1. Процедуры и функции

В языке Паскаль предусмотрены средства, позволяющие оформить последовательность операторов как подпрограмму. Это бывает необходимо, когда такая последовательность встречается уже более 2-х раз в программе или когда имеется возможность использовать некоторые фрагменты уже написанных ранее программ.

Для оформления подпрограмм последовательности операторов присваивается имя, и в дальнейшем при необходимости выполнить эту последовательность в программе упоминают имя последовательности. В языке Паскаль такую поименованную последовательность операторов называют процедурой.

Если в результате выполнения поименованной последовательности операторов получается только 1 результат, то имя этой последовательности можно использовать в выражении, и тогда такую последовательность называют функцией. В языке Паскаль имеется целый ряд встроенных в компилятор и распознаваемых без дополнительного описания стандартных функций, например, математические функции sin и cos.

Стандартные процедуры и функции. Для IBM при работе со встроенными процедурами необходимо в самом начале после слова *program* записать служебное слово *uses*, после которого указать имена подключаемых модулей например, *uses crt*. Раздел *uses* описывает список имен, подключаемых стандартных и пользовательских модулей.

crt – содержит средства управления дисплеем и клавиатурой ЭВМ;
printer – открывает доступ к печатающему устройству;
graph3 – позволяет использовать графику в Паскале версии 3.

Стандартные процедуры или функции:

gotoxy (x,y) – перемещает курсор в строку x позиции y (x,y – целые);
clrscr – очистка экрана;
delay (m_s) – задержка программы на m_s миллисекунд (m_s – *integer*);
pred (x) – предыдущее значение x (x – целые);
succ(x) – последующее число после x;
random – генератор случайных чисел (0-1);
random(x) – случайное число (0–x);
keypressed – функция, результат *true* – клавиша нажата, *false* – клавиша не нажата.

Помимо стандартных функций, программист имеет право вводить новые функции. Для этого в программе необходимо задать правило вычисления значения функции, т. е. провести ее описание. Процедуры также должны описываться. Описание процедуры и функции располагается в программе после описания переменных. Сами эти описания не предписывают выполнения действий, как операторы. Чтобы процедура была выполнена, в программе должен быть предусмотрен оператор вызова процедуры. Функция будет выполнена, если ее имя встретится в программе в каком-либо выражении.

Описание процедуры начинается с заголовка. За заголовком следует текст описания, во многом аналогичный тексту программы, т. к. строится по тем же правилам.

procedure имя процедуры (имя аргумента: тип; имя аргумента:тип**;** ***var*** имя результата: тип);

```
var .....
begin
.....
end;.
```

Без служебного слава var перед именем результата значение результата из процедуры не возвращается. Описание процедуры заканчивается служебным словом *end*, за которым следует точка с запятой.

```
procedure swap (var x,y:real);        алг обмен (вещ x,y)
var t:real;                           дано x,y
                                      надо x,y
begin                                 нач вещ t
  t:=x; x:=y; y:=t;                         t:=x; x:=y; y:=t
end;                                  кон
```

Для вызова процедуры в тексте программы указывается ее имя, а в скобках через запятую перечисляются имена переменных или конкретное числовое значение, необходимое для этой процедуры.

Пример.

```
program vv;
var x,y:real;
BEGIN
  x:=5; y:=2.3;
  swap (x,y);
  writeln(x,y)
END.
```

Ответ: x=2.3; y=5.

Процедура может и не содержать параметров. В этом случае заголовок имеет вид:

procedure имя процедуры;.

Оператор вызова такой процедуры имеет вид:

```
begin
  оператор;
  ostatok;
  .....
end.
procedure ostatok
begin
  g:=a div b;
  r:=a mod b;
end;
```

В результате ее выполнения получим значение переменных g и r (частного и остатка от деления целых чисел a и b).

При этом, конечно, переменные a, b, g, r должны быть описаны в программе, содержащей описание этой процедуры, а значения переменных a и b должны быть заданы до выполнения операторов процедуры. Описание функций аналогично описанию процедуры.

Заголовок описания функции имеет вид:

function имя функции (имя аргумента:тип):тип функции;

```
var ...
begin
...
end;.
```

Например:

```
function summa (var x,y:real):real;,
function summa (x,y:real):real;.
```

В этом заголовке тип, стоящий после круглой скобки, определяет тип переменной имени функции. Отличие от процедуры состоит также в том, что в тексте описания функции ***должен обязательно присутствовать оператор присваивания вида: имя функции:=выражение*** (summa:=s)**.**

```
program sum;
BEGIN
  sum:=x+y
END.
```

Пример: Программа вычисления факториала

```
program factorial;
var i,k:integer;
function fact (n:integer):integer;
begin
  if n<=1 then fact:=1
     else fact:=n*fact(n-1);
```

```
end;
BEGIN
  write ('ввод символа i '):readln(i);
  for k:=1 to i do
      writeln(fact(k));
END.
```

Задача 1. Оформить в виде процедуры алгоритм вычисления степени $y = x^n$ с натуральным показателем.

```
program ap;
var n:integer; x,y:real;
procedure s1 (k:integer; e:real; var d:real);
var i:integer;
begin
  d:=1;
  for i:=1 to k do d:=d*e;
end;
BEGIN
  writeln ('введи показатель степени');
  readln (n);
  writeln('введи значение x');
  readln(x);
  s1(n,x,y);
  writeln('y=',y)
END.
```

Задача 2. Оформить алгоритм вычисления степени $y = x^n$ в виде процедуры без параметров.

```
program p2;
var n:integer;
  x,y:real;
procedure s2;
var
  i:integer;
begin
  y:=1;
  for i:=1 to n do y:=y*x;
end;
BEGIN
  writeln ('введите показатель степени');
  readln(n);
  writeln('введите значение x');
  readln(x);
  s2;
  writeln('y=',y)
END.
```

В этом примере x, y – глобальные переменные по отношению к процедуре *s2*, а переменная i там – локальная.

Задача 3. Оформить в виде функции алгоритм вычисления степени $y = x^n$.

```
program p3;
var n:integer;
  x,z:real;
function s3 (n:integer; x:real):real;
var
  i:integer; y:=real;
begin
  y:=1;
  for i:=1 to n do y:=y*x;
```

```
  s3:=y;
end;

function s3:real;
var i:integer;
  y:real;
begin
  y:=1;
  for i:=1 to n do
  y:=y*x;
  s3:=y
end;

BEGIN
  writeln ('введи n');
  readln(n);
  writeln('введи x');
  readln(x);
  z:=s3(n,x);
  z:=s3;
  writeln('y=',z)
END.
```

Задача 3. Определите с помощью функции пользователя периметр восьмиугольника.

```
 program tem;
  var i:integer;
      x,y:array[1..8] of real;      p:real;
function dl(x1,y1,x2,y2:real):real;
   begin
      dl:=sqrt(sqr(x2-x1)+sqr(y2-y1));
   end;

BEGIN
  writeln('Введите координаты 8 точек  ');
  p:=0;
  for i:=1 to 8 do
     begin
       readln(x[i],y[i]);
       if i>1 then  p:=p+dl(x[i-1],y[i-1],x[i],y[i]);
     end;
  p:=p+dl(x[1],y[1],x[8],y[8]);
  writeln;
  writeln('Периметр равен ',p);
END.
```

Главная программа может содержать описание нескольких процедур и функций. Любая из них может быть вызвана в главной программе, или ее вызов может содержаться в теле другой процедуры, описанной в программе. Описание процедуры или функции может быть также приведено в теле другой процедуры или функции, т. е. возможны вложенные процедуры и функции. Главная программа может получать начальные данные, с которыми ей предстоит работать извне, читая их из файла *input*. Процедуры и функции обычно получают такие начальные данные из той программы или процедуры, которая их вызывает. Передача этих данных осуществляется через параметры. В операторе вызова процедуры или при вызове функции указываются фактические параметры, подставляемые вместо формальных параметров и используемые при вычислении. Аналогично результат может быть передан в вызывающую программу через

соответствующий параметр. Процедуры и функции могут также получать данные из файлов *input* и выдавать результат в файле *output*. Передача значений между вызывающей программой и процедурой может осуществляться также с помощью глобальных переменных.

Локальные и глобальные переменные. Переменные описываются в программе в разделе описания переменных. Если же программа содержит описание процедуры, то некоторые переменные могут быть описаны в этой процедуре. Переменные, описанные только в главной программе, называются глобальными. Такими переменными можно пользоваться и в главной программе, и в процедуре. Переменные, описанные в процедуре, называются локальными. Этими переменными можно пользоваться только в процедуре. Вне тела процедуры значения таких переменных не определены.

```
program sum ;
var p,q,s:integer;
procedure now;
begin
  p:=1
  q:=1;
end;
BEGIN
  p:=0;
  q:=0;
  now;
  s:=p+q;
  write (s);
END.
```

В результате выполнения оператора вызова процедуры *now* глобальным переменным p и q присваивается значение 1. Поэтому оператор *write(s)* выдаст значение 2. p и q – глобальные.

Рассмотрим пример, где p является локальной и q – глобальной.

```
program sum ;
var p,q,s:integer;
procedure now;
var p:=integer;
begin
  p:=1;
  q:=1;
end;
BEGIN
  p:=0;
  q:=0;
  now; s:=p+q;
  write (s);
END.
```

Теперь в результате выполнения оператора вызова процедуры значения глобальной переменной q и локальной переменной p будут равны 1. Но значения локальной переменной p теперь недоступно главной программе, т. к. оно считается не определенным вне процедуры. Поэтому при выполнении оператора $s:=p+q$ в качестве значения p будет взято значение глобальной переменной p, которое равно 0, поэтому будет $s=1$.

Если p и q будут описаны в процедуре, то они будут локальные внутри процедуры и не будут иметь ничего общего с глобальными переменными p и q, которые не описаны в основной программе. В результате выполнения такой программы будет напечатано 0.

Рассмотренные примеры показывают, что процедуры могут использовать и изменять значение глобальных переменных. Если процедура изменяет значение глобальной переменной, то говорят, что она имеет побочный эффект. Если такой эффект не был предусмотрен программистом, то будет ошибка и ее очень трудно обнаружить. Чтобы избежать непредусмотренных эффектов, то не забывайте описывать вновь

введенные переменные и не используйте одинаковых имен переменных в главной программе и процедурах и функциях.

```
program gl;      {В трех квадратных матрицах одинаковой размерности определить сумму }
const n=3;       {элементов в первом и последнем столбце}
type mas = array[1..n,1..n] of integer;
var i,j:integer; a,b,c:mas; a1,b1,c1:real;
procedure inpmas(var z:mas);
begin
  for i:=1 to n do
      begin
          for j:=1 to n do
              begin
                  z[i,j]:=random(20)-5; write (z[i,j]:9)
              end;
           writeln;
      end;
  writeln;
end;       {end inpmas}
procedure coun(z:mas;var s:real);
begin
  s:=0;
  for i:=1 to n do
      begin
          s:=s+z[i,1]+z[i,n];
      end;
end;       {end counddig}
BEGIN
  inpmas(a); inpmas(b); inpmas(c);
  coun(a,a1); writeln('sum a=',a1:7:2);
  coun(b,b1);writeln('sum b=',b1:7:2);
  coun(c,c1); writeln('sum c=',c1:7:2);
                              END.
```