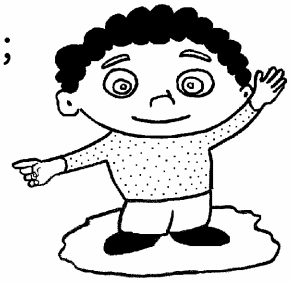
Рис. 4.4. Возврат гомеостата в положение равновесия.

4.11. Промоделируйте игру "Жизнь". Плоская поверхность разбита на клетки, которые ведут себя как автоматы, которые способны находиться в двух состояниях: "живой" или "мертвый". Клетка оживает при наличии 3 живых соседей. Если живых соседей 4 и больше, она умирает от перенаселенности. Если живых соседей меньше 2, она умирает от одиночества.

Представим двумерную сетку, в каждом узле — автомат, реализующий заданное правило. Используется программа ПР–4.9. Состояния клеток закодированы так: клетка "жива" - 1, "мертва" - 0. Вычисление числа "живых" соседей и установление "жива" данная клетка

или нет на следующем временном шаге осуществляется в процедуре Raschet и сохраняется в массиве x[i,j]. В массив x1[i,j] записывается состояние клеток на предыдущем временном шаге. В начале программы следует задать исходное распределение "живых" клеток.

```
uses crt; const N=23;                                { ПР - 4.9 }
type  z1 = record x,y,xy,x1,y1: real end;
     massiv = array[0..N+1,0..N+1] of integer;
var z,y,x,x1: massiv;  s,i,j,k,l,m: integer;
procedure Print;
var  i,j: integer;
begin clrscr;
For i:=1 to N do begin For j:=1 to N do begin
  If x[i,j]=1 then Write(' *');
  If x[i,j]=0 then Write('  ');
end; Writeln; end;
end;
procedure Oboznach;
var i,j: integer;
begin
 For i:=1 to N do For j:=1 to N do x1[i,j]:=x[i,j];
end;
procedure Raschet;
var i,j: integer;
begin
For i:=1 to N do For j:=1 to N do  begin
   S:=x1[i-1,j-1]+x1[i-1,j]+x1[i-1,j+1]+x1[i,j-1]+
       x1[i,j+1]+x1[i+1,j-1]+x1[i+1,j]+x1[i+1,j+1];
   If s=3 then x[i,j]:=1;
   If (s<2)or(s>3) then x[i,j]:=0;
end; end;
BEGIN
 For i:=1 to N do For j:=1 to N do x[i,j]:=0;
 x[14,18]:=1; x[14,17]:=1; x[14,16]:=1;
 x[15,15]:=1;  x[16,17]:=1; x[17,18]:=1; Print;
 Repeat delay(300); Oboznach;
 Raschet; Print;
 until keypressed;
END.
```



4.12. Создайте компьютерную модель абстрактной вычислительной машины (ВМ), состоящей из памяти, устройства управления, арифметико–логического устройства и устройства ввода–вывода. Па-

мять состоит из регистров (ячеек) с адресами 1, 2, 3, ..., в каждом из которых может быть записано целое положительное число, и устройства для хранения программы. Арифметико–логическое устройство может складывать и вычитать числа (операнды), хранящиеся в регистрах памяти. Устройство управления считывает программу из памяти, распознает команды и управляет работой всех устройств. Система команд виртуальной ВМ включает в себя операторы: 1) Registr(A1,X) — поместить в регистр с адресом A1 значение X; 2) Peresilka(A1,A2) — скопировать в регистр с адресом A2 содержимое регистра A1; 3) Summa(A1,A2,A3) — записать в регистр с адресом A3 сумму операндов, хранящихся в регистрах A1 и A2; 4) Raznost(A1,A2,A3) — записать в регистр с адресом A3 разность операндов, хранящихся в регистрах A1 и A2; 5) Vivod(A1) — вывод на экран числа, хранящегося в регистре с адресом A1; 6) Uslovie(A1,A2,m1) — условный переход: если операнд из регистра A1 больше операнда из регистра A2, то перейти к метке m1.

Текст программы, моделирующей рассмотренную абстрактную ВМ, представлен ниже (ПР–4.10).

```
uses dos, crt;                                    { ПР - 4.10 }
var r: array[1..256] of integer;
Label m1,m2,m3,m4,m5;
Procedure Registr(a,x: integer);
begin r[a]:=x; end;
Procedure Peresilka(a1,a2: integer);
begin r[a2]:=r[a1]; end;
Procedure Summa(a1,a2,a3: integer);
begin r[a3]:=r[a1]+r[a2]; end;
Procedure Raznost(a1,a2,a3: integer);
begin r[a3]:=r[a1]-r[a2]; end;
Procedure Vivod(a: integer);
begin writeln('Registr ',a,' = ',r[a]); end;
Function Uslovie(a1,a2: integer): boolean;
begin if r[a1]>r[a2] then Uslovie:=true else
Uslovie:=false; end;
BEGIN clrscr;
{ -------- текст программы ---------- }
Registr(1,7); Registr(2,6); Registr(3,2);
Summa(1,2,2); Raznost(2,3,3); Vivod(3);
{ ---------------------------------- }
ReadKey;
END.
```

4.13. Напишите программу для абстрактной ВМ, которая вычисляла бы выражение $x_1 + x_2 - x_3$. Операнды x_1, x_2, x_3 запишите в регистры с адресами 1, 2 и 3.

Представленная ниже программа находит значение $7 + 6 - 2$. Результат записывается в регистр 3, а затем выводится на экран.

```
Registr(1,7); Registr(2,6); Registr(3,2);
Summa(1,2,2); Raznost(2,3,3); Vivod(3);
```

4.14. Напишите программу для абстрактной ВМ, которая умножала бы два целых числа. Апробируйте ее на компьютерной модели.

Решением задачи является следующая программа:

```
Registr(1,5); Registr(2,6); Registr(3,1);
Registr(4,0); Peresilka(1,5); Raznost(2,3,2);
if Uslovie(3,2) then goto m2;
m1: Summa(5,1,5);
Raznost(2,3,2);
if Uslovie(2,4) then goto m1;
m2: Vivod(5);
```

4.15. Напишите программу для абстрактной ВМ, которая делила бы 2 целых числа с остатком. Апробируйте ее на компьютерной модели.

Один из вариантов решения выглядит так:

```
Registr(1,22); Registr(2,6); Registr(3,1); Registr(4,0);
Peresilka(1,5); Raznost(2,3,6);
m1:
Raznost(5,2,5); Summa(4,3,4); {Vivod(5);}
if Uslovie(5,6) then goto m1;
Vivod(4); Vivod(5);
```

4.16. Напишите программу для абстрактной ВМ, которая табулировала бы функцию $y = 5x - 3$; аргумент x изменяется от 1 до 20 с шагом 1. Апробируйте ее на компьютерной модели.

Эта программа представлена ниже:

```
Registr(1,1); Registr(2,5); Registr(3,1); Registr(4,0);
Registr(6,20); Registr(7,3);
m3:
Peresilka(1,5); Peresilka(2,8); Raznost(8,3,8);
if Uslovie(3,8) then goto m2;
```

```
m1: Summa(5,1,5); Raznost(8,3,8);
if Uslovie(8,4) then goto m1;
m2: Raznost(5,7,5); Vivod(5); Summa(1,3,1);
if Uslovie(6,1) then goto m3;
```

4.17. Напишите программу для абстрактной ВМ, которая вычисляла бы разность двух целых чисел (и положительных, и отрицательных). Знак и модуль числа кодируйте в отдельных регистрах.

4.18. Напишите программу для абстрактной ВМ, которая вычисляла бы квадрат целого числа.

4.19. Напишите программу для абстрактной ВМ, которая вычисляла бы выражение $xy - 2z$, где x, y, z — целые.

4.20. Напишите программу для абстрактной ВМ, которая вычисляла бы факториал числа.

4.21. Напишите программу для абстрактной ВМ, которая складывала бы два дробных числа с точностью до тысячных. Целую и дробную части числа (количество тысячных) запишите в разных регистрах.

4.22. Напишите программу для абстрактной ВМ, которая вычитала бы два дробных числа с точностью до тысячных.

5.1. Машина Поста состоит из ленты, разбитой на ячейки, и каретки, которая может считывать содержимое обозреваемой ячейки, стирать метки и ставить метки. Создайте компьютерную модель машины Поста, увеличивающей целое число на 2.

Программа, моделирующая работу машины Поста, представлена ниже (ПР–5.1).

```
uses crt, dos;                                    { ПР -- 5.1 }
const N=35; t=100;
var z,lenta: string; a, kom : array [1..N] of string;
    k, kk : array [1..N] of integer;
    x, p, i, ii : integer; Label m1, m2;
Procedure Programma;
Begin  {Программа МП: увеличение числа на 2}
x:=3;                                {координата головки}
lenta:='VVVVVVV----------------------------------';
kom[1]:='right'; k[1]:=2;
kom[2]:='if';    k[2]:=3; kk[2]:=1;
kom[3]:='metka'; k[3]:=4;
kom[4]:='right'; k[4]:=5;
kom[5]:='metka'; k[5]:=6;
kom[6]:='stop';  k[6]:=0;
end;
Procedure Pechat;
begin writeln; p:=p+1; For i:=1 to N do write(a[i],' ');
  writeln('| ',p,' |');
  For i:=1 to x-1 do write('--'); write('M');
  delay(200); end;
BEGIN clrscr; Programma;
   For i:=1 to N do begin a[i]:=copy(lenta,i,1); end;
   Pechat; ii:=1; m2: If Keypressed then goto m1;
   If kom[ii]='stop' then goto m1;
   If kom[ii]='left'  then begin x:=x-1; Pechat;
                               ii:=k[ii]; goto m2; end;
   If kom[ii]='right' then begin x:=x+1; Pechat;
                               ii:=k[ii]; goto m2; end;
   If kom[ii]='erase' then begin a[x]:='-'; Pechat;
                               ii:=k[ii]; goto m2; end;
   If kom[ii]='metka' then begin a[x]:='V'; Pechat;
                               ii:=k[ii]; goto m2; end;
```

```
   If kom[ii]='if' then begin z:=a[x];
   If z='-' then ii:=k[ii] else
   ii:=kk[ii]; goto m2; end; writeln;
   writeln('ОШИБКА В СТРОКЕ', ii);
m1: writeln; writeln('КОНЕЦ РАБОТЫ');
Repeat until KeyPressed;
END.
```

Алгоритм увеличения целого числа на 2 выглядит так:

```
3                                -- координата каретки
VVVVVVV--------------------------  лента
1 сместить вправо, команда 2
2 если пусто - команда 3, если метка - команда 1
3 поставить метку, команда 4
4 сместить вправо, команда 5
5 поставить метку, команда 6
6 остановить МП.
```

Для реализации этого алгоритма в процедуру Programma программы ПР – 5.1 необходимо вставить следующий код:

```
x:=3;                                {координата головки}
lenta:='VVVVVVV-----------------------------------';
kom[1]:='right'; k[1]:=2;
kom[2]:='if';    k[2]:=3; kk[2]:=1;
kom[3]:='metka'; k[3]:=4;
kom[4]:='right'; k[4]:=5;
kom[5]:='metka'; k[5]:=6;
kom[6]:='stop';  k[6]:=0;
```

Результат работы программы (7 + 2 = 9):

```
V V V V V V V - - - - - - - - - - - - - - - - - - - - | 1 |
----M
V V V V V V V - - - - - - - - - - - - - - - - - - - - | 2 |
------M
V V V V V V V - - - - - - - - - - - - - - - - - - - - | 3 |
--------M
V V V V V V V - - - - - - - - - - - - - - - - - - - - | 4 |
----------M
V V V V V V V - - - - - - - - - - - - - - - - - - - - | 5 |
------------M
V V V V V V V - - - - - - - - - - - - - - - - - - - - | 6 |
--------------M
V V V V V V V V - - - - - - - - - - - - - - - - - - - | 7 |
```

```
-------------M
V V V V V V V V - - - - - - - - - - - - - - - - - - - | 8 |
---------------M
V V V V V V V V V - - - - - - - - - - - - - - - - - - | 9 |
---------------M
```

5.2. Напишите программу для МП, вычитающую два целых числа, записанных через пустую клетку: VVVV–VVV. Каретка находится напротив пустой клетки.

Алгоритм вычитания целых чисел для машины Поста приведен ниже. В первых двух строчках указывается положение каретки и состояние ленты, на которой в унарной системе счисления записаны два числа (в данном случае 6 и 4). В результате исполнения программы на ленте останется число 2 в унарной системе счисления.

```
7                          -- координата каретки
VVVVVV-VVVV-------------------------   лента
1 сместить влево, команда 2
2 если пусто -- команда 1, если метка -- команда 3
3 удалить метку, команда 4
4 сместить вправо, команда 5
5 если пусто -- команда 4, если метка -- команда 6
6 удалить метку, команда 7
7 сместить вправо, команда 8
8 если пусто -- команда 9, если метка -- команда 1
9 остановить МП.
```

Для решения задачи в программу ПР–5.1 необходимо вставить код:

```
lenta:='VVVVVV-VV-----------------------------------';
kom[1]:='left';  k[1]:=2;
kom[2]:='if';    k[2]:=1; kk[2]:=3;
kom[3]:='erase'; k[3]:=4;
kom[4]:='right'; k[4]:=5;
kom[5]:='if';    k[5]:=4; kk[5]:=6;
kom[6]:='erase'; k[6]:=7;
kom[7]:='right'; k[7]:=8;
kom[8]:='if';    k[8]:=9; kk[8]:=1;
kom[9]:='stop';  k[9]:=0;
```

Результат работы компьютерной модели МП (6 - 2 = 4):

```
V V V V V V - V V - - - - - - - - - - - - - - - - - - | 1 |
------------M
V V V V V V - V V - - - - - - - - - - - - - - - - - - | 2 |
----------M
```

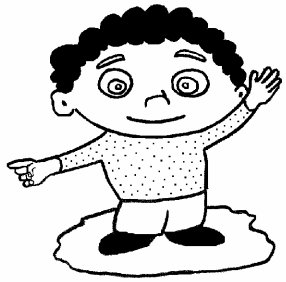
```
V V V V V - - V V - - - - - - - - - - - - - - - - - - | 3 |
----------M
V V V V V - - V V - - - - - - - - - - - - - - - - - - | 4 |
------------M
V V V V V - - V V - - - - - - - - - - - - - - - - - - | 5 |
--------------M
V V V V V - - - V - - - - - - - - - - - - - - - - - - | 6 |
--------------M
V V V V V - - - V - - - - - - - - - - - - - - - - - - | 7 |
----------------M
V V V V V - - - V - - - - - - - - - - - - - - - - - - | 8 |
--------------M
V V V V V - - - V - - - - - - - - - - - - - - - - - - | 9 |
------------M
V V V V V - - - V - - - - - - - - - - - - - - - - - - | 10 |
----------M
V V V V V - - - V - - - - - - - - - - - - - - - - - - | 11 |
--------M
V V V V - - - - V - - - - - - - - - - - - - - - - - - | 12 |
--------M
V V V V - - - - V - - - - - - - - - - - - - - - - - - | 13 |
----------M
V V V V - - - - V - - - - - - - - - - - - - - - - - - | 14 |
------------M
V V V V - - - - V - - - - - - - - - - - - - - - - - - | 15 |
--------------M
V V V V - - - - V - - - - - - - - - - - - - - - - - - | 16 |
----------------M
V V V V - - - - - - - - - - - - - - - - - - - - - - - | 17 |
----------------M
V V V V - - - - - - - - - - - - - - - - - - - - - - - | 18 |
------------------M
```

5.3. Напишите компьютерную программу, моделирующую машину Поста, которая уменьшает целое число на 2.

Пусть на ленте машины Поста записано число 6, каретка находится напротив левой ячейки (координата x =1). Для моделирования вычитания из любого целого числа 2 в процедуру Programma программы ПР–5.1 необходимо вставить следующий код:

```
x:=1; lenta:='VVVVVV----------------------------';
```

```
kom[1]:='right'; k[1]:=2;
kom[2]:='if';    k[2]:=3; kk[2]:=1;
kom[3]:='left';  k[3]:=4;
kom[4]:='erase'; k[4]:=5;
kom[5]:='left';  k[5]:=6;
kom[6]:='erase'; k[6]:=7;
kom[7]:='stop';
```



Результат работы программы (6 - 2 = 4):

```
V V V V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 1 |
M
V V V V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 2 |
--M
V V V V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 3 |
----M
V V V V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 4 |
------M
V V V V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 5 |
--------M
V V V V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 6 |
----------M
V V V V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 7 |
------------M
V V V V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 8 |
----------M
V V V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 9 |
----------M
V V V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 10 |
--------M
V V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 11 |
--------M
```

5.4. Напишите компьютерную программу, моделирующую машину Поста, которая складывает два целых числа.

Программа сложения целых чисел на машине Поста выглядит так:

```
5                                  -- координата каретки
VVVV-VVV---------------------------------- лента
1 поставить метку, команда 2
2 сместить вправо, команда 3
3 если пусто -- команда 4, если метка -- команда 2
4 сместить влево, команда 5
5 удалить метку, команда 6
6 остановить МП.
```

В программу ПР – 5.1 следует вставить код:

```
x:=5;              {сложение двух чисел}
lenta:='VVVV-VVV--------------------------------';
komand[1]:='metka'; k[1]:=2;
komand[2]:='right'; k[2]:=3;
komand[3]:='if';    k[3]:=4; kk[3]:=2;
komand[4]:='left';  k[4]:=5;
komand[5]:='erase'; k[5]:=6;
komand[6]:='stop';  k[6]:=0;
```

Результат работы программы (4 + 3 = 7):

```
V V V V - V V V - - - - - - - - - - - - - - - - - | 1 |
--------M
V V V V V V V V - - - - - - - - - - - - - - - - - | 2 |
--------M
V V V V V V V V - - - - - - - - - - - - - - - - - | 3 |
----------M
V V V V V V V V - - - - - - - - - - - - - - - - - | 4 |
------------M
V V V V V V V V - - - - - - - - - - - - - - - - - | 5 |
--------------M
V V V V V V V V - - - - - - - - - - - - - - - - - | 6 |
----------------M
V V V V V V V V - - - - - - - - - - - - - - - - - | 7 |
--------------M
V V V V V V V - - - - - - - - - - - - - - - - - - | 8 |
-------------M
```

5.5. Напишите компьютерную программу, моделирующую машину Поста, которая умножает целое число на 2.

В процедуру Programma следует вставить код:

```
x:=2;                          {Умножение числа на 2}
lenta:='-VVV-----------------------------------';
kom[1]:='right';  k[1]:=2;
kom[2]:='if';     k[2]:=3; kk[2]:=1;
kom[3]:='left';   k[3]:=4;
kom[4]:='erase';  k[4]:=5;
kom[5]:='right';  k[5]:=6;
kom[6]:='metka';  k[6]:=7;
kom[7]:='right';  k[7]:=8;
kom[8]:='metka';  k[8]:=9;
```

```
kom[9]:='left';  k[9]:=10;
kom[10]:='if';    k[10]:=11; kk[10]:=9;
kom[11]:='left';  k[11]:=12;
kom[12]:='if';    k[12]:=11; kk[12]:=13;
kom[13]:='erase'; k[13]:=14;
kom[14]:='left';  k[14]:=15;
kom[15]:='if';    k[15]:=20; kk[15]:=16;
kom[16]:='right'; k[16]:=17;
kom[17]:='if';    k[17]:=16; kk[17]:=18;
kom[18]:='right'; k[18]:=19;
kom[19]:='if';    k[19]:=6; kk[19]:=18;
kom[20]:='right'; k[20]:=21;
kom[21]:='if';    k[21]:=20; kk[21]:=22;
kom[22]:='right'; k[22]:=23;
kom[23]:='if';    k[23]:=24; kk[23]:=22;
kom[24]:='metka'; k[24]:=25;
kom[25]:='right'; k[25]:=26;
kom[26]:='metka'; k[26]:=27;
kom[27]:='stop';  k[28]:=0;
```

Результат работы программы ($3 \cdot 2 = 6$):

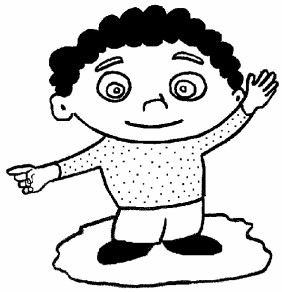
```
- V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 1 |
--M
- V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 2 |
----M
- V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 3 |
------M
- V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 4 |
--------M
- V V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 5 |
------M
- V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 6 |
------M
- V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 7 |
--------M
- V V - V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 8 |
--------M
- V V - V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 9 |
----------M
- V V - V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 10 |
----------M
- V V - V V - - - - - - - - - - - - - - - - - - - - - - - - - - - - - | 11 |
--------M
```

```
. . . . . .

- - - - V V V V - - - - - - - - - - - - - - - - - - | 39 |
---------------M
- - - - V V V V V - - - - - - - - - - - - - - - - - | 40 |
---------------M
- - - - V V V V V - - - - - - - - - - - - - - - - - | 41 |
-----------------M
- - - - V V V V V V - - - - - - - - - - - - - - - - | 42 |
-----------------M
```

Возможен другой способ решения:

```
x:=9; lenta:='-VVVVV--------------------------';
kom[1]:='metka'; k[1]:=2;
kom[2]:='right'; k[2]:=3;
kom[3]:='metka'; k[3]:=4;
kom[4]:='left';  k[4]:=5;
kom[5]:='if';    k[5]:=6; kk[5]:=4;
kom[6]:='left';  k[6]:=7;
kom[7]:='if';    k[7]:=6; kk[7]:=8;
kom[8]:='erase'; k[8]:=9;
kom[9]:='left'; k[9]:=10;
kom[10]:='if'; k[10]:=15; kk[10]:=11;
kom[11]:='right'; k[11]:=12;
kom[12]:='if'; k[12]:=11; kk[12]:=13;
kom[13]:='right'; k[13]:=14;
kom[14]:='if'; k[14]:=1; kk[14]:=13;
kom[15]:='stop'; k[15]:=0;
```



5.6. На информационной ленте машины Поста имеется массив из N меток, расположенных через 2 или 3 пробела. Необходимо сжать массив так, чтобы между метками не было пробелов.

5.7. На ленте машины Поста записано произвольное количество целых неотрицательных чисел, разделенных одной ячейкой: –VV–VVV–V–VV–. Напишите программу, складывающую эти числа.

5.8. На ленте машины Поста записано число x в унарной системе счисления. Напишите программу, определяющую остаток от деления этого числа: 1) на 2; 2) на 3.

5.9. На ленте машины Поста записано m массивов, разделенных одной ячейкой: –VV–VVV–V–VV–. Напишите программу, определяющую число m.

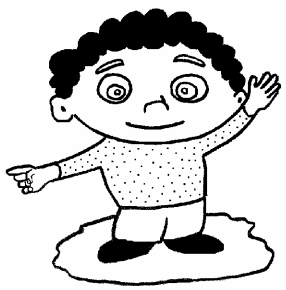
6.1. Машина Тьюринга состоит из бесконечной ленты и головки, которая перемещается относительно ленты, стирает символы и ставит новые символы. Напишите программу, моделирующую работу машины Тьюринга, которая увеличивает заданное число на 2. Оформите программу в виде таблицы.

Состояние ленты машины Тьюринга задается так:

$$lenta :=' \_111111\_\_\_\_\_';$$

Положение головки определяется переменной m, исходное состояние машины Тьюринга – переменной q : $q :=' 1'; m := 2;$. Программа для машины Тьюринга записывается в виде последовательности команд, представленных в общепринятом формате: "(Состояние q_1) (Читаю символ s_1) => (Новое состояние q_2) (Напечатать символ s_1) (Сместить головку вправо (R), влево (L) или остановиться (S))". Ниже приведена компьютерная программа ПР–6.1, моделирующая работу машины Тьюринга.

```
uses crt, graph;                              { ПР - 6.1 }
const N=50;
Var i,k,m,s,flag : integer;
    x1,x2,x4,x5,x6,q,lenta: string;
    kom,a : array[1..N] of string; Label m1;
BEGIN
clrscr; m:=2; { положение головки }
lenta:='_111111__________________';
q:='1';       { Машина Тьюринга: Увеличение числа на 2 }
kom[1]:='11>11R';
kom[2]:='1_>21R';
kom[3]:='2_>21S';
For i:=1 to N do a[i]:=copy(lenta,i,1);
Repeat flag:=0; s:=s+1;
  For i:=1 to N do begin
   x1:=copy(kom[i],1,1); x2:=copy(kom[i],2,1);
   x4:=copy(kom[i],4,1); x5:=copy(kom[i],5,1);
   x6:=copy(kom[i],6,1);
   If (flag=0)and(x1=q)and(x2=a[m]) then
     begin q:=x4; a[m]:=x5;
     If x6='R' then m:=m+1;
     If x6='L' then m:=m-1;
     If x6='S' then goto m1;
     flag:=1; end; end;
```



```
   m1: k:=k+1;
For i:=1 to 25 do write(a[i],' ');
writeln('   ',q,'   k=',k); delay(5);
For i:=1 to m-1 do write('=='); write('|'); writeln;
until x6='S'; Readkey;
END.
```

Программа для машины Тьюринга, увеличивающая целое число на 2 состоит из 3 команд:

```
kom[1]:='11>11R'; kom[2]:='1_>21R'; kom[3]:='2_>21S';
```

Результат ее использования:

```
_ 1 1 1 1 1 1 _ _  q=1   k=1
====|
_ 1 1 1 1 1 1 _ _  q=1   k=2
======|
_ 1 1 1 1 1 1 _ _  q=1   k=3
========|
_ 1 1 1 1 1 1 _ _  q=1   k=4
==========|
_ 1 1 1 1 1 1 _ _  q=1   k=5
============|
_ 1 1 1 1 1 1 _ _  q=1   k=6
==============|
_ 1 1 1 1 1 1 1 _  q=2   k=7
================|
_ 1 1 1 1 1 1 1 1 1  q=2   k=8
==================|
```

6.2. Напишите программу для МТ, складывающую два целых числа, заданных набором единиц.

Пусть начальное состояние информационной ленты машины Тьюринга: _11111_1111__, головка находится напротив левой единицы. МТ находится в состоянии 1. *Программа выглядит так:*

q	"1"	"_"
1	$11R$	$21R$
2	$21R$	3_L
3	3_S	

В программу ПР – 6.1. следует вставить код:

```
lenta:='_11111_1111______';
m:=2;   q:='1';   kom[1]:='11>11R'; kom[2]:='1_>21R';
kom[3]:='21>21R'; kom[4]:='2_>3_L'; kom[5]:='31>3_S';
```

Результат работы машины Тьюринга $(5 + 4 = 9)$ *представлен ниже:*

```
_ 1 1 1 1 1 _ 1 1 1 1 _ _ _ _  q=1   k=1
====|
_ 1 1 1 1 1 _ 1 1 1 1 _ _ _ _  q=1   k=2
======|
_ 1 1 1 1 1 _ 1 1 1 1 _ _ _ _  q=1   k=3
========|
_ 1 1 1 1 1 _ 1 1 1 1 _ _ _ _  q=1   k=4
==========|
_ 1 1 1 1 1 _ 1 1 1 1 _ _ _ _  q=1   k=5
============|
_ 1 1 1 1 1 1 1 1 1 1 _ _ _ _  q=2   k=6
==============|
_ 1 1 1 1 1 1 1 1 1 1 _ _ _ _  q=2   k=7
================|
_ 1 1 1 1 1 1 1 1 1 1 _ _ _ _  q=2   k=8
==================|
_ 1 1 1 1 1 1 1 1 1 1 _ _ _ _  q=2   k=9
====================|
_ 1 1 1 1 1 1 1 1 1 1 _ _ _ _  q=2   k=10
======================|
_ 1 1 1 1 1 1 1 1 1 1 _ _ _ _  q=3   k=11
====================|
_ 1 1 1 1 1 1 1 1 1 1 _ _ _ _  q=3   k=12
====================|
```

6.3. На ленте МТ – конечный набор единиц: _11111__. Напишите программу, которая ставит звездочки вместо первой и последней единицы, остальные стирает. Оформите программу в виде таблицы.

Пусть машина Тьюринга находится в состоянии 1. *В компьютерную программу ПР–6.1 необходимо вставить следующий код:*

```
lenta:=('_1111111111_______');
m:=1; q:='1';
kom[1]:='1_>1_R'; kom[2]:='2_>2_L';
kom[3]:='3_>3_S'; kom[4]:='11>21R';
kom[5]:='21>2*R'; kom[6]:='31>3*L';
kom[7]:='2*>3*L'; kom[8]:='3*>3_L';
```

Результат исполнения программы представлен ниже:

```
_ 1 1 1 1 1 1 1 1 1 1 _ _ _ _  q=1   k=1
==|
```

```
_ 1 1 1 1 1 1 1 1 1 _ _ _ _  q=2   k=2
====|
_ 1 * 1 1 1 1 1 1 1 _ _ _ _  q=2   k=3
======|
_ 1 * * 1 1 1 1 1 1 _ _ _ _  q=2   k=4
========|
_ 1 * * * 1 1 1 1 1 _ _ _ _  q=2   k=5
==========|
_ 1 * * * * 1 1 1 1 _ _ _ _  q=2   k=6
============|
_ 1 * * * * * 1 1 1 _ _ _ _  q=2   k=7
==============|
_ 1 * * * * * * 1 1 _ _ _ _  q=2   k=8
================|
_ 1 * * * * * * * 1 _ _ _ _  q=2   k=9
==================|
_ 1 * * * * * * * * _ _ _ _  q=2   k=10
====================|
_ 1 * * * * * * * * * _ _ _  q=2   k=11
======================|
_ 1 * * * * * * * * * _ _ _  q=2   k=12
====================|
_ 1 * * * * * * * * * _ _ _  q=3   k=13
==================|
_ 1 * * * * * * * _ * _ _ _  q=3   k=14
================|
_ 1 * * * * * * _ _ * _ _ _  q=3   k=15
==============|
_ 1 * * * * * _ _ _ * _ _ _  q=3   k=16
============|
_ 1 * * * * _ _ _ _ * _ _ _  q=3   k=17
==========|
. . . . . . . . . . . . . . . . . . .
_ 1 * * _ _ _ _ _ _ * _ _ _  q=3   k=19
======|
_ 1 * _ _ _ _ _ _ _ * _ _ _  q=3   k=20
====|
_ 1 _ _ _ _ _ _ _ _ * _ _ _  q=3   k=21
==|
_ * _ _ _ _ _ _ _ _ * _ _ _  q=3   k=22
|
_ * _ _ _ _ _ _ _ _ * _ _ _  q=3   k=23
|
```

6.4. На ленте МТ – конечный набор единиц: _111111__. Напишите программу, которая заменяет единицы звездочками. Головка – левее первой единицы. Программу оформите в виде таблицы.

Программа для машины Тьюринга имеет вид:

q	" _ "	"1"	" $*$ "
1	1_R	$3*R$	
2	2_L	$2*R$	$3*L$
3	3_S	$2*R$	$3*L$

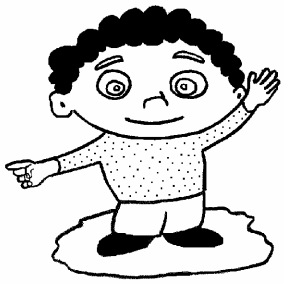
Для решения этой задачи в программу ПР–6.1 следует вставить код:

```
lenta:='__1111111111____';
m:=1; q:='1';
kom[1]:='1_>1_R';
kom[2]:='2_>2_L';
kom[3]:='3_>3_S';
kom[4]:='11>3*R';
kom[5]:='21>2*R';
kom[6]:='31>2*R';
kom[7]:='2*>3*L';
kom[8]:='3*>3*L';
```

Результат работы программы представлен ниже:

```
_ _ 1 1 1 1 1 1 1 1 1 1 _ _ _  q=1   k=1
==|
_ _ 1 1 1 1 1 1 1 1 1 1 _ _ _  q=1   k=2
====|
_ _ * 1 1 1 1 1 1 1 1 1 _ _ _  q=3   k=3
======|
_ _ * * 1 1 1 1 1 1 1 1 _ _ _  q=2   k=4
========|
_ _ * * * 1 1 1 1 1 1 1 _ _ _  q=2   k=5
==========|
_ _ * * * * 1 1 1 1 1 1 _ _ _  q=2   k=6
============|
_ _ * * * * * 1 1 1 1 1 _ _ _  q=2   k=7
==============|
_ _ * * * * * * 1 1 1 1 _ _ _  q=2   k=8
================|
_ _ * * * * * * * 1 1 1 _ _ _  q=2   k=9
==================|
```

```
_ _ * * * * * * * * 1 1 _ _ _  q=2   k=10
====================|
_ _ * * * * * * * * * 1 _ _ _  q=2   k=11
======================|
_ _ * * * * * * * * * * _ _ _  q=2   k=12
========================|
_ _ * * * * * * * * * * * _ _ _  q=2   k=13
======================|
_ _ * * * * * * * * * * * _ _ _  q=3   k=14
====================|
_ _ * * * * * * * * * * * _ _ _  q=3   k=15
==================|
_ _ * * * * * * * * * * * _ _ _  q=3   k=16
================|
_ _ * * * * * * * * * * * _ _ _  q=3   k=17
==============|
_ _ * * * * * * * * * * * _ _ _  q=3   k=18
============|
_ _ * * * * * * * * * * * _ _ _  q=3   k=19
==========|
_ _ * * * * * * * * * * * _ _ _  q=3   k=20
========|
_ _ * * * * * * * * * * * _ _ _  q=3   k=21
======|
_ _ * * * * * * * * * * * _ _ _  q=3   k=22
====|
_ _ * * * * * * * * * * * _ _ _  q=3   k=23
==|
_ _ * * * * * * * * * * * _ _ _  q=3   k=24
==|
```



6.5. На ленте МТ – последовательность _ABBAABAB___. Головка расположена напротив левого символа. Напишите программу, чтобы МТ группировала символы "A" в правой части строки, а вместо них ставила звездочки. Программу оформите в виде таблицы.

В компьютерную программу ПР – 6.1 следует вставить код:

```
lenta:='_ABBAABAB_______';
m:=1; q:= '1';
kom[1]:='1_>1_R';  kom[2]:='2_>3|R';
kom[3]:='3_>4AL';  kom[4]:= '4_>1_R';
kom[5]:='1A>2*R';  kom[6]:='2A>2AR';
kom[7]:='3A>3AR';  kom[8]:='4A>4AL';
```

```
kom[9]:='1B>1BR';  kom[10]:='2B>2BR';
kom[11]:='4B>4BL'; kom[12]:='1*>1*R';
kom[13]:='2*>2*R'; kom[14]:='4*>4*L';
kom[15]:='1|>1|S'; kom[16]:='2|>3|R';
kom[17]:='4|>4|L';
```

Сама программа для машины Тьюринга выглядит так:

q	"_"	"A"	"B"	"$*$"	"$\|$"
1	1_R	$2*R$	$1BR$	$1*R$	$1\|S$
2	$3\|R$	$2AR$	$2BR$	$2*R$	$3\|R$
3	$4AL$	$3AR$			
4	1_R	$4AL$	$4BL$	$4*L$	$4\|L$

Результат исполнения программы:

```
_ A B B A A B A B _ _ _ _ _ _ _  q=1   k=1
==|
_ * B B A A B A B _ _ _ _ _ _ _  q=2   k=2
====|
_ * B B A A B A B _ _ _ _ _ _ _  q=2   k=3
======|
_ * B B A A B A B _ _ _ _ _ _ _  q=2   k=4
========|
_ * B B A A B A B _ _ _ _ _ _ _  q=2   k=5
==========|
_ * B B A A B A B _ _ _ _ _ _ _  q=2   k=6
============|
_ * B B A A B A B _ _ _ _ _ _ _  q=2   k=7
==============|
_ * B B A A B A B _ _ _ _ _ _ _  q=2   k=8
================|
_ * B B A A B A B _ _ _ _ _ _ _  q=2   k=9
==================|
_ * B B A A B A B | _ _ _ _ _ _  q=3   k=10
====================|
_ * B B A A B A B | A _ _ _ _ _  q=4   k=11
==================|
_ * B B A A B A B | A _ _ _ _ _  q=4   k=12
================|

. . . . . . . . . . . . . . . . .
_ * B B * * B * B | A A A A _  q=1   k=100
================|
```

```
_ * B B * * B * B | A A A A _  q=1   k=101
==================|
_ * B B * * B * B | A A A A _  q=1   k=102
==================|
```

6.6. На ленте МТ – число в десятичной системе счисления, например, 134999. Напишите программу, увеличивающую его на 1 и оформите ее в виде таблицы.

Программа для машины Тьюринга выглядит так:

q	"_"	"1"	"2"	.....	"5"	"6"	"7"	"8"	"9"	"0"
1	2_L	$11R$	$12R$	.....	$15R$	$16R$	$17R$	$18R$	$19R$	$10R$
2	$21S$	$22S$	$23S$	.....	$26S$	$27S$	$28S$	$29S$	$20L$	$21S$

В программу ПР – 6.1 следует вставить код:

```
lenta:='_134999________';
m:=2; q:= '1';
kom[1]:='1_>2_L';  kom[2]:='2_>2_S';
kom[3]:='11>11R';  kom[4]:='21>22S';
kom[5]:='12>12R';  kom[6]:='22>23S';
kom[7]:='13>13R';  kom[8]:='23>24S';
kom[9]:='14>14R';  kom[10]:='24>25S';
kom[11]:='15>15R'; kom[12]:='25>26S';
kom[13]:='16>16R'; kom[14]:='26>27S';
kom[15]:='17>17R'; kom[16]:='27>28S';
kom[17]:='18>18R'; kom[18]:='28>29S';
kom[19]:='19>19R'; kom[20]:='29>20L';
```

Результат работы программы выглядит так:

```
_ 1 3 4 9 9 9 _ _ _ _ _ _ _ _ _  q=1   k=1
====|
_ 1 3 4 9 9 9 _ _ _ _ _ _ _ _ _  q=1   k=2
======|
_ 1 3 4 9 9 9 _ _ _ _ _ _ _ _ _  q=1   k=3
========|
_ 1 3 4 9 9 9 _ _ _ _ _ _ _ _ _  q=1   k=4
==========|
_ 1 3 4 9 9 9 _ _ _ _ _ _ _ _ _  q=1   k=5
============|
```

```
_ 1 3 4 9 9 9 _ _ _ _ _ _ _ _ _  q=1   k=6
==============|
_ 1 3 4 9 9 9 _ _ _ _ _ _ _ _ _  q=2   k=7
============|
_ 1 3 4 9 9 0 _ _ _ _ _ _ _ _ _  q=2   k=8
==========|
_ 1 3 4 9 0 0 _ _ _ _ _ _ _ _ _  q=2   k=9
========|
_ 1 3 4 0 0 0 _ _ _ _ _ _ _ _ _  q=2   k=10
======|
_ 1 3 5 0 0 0 _ _ _ _ _ _ _ _ _  q=2   k=11
======|
```

6.7. На ленте МТ — целое число в двоичной системе счисления, например, 101101. Напишите программу, уменьшающую его на 1.

6.8. Имеется массив из открывающихся и закрывающихся скобок: "$)()(())...)()()())(($". Постройте МТ, удаляющую пары взаимных скобок "$($" и "$)$".

6.9. На ленте МТ — целое число x в унарной системе счисления $\_1111\_\_$. Вычислите функцию $y = y(x)$, заданную следующим образом: $y(x) = x - 1$, если $x > 4$; иначе — $y(x) = 3x$.

6.10. На ленте МТ имеются числа $01021021102...$, записанные в произвольном порядке. Напишите программу для МТ, которая располагает эти числа по возрастанию: $000...111...222...$

6.11. На ленте МТ слово вида "$|||...|| * |||...|$", где "$*$" — один из знаков "$+$" или "$-$". Напишите программу, складывающую или вычитающую числа в унарной системе счисления.

6.12. Дано слово "$abacbabcac...$". Создайте МТ, удаляющую все "a" и сжимающую слово.

6.13. На ленте МТ двоичное число $101...11$. Умножьте его на 4.

6.14. На ленте МТ слово вида $101...11$. Напишите программу, определяющую является ли число степенью двойки $(1,\ 10,\ 100$ и т.д.$)$.

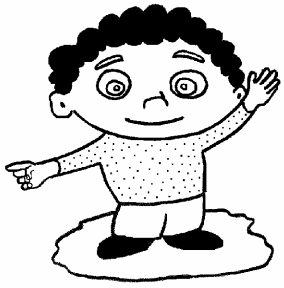
6.15. На ленте МТ слово P на алфавите $A = \{a, b, c\}$. Создайте МТ, заменяющую на "d" каждый второй символ в слове P.

6.16. На ленте МТ слово P на алфавите $A = \{a, b, c\}$. Сконструируйте МТ, которая стирает первый символ и записывает его правее последнего.

7.1. Напишите программу, позволяющую автоматически реализовать нормальный алгоритм Маркова, обрабатывающий входное слово с помощью системы подстановок. Например, дано слово из алфавита $\{a, b, c, d\}$, следует расположить буквы в алфавитном порядке.

Нормальная система подстановок осуществляется так: сначала выполняется первая подстановка ($x[1]$ заменяется на $y[1]$), слово переписывается. Затем — снова первая подстановка; если невозможно — вторая ($x[2]$ заменяется на $y[2]$); если вторая не проходит, — третья. Слово переписывается. Снова первая подстановка, если невозможно — вторая; если невозможно — третья. Слово переписывается. Система подстановок, позволяющая расположить буквы в алфавитном порядка, представлена ниже:

```
slovo:='dabadbcadcbd';
----------------------
x[1]:='ba';  y[1]:='ab';
x[2]:='ca';  y[2]:='ac';
x[3]:='da';  y[3]:='ad';
x[4]:='cb';  y[4]:='bc';
x[5]:='db';  y[5]:='bd';
x[6]:='dc';  y[6]:='cd';
```



Для автоматического выполнения нормального алгоритма Маркова используется программа ПР – 7.1.

```
uses crt, graph;                                { ПР - 7.1 }
const Chislo_podstan=30; label m1;
var i,j,k,m,p,s,flag: integer; x1,x2,x4,x5,x6,q: string;
    x,y: array[1..100] of string; slovo, slovo1: string;
Procedure Podstanovka(j:integer); label m;
begin flag:=0; i:=0;
Repeat inc(i);
  If copy(slovo,i,length(x[j]))=x[j] then begin flag:=1;
  slovo1:=copy(slovo,1,i-1)+y[j]+copy(slovo,
  i+length(x[j]),length(slovo)-i-length(x[j])+1);
  slovo:=slovo1;
  If (x[j]='')or(y[j]='') then flag:=0;
  goto m; end;
until i>length(slovo); m:
If flag=1 then writeln(k,' ',slovo,' | подстановка ',j);
end;
```

```
BEGIN clrscr;
slovo:='dabadbcadcbd';
writeln(slovo);
{====== Система подстановок =======}
x[1]:='ba';   y[1]:='ab';
x[2]:='ca';  y[2]:='ac';
x[3]:='da';   y[3]:='ad';
x[4]:='cb';   y[4]:='bc';
x[5]:='db';  y[5]:='bd';
x[6]:='dc';  y[6]:='cd';
m1: k:=k+1; delay(50);
For p:=1 to Chislo_podstan do
  begin Podstanovka(p);
  If flag=1 then goto m1; end;
Readkey;
END.
```

Результат работы программы:

```
1 daabdbcadcbd | подстановка 1
2 daabdbacdcbd | подстановка 2
3 daabdabcdcbd | подстановка 1
4 adabdabcdcbd | подстановка 3
5 aadbdabcdcbd | подстановка 3
6 aadbadbcdcbd | подстановка 3
7 aadabdbcdcbd | подстановка 1
8 aaadbdbcdcbd | подстановка 3
9 aaadbdbcdbcd | подстановка 4
10 aaabddbcdbcd | подстановка 5
11 aaabdbdcdbcd | подстановка 5
 . . . . . .
16 aaabbbddcdcd | подстановка 5
17 aaabbbdcddcd | подстановка 6
18 aaabbbcdddcd | подстановка 6
19 aaabbbcddcdd | подстановка 6
20 aaabbbcdcddd | подстановка 6
21 aaabbbccdddd | подстановка 6
```

7.2. Дана последовательность скобок. С помощью нормальной системы подстановок Маркова определите правильность скобочной структуры.

Чтобы реализовать нормальную систему подстановок Маркова, в программу ПР – 7.1 следует вставить код:

```
slovo:='()()(())(()())(())';
---------------------------
x[1]:='**';   y[1]:='*';
x[2]:='()*';  y[2]:='*';
x[3]:='*()';  y[3]:='*';
x[4]:='(*)';  y[4]:='*';
x[5]:='()';  y[5]:='*';
```

Результат исполнения программы:

```
1 *()(())(()())(()) | подстановка 5
2 *(())(()())(()) | подстановка 3
3 *(*)(()())(()) | подстановка 5
4 **(()())(()) | подстановка 4
5 *(()())(()) | подстановка 1
6 *(*())(()) | подстановка 5
7 *(*)(()) | подстановка 3
8 **(()) | подстановка 4
9 *(()) | подстановка 1
10 *(*) | подстановка 5
11 ** | подстановка 4
12 * | подстановка 1
```

7.3. Напишите программу, автоматически реализующий нормальный алгоритм Маркова, который переводит число из двоичной системы счисления в унарную.

В программу ПР–7.1 следует вставить код:

```
slovo:='10011';
------------------
x[1]:='|0'; y[1]:='0||';
x[2]:='1';  y[2]:='0|';
x[3]:='0|';  y[3]:='|';
```

Результат решения задачи представлен ниже.

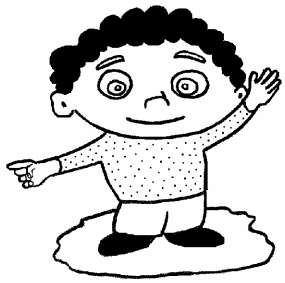
```
1 0|0011 | подстановка 2
2 00||011 | подстановка 1
3 00|0||11 | подстановка 1
4 000||||11 | подстановка 1
5 000||||0|1 | подстановка 2
6 000|||0|||1 | подстановка 1
7 000||0|||||1 | подстановка 1
8 000|0|||||||1 | подстановка 1
```

```
9 0000|||||||||1 | подстановка 1
10 0000|||||||||0| | подстановка 2
11 0000|||||||||0||| | подстановка 1
12 0000|||||||0||||| | подстановка 1
13 0000||||||0||||||| | подстановка 1
14 0000|||||0||||||||| | подстановка 1
15 0000||||0||||||||||| | подстановка 1
16 0000|||0|||||||||||| | подстановка 1
17 0000||0|||||||||||||| | подстановка 1
18 0000|0|||||||||||||||| | подстановка 1
19 00000|||||||||||||||||| | подстановка 1
20 0000|||||||||||||||||| | подстановка 3
21 000|||||||||||||||||| | подстановка 3
22 00|||||||||||||||||| | подстановка 3
23 0|||||||||||||||||| | подстановка 3
24 |||||||||||||||||| | подстановка 3
```

7.4. Напишите программу, автоматически реализующий нормальный алгоритм Маркова, который складывает два целых числа.

В программу ПР – 7.1 следует вставить код:

```
slovo:='8eight+5five';
-----------------------
x[1]:='1one'; y[1]:='|';
x[2]:='2two'; y[2]:='||';
x[3]:='3three'; y[3]:='|||';
x[4]:='4four'; y[4]:='||||';
x[5]:='5five'; y[5]:='|||||';
x[6]:='6six'; y[6]:='||||||';
x[7]:='7seven'; y[7]:='|||||||';
x[8]:='8eight'; y[8]:='||||||||';
x[9]:='9nine'; y[9]:='|||||||||';
x[10]:='|+|'; y[10]:='||';
x[11]:='||||||||||'; y[11]:='10';
x[12]:='0|||||||||'; y[12]:='9';
x[13]:='0||||||||'; y[13]:='8';
x[14]:='0|||||||'; y[14]:='7';
x[15]:='0||||||'; y[15]:='6';
x[16]:='0|||||'; y[16]:='5';
x[17]:='0||||'; y[17]:='4';
x[18]:='0|||'; y[18]:='3';
x[19]:='0||'; y[19]:='2';
x[20]:='0|'; y[20]:='1';
```



```
x[21]:='|||||||||'; y[21]:='9';
x[22]:='||||||||'; y[22]:='8';
x[23]:='|||||||'; y[23]:='7';
x[24]:='||||||'; y[24]:='6';
x[25]:='|||||'; y[25]:='5';
x[26]:='||||'; y[26]:='4';
x[27]:='|||'; y[27]:='3';
x[28]:='||'; y[28]:='2';
x[29]:='|'; y[29]:='1';
```

Результат выполнения подстановок Маркова.

```
slovo:='8eight+5five';
------------------------------
1 8eight+||||| | подстановка 5
2 ||||||||+||||| | подстановка 8
3 |||||||||||||| | подстановка 10
4 10||| | подстановка 11
5 13 | подстановка 18
```

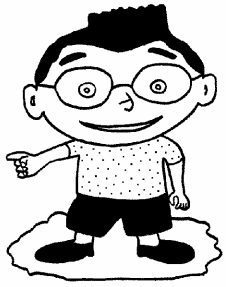
7.5. Напишите программу, автоматически реализующий нормальный алгоритм Маркова, который умножает два числа.

В программу ПР – 7.1 следует вставить код:

```
slovo:='1111*111';
------------------
x[1]:='*11'; y[1]:='A*1';
x[2]:='*1'; y[2]:='A';
x[3]:='1A'; y[3]:='A1B';
x[4]:='BA'; y[4]:='AB';
x[5]:='B1'; y[5]:='1B';
x[6]:='A1'; y[6]:='A';
x[7]:='AB'; y[7]:='B';
x[8]:='B'; y[8]:='1';
```

Результат работы программы:

```
slovo:='1111*111';
------------------------
1 1111A*11 | подстановка 1
2 1111AA*1 | подстановка 1
3 1111AAA | подстановка 2
4 111A1BAA | подстановка 3
5 11A1B1BAA | подстановка 3
6 1A1B1B1BAA | подстановка 3
```



```
7 A1B1B1B1BAA | подстановка 3
8 A1B1B1B1ABA | подстановка 4
9 A1B1B1BA1BBA | подстановка 3
10 A1B1B1AB1BBA | подстановка 4
11 A1B1BA1BB1BBA | подстановка 3
. . . . . .
61 111111111BBB | подстановка 8
62 1111111111BB | подстановка 8
63 11111111111B | подстановка 8
64 111111111111 | подстановка 8
```

Другой способ решения задачи:

```
slovo:='1111*111';
--------------------
x[1]:='1*'; y[1]:='X';
x[2]:='_1'; y[2]:='1_Z';
x[3]:='Z1'; y[3]:='1Z';
x[4]:='1X'; y[4]:='X_';
x[5]:='X'; y[5]:='';
x[6]:='_'; y[6]:='';
x[7]:='Z'; y[7]:='1';
```

7.6. Имеется число в четверичной системе счисления. Предложите систему нормальных подстановок, которая переводит это число в двоичную систему счисления. Апробируйте решение на компьютере.

В программу ПР – 7.1 следует вставить код:

```
slovo:='*3021032';
----------------------
x[1]:='*0';  y[1]:='00*';
x[2]:='*1';  y[2]:='01*';
x[3]:='*2';  y[3]:='10*';
x[4]:='*3';  y[4]:='11*';
x[5]:='*';   y[5]:=' ';
```

7.7. Дано двоичное число. Предложите систему нормальных подстановок, которая инвертирует все 0 и 1. Апробируйте решение на компьютере.

В программу ПР – 7.1 следует вставить код:

```
slovo:='*011010101';
--------------------
x[1]:='*0';  y[1]:='1*';
x[2]:='*1';  y[2]:='0*';
x[3]:='*';   y[3]:=' ';
```

7.8. Дано число в унарной системе счисления от 1 до 15. Предложите систему нормальных подстановок, которая представляет его как сумму степеней числа 2. Апробируйте решение на компьютере.

В программу ПР – 7.1 следует вставить код:

```
slovo:='||||||||||||_';
------------------------
x[1]:='||||||||';    y[1]:='8+';
x[2]:='||||';        y[2]:='4+';
x[3]:='||';          y[3]:='2+';
x[4]:='|';           y[4]:='1+';
x[5]:='+_';          y[5]:=' ';
```

7.9. Имеется слово '$bab\_ba\_aa\_babb\_aba$'. Создайте нормальный алгоритм Маркова, который символы 'a' переносит влево, символы 'b' — вправо, а пробелы оставляет посередине. Промоделируйте на ПЭВМ. (Ответ: 1) 'ba' $=>$ 'ab'; 2) '$b\_$' $=>$ '$\_b$'; 3) '$\_a$' $=>$ '$a\_$').

7.10. Имеется слово P на алфавите $A = \{a, b, c, d\}$. Создайте нормальный алгоритм Маркова, который кодирует это слово. Апробируйте решение задачи на компьютере. (Ответ: 1) 'a' $=>$ '$00-$'; 2) 'b' $=>$ '$01-$'; 3) 'c' $=>$ '$10-$'; 4) 'd' $=>$ '$11-$'; 5) 'e' $=>$ '$111-$').

7.11. В слове P на алфавите $A = \{a, b, c, d\}$ с помощью подстановок Маркова замените первое вхождение подслова bb на ddd и удалите все вхождения символа c. (Ответ: 'c' $=>$; 'bb' $|=>$ 'ddd'. Символ $|=>$ означает: выполнить подстановку и остановиться.)

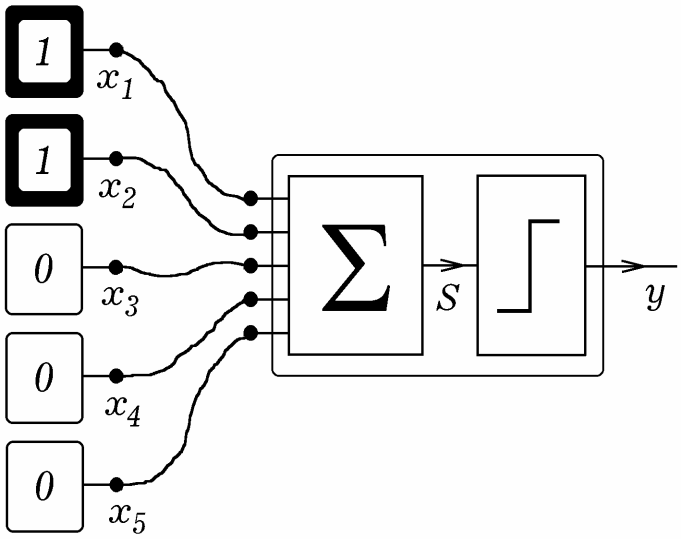
7.12. В слове P на алфавите $A = \{a, b\}$ с помощью подстановок Маркова удалите первый символ. Пустое слово не следует менять. (Ответ: '$*a$' $|=>$; '$*b$' $|=>$; '$*$' $|=>$; $|=>$ '$*$'. Последняя подстановка означает: слева приписать '$*$' и остановиться).

7.13. Дано слово P на алфавите $A = \{a, b\}$. С помощью подстановок Маркова припишите символ a к концу слова P справа. (Ответ: '$*a$' $=>$ '$a*$'; '$*b$' $=>$ '$b*$'; '$*$' $|=>$ 'a'; $=>$ 'a').

7.14. Имеется слово "$|||...||*|||...|$", где "$*$" — один из знаков "$+$" или "$-$". Предложите систему нормальных подстановок, реализующих операции сложения или вычитания в унарной системе счисления.

7.15. Пусть P — непустое слово на алфавите $A = \{0, 1, 2, 3\}$. Рассматривая его как запись натурального числа в четверичной системе счисления, получите запись этого числа в двоичной системе счисления. (Ответ: "$*0$" $=>$ "$00*$"; "$*1$" $=>$ '$01*$'; '$*2$' $=>$ '$10*$'; '$*3$' $=>$ '$11*$'; '$*$' $|=>$; $=>$ '$*$').

8.1. На базе формального нейрона создайте компьютерную модель однослойного персептрона, который бы осуществлял распознавание образов и классификацию объектов на два класса.



Класс	*Объекты*	*y*
1	*1,0,0,0,0;*	*1*
1	*1,1,0,0,0;*	*1*
1	*1,1,1,0,0;*	*1*
2	*0,0,1,1,0;*	*0*
2	*0,0,1,1,1;*	*0*
2	*0,0,0,1,0;*	*0*

Рис. 8.1. Однослойный персептрон на основе нейрона.

Под формальным нейроном понимают гипотетический автомат с n входами $x_1, x_2, ..., x_n$ и одним выходом y, характеризующийся порогом h и весами $\omega_1, \omega_2, ..., \omega_n$. Он состоит из суммирующего и порогового элементов (рис. 8.1). Его выход возбужден $(y = 1)$, когда сумма всех весов возбужденных входов превышает порог срабатывания: $\omega_1 x_1 + \omega_2 x_2 + ... + \omega_n x_n > h$. В противном случае выход не возбужден $(y = 0)$. Это можно записать так:

$$y = \begin{cases} 1, & \text{если } \omega_1 x_1 + \omega_2 x_2 + ... + \omega_n x_n \geq h, \\ 0, & \text{в противном случае.} \end{cases}$$

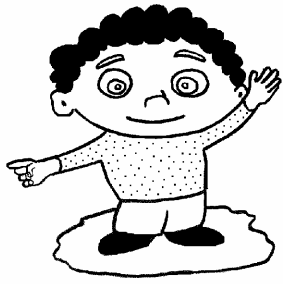
Если вес i–го входа положительный $(\omega_i > 0)$, то вход возбуждающий, если отрицательный $(\omega_i < 0)$, — вход тормозящий. Выход искусственного нейрона может находиться в двух состояниях, поэтому он может разделять объекты только на два класса. Чтобы симитировать работу нейрона достаточно найти взвешенную сумму его входов и использовать оператор условного перехода.

```
uses crt;                                          { ПР - 8.1 }
const N=6; h=0.2;
x: array[1..N,1..5]of integer=((1,0,0,0,0),(1,1,0,0,0),
      (1,1,1,0,0),(0,0,1,1,0),(0,0,1,1,1),(0,0,0,1,0));
w: array[1..5] of real=(1,1,1,-1,-1);
var m,i,j,k,y,DV,MV,EC : integer; S: real;
```

```
BEGIN  clrscr;
For m:=1 to N do
  begin S:=0;
    For i:=1 to 5 do S:=S+w[i]*x[m,i];
    If S>h then y:=1 else y:=0;
    Writeln('Объект ',m,': ','S= ',S:2:1,'; y= ',y);
  end;
Readkey;
END.
```

Так работает программа ПР – 8.1: формальному нейрону последовательно предъявляются 6 объектов, образующих два класса: $K_1 = \{10000, 11000, 11100\}$ *и* $K_2 = \{00111, 00111, 00010\}$. *Веса входов подобраны так:* $\omega_i = (1, 1, 1, -1, -1)$, *порог срабатывания равен* $h = 0,2$. *После запуска программы получаем:*

```
Объект 1: S=  1.0; y= 1
Объект 2: S=  2.0; y= 1
Объект 3: S=  3.0; y= 1
Объект 4: S= -1.0; y= 0
Объект 5: S= -1.0; y= 0
Объект 6: S= -1.0; y= 0
```



8.2. Рассчитайте нейросеть с 2 входами и 4 нейронами (рис. 8.2.1). При подаче на вход сигналов $o[1] = 00$, $o[2] = 01$, $o[3] = 10$, $o[4] = 11$, на выходах нейронов должно появиться $y[1, j] = 1000$, $y[2, j] = 0100$, $y[3, j] = 0010$, $y[4, j] = 0001$, то есть соответствующий нейрон должен быть возбужден.

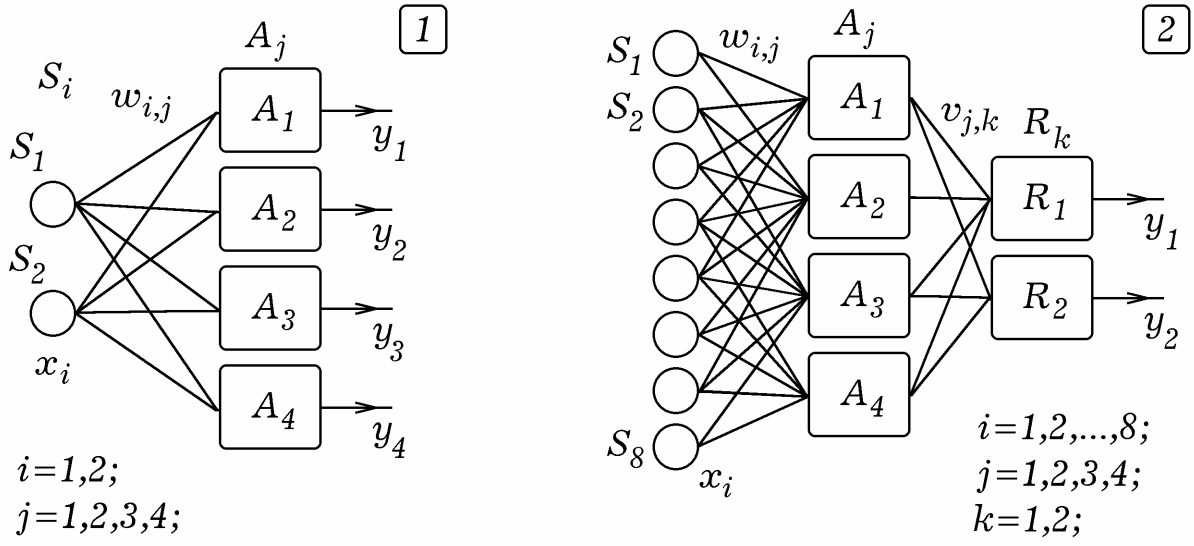
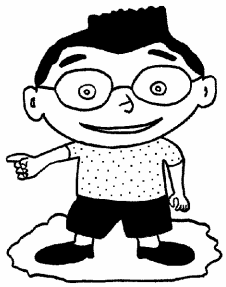


Рис. 8.2. Нейросети: 1) 2 входа, 4 нейрона; 2) 8 входов, 4 ассоциативных и 2 реагирующих элементов.

Написать программу, моделирующую работу этой нейросети, несложно. Веса синапсов и порог срабатывания нейрона задаются матрицами: $w[1,j] = (-1,-2,1,1)$, $w[2,j] = (-1,1,-2,1)$, $h[j] = (-0.5,0,0,1.5)$. *Решением задачи является программа ПР – 8.2.*

```
uses crt;                                          { ПР - 8.2 }
const
   w: array[1..2,1..4]of real =((-1,-2,1,1),(-1,1,-2,1));
   h: array[1..4]of real =(-0.5,0,0,1.5);
var i,j,y1,y2,y3,y4,x1,x2 : integer;
Function Neiron_2(j,a1,a2:integer):integer;
begin
If w[1,j]*a1+w[2,j]*a2>h[j]
              then Neiron_2:=1 else Neiron_2:=0;
end;
BEGIN clrscr;
For x1:=0 to 1 do
  For x2:=0 to 1 do
    begin
      y1:=Neiron_2(1,x1,x2);
      y2:=Neiron_2(2,x1,x2);
      y3:=Neiron_2(3,x1,x2);
      y4:=Neiron_2(4,x1,x2);
      writeln(x1,x2,' | ',y1,y2,y3,y4);
    end;
Readkey;
END.
```



8.3. Промоделируйте персептрон (двухслойную нейросеть) с 8 входами (сенсорами), 4 ассоциативными элементами и 2 реагирующими элементами (рис. 8.2.2). При предъявлении объектов $o[1] = 11110000$, $o[2] = 00001111$, $o[3] = 00111100$, $o[4] = 11000011$ на выходах персептрона должны появиться сигналы $y[1,j] = 00$, $y[2,j] = 01$, $y[3,j] = 10$, $y[4,j] = 11$.

Программа ПР – 8.3, моделирующая работу этой нейросети, представлена ниже.

```
uses dos, crt;                                     { ПР - 8.3 }
type  Neiron_8 = record
      w : array[1..8]of real; h: real; end;
      Neiron_4 = record
      v : array[1..4]of real; h: real; end;
const
```

```
Ne8: array[1..4]of Neiron_8=(
                        (w:(1,1,1,1,-1,-1,-1,-1); h: 2.5),
                        (w:(-1,-1,-1,-1, 1,1,1,1); h: 2.5),
                        (w:(-1,-1,1,1,1,1,-1,-1); h: 2.5),
                        (w:(1,1,-1,-1,-1,-1,1,1); h: 2.5));
Ne4: array[1..2]of Neiron_4=((v:(-1,0,1,1); h: 0.5),
                             (v:(-1,1,0,1); h: 0.5));
objekt: array[1..4,1..8]of integer =((1,1,1,1,0,0,0,0),
                                     (0,0,0,0,1,1,1,1),
                                     (0,0,1,1,1,1,0,0),
                                     (1,1,0,0,0,0,1,1));
var i,j,k,o,z1,z2 : integer; S: real;
    x: array[1..8]of integer;  y: array[1..4]of integer;
Function Neiron8(j:integer; a:array of integer):integer;
begin S:=0;
  For i:=1 to 8 do S:=S+Ne8[j].w[i]*objekt[o,i];
  If S>Ne8[j].h then Neiron8:=1 else Neiron8:=0;
end;
Function Neiron4(j: integer; a :array of integer):integer;
begin S:=0;
  For i:=1 to 4 do S:=S+Ne4[j].v[i]*y[i];
  If S>Ne4[j].h then Neiron4:=1 else Neiron4:=0;
end;
BEGIN clrscr;
For o:=1 to 4 do begin
    y[1]:=Neiron8(1,objekt[o,i]);
    y[2]:=Neiron8(2,objekt[o,i]);
    y[3]:=Neiron8(3,objekt[o,i]);
    y[4]:=Neiron8(4,objekt[o,i]);
    z1:=Neiron4(1,y[k]); z2:=Neiron4(2,y[k]);
    For i:=1 to 8 do write(objekt[o,i],' ');
    For j:=1 to 4 do write(' : ',y[j],' ');
    writeln(' | ',z1,' ',z2);
  end; Readkey;
END.
```

8.4. Имеется однослойная нейросеть с 9 входами и 4 нейронами. Напишите программу, вычисляющую веса w так, чтобы сеть распознавала 8 объектов, представленные на рис. 8.3.

Объекты закодируем так: $o[1] = 000010111$, $o[2] = 111101000$ *и т.д. Им соответствуют следующие сигналы на выходах нейронов:*

$$y[1,j] = 0000,\ y[2,j] = 1111,\ y[3,j] = 1001,\ y[4,j] = 0110,$$
$$y[5,j] = 1010,\ y[6,j] = 0101,\ y[7,j] = 1100,\ y[8,j] = 0011.$$

Необходимо последовательно предъявлять нейросети объекты $o[1]$, $o[2]$, ..., $o[8]$ *и изменять веса в соответствии с правилом: 1) если вход* $x[i]$ *и выход* $y[j]$ *одновременно возбуждены, то вес связи* $w[i,j]$ *увеличивается; 2) если вход* $x[i]$ *возбужден, а выход* $y[j]$ *невозбужден, то вес связи* $w[i,j]$ *уменьшается. Порог срабатывания для всех нейронов будем считать равным 0. Используется программа ПР–8.4. В результате расчетов получается матрица весов:*

$$w[i,1] := (1,1,1,-1,-0.5,0,0,-1,0);$$
$$w[i,2] := (-1,1,1,1,-0.5,2,-2,-1,0);$$
$$w[i,3] := (1,1,-1,1,-1.5,0,0,1,-2);$$
$$w[i,4] := (1,-1,1,1,0.5,0,0,-1,0).$$

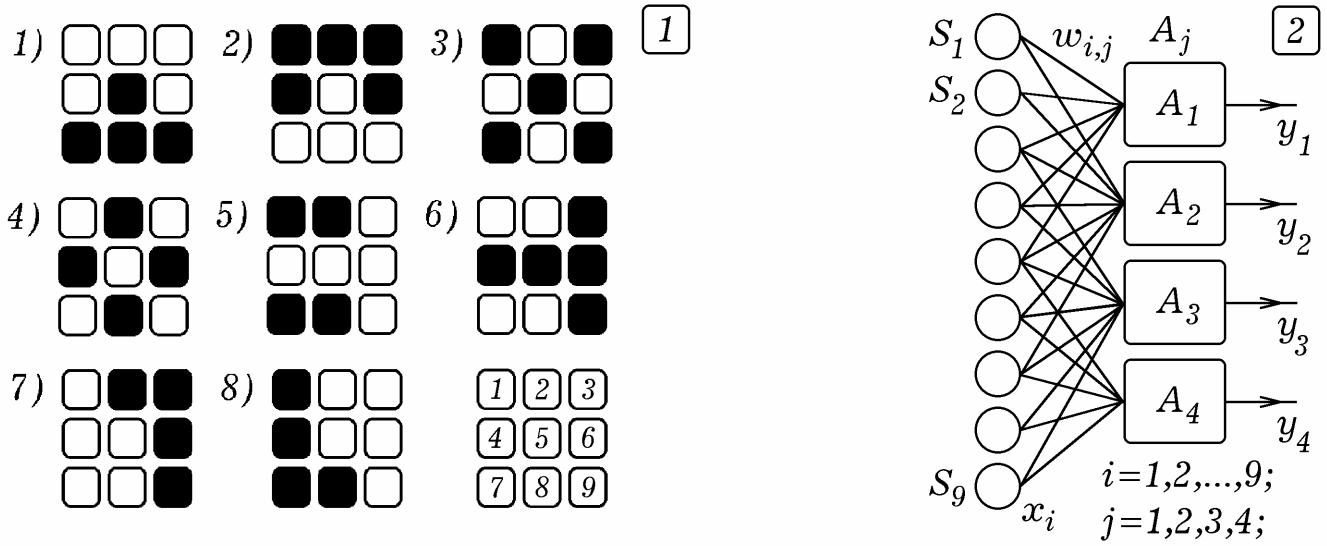


Рис. 8.3. Объекты для распознавания. Схема нейросети.

```
uses crt;                                  { ПР - 8.4 }
const
x: array[1..8,1..9] of integer=((0,0,0,0,1,0,1,1,1),
            (1,1,1,1,0,1,0,0,0),(1,0,1,0,1,0,1,0,1),
            (0,1,0,1,0,1,0,1,0),(1,1,0,0,0,0,1,1,0),
            (0,0,1,1,1,1,0,0,1),(0,1,1,0,0,1,0,0,1),
                                (1,0,0,1,0,0,1,1,0));
y0: array[1..8,1..4] of integer=((0,0,0,0),(1,1,1,1),
                        (1,0,0,1),(0,1,1,0),(1,0,1,0),
                       (0,1,0,1),(1,1,0,0),(0,0,1,1));
var i,j,o,t: integer; Sw : real;
    S : array[1..4] of real; y : array[1..4] of real;
    w : array[1..9,1..4] of real;
BEGIN Clrscr;
Repeat inc(t);
  For o:=1 to 8 do For j:=1 to 4 do
  For i:=1 to 9 do begin
```

```
  If x[o,i]*y0[o,j]=1 then w[i,j]:=w[i,j]+0.01;
  If (x[o,i]=1)and(y0[o,j]=0) then w[i,j]:=w[i,j]-0.01;
  end;
until (t>50)or(KeyPressed);
For i:=1 to 9 do For j:=1 to 4 do Sw:=Sw+w[i,j];
For i:=1 to 9 do begin
    For j:=1 to 4 do write(' w',i,j,'=',w[i,j]:2:1);
    writeln; end;
{=== ПРОВЕРКА ===}
For o:=1 to 8 do begin For j:=1 to 4 do begin
 S[j]:=0;
 For i:=1 to 9 do S[j]:=S[j]+x[o,i]*w[i,j];
 If S[j]>0 then y[j]:=1 else y[j]:=0;
 write(' ',{S[j]:1:2,' ',}y[j]:1:0);
 end; writeln(' objekt ',o);
 end; Readkey;
END.
```

8.5. Создайте компьютерную модель трехслойного персептрона, содержащего 3 ассоциативных и 2 реагирующих элементов. Подберите веса связей так, чтобы персептрон осуществлял классификацию объектов на 4 класса.

Под персептроном понимают обучаемую нейросеть, состоящую из датчиков, ассоциативных и реагирующих элементов с заданной матрицей весовых коэффициентов. В многослойных персептронах присутствуют дополнительные слои ассоциативных элементов. В общем случае персептрон оперирует с цифровыми образами объектов, каждый из которых представим в виде массива нулей и единиц.

Рассмотрим трехслойный персептрон (рис. 8.4), состоящий из слоя сенсоров или датчиков D_i $(i = 1, 2, ..., 9)$, слоя ассоциативных элементов A_j $(j = 1, 2, 3)$, и слоя реагирующих элементов R_k $(k = 1, 2)$. Если уровень воздействия на датчик превышает некоторое пороговое значение h, то на его выходе появляется 1, а иначе — 0. Ассоциативный элемент работает как формальный нейрон: на выходе 1, когда сумма всех весов возбужденных входов превышает порог срабатывания; в противном случае на выходе 0. Веса входов $w_{i,j}$ принимают значения $-1, 0$ или 1. Реагирующий R–элемент работает так: когда сумма всех весов возбужденных входов положительна, на выходе 1, а когда отрицательна, — на выходе 0. Веса входов $v_{j,k}$ реагирующего элемента могут принимать произвольные значения. Выход реагирующего элемента имеет два состояния 0 и 1, поэтому персептрон с двумя решающими элементами может классифицировать объекты на четыре класса $(2^2 = 4)$, соответствующие выходным сигналам $00, 01, 10, 11$.

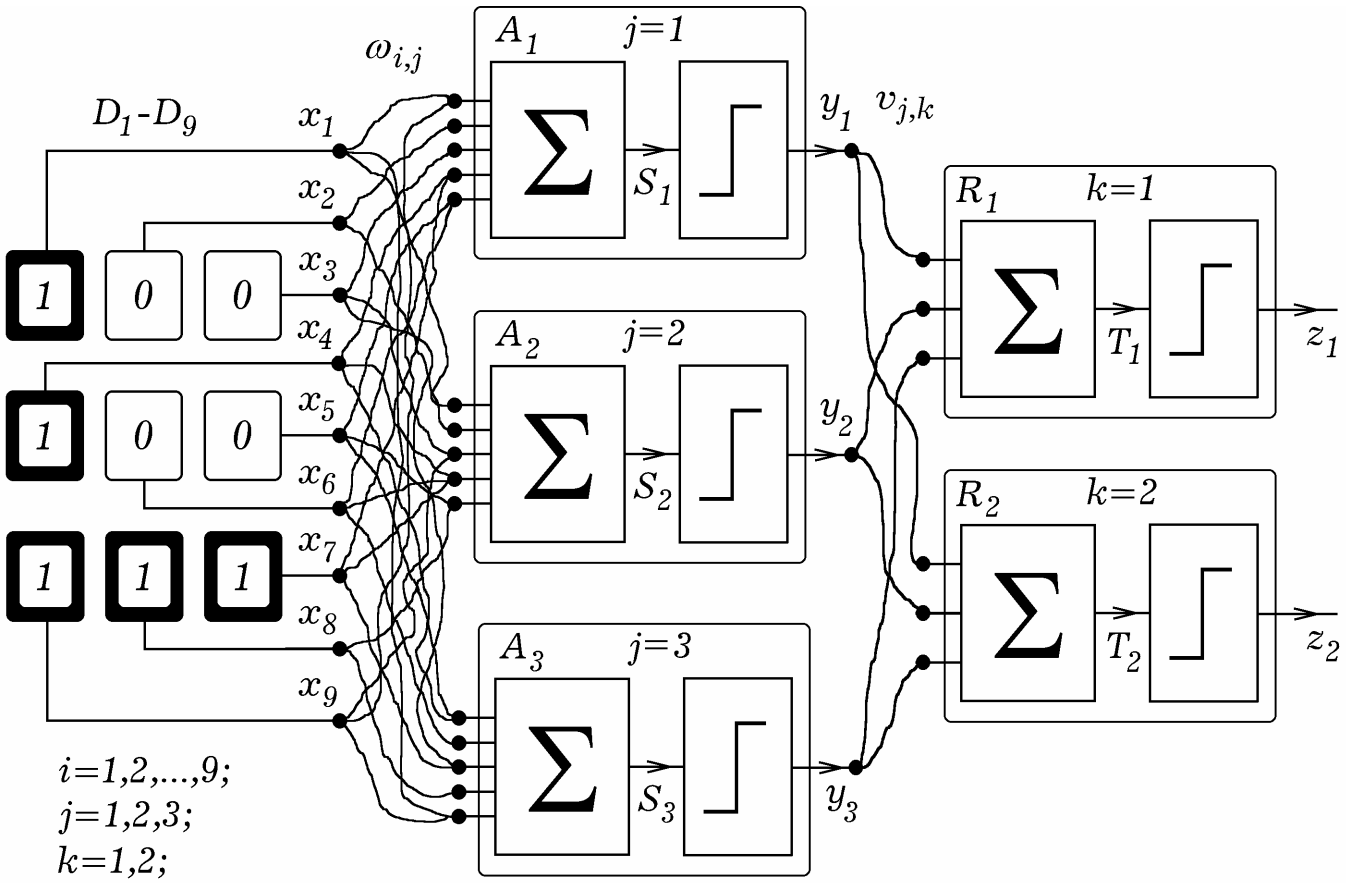


Рис. 8.4. Схема трехслойного персептрона.

Рассмотрим программу ПР–8.5, моделирующую работу трехслойного персептрона, изображенного на рисунке. Допустим, персептрон должен различать четыре объекта, изображенные на рис. 8.5.

Объект 1			*Объект 2*			*Объект 3*			*Объект 4*			*Датчики*		
0	1	0	0	0	0	1	0	0	1	1	1	x_1	x_2	x_3
0	1	0	1	1	1	1	0	0	0	0	1	x_4	x_5	x_6
0	1	0	0	0	0	1	1	1	0	0	1	x_7	x_8	x_9

Рис. 8.5. Объекты для распознавания.

Учитывая расположение датчиков $D_i (i = 1, 2, ..., 9)$, *предъявляемые персептрону объекты кодируются так:*

$$O_1 = (0,1,0,0,1,0,0,1,0),\ O_2 = (0,0,0,1,1,1,0,0,0),$$
$$O_3 = (1,0,0,1,0,0,1,1,1),\ O_4 = (1,1,1,0,0,1,0,0,1)).$$

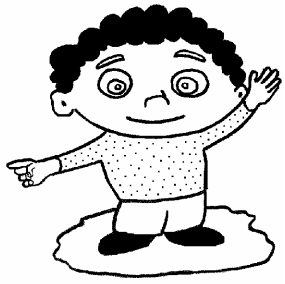
Для того, чтобы персептрон правильно классифицировал объекты, необходимо задать веса связей, соединяющих датчики с ассоциативными элементами и ассоциативные элементы с реагирующими элементами, следующим образом: $w_{ij} = (0,-1,1,1,0,1,-1,-1,0)$, $(0,1,1,-1,0,1,-1,-1,0)$, $(1,0,1,-1,0,0,1,-1,1))$; $v_{jk} = (2,2,-2)$ $(-2,2,2)$. *Порог срабатывания ассоциативных элементов* h *равен* $0,5$.

```
uses crt;                                        { ПР - 8.5 }
const x: array[1..9] of integer=(1,0,0,
                                 1,0,0,
                                 1,1,1);
w: array[1..3,1..9] of real=((0,-1,1,1,0,1,-1,-1,0),
      (0,1,1,-1,0,1,-1,-1,0),(1,0,1,-1,0,0,1,-1,1));
v: array[1..2,1..3] of real=((2,2,-2),(-2,2,2));
var m,i,j,k,DV,MV,EC : integer;
    S,y : array[1..3] of real;
    T,z : array[1..2] of real;
BEGIN clrscr;
For j:=1 to 3 do begin
  S[j]:=0;
  For i:=1 to 9 do
  S[j]:=S[j]+w[j,i]*x[i];
  If S[j]>0.5 then y[j]:=1 else y[j]:=0;
  Writeln('j= ',j,'| S= ',S[j]:2:1,'| y= ',y[j]:1:1);
  end;
For k:=1 to 2 do begin T[k]:=0;
For j:=1 to 3 do T[k]:=T[k]+v[k,j]*y[j];
If T[k]>0 then z[k]:=1 else z[k]:=0;
  Writeln('k=',k,'| T= ',T[k]:2:1,
                ' | z= ',z[k]:1:0);
  end; Readkey;
END.
```

Результат работы программы представлен ниже. Из распечатки видно, какие значения принимают величины S_j, T_k, а также выходы y_j и z_k в случае, когда персептрону предъявляются объекты O_1, O_2, O_3, O_4. Объекту O_1 соответствуют состояния $z_1 = 0, z_2 = 0$, объекту O_2 — состояния $z_1 = 1, z_2 = 0$ и т.д. Таким образом, наша компьютерная модель персептрона распознает рассмотренные выше объекты. Вы можете дальше поэкспериментировать: изменить веса связей, предъявить персептрону другие объекты, промоделировать персептрон с тремя реагирующими элементами.

```
====== ОБЪЕКТ 1
j= 1 | S= -2.0 | y= 0.0
j= 2 | S= 0.0 | y= 0.0
j= 3 | S= -1.0 | y= 0.0
k= 1 | T= 0.0 | Выход z= 0
k= 2 | T= 0.0 | Выход z= 0
====== ОБЪЕКТ 2
j= 1 | S= 2.0 | y= 1.0
```

```
j= 2 | S= 0.0 | y= 0.0
j= 3 | S= -1.0 | y= 0.0
k= 1 | T= 2.0 | Выход z= 1
k= 2 | T= -2.0 | Выход z= 0
====== ОБЪЕКТ 3
j= 1 | S= -1.0 | y= 0.0
j= 2 | S= -3.0 | y= 0.0
j= 3 | S= 1.0 | y= 1.0
k= 1 | T= -2.0 | Выход z= 0
k= 2 | T= 2.0 | Выход z= 1
====== ОБЪЕКТ 4
j= 1 | S= 1.0 | y= 1.0
j= 2 | S= 3.0 | y= 1.0
j= 3 | S= 3.0 | y= 1.0
k= 1 | T= 2.0 | Выход z= 1
k= 2 | T= 2.0 | Выход z= 1
```



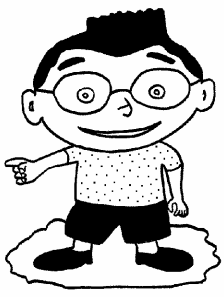
8.6. Сконструируйте нейросеть с пятью входами и одним выходом, которая различала бы два класса объектов $K_1 = \{10100,\ 11000,\ 01100,\ 11100\}$ и $K_2 = \{00010,\ 00001,\ 00011\}$. Задайте веса связей, промоделируйте ее работу на компьютере.

8.7. Сконструируйте нейросеть с шестью входами и двумя выходами, которая различала бы три класса объектов $K_1 = \{110000, 011000, 101000,\ 111000\}$, $K_2 = \{000110,\ 000101,\ 000011,\ 000111\}$ и K_3, в который входят все остальные объекты. Задайте веса связей, промоделируйте ее работу на компьютере.

9.1. Напишите программу, моделирующую парную игру человека (игрок A) с компьютером (игрок B). Перед каждым ходом программа формирует матрицу игры так, чтобы цена игры принимала бы случайное значение в интервале от -5 до 5. Ходы человека соответствуют строкам, ходы компьютера — столбцам матрицы игры. Компьютер делает ходы случайно. Сыграйте несколько раз, используя стратегии: 1) случайный выбор хода; 2) выбор хода, который может принести наибольший выигрыш при ошибке игрока B; 3) выбор хода, который минимизирует проигрыш при наименее благоприятных действиях игрока B (стратегия минимакса).

Программа ПР-9.1, моделирующая игру человека с компьютером представлена ниже.

```
uses dos, crt;                                    { ПР - 9.1 }
const N=4;
var x,i,j,k,hodA,hodB,code,S: integer; hodAstr : string;
cena : array[1..N,1..N] of integer;
BEGIN clrscr; Randomize;
Repeat inc(k);
For j:=1 to N do begin For i:=1 to N do begin
  cena[i,j]:=round(random(100)/10-5);
  write(cena[i,j],' | ');
end; writeln('TVOI HOD A',j);
end; writeln('TVOI HOD A (1,2,3,4)');
hodAstr:=readkey; val(hodAstr,hodA,code);
hodB:=round(random(34)/10)+1;
S:=S+cena[hodB,hodA];
writeln('tvoi hod A',hodA,'
   hod komputera B',hodB);
writeln('VIIGRISH A',
cena[hodB,hodA],'SUM A ',S);
until (keypressed)or(k>10); Readkey;
END.
```



9.2. Промоделируйте парную игру человека (игрок A) с компьютером (игрок B), при которой компьютер выбирает наихудший для вас вариант. Матрица игры формируется случайно из интервала от -5 до 5. Поиграйте с компьютером, используя стратегию минимакса.

9.3. Промоделируйте парную игру человека (игрок A) с компьютером (игрок B), при которой компьютер выбирает наихудший для вас вариант. Матрица игры формируется случайно из интервала от -3 до 8. Поиграйте с компьютером, используя стратегию минимакса.

Программа ПР–9.2, моделирующая эту игру, представлена ниже.

```
uses dos, crt;                                   { ПР - 9.2 }
const N=4;
var x,i,j,k,jj,hodA,hodB,code,S,min : integer;
hodAstr : string;
cena:array[1..N,1..N] of integer;
max: array[1..N] of integer;
BEGIN clrscr; Randomize;
Repeat inc(k);
For j:=1 to N do begin
    For i:=1 to N do begin
      cena[i,j]:=round(random(100)/10-5);
      write(cena[i,j],' | ');
     end; writeln;
end; writeln('TVOI XOD A (1, 2, 3, 4, 5)');
hodAstr:=readkey; val(hodAstr,hodA,code);
hodB:=1; min:=cena[1,hodA];
For i:=1 to N do if cena[i,hodA]<min then
  begin min:=cena[i,hodA]; hodB:=i; end;
S:=S+cena[hodB,hodA];
writeln('tvoi hod A',hodA,' hod PC B ',hodB);
writeln('VIIGR A',cena[hodB,hodA],'SUM A ',S);
until (keypressed)or(k>8); Readkey;
END.
```

9.4. Три человека A, B и C стали свидетелями ограбления банка. В процессе расследования они заявили: A : преступники уехали на синей "Тойоте"; B : грабители были на красном "BMW"; C : у преступников был "Форд", но не синий. Каждой свидетель ошибся один раз. Какой автомобиль был у преступников?

Утверждения "цвет машины синий", "цвет машины красный" обозначим через S и K. Утверждения "марка машины Тойота", "марка машины BMW", "марка машины Форд" обозначим через T, B и F. Получаем логические уравнения: $\overline{S}T + S\overline{T} = 1$, $\overline{K}B + K\overline{B} = 1$, $SF + \overline{SF} = 1$, $TB + TF + BF = 0$, $SK = 0$. Путем перебора всех вариантов следует найти такие значения S, K, T, B и F, при которых перечисленные 5 уравнений превращаются в истинные высказывания. Для этого используется программа ПР – 9.3. Ответ: красная Тойота.

```
uses crt;                                          { ПР - 9.3 }
var S,T,K,B,F : boolean; n :integer;
BEGIN clrscr; n:=0;
for S:=false to true do for T:=false to true do
for K:=false to true do for B:=false to true do
for F:=false to true do begin
  if (S)and(not(T))or(not(S)and(T)) then n:=n+1;
  if (K)and(not(B))or(not(K)and(B)) then n:=n+1;
  if (S)and(F)or(not(S)and(not(F)))
                          then n:=n+1;
  if (T)and(B)or((T)and(F))or
       ((B)and(F))=false then n:=n+1;
  if (S)and(K)=false then n:=n+1;
writeln(S,' ',T,' ',K,' ',B,' ',F,' ',n); n:=0;
end; readkey;
END.
```

9.5. Археологи А, Б и В нашли монету. Каждый высказал по 2 предположения: А: монета греческая, 5 век; Б: монета испанская, 3 век; В: монета не греческая, 4 век. Каждый из археологов прав только в одном из двух предположений. Где и когда была выпущена монета?

Обозначим простые высказывания: G — монета греческая, I — монета испанская, P — 5 век, C — четвертый век, T — третий век. Высказывания: $G\overline{P}+\overline{G}P=1$, $I\overline{T}+\overline{I}T=1$, $\overline{G}\,\overline{C}+G\,C=1$. Монета не может быть изготовлена в двух государствах и двух веках: $GI=0$, $PC=0$, $PT=0$, $CT=0$. Для решения используется программа ПР – 9.4, в которой перебираются все варианты и автоматически находится приемлемый.

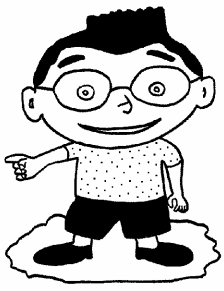
```
uses crt;                                          { ПР - 9.4 }
var G,P,I,T,C: boolean; k,l,m,n,o: integer;
BEGIN clrscr; k:=0;
for g:=false to true do for p:=false to true do
for t:=false to true do for c:=false to true do
for i:=false to true do begin
if ((g)and(not(p))or(not(g)and(p))=true) then k:=k+1;
if ((i)and(not(t)))or(not(i)and(t))=true then k:=k+1;
if (not(g)and(not(c)))or((g)and(c))=true then k:=k+1;
if(g)and(i)=false then k:=k+1;
if(p)and(c)=false then k:=k+1;
if(p)and(t)=false then k:=k+1;
if(c)and(t)=false then k:=k+1;
writeln(g,' ',p,' ',t,' ',c,' ',i,' ',k,' ');
```

```
k:=0; end; readkey;
END.
```

9.6. Кто-то разбил вазу. Коля, Миша и Саша сказали: Коля: "Миша не разбивал вазу, его разбил Саша"; Миша: "Я не разбивал вазу и Коля тоже"; Саша: "Я не разбивал вазу, ее разбил Миша". Оказалось, что один из них сказал правду в обоих случаях, другой — оба раза обманул, а третий — один раз сказал правду, другой раз соврал. Кто разбил вазу?

Утверждения "Коля разбил вазу", "Миша разбил вазу", "Саша разбил вазу" обозначим через K, M, S. *Получаем уравнения:* $\overline{MK} + \overline{MS} + \overline{S}M = 1$, $M\overline{K} + \overline{M}K + MS + \overline{MS} = 1$, $MK + M\overline{S} + S\overline{M} = 1$, $MK + MS + KS = 0$. *В программе ПР – 9.5 перебираются все варианты. Ответ: вазу разбил Миша.*

```
uses crt;                                          { ПР - 9.5 }
var M,K,S : boolean; n :integer;
BEGIN clrscr; n:=0;
for M:=false to true do
for K:=false to true do
for S:=false to true do begin
if (not(M)and(not(K)))or(not(M)and(S))or(not(S)and(M))
                                                  then n:=n+1;
if ((M)and(K))or((M)and(not(S)))or((S)and(not(M)))
                                             then n:=n+1;
if (M)and(K)or((M)and(S))or((K)and(S))=false
                                           then n:=n+1;
if (M)and(not(K))or(not(M)and(K))or
((M)and(S))or(not(M)and(not(S))) then n:=n+1;
writeln(M,' ',K,' ',S,' ',n); n:=0;
end; Readkey;
END.
```



9.7. В ряд стоят четыре дома, в каждом живет по одному человеку: Алексей, Егор, Михаил и Виктор. У всех разные профессии: фермер, пчеловод, рыбак и учитель. Известно следующее: 1) рыбак живет правее фермера; 2) Виктор — не пчеловод; 3) Алексей живет левее Виктора; 4) рыбак живет рядом с Егором; 5) фермер живет правее пчеловода; 6) учитель живет рядом с рыбаком; 7) пчеловод живет через дом от рыбака; 8) фермер живет левее Алексея. Кто где живет?

Буквами A, E, M *и* V *обозначим номера домов, в которых живут Алексей, Егор, Михаил и Виктор. Буквами* F, P, R *и* U *обозначим номера домов, в которых живут фермер, пчеловод, рыбак и учитель.*

Все эти переменные принимают значения $1, 2, 3$ *или* 4. *Получаем:* $R > F, V \neq P, A < V, |R-E| = 1, F > P, |U-R| = 1, |P-R| = 2, F < A$. *Так как в каждом доме не может быть два человека, то* $A \neq E$, $A \neq V$, $A \neq M$, $E \neq V$, $E \neq M$, $V \neq M$, $R \neq P$, $R \neq F$, $R \neq U$, $P \neq F$, $P \neq U$, $F \neq U$. *В программе ПР – 9.6 перебираются все варианты и находится тот, при котором все* 20 *условий выполняются. Ответ: 1) пчеловод Михаил; 2) фермер Егор; 3) рыбак Алексей; 4) учитель Виктор.*

```
uses crt;                                              { ПР - 9.6 }
var A,E,V,M,R,P,F,U,N : integer;
BEGIN clrscr; n:=0;
for A:=1 to 4 do for E:=1 to 4 do for V:=1 to 4 do
for M:=1 to 4 do for R:=1 to 4 do for P:=1 to 4 do
for F:=1 to 4 do for U:=1 to 4 do begin
if (F>P)and(R>F)and(ABS(U-R)=1)and(ABS(R-P)=2)
                                                    then N:=N+1;
if (A>F)and(V>A)and(V<>P)and(ABS(E-R)=1) then N:=N+1;
if (A<>E)and(A<>V)and(A<>m)and(E<>V)
             and(E<>M)and(V<>M) then N:=N+1;
if (R<>P)and(R<>F)and(R<>U)and(P<>F)
             and(P<>U)and(F<>U) then N:=N+1;
if N>3 then write(A,E,V,M,R,P,F,U,
                  ' ',n,'| '); n:=0;
end; Readkey;
END.
```

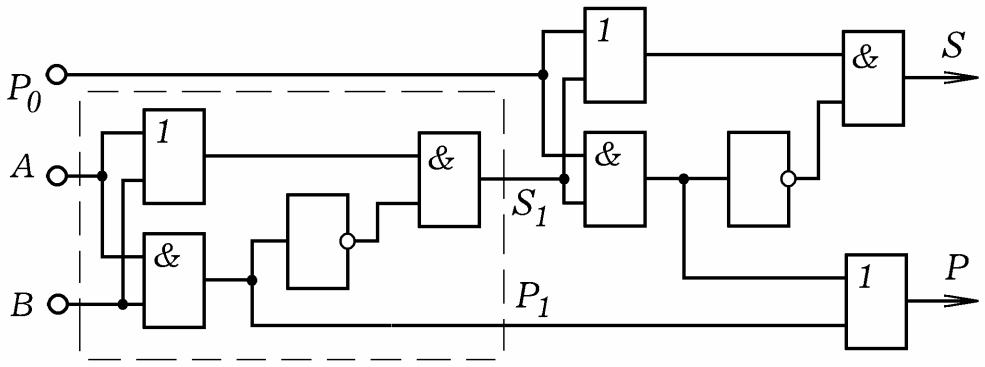
9.8. Сумматор (рис. 9.1) состоит из двух полусумматоров, обведенных штриховой линией. На входы A и B подаются складываемые биты, на выходе S получается их сумма. На вход P_0 подается цифра переноса из младшего разряда, на выходе P появляется цифра переноса в старший разряд. Промоделируйте работу сумматора при всевозможных комбинациях входных сигналов, получите таблицу истинности.

Для решения задачи используется программа ПР–9.7.

```
uses crt;                                              { ПР - 9.7 }
var A,B,S1,S,P1,P0,P : boolean;
BEGIN clrscr;
for A:=false to true do for B:=false to true do
for P0:=false to true do begin
P1:=A and B;
S1:=(A or B)and(not(A and B));
```

```
P:=(S1 and P0) or P1;
S:=(S1 or P0)and(not(S1 and P0));
writeln(A,' ',B,' ',P0,' S=',S,' P=',P);
end; ReadKey;
END.
```

Можно проанализировать работу полусумматора (на рис. 9.1 он обведен пунктиром).



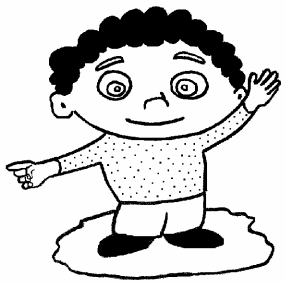


Рис. 9.1. Схема сумматора.

9.9. Имеются две кучки камней: в одной — 1, а в другой — 4 камня. Два игрока имеют неограниченное число камней, каждый ходит по очереди. Ход заключается в том, что игрок добавляет к одной из кучек 3 камня, либо увеличивает их число в 3 раза. Игрок выигрывает, если после его хода суммарное число камней в обоих кучках достигнет 22 или более. Кто выиграет при безошибочной игре: тот кто сделает первый ход или другой игрок? Каким должен быть первый ход выигрывающего игрока? Составьте полное дерево игры.

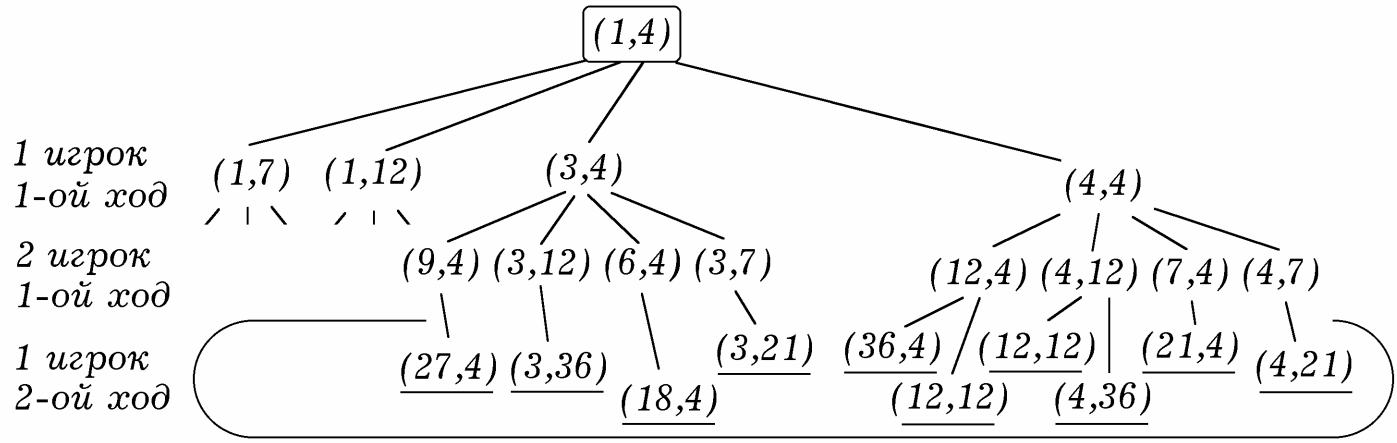


Рис. 9.2. Неполное дерево игры.

Неполное дерево игры приведено на рис.9.2. Выигрывает игрок, делающий первый ход и переводящий систему от (1,4) в состояние (3,4) и (4,4). Выигрышные ходы подчеркнуты. Самостоятельно составьте полное дерево игры.

9.10. Имеются две кучки камней: в одной — 6, а в другой — 5 камней. Два игрока имеют неограниченное число камней, каждый

ходит по очереди. Ход заключается в том, что игрок удваивает или утраивает количество камней в одной кучке. Игрок выигрывает, если после его хода суммарное число камней в обоих кучках достигнет 48 или более. Кто выиграет при безошибочной игре: тот кто сделает первый ход или другой игрок? Каким должен быть первый ход выигрывающего игрока? Составьте дерево игры.

9.11. Имеется веревка длиной 18 см. Два игрока ходят по очереди. Ход заключается в том, что игрок отрезает от веревки 4 или 5 см. Выигрывает тот, на чьем ходе закончится веревка (выигрышный ход может быть меньше 4 см). Кто выиграет при безошибочной игре: тот, кто сделает первый ход или тот, кто сходит вторым? Постройте дерево игры.

9.12. Напишите программу, которая работает так: компьютер случайно загадывает число x от 1 до 256. Вы пытаясь угадать, вводите число a. Компьютер отвечает x "больше" или "меньше" a. Вы снова вводите a и т.д. Какова правильная стратегия игры? Сколько вопросов достаточно задать компьютеру, чтобы обязательно угадать число?

9.13. Создайте компьютерную программу, моделирующую работу автомата (рис. 9.3), состоящего из элементов И, ИЛИ, НЕ и двух задерживающих элементов Z_1 и Z_2.

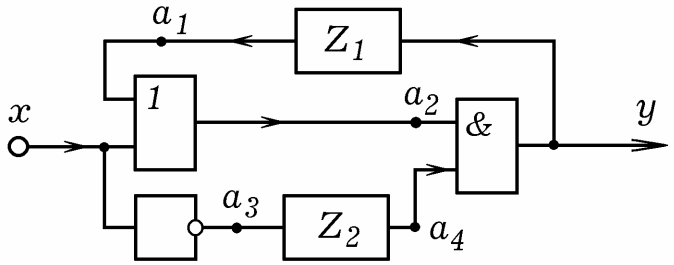


Рис. 9.3. Схема цепи с задерживающими элементами.

```
uses crt;                                              { ПР - 9.8 }
var y1,i : integer; s : string;
x,y,z1,z2,a1,a2,a3,a4 : boolean;
BEGIN clrscr; z1:=true; z2:=true;
s:='00110101001111101111001';
For i:=1 to length(s) do begin
If s[i]='1' then x:=true else x:=false;
a3:=not(x); a4:=z1; z1:=a3; a1:=z2; z2:=y;
a2:=(a1)or(x); y:=(a2)and(a4);
If y=true then y1:=1 else y1:=0;
writeln(s[i],' ',a1,' ',a2,' ',a3,' ',y1);
end; readkey;
END.
```

10. ДОПОЛНИТЕЛЬНЫЕ ЗАДАЧИ

10.1. В сообщении буквы A, B, C, D встречаются с вероятностями $p_A = 0,45$, $p_B = 0,22$, $p_C = 0,14$, $p_D = 0,19$. Используются два кода: 1) A - 00, B - 01, C - 10, D - 11; 2) A - 0, B - 10, C - 110, D - 111. Определите избыточность этих кодов.

10.2. Буквы некоторого алфавита A используются с вероятностями $1/6, 1/6, ...1/6$. Постройте код Шеннона–Фано, вычислите его среднюю длину L, эффективность η и относительную избыточность Q.

10.3. Буквы некоторого алфавита A используются с вероятностями $1/2; 1/4; 1/8; 1/16; 1/32; 1/32$. Постройте код Шеннона–Фано, определите его среднюю длину L и относительную избыточность Q.

10.4. Буквы некоторого алфавита A используются с вероятностями $7/18; 1/6; 1/6; 1/6; 1/9$. Постройте код Шеннона–Фано и код Хаффмена. Определите среднюю длину L и относительную избыточность Q для того и другого кода.

10.5. Имеются весы с двумя чашечками и 27 одинаковых монет, одна из них фальшивая (легче других). Сколько взвешиваний достаточно произвести, чтобы найти фальшивую монету?

Рассмотрим ситуацию с 3 монетами. Первую монету положим на левую чашечку, вторую — на правую. Если весы в равновесии, — фальшивой является третья монета. Если нет, — фальшивая та, что легче. Априорная энтропия $H_0 = \log_2 3 \approx 1,58$ *бит, апостероидная (после одного взвешивания) —* $H_1 = \log_2 1 = 0$, *неопределенность исчезла полностью. Полученное количество информации:* $I = H_0 - H_1 = \log_2 3 \approx 1,58$ *бит. Это максимальное количество информации которое может быть получено в результате одного взвешивания.*

Разделим все 27 монет на три кучки по 9 монет. На одну чашечку весов положим одну кучку, а на другую — вторую кучка. Если весы в равновесии, то фальшивая монета в третьей кучке, если нет — в той кучке, которая легче. Неопределенность уменьшилась; было: одна из 27, стало: одна из 9. Априорная энтропия $H_0 = \log_2 27 \approx 4,75$ *бит, апостероидная (после одного взвешивания) —* $H_1 = \log_2 9 = 3,17$. *Полученное количество информации:* $I_1 = H_0 - H_1 = \log_2 27 - \log_2 9 = \log_2(27/9) \approx 1,58$ *бит. Количество требуемых взвешиваний:* $n = \log_2 27/\log_2 3 \approx 4,75/1,58 = 3$, $N = 3$. *Если бы монет было не 27, а 30, то трех взвешиваний было бы недостаточно:* $n = \log_2 30/\log_2 3 \approx 4,91/1,58 = 3,11$. *Округляя с избытком получаем:* $N = 4$.

10.6. Напишите программу, моделирующую решение предыдущей задачи. Она должна: 1) спрашивать общее количество монет, включая фальшивую, 2) нумеровать их случайным образом; 3) спрашивать, какие монеты вы кладете на левую чашечку, какие на правую; 4) сообщать результат взвешивания; 5) моделировать второе, третье и последующие взвешивания. Человек, решающий задачу, должен определить номер фальшивой монеты и ввести его в компьютер. Программа сообщает, какая монета фальшивая на самом деле.

10.7. Имеется совокупность 30 объектов, каждый из которых характеризуется двумя величинами. Напишите программу, которая осуществляла бы кластеризацию (автоматическую классификацию) объектов, разделяя их на классы.

Кластеризация иерархического типа состоит в последовательном объединении групп объектов, начиная с самых близких и похожих друг на друга. Рассмотрим множество из n объектов $O_1(x_1, y_1)$, $O_2(x_2, y_2)$, ..., $O_n(x_n, y_n)$, каждый из которых характеризуется двумя признаками X и Y. В пространстве признаков XOY каждому объекту соответствует точка. В качестве меры близости двух объектов O_i и O_j обычно выбирают геометрическое расстояние между этими точками:

$$L_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

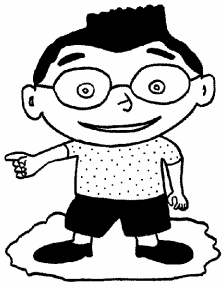
На начальном этапе считают, что каждый объект образует кластер с массой 1. Для каждой пары точек определяют меру близости L_{ik} и, сравнивая их друг с другом, находят наиболее близко расположенные объекты, которые объединяются в один кластер с массой 2. Координаты нового кластера вычисляются как средние взвешенные координат объединенных кластеров. Затем процесс повторяется. При слиянии двух кластеров O_i и O_j (i меньше j) с весовыми коэффициентами p_i и p_j получается кластер O'_i, имеющий следующие координаты и весовой коэффициент соответственно:

$$x'_i = \frac{p_i x_i + p_j x_j}{p_i + p_j}, \qquad y'_i = \frac{p_i y_i + p_j y_j}{p_i + p_j}, \qquad p'_i = p_i + p_j.$$

В результате s шагов, на каждом из которых два кластера объединяются в один, получается разбиение на $n - s$ кластеров. Результат использования программы ПР–10.1 для кластеризации 30 двумерных объектов представлен на рис. 10.1. Если отключить графический режим, закомментировать соответствующие операторы и раскомментировать другие, то на экран будет выведен список объектов с указанием класса, к которому он отнесен.

```
uses  crt, dos, graph;                          { ПР - 10.1 }
const Max=30; Ch_kl=3; U=15;
Data1: array[1..Max]of real=(4,3,5,6,5,7,8,4,10,13,
9,11,-2,-6,-7,-9,-8,-11,-7,-12,4,7,3,5,6,8,5,7,10,13);
Data2: array[1..Max]of real=(-2,-8,-3,-9,-7,-5,-12,-11,
 -6,-10,-8,-7,9,7,4,6,8,10,6,4,8,7,10,11,6,7,4,11,6,5);
Var x,y,p : array[1..Max] of real;
    klass : array[1..Max] of integer;
    L: array[1..Max,1..Ch_kl] of real;
    i,j,k,m,N,t,Gd,Gm: integer; a,L1: real;
BEGIN
Gd:=Detect; InitGraph(Gd, Gm, 'c:\bp\bgi'); N:=Max;
For i:=1 to N do begin
    klass[i]:=i; p[i]:=1;
    x[i]:=Data1[i]; y[i]:=Data2[i]; end;
Repeat a:=10000;
  For i:=1 to N do For j:=i+1 to N do begin
        L1:=sqr(x[i]-x[j])+sqr(y[i]-y[j]);
        If L1<a then begin a:=L1; k:=i; m:=j; end;
      end; dec(N);
{ writeLn('Объекты ',k,' и ',m,' объединяются
                                в новый кластер ',k);}
{ line(320+round(x[k]*U),240-round(y[k]*U),
  320+round(x[m]*U),240-round(y[m]*U));}
  y[k]:=(y[k]*p[k]+y[m]*p[m])/(p[k]+p[m]);
  x[k]:=(x[k]*p[k]+x[m]*p[m])/(p[k]+p[m]);
  p[k]:=p[k]+p[m];
{ writeLn('Его координаты ', x[k]:4:2,
                          y[k]:8:2,' вес ',p[k]:2:0);}
  For i:=1 to N do If i>=m then begin
      x[i]:=x[i+1]; y[i]:=y[i+1];
      p[i]:=p[i+1]; klass[i]:=klass[i+1]; end;
until N=Ch_kl;
For i:=1 to Max do begin a:=1000;
  For j:=1 to Ch_kl do begin
  L[i,j]:=sqr(Data1[i]-x[j])+sqr(Data2[i]-y[j]);
  If L[i,j]<a then begin a:=L[i,j]; klass[i]:=j;
  end;
end; end;
For i:=1 to Max do write(' | ob[',i,']=',klass[i]);
line(0,240,640,240); line(320,0,320,480);
For i:=1 to Max do circle(320+round(Data1[i]*U),
```

```
                          240-round(Data2[i]*U),2);
For i:=1 to N do circle(320+round(x[i]*U),
                          240-round(y[i]*U),3);
For i:=1 to Max do
 For j:=1 to Ch_kl do If klass[i]=j then
 line(320+round(x[j]*U),240-round(y[j]*U),
 320+round(Data1[i]*U),240-round(Data2[i]*U));
Repeat until keypressed; CloseGraph;
END.
```



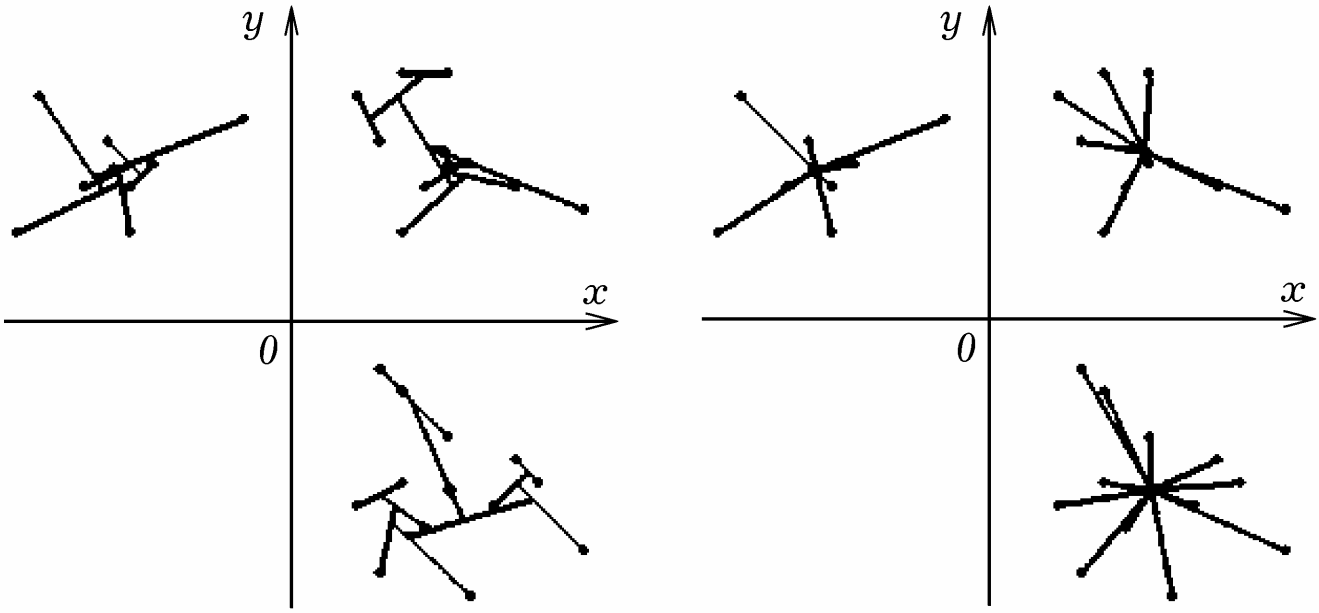


Рис. 10.1. Результат классификации объектов на 3 кластера.

10.8. Напряжение на входе АЦП изменяется по закону $u = Ae^{-bt}$. Число уровней напряжения равно 512. Сохраните в массиве $u[i]$ результат оцифровки. Методом покоординатного спуска восстановите значения A и b. Постройте график исходного сигнала, точки после оцифровки, график восстановленного сигнала.

```
uses crt, graph;                                  { ПР - 10.2 }
var i: integer; A, b, nev, nevyazka : real;
u:array[1..100]of integer;
BEGIN clrscr;
 for i:=1 to 100 do u[i]:=round(512*exp(-0.05*i));
 for i:=1 to 100 do write(u[i],' '); A:=100; b:=0.001;
 Repeat nevyazka:=0;
 for i:=1 to 100 do
          nevyazka:=nevyazka+abs(u[i]-A*exp(-b*i));
 A:=A+0.5; nev:=0;
 for i:=1 to 100 do nev:=nev+abs(u[i]-A*exp(-b*i));
```

```
 if nev>nevyazka then A:=A-0.5; nevyazka:=0;
 for i:=1 to 100 do
     nevyazka:=nevyazka+abs(u[i]-A*exp(-b*i));
 b:=b+0.00001; nev:=0;
 for i:=1 to 100 do
                nev:=nev+abs(u[i]-A*exp(-b*i));
 if nev>nevyazka then b:=b-0.00001;
 writeln(A,' ', b,' ',nevyazka);
 until nevyazka<0.1;
 Repeat until KeyPressed;
END.
```

10.9. Убедитесь в справедливости теоремы Котельникова, для этого оцифруйте входной гармонический сигнал частотой f и восстановите его, соединяя точки отрезками. Что получается, когда частота отсчетов больше $2f$? меньше $2f$?

10.11. Вычислите объем несжатого видеофайла длительностью 3 минуты, если разрешение экрана 640×480, частота кадров 30 Гц, используется 256 цветов.

10.12. Создайте архиватор, который архивирует строку 111 ... 1000 ... 0111 ... 1 по принципу "значение бита — число повторов" (например, так: 1-4-0-4-1-4-0-3 ...) и затем восстанавливает ее.

ЛИТЕРАТУРА

1. Аветисян, Р.Д., Аветисян Д.О. Теоретические основы информатики [Текст] / Р. Д. Аветисян, Д. О. Аветисян. — М.: РГНУ, 1997 — 167 с.
2. Акулов, О. А. Информатика: базовый курс: Учебник для студентов вузов, бакалавров, магистров, обучающихся по направлениям 552800, 65460 "Информатика и вычислительная техника" [Текст] / О. А. Акулов, Н. В. Медведев. — М.: Омега–Л, 2004. — 552 с.
3. Глушков, В. М. Введение в кибернетику [Текст] / В. М. Глушков. — Киев: Изд–во Академии наук Украинской ССР, 1964. — 324 с.
4. Гуц, А.К. Математическая логика и теория алгоритмов: Учебное пособие [Текст] / А. К. Гуц. — Омск: Изд–во Наследие. Диалог – Сибирь, 2004. — 108 с.
5. Душин, В. К. Теоретические основы информационных процессов и систем: Учебник [Текст] / В. К. Душин. — Издательско–торговая корпорация "Дашков и К^{o}", 2003. — 348 с.
6. Лидовский, В. В. Теория информации: Учебное пособие [Текст] / В. В. Лидовский. — М.: Компания Спутник+, 2004. — 111 с.
7. Майер Р.В. Как стать компьютерным гением или книга о информационных системах и технологиях [Текст] / Р. В. Майер. — Глазов, 2008. — 204 с.
8. Майер Р.В. Задачи, алгоритмы, программы [Электронный ресурс]/ Р. В. Майер. — Глазов, 2010. (http://maier-rv.glazov.net или http://komp-model.narod.ru)
9. Могилев, А. В. Информатика: Учебн. пособие для студ. пед. вузов. [Текст] / А. В. Могилев, Н. И. Пак, Е. К. Хеннер. — М.: Издательский центр "Академия", 2003. — 816 с.
10. Муттер, В.М. Основы помехоустойчивой телепередачи информации. [Текст] / В. М. Муттер. — Л.: Энергоиздат. Ленингр. отд–ние, 1990. — 288 с.
11. Петцольд, Ч. Код [Текст] / Ч. Петцольд. — М.: Издательско–торговый дом "Русская редакция", 2001. — 512 с.
12. Рыжиков, Ю. В. Информатика: лекции и практикум [Текст] / Ю. В. Рыжиков. — СПб.: КОРОНАпринт, 2000. — 256 с.
13. Стариченко, Б. Е. Теоретические основы информатики: Учебное пособие для вузов [Текст] / Б. Е. Стариченко. — М.: Горячая линия – Телеком, 2003. — 312 с.
14. Стратанович, Р.Л. Теория информации [Текст] / Р. Л. Стратанович. — М.: Сов. радио, 1975. — 424 с.
15. Успенский, В. А. Теория алгоритмов: основные открытия и приложения [Текст] / В. А. Успенский, А. Л. Семенов. — М.: Наука. Гл. ред. физ.–мат. лит., 1987. — 288 с.
16. Чисар, И. Теория информации: Теоремы кодирования для дискретных систем без памяти [Текст] / И. Чисар, Я. Кернер. — М.: Мир, 1985. — 395 с.
17. Энциклопедия кибернетики. В 2–х томах. [Текст] // Отв. ред. В. М. Глушков. — Киев: Гл. ред. Украинской Советской энциклопедии, 1974. — Т. 1. – 607 с. Т. 2. – 620 с.

СОДЕРЖАНИЕ

Учебное издание

Майер Роберт Валерьевич

Теоретические основы информатики.
Задачи и программы на языке Pascal.

Отпечатано с оригинал–макета автора
в авторской редакции.

Подписано в печать 22.03.11. Напечатано на ризографе.
Формат 60 x 90 1/16. Усл. печ.л. 4,19. Учетн.–изд.л. 4,7.

ГОУ ВПО "Глазовский государственный педагогический институт
им. В. Г. Короленко".
427621, Удмуртия, г. Глазов, ул. Первомайская, 25.
