Федеральное агентство по образованию
Томский государственный университет систем
управления и радиоэлектроники
Кафедра высшей математики (ВМ)

Приходовский М.А.

ПРОГРАММИРОВАНИЕ АЛГЕБРАИЧЕСКИХ ЗАДАЧ НА ЯЗЫКЕ ПАСКАЛЬ С ПОМОЩЬЮ КОМБИНИРОВАННЫХ ТИПОВ И ДИНАМИЧЕСКИХ ПЕРЕМЕННЫХ

Методическое пособие

2006

В данном пособии рассматриваются методы программирования ряда математических задач на Паскале с применением комбинированных типов переменных и с динамическими переменными. Изложение дополнено примерами программ по соответствующим темам.
Может применяться при изучении информатики и высшей математики на первом и втором курсах.

Содержание.

Введение.

При программировании многих математических задач на языке Паскаль, рациональность алгоритмов может достигаться с помощью применения комбинированных типов переменных и динамических переменных.

Задавая массив обычным образом, в виде a:array[1..n] of real , константу n нужно было бы задать до начала работы программой, и мы заранее ограничили бы возможность решения задач большего, чем n, порядка. В то же время, увеличивая порядок массива, заданный в заголовке программы, придётся резервировать чрезмерно много машинной памяти. Использование массивов, состоящих из динамических переменных, устраняет эти проблемы. Достигается решение систем уравнений с произвольно большим количеством уравнений и переменных. При этом максимальный размер ограничен лишь возможностями используемого компьютера и может быть задан пользователем уже во время работы с программой.

В данном пособии применение динамических переменных будет рассмотрено на примере некоторых основных алгоритмов линейной алгебры: операции над векторами, операции над матрицами произвольных размеров; вычисление определителей матриц; решение систем линейных алгебраических уравнений произвольного порядка с невырожденной квадратной матрицей; нахождение обратной матрицы произвольного порядка.

§1. Комбинированный тип переменных.

Кроме четырёх основных типов переменных: integer, real, char, boolean, в Паскале существует «комбинированный» тип переменных, отличающийся тем, что переменная составлена из нескольких полей (независимых ячеек)

машинной памяти. При этом каждое поле содержит переменную одного из известных типов.

Пример.

```
type vector=record x:real; y:real; z:real; end;
var a,b,c:vector;
begin writeln(' Введите координаты 1 и 2 вектора ');
read(a.x, a.y, a.z); read(b.x, b.y, b.z);
end.
```

Без применения комбинированных типов было бы необходимо вводить следующий двумерный массив:

```
var a:array[1..2,1..3]of real;
begin writeln(' Введите координаты 1 и 2 вектора ');
for i:=1 to 2 do for j:=1 to 3 do read(a[i,j]);
end.
```

Процедура присвоения может быть выполнена целиком для всей переменной комбинированного типа, например, *b:=a* означает, что автоматически будут выполнены операции:

b.x:=a.x; b.y:=a.y; b.z:=a.z;

Задавая векторы обычным образом (с помощью массивов), необходимо было бы присваивать значения каждой координаты с помощью цикла.

Также комбинированный тип удобен, если нужно задать массив только для каких-либо выборочных данных. В задачах со многими параметрами часто встречаются системы, где основная матрица содержит много нулей. При задании таких «разрежённых» матриц нерационально хранить информацию обо всех элементах матрицы, достаточно только записать информацию о ненулевых элементах. Один из вариантов решения этой

машинной памяти. При этом каждое поле содержит переменную одного из известных типов.

Пример.

```
type vector=record x:real; y:real; z:real; end;
var a,b,c:vector;
begin writeln(' Введите координаты 1 и 2 вектора ');
read(a.x, a.y, a.z); read(b.x, b.y, b.z);
end.
```

Без применения комбинированных типов было бы необходимо вводить следующий двумерный массив:

```
var a:array[1..2,1..3]of real;
begin writeln(' Введите координаты 1 и 2 вектора ');
for i:=1 to 2 do for j:=1 to 3 do read(a[i,j]);
end.
```

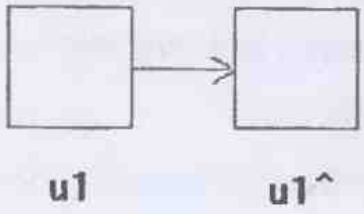
Процедура присвоения может быть выполнена целиком для всей переменной комбинированного типа, например, *b:=a* означает, что автоматически будут выполнены операции:

```
b.x:=a.x; b.y:=a.y; b.z:=a.z;
```

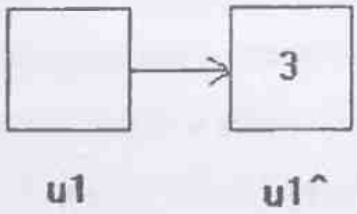
Задавая векторы обычным образом (с помощью массивов), необходимо было бы присваивать значения каждой координаты с помощью цикла.

Также комбинированный тип удобен, если нужно задать массив только для каких-либо выборочных данных. В задачах со многими параметрами часто встречаются системы, где основная матрица содержит много нулей. При задании таких «разрежённых» матриц нерационально хранить информацию обо всех элементах матрицы, достаточно только записать информацию о ненулевых элементах. Один из вариантов решения этой

присваиваст этой переменной значение 3. Действие перечисленных операторов можно представить с помощью схемы:



new(u1);



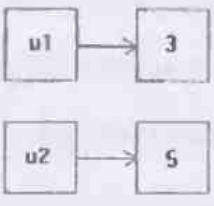
u1^:=3;

Далее создаётся переменная u2^ с помощью оператора new(u2).

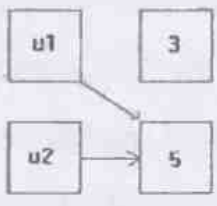
При работе с динамическими переменными в разделе описания переменных var не нужно указывать переменные целого типа, с которыми будет работать программа, потому что они порождаются с помощью оператора new непосредственно в процессе работы программы.

Действие оператора присвоения для переменных ссылочного типа.

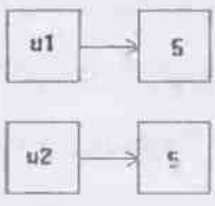
Оператор u1:=u2 означает, что ссылка u1 станет указывать на ту же самую переменную типа integer, на которую указывает u2. При этом, если до выполнения этого оператора ссылка u1 указывала на какое-либо значение, то информация об том значении станет недоступна. Оператор u1^:=u2^ означает, что переменная, на которую указывает u1, примет значение той переменной, на которую указываст u2.



до присвоения



если u1:=u2



если u1^:=u2^

Приведённая выше схема поясняет действие этих операторов.

Таким образом, есть два способа сделать так, чтобы переменная u1 ссылочного типа указывала на какой-нибудь объект (или переменную). Нужно либо создать этот объект с помощью оператора new(u1), либо присвоить u1:=u2 (при этом предполагается, что объект, на который указывает u2, существует, то есть был создан раньше).

Но переменная ссылочного типа может указывать также и на переменную комбинированного типа. В этом случае u1^ - переменная комбинированного типа, состоящая из нескольких полей. Это значит, что оператор u1^:=3 не имеет смысла, так как числовые значения может принимать каждое поле памяти отдельно, например, u1^.x , u1^.y , u1^.z . Необходимо после символов u1^ добавлять, в какую именно ячейку памяти записать то или иное значение, например u1^.x:=3 .

```
program a2; uses crt;
type uk=^vect; vect= record x,y,z:real; end;
var u1,u2:uk;
begin clrscr;
new(u1); u1^.x:=3; u1^.y:=5; u1^.z:=2;
end.
```

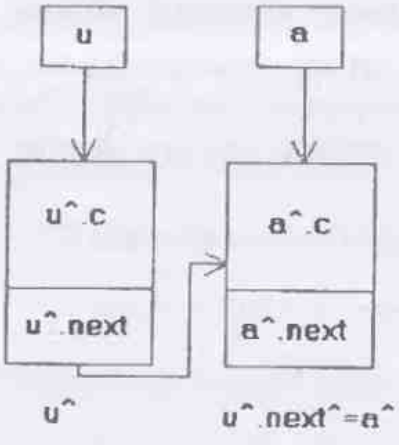
После действия этой программы ссылка u1 указывает на вектор (3,5,2).
Для задания пустой ссылки применяется оператор u:=nil. Для удаления переменной, на которую указывает ссылка, - оператор dispose(u).

§3. Однонаправленный и двунаправленный список

Одно из полей в переменной комбинированного типа может являться переменной ссылочного типа, причём, если ссылка будет на тот же

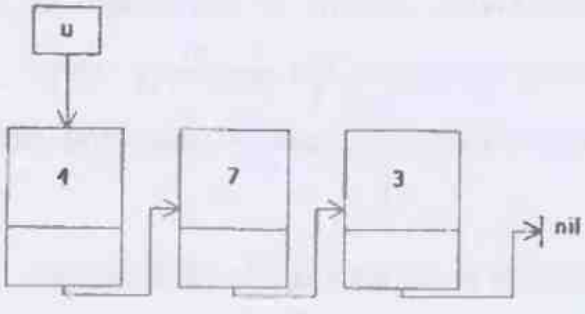
комбинированный тип, появляется возможность объединять переменные в цепочки, таблицы или сложные структуры. Например, так можно задать построение вектора из 2 координат:

```
program a2; uses crt;
type uk=^element; element=record c:integer; next:uk; end;
var u1,u2:uk;
begin  new(u1); u1^.c:=3; new(u1^.next);
u2:=u1^.next; u2^.c:=4; u2^.next:=nil;
end.
```



На этой схеме показана структура из двух указателей и двух переменных комбинированного типа, объединённых в последовательность.

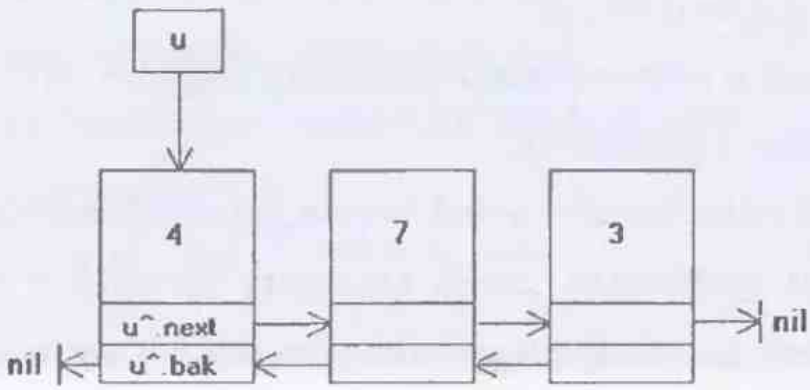
Структура динамических переменных для вектора из трёх координат:



Если задать тип динамических переменных следующим образом:

Type uk=^vect; vect=record c:real; next:uk; **bak:uk;** end;

то получим «двунаправленный» список, структуру которого можно представить с помощью схемы:



При этом процедура построения массива из этих динамических переменных задаётся так:

procedure make(n:integer); var j:integer;

begin new(u); ukaz:=u; ***u^.bak:=nil;***

for j:=1 to n-1 do begin new(u^.next); ***u^.next^.bak:=u;*** *u:=u^.next;*

end; u^.next:=nil; end;

Для того, чтобы не задавать движение указателей каждый раз, когда нужно записать, или наоборот, узнать значение какой-либо координаты, введём процедуру записи значения на место координаты i.

procedure w(i1:integer;x1:real); var j:integer;

begin if(i1>1) then for j:=1 to i1-1 do u:=u^.next;

u^.c:=x1; end;

До выполнения этой процедуры вспомогательный указатель u установим туда же, куда указывает основной указатель вектора a. Если нам нужно записать значение первой координаты, то цикл не выполняется и значение будет записано туда, куда указывает u, а если координата с номером i, то производится сдвиг указателя u на i-1 позиций с помощью цикла

for j:=1 to i1-1 do u:=u^.next;

Теперь, если в теле программы написать оператор w(2, 10) , то вторая координата примет значение 10.

Процедура make создаёт пустой массив, затем с помощью процедуры w (от слова write) происходит запись вводимых значений в ячейки памяти. Приведём пример программы, позволяющей строить 2 вектора с произвольно большим количеством координат и находить их скалярное произведение:

```
program a2; uses crt;
type z1=^z; z=record c:real; next:z1; end;
var n,i:integer; u,ukaz,a,b:z1; s1,s2,s,x:real;
procedure w(i1:integer;x1:real); var j:integer;
begin if(i1>1) then for j:=1 to i1-1 do u:=u^.next;
u^.c:=x1;  end;
function r(i1:integer):real; var j:integer;
begin if(i1>1) then for j:=1 to i1-1 do u:=u^.next;
r:=u^.c;  end;
procedure make(n:integer); var j:integer;
begin new(u); ukaz:=u;
for j:=1 to n-1 do begin new(u^.next); u:=u^.next;
end; u^.next:=nil;
end;
```

```
begin clrscr;
writeln('Vvedite n'); read(n);
make(n); a:=ukaz; make(n); b:=ukaz;
writeln('Vvedite 2 vectors');
for i:=1 to n do begin read(x); u:=a; w(i,x); end;
for i:=1 to n do begin read(x); u:=b; w(i,x); end;
s:=0; for i:=1 to n do begin u:=a; s1:=r(i); u:=b; s2:=r(i);
s:=s+s1*s2; end; writeln;
writeln('Scalarn proizv ravno ', s:6:0);  readln;
end.
```

Таким же образом, как векторы, можно задавать многочлены произвольной степени. Среди основных функций языка Паскаль не предусмотрена функция возведения в произвольную степень, а для вычисления значений многочлена необходима такая операция, поэтому введём функцию вычисления целой степени любого действительного числа.

```
function st(a:real;b:integer):real;
var i:integer; q:real;
begin q:=1; for i:=1 to b do q:=q*a;
st:=q;end;
```

Для задания массива из динамических переменных, содержащих информацию о коэффициентах многочлена, можно применять следующий тип:

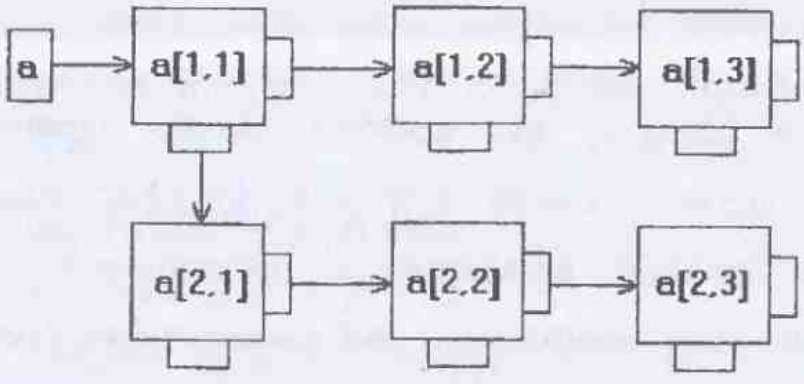
```
type s=^mn;
mn=record c:real; back:s; next:s; deg:integer; end;
```

§4 действия над матрицами

Определим новый тип переменных и построим «пустой» массив из динамических переменных, который будет содержать систему ссылок и указателей каждого элемента на следующий в строке и следующий в столбце.

```
type uk=^element;
element=record c:real; n:uk; pr:uk; end;
var uk1,uk2:uk;
```

Схема строения динамических переменных:



Можно также создавать систему ссылок на все четыре соседних элемента: верхний, нижний, левый и правый. При этом указатели смогут перемещаться в любом направлений на необходимое количество шагов.

```
type uk=^element;
element=record c:real; n:uk; v:uk; l:uk; pr:uk; end;
var uk1,uk2:uk;
```

Таким образом, переменная типа element – это переменная, представляющая собой блок, состоящий из 5 частей – числового значения и 4 ссылок на соседние элементы. Uk1 , uk2 – указатели на переменные типа element, также необходим некоторый «главный» указатель, который будет

всегда указывать на элемент a_{11} и с помощью которого функции записи и чтения значений всегда смогут выйти на любой элемент a_{ij}.

Теперь напишем процедуру построения «пустой» матрицы размера $m \times n$. Эту процедуру назовём make, в теле программы достаточно будет написать только одно обращение к make после основного begin, чтобы построить массив со всеми ссылками элементов друг на друга, куда в дальнейшем будем записывать значения элементов.

```
procedure make(m,n:integer);var i1,i2:integer;
begin new(glavn);
uk:=glavn;
uk^.v:=nil; uk^.l:=nil, (ставятся пустые ссылки на верхний и левый элемент)
for i1:=1 to n-1 do begin  new(uk^.p); uk^.p^.l:=uk; uk:=uk^.p;
uk^.v:=nil; end;
uk^.p:=nil;
for i1:=1 to n-1 do uk:=uk^.l;
 for i2:=1 to m-1 do begin uk1:=uk; new(uk^.n); uk:=uk^.n;
uk1^.n:=uk; uk^.v:=uk1;
 for i1:=1 to n-1 do begin
new(uk^.p); uk^.p^.l:=uk; uk:=uk^.p;
  uk1:=uk1^.p; uk1^.n:=uk; uk^.v:=uk1; end;
uk^.p:=nil;
 for i1:=1 to n-1 do begin uk:=uk^.l; uk1:=uk1^.l; end;
end.
 for i1:=1 to n-1 do begin uk^.n:=nil; uk:=uk^.p; end;
uk^.n:=nil;
```

```
for i1:=1 to n-1 do uk:=uk^.l; for i1:=1 to m-1 do uk:=uk^.v; end;
```

Процедура записи значения элемента в динамический массив. Так же, как и ранее для векторов, создаём процедуру записи значения, чтобы в дальнейшем не программировать движение указателей по элементам массива, а просто указывать индексы элемента и вводимое значение. Тогда программирование самого алгоритма решения математической задачи будет значительно компактнее и нагляднее.

```
procedure w(a1:uk; i1,j1:integer;x1:real); var i,j:integer;
begin u:=a1;
if(i1>1) then for i:=1 to i1-1 do u:=u^.n;
if(j1>1) then for i:=1 to j1-1 do u:=u^.pr;
u^.c:=x1;  end;
```

Теперь при вводе элементов можно с помощью одного оператора w(a,i,j,c); сразу записывать их в нужную ячейку динамического массива:

```
for l:=1 to n do begin  for j:=1 to m do begin
 read(c); w(a,i,j,c);  end; end;
```

(При этом элемент с индексами i и j должен существовать в данном динамическом массиве, иначе программа сообщит о некорректной операции и прекратит работу). Также создадим функцию, которая будет выдавать значения по указанным индексам элемента в массиве (матрице).

```
function r(a1:uk; i1,j1:integer):real; var i,j:integer;
begin u:=a1;
if(i1>1) then for i:=1 to i1-1 do u:=u^.n;
if(j1>1) then for j:=1 to j1-1 do u:=u^.pr;
r:=u^.c;  end;
```

Умножение матриц произвольных размеров

```
program a2; uses crt;
type uk=^element; element=record c:real; pr:uk;n:uk; end;
var m1,n1,m2,n2,i,j:integer; u,ukaz,a,b,c:uk; s1,s2,s,x:real;
procedure w(a1:uk; i1,j1:integer;x1:real); var i,j:integer;
begin u:=a1;
if(i1>1) then for i:=1 to i1-1 do u:=u^.n;
if(j1>1) then for i:=1 to j1-1 do u:=u^.pr;
u^.c:=x1;  end;
function r(a1:uk; i1,j1:integer):real; var i,j:integer;
begin u:=a1;
if(i1>1) then for i:=1 to i1-1 do u:=u^.n;
if(j1>1) then for j:=1 to j1-1 do u:=u^.pr;
r:=u^.c;  end;
procedure make(m,n:integer); var i,j:integer; u1,u2:uk;
begin new(u1); ukaz:=u1; u2:=u1;
for j:=1 to n-1 do begin new(u1^.pr); u1:=u1^.pr; end; u1^.pr:=nil;
for i:=1 to m-1 do begin new(u1); u2^.n:=u1; u2:=u2^.n;
for j:=1 to n-1 do begin new(u1^.pr); u1:=u1^.pr; end; u1^.pr:=nil;
end; u2^.n:=nil;end;
procedure mult; var k:integer; x,x1,x2:real;
begin make(m1,n2);
for i:=1 to m1 do begin  for j:=1 to n2 do begin
 x:=0; for k:=1 to n1 do begin x1:=r(a,i,k); x2:=r(b,k,j);
 x:=x+x1*x2; end; w(c,i,j,x);
```

```
end; end;
 writeln('Proizvedenie:');
 for i:=1 to m1 do begin for j:=1 to n2 do begin
 write(r(c,i,j):4:0); end; writeln; end;
end;
begin clrscr;
writeln('Vvedite m1,n1'); read(m1,n1);make(m1,n1); a:=ukaz;
writeln('Vvedite m2,n2'); read(m2,n2);make(m2,n2); b:=ukaz;
writeln('Vvedite 1 matrix');
 for i:=1 to m1 do begin for j:=1 to n1 do begin
 read(x); w(a,i,j,x); end; end;
writeln('Vvedite 2 matrix');
 for i:=1 to m2 do begin for j:=1 to n2 do begin
 read(x); w(b,i,j,x); end; end;
(* for i:=1 to m1 do begin for j:=1 to n1 do begin
 u:=a; write(r(i,j):4:0); end; writeln; end;
 for i:=1 to m2 do begin for j:=1 to n2 do begin
 u:=b; write(r(i,j):4:0); end; writeln; end;*)
 if(m2=n1) then mult else writeln('Razmer ne soglasovan');
 readln; end
```

§5. Различные алгебраические задачи.

В этом параграфе рассматриваются следующие задачи: нахождение значения определителя, решение систем линейных алгебраических уравнений (метод Гаусса), нахождение обратной матрицы. Приведены примеры программ.

Нахождение определителя порядка n. Наиболее рациональным решением будет поиск значения определителя с помощью приведения матрицы к треугольному виду методом Гаусса.

Описание процедур и функций, алгоритм программы.

В начале программы идёт обращение к процедуре *make*, которая строит массив из динамических переменных, затем вводятся n^2 элементов. Далее программа обращается к процедуре *gauss*, которая в свою очередь вызывает процедуру *privod* (приведение всех строк, расположенных ниже чем i, то есть получение нулей ниже элемента a_{ii}). Но нужно учесть, что в процессе выполнения алгоритма, на диагонали может оказаться нулевой элемент, и нужно будет поменять местами строки, чтобы получить $a_{ii} \neq 0$. А для наименьшей вычислительной погрешности лучше, чтобы на диагонали оказывался максимальный из элементов $a_{ii},\ldots,a_{ni}$. Для этого можно менять местами строки, учитывая, конечно, количество таких замен, чтобы не изменился знак определителя. Такая модификация алгоритма Гаусса достигается с помощью процедур *big* и *change*, первая из которых ищет номер строки, где расположен наибольший из элементов, находящихся ниже a_{ii}, а вторая – меняет местами строки. Затем следует выполнение процедуры *minus* (эта процедура к строке номер j прибавляет строку номер i, умноженную на коэффициент c1). Естественно, если необходимо вычесть одну строку из другой, просто полагаем c1= -1.

В процессе всех преобразований работают процедура *procedure w(a1:uk; i1,j1:integer;x1:real)* и функция *function r(a1:uk; i1,j1:integer):real*. Первая из них выполняет запись значения элемента x1 в строку i1, стобец j1 матрицы,

на которую ссылается указатель a1. Вторая – считывание значения элемента на пересечении строки i1 и столбца j1.

```
program a4; uses crt;
type uk=^element; element=record c:real; pr:uk;n:uk; end;
var sc,maxn,n,i,j,ji:integer; u,ukaz,a,b,c:uk; maxc,s1,s2,s,x,det:real;
function st(q1:real;q2:integer):real;
var i1:integer;q3:real;
begin q3:=1;
for i1:=1 to q2 do q3:=q3*q1; st:=q3; end;
procedure w(a1:uk; i1,j1:integer;x1:real); var i,j:integer;
begin u:=a1;
if(i1>1) then for i:=1 to i1-1 do u:=u^.n;
if(j1>1) then for i:=1 to j1-1 do u:=u^.pr;
u^.c:=x1;  end;
function r(a1:uk; i1,j1:integer):real; var i,j:integer;
begin u:=a1;
if(i1>1) then for i:=1 to i1-1 do u:=u^.n;
if(j1>1) then for j:=1 to j1-1 do u:=u^.pr;
r:=u^.c;  end;
procedure make(m,n:integer); var i,j:integer; u1,u2:uk;
begin new(u1); ukaz:=u1; u2:=u1;
for j:=1 to n-1 do begin new(u1^.pr); u1:=u1^.pr; end; u1^.pr:=nil;
for i:=1 to m-1 do begin new(u1); u2^.n:=u1; u2:=u2^.n;
for j:=1 to n-1 do begin new(u1^.pr); u1:=u1^.pr; end; u1^.pr:=nil;
end; u2^.n:=nil; end;
procedure change(j1,i1:integer); var i2:integer; x:real;
```

```
begin for i2:=1 to n+1 do begin
x:=r(a,i1,i2); w(a,i1,i2,r(a,j1,i2)); w(a,j1,i2,x); end; end;
procedure minus(j1,i1:integer; c1:real); var i2:integer; x:real;
begin for i2:=1 to n+1 do begin
x:=r(a,i1,i2); w(a,j1,i2,r(a,j1,i2)-c1*x); end; end;
procedure big(i1:integer); var n1,i2:integer; c1:real;
begin n1:=i1; c1:=abs(r(a,i1,i1));
for i2:=i1+1 to n do begin
if(abs(r(a,i2,i1))>c1)then begin c1:=abs(r(a,i2,i1)); n1:=i2; end; end;
maxn:=n1; maxc:=c1; end;
procedure privod(i1:integer); var i2:integer; c1:real;  begin
for i2:=i1+1 to n do begin c1:=r(a,i2,i1)/r(a,i1,i1); minus(i2,i1,c1); end; end;
procedure gauss; var i2:integer;
begin sc:=0;
for ji:=1 to n-1 do begin big(ji);
if(maxn<>ji) then begin sc:=sc+1; change(ji,maxn); end;
privod(ji); end;
det:=1; for i2:=1 to n do det:=det*r(a,i2,i2); det:=det*st(-1,sc);
writeln('Determinant is  ',det:4:1);
end;
begin clrscr;
writeln('Vvedite n'); read(n);make(n,n); a:=ukaz;
writeln('Vvedite matrix');
for i:=1 to n do begin  for j:=1 to n do begin
 read(x); w(a,i,j,x); end;  end;
gauss; readln; end.
```

Системы линейных алгебраических уравнений. Для наглядности алгоритма рассмотрим системы с невырожденной основной квадратной матрицей.

Для решения этой задачи можно использовать те же процедуры, что и в предыдущей программе, но их необходимо видоизменить, так, например, добавить столбец для задания расширенной матрицы. Кроме того, после приведения основной матрицы к треугольному виду, нужно будет ввести другую процедуру - не перемножать элементы главной диагонали, а находить n неизвестных (процедура *resh*). Для хранения значений найденных неизвестных вводится «n+2»-й столбец матрицы.

```
program a5; uses crt;
type uk=^element; element=record c:real; pr:uk;n:uk; end;
var sc,maxn,n,i,j,ji:integer; u,ukaz,a,b,c:uk; maxc,s1,s2,s,x,det:real;
function st(q1:real;q2:integer):real;
var i1:integer;q3:real;
begin q3:=1;
for i1:=1 to q2 do q3:=q3*q1; st:=q3; end;
procedure w(a1:uk; i1,j1:integer;x1:real); var i,j:integer;
begin u:=a1;
if(i1>1) then for i:=1 to i1-1 do u:=u^.n;
if(j1>1) then for i:=1 to j1-1 do u:=u^.pr;
u^.c:=x1;  end;
function r(a1:uk; i1,j1:integer):real; var i,j:integer;
begin u:=a1;
if(i1>1) then for i:=1 to i1-1 do u:=u^.n;
if(j1>1) then for j:=1 to j1-1 do u:=u^.pr;
```

```
r:=u^.c;  end;
procedure make(m,n:integer); var i,j:integer; u1,u2:uk;
begin new(u1); ukaz:=u1; u2:=u1;
for j:=1 to n-1 do begin new(u1^.pr); u1:=u1^.pr; end; u1^.pr:=nil;
for i:=1 to m-1 do begin new(u1); u2^.n:=u1; u2:=u2^.n;
for j:=1 to n-1 do begin new(u1^.pr); u1:=u1^.pr; end; u1^.pr:=nil;
end; u2^.n:=nil; end;
procedure change(j1,i1:integer); var i2:integer; x:real;
begin for i2:=1 to n+1 do begin
x:=r(a,i1,i2); w(a,i1,i2,r(a,j1,i2)); w(a,j1,i2,x); end; end;
procedure minus(j1,i1:integer; c1:real); var i2:integer; x:real;
begin for i2:=1 to n+1 do begin
x:=r(a,i1,i2); w(a,j1,i2,r(a,j1,i2)-c1*x); end; end;
procedure big(i1:integer); var n1,i2:integer; c1:real;
begin n1:=i1; c1:=abs(r(a,i1,i1));
for i2:=i1+1 to n do begin
if(abs(r(a,i2,i1))>c1)then begin c1:=abs(r(a,i2,i1)); n1:=i2; end; end;
maxn:=n1; maxc:=c1; end;
procedure privod(i1:integer); var i2:integer; c1:real;  begin
for i2:=i1+1 to n do begin c1:=r(a,i2,i1)/r(a,i1,i1); minus(i2,i1,c1); end; end;
procedure resh;  var i1,i2:integer; x1,c1:real;
begin x:=r(a,n,n+1)/r(a,n,n); w(a,n,n+2,x);
for i1:=n-1 downto 1 do begin x1:=0;
for i2:=i1+1 to n do x1:=x1+r(a,i1,i2)*r(a,i2,n+2);
x:=(r(a,i1,n+1)-x1)/r(a,i1,i1); w(a,i1,n+2,x); end;
writeln('Reshenie of sistema: ');
```

```
for i1:=1 to n do writeln('x_',i1,' = ',r(a,i1,n+2):5:2);end;
procedure gauss; var i2:integer;
begin sc:=0; for ji:=1 to n-1 do begin big(ji);
if(maxn<>ji) then begin sc:=sc+1; change(ji,maxn); end;
privod(ji); end;
det:=1; for i2:=1 to n do det:=det*r(a,i2,i2); det:=det*st(-1,sc);
writeln('Determinant of base matrix is ',det:4:1); end;
begin clrscr;
writeln('Vvedite n'); read(n);make(n,n+2); a:=ukaz;
writeln('Vvedite matrix');
for i:=1 to n do begin for j:=1 to n+1 do begin
 read(x); w(a,i,j,x); end; end;
gauss; if(abs(det)<0.00001) then
writeln('Determinant is ',det:5:1,' net reshenij.')
else resh; readln; readln; end.
```

Заметим, что возможно также решение системы методом Крамера, при этом вычисление определителей будет использоваться в программе в качестве процедуры.

Используя процедуры и функции, введённые в предыдущих программах, можно задать алгоритм нахождения обратной матрицы с помощью решения n систем линейных уравнений, рассматривая в качестве правой части столбцы единичной матрицы. Но поскольку при решении системы основная матрица преобразуется, а нам нужно решать n систем, то при вводе запишем элементы в виде матрицы C, а затем будем копировать с помощью процедуры *copy* в матрицу A при решении каждой новой системы уравнений.

```
program a6; uses crt;
type uk=^element; element=record c:real; pr:uk;n:uk; end;
var sc,maxn,n,i,j,ji:integer; u,ukaz,a,b,c:uk; maxc,s1,s2,s,x,det:real;
function st(q1:real;q2:integer):real;
var i1:integer;q3:real;
begin q3:=1;
for i1:=1 to q2 do q3:=q3*q1; st:=q3; end;
procedure w(a1:uk; i1,j1:integer;x1:real); var i,j:integer;
begin u:=a1;
if(i1>1) then for i:=1 to i1-1 do u:=u^.n;
if(j1>1) then for i:=1 to j1-1 do u:=u^.pr;
u^.c:=x1;  end;
function r(a1:uk; i1,j1:integer):real; var i,j:integer;
begin u:=a1;
if(i1>1) then for i:=1 to i1-1 do u:=u^.n;
if(j1>1) then for j:=1 to j1-1 do u:=u^.pr;
r:=u^.c;  end;
procedure make(m,n:integer); var i,j:integer; u1,u2:uk;
begin new(u1); ukaz:=u1; u2:=u1;
for j:=1 to n-1 do begin new(u1^.pr); u1:=u1^.pr; end; u1^.pr:=nil;
for i:=1 to m-1 do begin new(u1); u2^.n:=u1; u2:=u2^.n;
for j:=1 to n-1 do begin new(u1^.pr); u1:=u1^.pr; end; u1^.pr:=nil;
end; u2^.n:=nil; end;
procedure change(j1,i1:integer); var i2:integer; x:real;
begin for i2:=1 to n+1 do begin
x:=r(a,i1,i2); w(a,i1,i2,r(a,j1,i2)); w(a,j1,i2,x); end; end;
```

```
procedure minus(j1,i1:integer; c1:real); var i2:integer; x:real;
begin for i2:=1 to n+1 do begin
x:=r(a,i1,i2); w(a,j1,i2,r(a,j1,i2)-c1*x); end; end;
procedure big(i1:integer); var n1,i2:integer; c1:real;
begin n1:=i1; c1:=abs(r(a,i1,i1));
for i2:=i1+1 to n do begin
if(abs(r(a,i2,i1))>c1)then begin c1:=abs(r(a,i2,i1)); n1:=i2; end; end;
maxn:=n1; maxc:=c1; end;
procedure privod(i1:integer); var i2:integer; c1:real; begin
for i2:=i1+1 to n do begin c1:=r(a,i2,i1)/r(a,i1,i1); minus(i2,i1,c1); end; end;
procedure resh; var i1,i2:integer; x1,c1:real;
begin x:=r(a,n,n+1)/r(a,n,n); w(a,n,n+2,x);
for i1:=n-1 downto 1 do begin x1:=0;
for i2:=i1+1 to n do x1:=x1+r(a,i1,i2)*r(a,i2,n+2);
x:=(r(a,i1,n+1)-x1)/r(a,i1,i1); w(a,i1,n+2,x); end; end;
procedure copy; var i1,j1:integer; begin
for i1:=1 to n do for j1:=1 to n do w(a,i1,j1,r(c,i1,j1)); end;
procedure gauss; var i2:integer;
begin sc:=0;
for ji:=1 to n-1 do begin big(ji);
if(maxn<>ji) then begin sc:=sc+1; change(ji,maxn); end;
privod(ji); end;
det:=1; for i2:=1 to n do det:=det*r(a,i2,i2); det:=det*st(-1,sc);
end;
begin clrscr;
writeln('Vvedite n'); read(n); make(n,n+2); a:=ukaz;
```

```
make(n,n); b:=ukaz; make(n,n); c:=ukaz;
writeln('Vvedite matrix');
for i:=1 to n do begin for j:=1 to n do begin
 read(x); w(c,i,j,x); end; end;
copy; for j:=1 to n do if(j=i) then w(a,j,n+1,1) else w(a,j,n+1,0); gauss;
if(abs(det)<0.00001) then
writeln('Determinant is ',det:4:1,' net obr matr.')
else begin
for i:=1 to n do begin
copy; for j:=1 to n do if(j=i) then w(a,j,n+1,1) else w(a,j,n+1,0); gauss;
resh; for j:=1 to n do w(b,j,i,r(a,j,n+2)); end;
writeln('Obratn matrix: ');
for i:=1 to n do begin
for j:=1 to n do write(r(b,i,j):6:2);
writeln; end; end; readln; end.
```

Литература

1. Абрамов В.Г., Трифонов Н.П., Трифонова Г.Н. Введение в язык Паскаль. Москва, «Наука», 1988.
2. Зубов В.С. Программирование на языке Turbo Pascal. Москва, 1997.
3. Йенсен, Вирт. Паскаль: руководство для пользователя. Москва,1993.
4. Перминов О.Н. Программирование на языке Паскаль. Москва, 1988.

Тираж 100. Заказ № 823
Томский государственный университет
систем управления и радиоэлектроники
634050, г. Томск, пр. Ленина, 40