Тема:

Подпрограммы в ABCPascal

Подпрограмма - это часть программы, описывающая некоторый алгоритм, который можно многократно использовать, обращаясь к нему из различных точек программы.

Применение подпрограмм дает возможность уменьшать число повторений одной и той же последовательности операторов.

Это позволяет получить более логичный процесс программирования.

Выполнение программы, имеющую подпрограмму, начинается с выполнения основной программы. Как только в программе идет обращение к подпрограмме, данные из основной программы (входные данные) передаются в подпрограмму, которая начинает выполняться.

Затем результаты подпрограммы (выходные данные) передаются в основную программу в то место, откуда был сделан вызов подпрограммы, и продолжает выполняться основная

Подпрограмма оформляется подобно основной программе, т.е. состоит из заголовка, раздела описаний, раздела операторов.

Операторы подпрограммы, окаймленные операторными скобками begin..end, называются *телом* этой подпрограммы.

Все имена, представленные в разделе описаний основной программы, называются *глобальными*. Они действуют как в разделе операторов основной программы, так и в любой подпрограмме.

Имена, представленные в разделе описаний подпрограммы, называют *локальными*. Они действуют только в рамках подпрограммы и недоступны операторам основной программы.

В языке Паскаль имеется два вида подпрограмм:

1. *процедура* (PROCEDURE),
2. *функция* (FUNCTION)

Функции являются частным случаем процедур и принципиально отличаются от них следующим:

1. результат выполнения функции - одно значение, а процедуры - одно или несколько;

2. результат выполнения функции передается в основную программу как значение имени этой функции, а результаты выполнения процедуры - как значения ее параметров.

Процедуры

Описание процедуры имеет вид:

```
procedure имя(список формальных
          параметров);
раздел описаний
begin
операторы
end;
```

Список формальных параметров состоит из одной или нескольких секций, разделенных символом ";". Каждая секция состоит из списка переменных, перечисляемых через запятую, после которого следуют двоеточие и тип.

Каждая секция может предваряться служебным словом **var**, что указывает на то, что параметры передаются по ссылке.

Список формальных параметров вместе с окружающими скобками может

Раздел описаний процедуры или функции устроен так же, как и раздел описаний основной программы.

Здесь описываются так называемые *локальные* переменные и константы, типы а также вложенные процедуры и функции. Все такие локальные объекты доступны лишь внутри данной подпрограммы и не видны извне.

Пример . Процедура меняет местами первый и последний, второй и предпоследний и т.д. элементы массива.

```
procedure Reverse(var a: array
[1..100] of integer; n: integer);
var i,v: integer;
begin
for i:=1 to n div 2 do
begin
v:=a[i];
a[i]:=a[n-i+1];
a[n-i+1]:=v;
end;
end;
```

Функции

Описание функции имеет вид:

```
   function имя(список формальных
параметров): тип возвращаемого значения;

   раздел описаний
   begin
   операторы
   end;
```

Внутри тела функции имя этой функции можно использовать как специальную переменную, которой необходимо присвоить возвращаемое значение. Например:

```
function Add(a,b: real): real;
begin
Add:=a+b;
end;
```

Имя функции может быть использовано с целью возврата значения только в левой части оператора присваивания.

Если имя функции встречается в выражении, то это трактуется как рекурсивный вызов этой функции.

Вместо имени функции, внутри тела функции можно использовать другую специальную переменную с именем Result. В отличие от имени функции, переменную Result можно использовать и в выражениях:

```
function Min(var a: array [1..100] of
          real; n: integer): real;
var i: integer;
begin
Result:=a[1];
for i:=1 to n do
if a[i]<Result then Result:=a[i];
end;
```

Если внутри функции не присвоить имени функции или переменной Result некоторое значение, то функция вернет в результате своего вызова непредсказуемое значение.

Параметры процедур и функций

Параметры, указываемые при описании подпрограммы, называются *формальными*. Параметры, указываемые при вызове подпрограммы, называются *фактическими*.

Если формальный параметр описан со служебным словом **var**, то его называют *параметром-переменной* и говорят, что он передается *по ссылке*.

Если же параметр описан без слова **var**, то его называют *параметром-значением* и говорят, что он передается *по значению*.

Если параметр передается по значению, то при вызове подпрограммы значения фактических параметров присваиваются соответствующим формальным параметрам. При этом изменение формального параметра не приводит к изменению фактического.

Если параметр передается по ссылке, то при вызове подпрограммы фактический параметр заменяет собой в теле процедуры соответствующий ему формальный параметр. В итоге любые изменения формального параметра-переменной внутри процедуры приводят к соответствующим изменениям фактического параметра.

При передаче параметра по ссылке в подпрограмму передается адрес фактического параметра.

Поэтому если параметр занимает много памяти (массив, запись), то обычно он также передается по ссылке.

В результате в процедуру передается не сам параметр, а его адрес, что экономит память и время работы.

В качестве фактического параметра-значения можно указывать любое выражение, тип которого совпадает с типом формального параметра или неявно к нему приводится.

В качестве фактического параметра-переменной можно указывать только переменную, тип которой в точности совпадает с типом формального параметра.

Параметр может быть также описан со служебным словом **const** (вместо **var**). Это означает, что менять его в подпрограмме запрещено.

Локальные и глобальные переменные

Переменные, описанные в разделе описаний подпрограммы, называются ее *локальными* переменными. Переменные же, описанные вне подпрограммы, называются *глобальными* по отношению к ней.

Параметры подпрограммы считаются ее локальными переменными.

Если имя локальной переменной совпадает с именем глобальной переменной, то локальная переменная скрывает глобальную, так что к глобальной переменной нельзя обратиться внутри

Пример 1. Вычислить число сочетаний

$$C_n^m = \frac{n!}{m!(n-m)!}$$

Для вычисления факториала использовать функцию.

```
var
 C:real;
 n,m:byte;

 function F(x:byte):longint;
 var i:byte;
 begin
 f:=1;
 for i:=2 to x do
     result:=result*i;
      end;

BEGIN
Writeln('n,m');Readln(n,m);
c:=f(n)/(F(m)*f(n-m));
Writeln('C=',c);
end.
```

Пример 2. Даны три массива. Вычислить сумму элементов каждого из них.

Для ввода и вывода массивов использовать процедуры, для вычисления суммы использовать функцию.

```
type
   massiv=array [1..30] of integer;
var
  n1,n2,n3,i:byte; // n1,n2,n3 - размерности массивов
  a,b,c:massiv;

procedure vvod(var m:byte; var x:massiv);
begin
write('размерность = ');
readln(m);
Writeln('элементы');
for i:=1 to m do
  begin
  write(i,' элемент ');
  readln(x[i]);
  end;
end;

procedure vyvod(m:byte; x:massiv);
begin
write('(');
for i:=1 to m do
 write(x[i]:6);
writeln(')');
end;
```

```
function Summa(m:byte; x:massiv):integer;
begin
result:=0;
for i:=1 to m do
  result:=result+x[i];
end;

BEGIN
Writeln('Введите массив A');
vvod(n1,a);
Writeln('Введите массив B');
vvod(n2,b);
Writeln('Введите массив C');
vvod(n3,c);
clrscr;
Writeln(' Результаты:');
Write('Массив A=');
vyvod(n1,A);
Writeln('Сумма его элементов равна ', summa(n1,a));
Writeln;

Write('Массив B=');
vyvod(n2,b);
Writeln('Сумма его элементов равна ', summa(n2,b));
Writeln;

Write('Массив C=');
vyvod(n3,c);
Writeln('Сумма его элементов равна ', summa(n3,c));
end.
```

Пример 3. Из диапазона целых чисел от n до m выписать простые числа.

```
var
 n,m,i:integer;

function Prost_Chislo(t:integer):boolean;
 var j:integer;
 begin
 result:=true;
 j:=2;
 While result and (j<t) do
 if t mod j =0 then result:=false else inc(j);
 end;

 begin
 Writeln('n,m');readln(n,m);
 Writeln('Простые числа в этом диапазоне');
 for i:=n to m do
   if Prost_Chislo(i) then writeln(i);
 end.
```

Задание для самостоятельного выполнения

Задача 1.Квадратное уравнение задано своими коэффициентами a,b,c. Разработать программу его решения. Вычисление дискриминанта выполнить с использованием функции.

Задача 2. В квадратных матрицах A и B посчитать сумму элементов главной диагонали. Ввод и вывод матриц оформить в виде процедур, подсчет суммы элементов главной диагонали – в виде функции.