**Процедуры и функции**

В языке Паскаль, как и в большинстве языков программирования, предусмотрены средства, позволяющие оформлять вспомогательный алгоритм как подпрограмму. Это бывает необходимо тогда, когда какой-либо подалгоритм неоднократно повторяется в программе или имеется возможность использовать некоторые фрагменты уже разработанных ранее алгоритмов. Кроме того, подпрограммы применяются для разбиения крупных программ на отдельные смысловые части в соответствии с модульным принципом в программировании.

Для использования подалгоритма в качестве подпрограммы ему необходимо присвоить имя и описать алгоритм по правилам языка Паскаль. В дальнейшем, при необходимости вызвать его в программе, делают вызов подпрограммы упоминанием в нужном месте имени соответствующего подалгоритма со списком входных и выходных данных. Такое упоминание приводит к выполнению входящих в подпрограмму операторов, работающих с указанными данными. После выполнения подпрограммы работа продолжается с той команды, которая непосредственно следует за вызовом подпрограммы.

В языке Паскаль имеется два вида подпрограмм - **процедуры** и **функции**.

Процедуры и функции помещаются в раздел описаний программы. Для обмена информацией между процедурами и функциями и другими блоками программы существует механизм **входных** и **выходных параметров**. Входными параметрами называют величины, передающиеся из вызывающего блока в подпрограмму (исходные данные для подпрограммы), а выходными - передающиеся из подрограммы в вызывающий блок (результаты работы подпрограммы).

Одна и та же подпрограмма может вызываться неоднократно, выполняя одни и те же действия с разными наборами входных данных. Параметры, использующиеся при записи текста подпрограммы в разделе описаний, называют **формальными**, а те, что используются при ее вызове - **фактическими**.

**Описание и вызов процедур и функций**

Структура описания процедур и функций до некоторой степени похожа на структуру Паскаль-программы: у них также имеются заголовок, раздел описаний и исполняемая часть. Раздел описаний содержит те же подразделы, что и раздел описаний программы: описания констант, типов, меток, процедур, функций, перменных. Исполняемая часть содержит собственно операторы процедур.

Формат описания процедуры имеет вид:

procedure *имя процедуры* (*формальные параметры*);

 *раздел описаний процедуры*

begin

 *исполняемая часть процедуры*

end;

Формат описания функции:

function *имя функции* (*формальные параметры*):*тип результата*;

 *раздел описаний функции*

begin

 *исполняемая часть функции*

end;

Формальные параметры в заголовке процедур и функций записываются в виде:

var *имя праметра*: *имя типа*

и отделяются друг от друга точкой с запятой. Ключевое слово var может отсутствовать (об этом далее). Если параметры однотипны, то их имена можно перечислять через запятую, указывая общее для них имя типа. При описании параметров можно использовать только стандартные имена типов, либо имена типов, определенные с помощью команды type.Список формальных параметров может отсутствовать.

Вызов процедуры производится оператором, имеющим следующий формат:

*имя процедуры*(*список фактических параметров*);

*Список фактических параметров* - это их перечисление через запятую. При вызове фактические параметры как бы подставляются вместо формальных, стоящих на тех же местах в заголовке. Таким образом происходит передача входных параметров, затем выполняются операторы исполняемой части процедуры, после чего происходит возврат в вызывающий блок. Передача выходных параметров происходит непосредственно во время работы исполняемой части.

Вызов функции в Турбо Паскаль может производиться аналогичным способом, кроме того имеется возможность осуществить вызов внутри какого-либо выражения. В частности имя функции может стоять в правой части оператора присваивания, в разделе условий оператора if и т.д.

Для передачи в вызывающий блок выходного значения функции в исполняемой части функции перед возвратом в вызывающий блок необходимо поместить следующую команду:

*имя функции* := *результат*;

При вызове процедур и функций необходимо соблюдать следущие правила:

* количество фактических параметров должно совпадать с количеством формальных;
* соответствующие фактические и формальные параметры должны совпадать по порядку следования и по типу.

Заметим, что имена формальных и фактических параметров могут совпадать. Это не приводит к проблемам, так как соответствующие им переменные все равно будут различны из-за того, что хранятся в разных областях памяти. Кроме того, все формальные параметры являются временными переменными - они создаются в момент вызова подпрограммы и уничтожаются в момент выхода из нее.

Рассмотрим использование процедуры на примере программы поиска максимума из двух целых чисел.

var x,y,m,n: integer;

procedure MaxNumber(a,b: integer; var max: integer);

begin

 if a>b then max:=a else max:=b;

end;

begin

 write('Введите x,y ');

 readln(x,y);

 MaxNumber(x,y,m);

 MaxNumber(2,x+y,n);

 writeln('m=',m,'n=',n);

end.

Аналогичную задачу, но уже с использованием функций, можно решить так:

var x,y,m,n: integer;

function MaxNumber(a,b: integer): integer;

 var max: integer;

begin

 if a>b then max:=a else max:=b;

 MaxNumber := max;

end;

begin

 write('Введите x,y ');

 readln(x,y);

 m := MaxNumber(x,y);

 n := MaxNumber(2,x+y);

 writeln('m=',m,'n=',n);

end.