Процедуры и функции в Паскале. Рекурсия

Подпрограммы

Часто в задаче требуется повторить определенную последовательность операторов в разных частях программы. Для того, чтобы описывать эту последовательность один раз, а применять многократно, в языках программирования применяются подпрограммы.

Подпрограмма - специальным образом оформленный блок программы, для дальнейшего его многократного использования в основной программе

Использование подпрограмм позволяет реализовать один из самых современных методов программирования - структурное программирование

Подпрограммы решают три важные задачи, значительно облегчающие программирование:

1. избавляют от необходимости многократно повторять в тексте программы аналогичные фрагменты, т.е. сократить объём программы;
2. улучшат структуру программы, облегчая понимание при разборе;
3. уменьшают вероятность появления ошибок, повышают устойчивость к ошибкам программирования и непредвиденным последствиям при модификации.

Процедуры и функции

В языке Паскаль существует два вида подпрограмм: **процедура (PROCEDURE)** и **функция (FUNCTION)**. Процедуры и функции в Паскале объявляются в разделе описания за разделом переменных.

```
Program ИмяПрограммы;
VAR … // раздел описания переменных главной программы;

procedure ИмяПроцедуры;
var …
begin
…//Тело процедуры
end;

begin
//тело главной программы
end.
```

У функций и процедур существуют **параметры** (переменные, которые передают какое - либо значение). Они бывают двух видов:

1) **Формальные** - те, которые находятся в описании подпрограммы

2) **Фактические** - те, которые передаются из основной программы в функцию или процедуру.

Фактические параметры должны соответствовать формальным по количеству, порядку следования и типу.

Также у подпрограммы существую **переменные**. с которыми она в дальнейшем работает. Они делятся опять же на два типа:

1) **Глобальные переменные**, то есть действующие во всей программе

2**) Локальные** - те, которые действуют только в процедуре или функции

Процедуры

Процедуры используются в случаях, когда в подпрограмме **необходимо получить несколько результатов**.

В языке Паскаль существует два вида процедур: **процедуры с параметрами и без параметров**.

Обращение к процедуре осуществляется по имени процедуры, за которым могут быть указаны фактические параметры.

При вызове процедуры устанавливается взаимно однозначное соответствие между фактическими и формальными параметрами, затем управление передается процедуре.

После выполнения процедуры управление передается следующему, после вызова процедуры, оператору вызывающей программы.

Пример 1. Процедура без параметров, которая печатает строку из 60 звездочек.

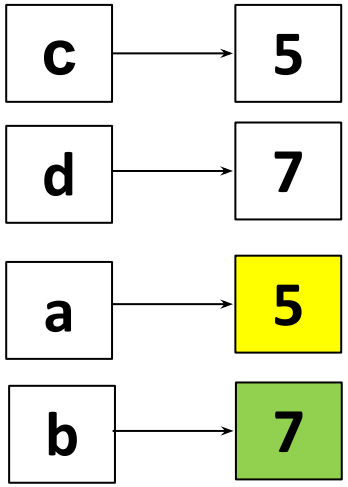
```
procedure  pr;
   var      i :  integer ;
   begin
       for i :=1 to 60 do write (‘ * ');    writeln;
   end;

begin
   pr;
end.
```

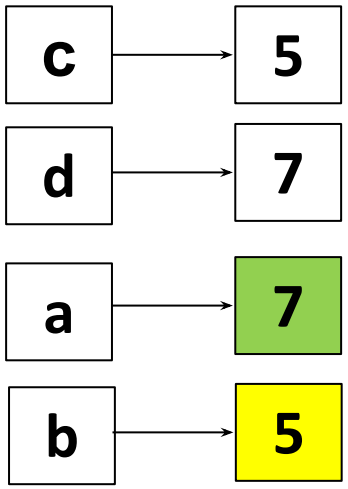
Пример 2. Составить программу обмена местами двух чисел c=5 и d=7

```
program obmenDan;
var c,d:integer;
procedure obmen ( a,b:integer);
var m:integer;
begin
    m:=a; a:=b; b:=m;
writeln(a,b);
end;
begin
      writeln ('Введите 2 числа: ');
       readln(c,d);
       obmen(c,d);
       writeln(c,' ',d);
end.
```

1) при вызове процедуры obmen с двумя параметрами 5 и 7, в переменные a и b помещаются тоже числа 5 и 7 соответственно:



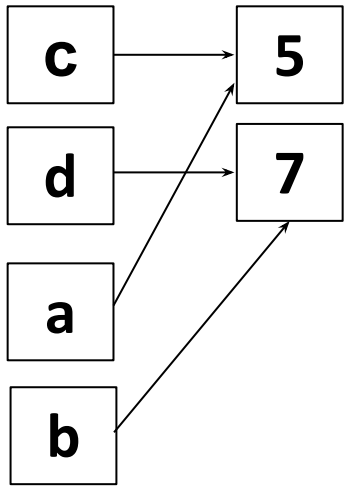
2) далее в процедуре осуществляется перестановка значений ячеек памяти a и b:



3) но в переменных c и d данные не поменялись, т.к. они находятся в других ячейках памяти

Для того чтобы переменные **c** и **d**, **a** и **b** ссылались на одни и те же ячейки памяти (если изменятся значения **a** и **b**, то изменятся значения и **c, d**) необходимо при описании формальных параметров, перед нужными переменными добавить слово **VAR**:

procedure obmen (**var a,b:integer**);



Пример 3.

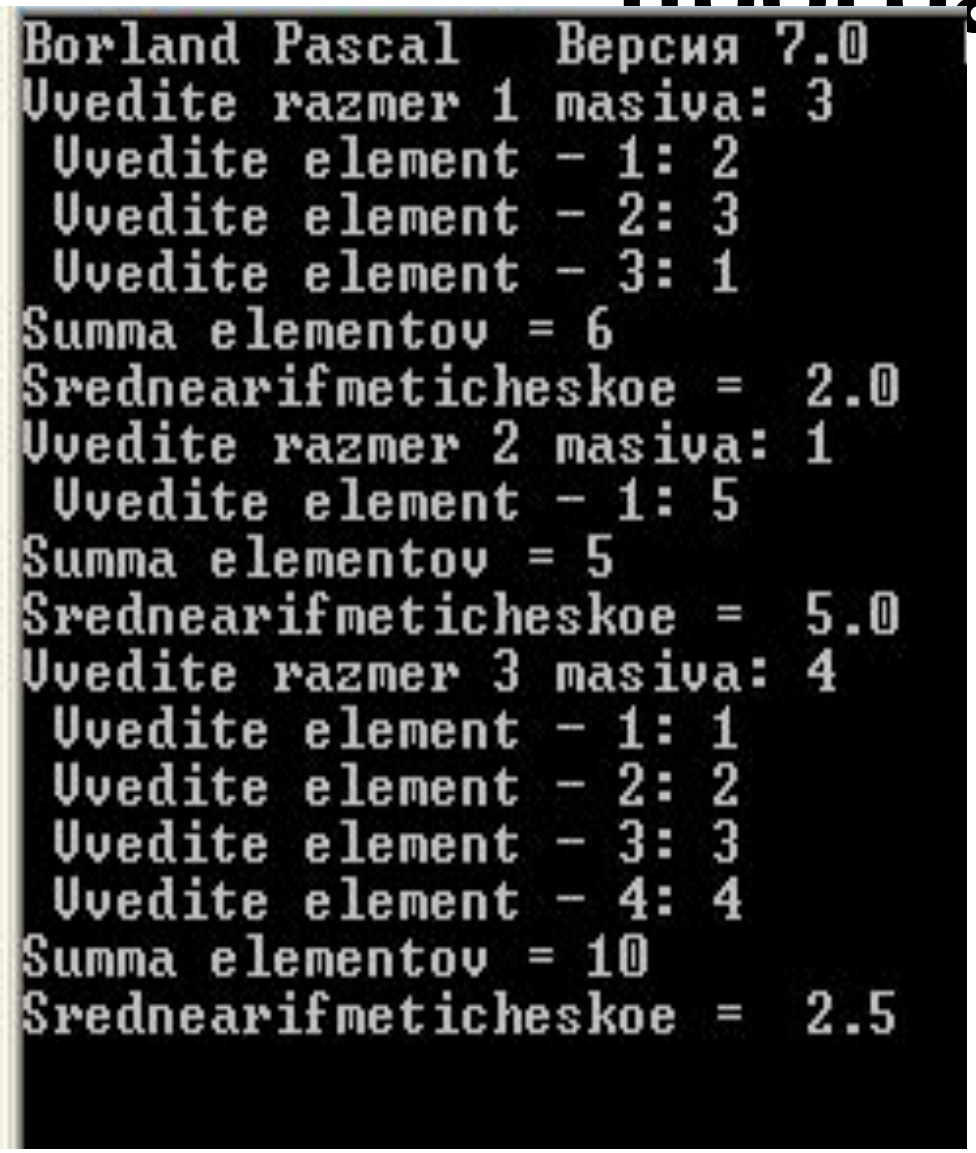
Даны 3 различных массива целых чисел (размер каждого не превышает 15). В каждом массиве **найти сумму элементов и среднеарифметическое значение**

```
program proc;
 var i , n , sum: integer;
 sr : real;
 procedure work (r:integer; var s:integer; var s1:real);
 var mas : array [1..15] of integer ;
  j : integer;
  begin
       s:=0;
       for j:=1 to r do
       begin
            read (mas[j]);  s:=s+mas [j];
       end;
       s1:=s/r;
   end;
```

{ главная программа}

```
begin
    for i:=1 to 3 do
    begin
        write ('Vvedite razmer ',i, ' masiva: ');
        readln(n);
        work (n, sum, sr); {вызов процедуры work}
        writeln ('Summa elementov = ',sum);
        writeln ('Srednearifmeticheskoe = ',sr:4:1);
    end;
end.
```

Результат работы программы:



В программе трижды вызывается процедура **work**, в которой ***формальные переменные r, s, s1*** заменяются ***фактическими n, sum, sr.*** Процедура выполняет ввод элементов массива, вычисляет сумму и среднее значение. **Переменные s и s1 возвращаются в главную программу**, поэтому перед их описанием ставится служебное слово **var**. Локальные параметры **mas, j** действуют только в процедуре.

Функции в Паскале

Набор встроенных функций в языке Паскаль достаточно широк (ABS, SQR, TRUNC и т.д.). Если в программу включается новая, нестандартная функция, то ее необходимо описать в тексте программы, после чего можно обращаться к ней из программы. **Обращение к функции осуществляется в правой части оператора присваивания, с указанием имени функции и фактических параметров.** Функция может иметь собственные локальные константы, типы, переменные, процедуры и функции. Описание функций в Паскале аналогично описанию процедур.

Отличительные особенности функций:

- результат выполнения - одно значение, которое присваивается имени функции и передается в основную программу;

- имя функции может входить в выражение как операнд.

Пример 4. Написать подпрограмму-функцию степени a^x, где a, x – любые числа. Воспользуемся формулой: $a^x = e^{x \ln a}$

```
program p2;
 var  f,  b, s, t, c, d : real; { глобальные переменные}
 function  stp (a, x : real) : real;
    var   y : real;  { локальные переменные}
    begin
          y := exp (x * ln ( a)) ;
          stp:= y;{присвоение имени функции результата   вычислений
          подпрограммы}
     end;  { описание функции закончено }
begin
   d:= stp  (2.4, 5); {вычисление степеней  разных  чисел и переменных}
   writeln (d,  stp (5,3.5));
   read (f, b, s, t);    c := stp  (f, s)+stp  (b, t);
   writeln (c);
end.
```

Описание подпрограмм

Подпрограмма — часть программы, оформленная в виде отдельной синтаксической конструкции и снабжённая именем (самостоятельный программный блок), для решения отдельных задач.

Процедуры

Описание процедуры:

```
procedure<имя> (<список формальных параметров>)
 {раздел выполнения локальных имён}
Begin
 {раздел выполнения операторов}
End;
```

Вызов процедуры:

<имя >(<список фактических переменных>);

Функции

Описание функции:

```
function<имя> (<список формальных параметров>): тип;
 {раздел описания локальных имён}
Begin
 {раздел выполняемых операторов}
<Имя функции>:=<значение>;
{обязательный параметр}
End;
```

Вызов функции:

< оператор>:= <имя функции>
(<список фактических переменных>);

1. В правой части оператора присваивания.
2. В выражении, стоящем в условии оператора разветвления.
3. В процедуре вывода, как результат работы функции.

Рекурсия

Процедуры и функции в Паскале могут вызывать сами себя, т.е. обладать свойством рекурсивности. Рекурсивная функция обязательно должна содержать в себе условие окончания рекурсивности, чтобы не вызвать зацикливания программы. При каждом рекурсивном вызове создается новое множество локальных переменных. То есть переменные, расположенные вне вызываемой функции, не изменяются.

Пример 5. Составить рекурсивную функцию, вычисляющую факториал числа **n** следующим образом: **n! = 1 , если n= 1**

n!= (n -1)! · n , если n > 1

```
function   f ( n : integer): integer;
begin
     if  n = 1 then  f := 1  else f := n * f ( n -1 );
    {функция f вызывает саму себя}
end;
```

Задачи

1)Найти площадь круга с использованием процедуры ифункции.

2)Найти НОД и НОК

3) Найти 1!+2!+…+n!

4) упорядочить значения трёх переменных a, b и c в порядке их убывания