




# Функции и процедуры

- Инструмент структурирования программ
- Два типа подпрограмм
- Описание
- Локальные и глобальные переменные
- Параметры: формальные и фактические
- Два класса формальных параметров

- 
- Процедурно-ориентированные языки имеют средства структурирования программ.
  - Структурирование предполагает расчленение программы на относительно самостоятельные фрагменты
  - Нисходящее программирование – метод конструирования сложных программ



## Отличие функции от процедуры

- Процедура может возвращать значение обработки (вычислений), если параметр объявляется с атрибутом VAR как параметр-переменная. Процедура на выходе может выдавать несколько значений или ни одного.
- Вызов функции можно использовать в списках параметров оператора WRITE (невозможно для процедуры).
- В теле функции результат вычислений обязательно присваивается переменной, имя которой совпадает с именем функции.



## Описание подпрограммы

После раздела переменных и констант и  
до начала основной части

### Структура

Структура подпрограммы такая же как основной программы

<заголовок функции(процедуры)>


<раздел описаний переменных и констант>

**begin**

<операторы>

**end;**

Переменные локальные и глобальные



Глобальные переменные объявляются в основной программе. Доступны основной и всем ее подпрограммам.

Локальные объявлены внутри подпрограммы и доступны только ей самой.

Одноименные глобальные и локальные переменные – разные переменные.

Обращение к таким переменным в теле подпрограммы трактуется как к локальным (глобальные не доступны).



**FUNCTION**<имя>(<параметры>):<тип ф-ции>

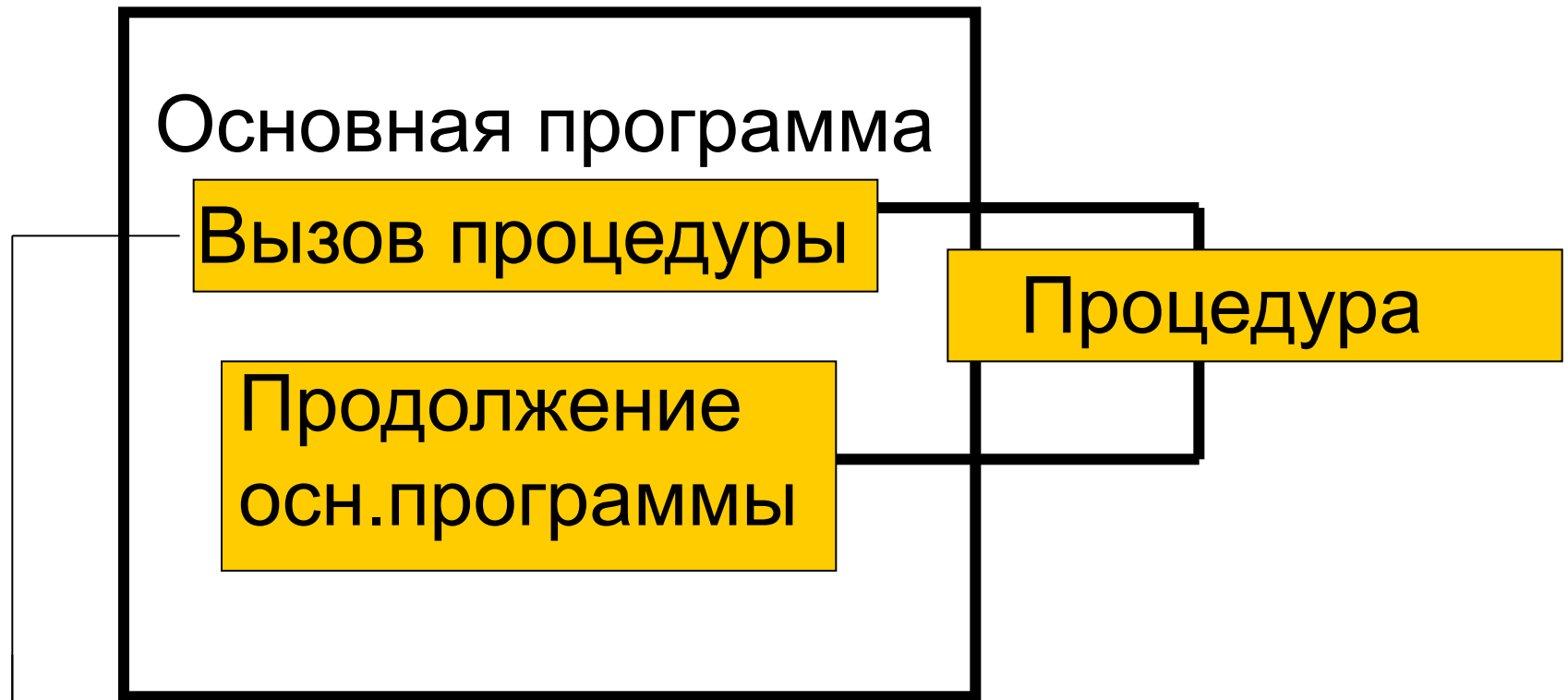
**FUNCTION** beta (a, b : real; c : integer) : real

**PROCEDURE**<имя>(<параметры>)


**PROCEDURE** vsp (x, y: integer; var m, n: real)

Формальные параметры

# Вызов процедуры (функции)



Указание имени и в круглых скобках списка фактических параметров



Количество и тип фактических параметров  
должны совпадать с количеством и типом  
формальных параметров

**d:=beta(3, 4, 7.5)**

3,4,7.5 фактические параметры (константы),  
перечисляются через запятую







```
const
```

```
a:integer=5; b:integer=7 ;
```

```
procedure udv (var c:integer;d:integer);
```

```
begin
```

```
    c:=2*c;
```

```
    d:=2*d;
```

```
    writeln('udvoennoe:',c:5,d:5);
```

```
end;
```


```
begin
```

```
writeln (' Ishodnoe:',a:5,b:5);
```

```
udv (a,b);
```

```
writeln(' rezultat:',a:5,b:5); readln
```

```
end.
```



```
var a,b:integer;  
function max(i,j:integer):integer;  
begin  
    if i>j then max:=i else max:=j  
    end;  
begin  
    writeln('Vvedite a,b');  
    readln(a,b);  
    writeln('Maxim=',max(a,b));  
    readln;  
end.
```



**program Factorials; {Ctrl+Break Enter}**

**var n:integer;**

**Function Factorial (k:integer):longint;**

**var i: integer; f: longint;**

**begin f:=1;**

**for i:=1 to k do f:=f\*i;**

**factorial:=f; end;**

**begin**

**repeat**

**writeln('Vvedite n'); readln(n);**

**if n<0 then writeln('Error')**

**else writeln(n,'!=',factorial(n));**

**until eof**

**end.**

**Program sochet;**  
**{C =n!/(m!\*(n-m)! ) }**  
**uses CRT;**  
**var n,m:integer; a,b,c,d: longint;**  
**Function Factorial (k:integer):longint;**  
    **var i: integer; f: longint;**  
    **begin**  
        **f:=1;**  
        **for l:=1 to k do f:=f\*i;**  
        **factorial:=f;**  
    **end;**

```
begin
Clrscr;
  writeln('Vvedite kol sochet iz n po m');
readln(n,m);
  a:=Factorial(n);
  b:=Factorial(m);
  c:=Factorial(n-m);
  d:=a div (b*c);
writeln('Kol sochet iz',n:2,' po',m:2,' =',d:3);
readln;
end.
```

```
const raz=20;  
var  
n,m: integer;  
massiv: array[1..raz,1..raz] of integer;  
x,y: integer;  
procedure massiv_out (l,k: integer);  
    var    i,j: integer;  
    begin  
        for i:=1 to l do begin  
            for j:=1 to k do  
                write(massiv[i,j]:6);  
                writeln          end;  
        end;
```

```
begin  
writeln('Vvedite razmer massiva N x M');  
readln(n,m);  
for x:=1 to n do  
for y:=1 to m do  
massiv[x,y]:=1;  
massiv_out(n,m);  
readln;  
end.
```





# Рекурсия

**Процедура (функция) может обращаться к другой процедуре(функции).**

**Вызов функции из нее самой называется рекурсией.**

Recurrence – повторение или возвращение



**program Factorials;**

**var n:integer;**

**Function Factorial (n:integer):real;**

**begin**

**if n=0 then factorial:=1**

**else factorial:=n\*factorial(n-1)**

**end;**

**begin**

**repeat**


**writeln('Vvedite n'); readln(n);**

**if n<0 then writeln('Error')**

**else writeln('n!=',factorial(n));**

**until eof**

**end.**



```
var k:integer;  
function Fibon(k:integer):integer;  
begin  
    if k=0 then fibon:=0;  
    if (k=1) or(k=2) then fibon:=1  
    else fibon:=fibon(k-2)+fibon(k-1)  
    end;  
begin  
repeat  
    writeln('Vvedite k'); readln(k);  
until k>=0;  
    writeln(k:4,'number Fibon=',fibon(k));  
    readln end.
```