

Подпрограммы pascal



Подпрограммы

```
graph TD; A[Подпрограммы] --- B[Функции  
возвращает 1 значение]; A --- C[Процедуры  
ничего не возвращают]
```

Функции
возвращает 1 значение

Процедуры
ничего не возвращают

Пример программы с процедурой:

Нахождение факториала числа.

```
program f;
var
  x,y: integer;
procedure factorial(n: integer);
var
  fact,i: integer;
begin
  fact:=1;
  for i:=1 to n do
  begin
    fact:=fact*i;
  end;
  writeln(n, '!=' , fact);
end;

begin
  write('Введите число: ');
  readln(x);
  factorial(x);
  factorial(2*x);
  readln(y);
  factorial(y);
end.
```

Пример программы с функцией:

Вычисление факториала:

```
program f1;
```

```
var a:integer;
```

```
S: Longint;
```

```
Function Factorial(N:Byte):Longint; // определение функции
```

```
Var fact:longint; i:byte;
```

```
Begin
```

```
    fact:=n;
```

```
    For i:=n-1 downto 2 do
```

```
        fact:=fact*i;
```

```
    Factorial:=fact; //имени функции обязательно присваиваем //значение,  
которое она будет возвращать
```

```
End;
```

```
begin
```

```
    write('Введите число:');
```

```
    readln(a);
```

```
    writeln('факториал вашего числа равен ', Factorial(a));
```

```
    S:= Factorial(a);
```

```
    S:= Factorial(a)+ 3*Factorial(a*2);
```

```
end.
```

Объявление

Функция

```
function <имя_функции>  
  [(<список_параметров>)]:<тип_результата>;
```

```
function <имя_функции>  
  [(<параметры>)]:<тип_результата>; forward;
```

Объявление

□ Процедура

```
procedure <имя_процедуры>  
  [(<список_параметров>)];
```

```
procedure <имя_процедуры>  
  [(<список_параметров>)]; forward;
```

Описание подпрограммы

```
[ uses <имена_подключаемых_модулей>;]
[ label <список_меток>;]
[ const <имя_константы> = <значение_константы>;]
[ type <имя_типа> = <определение_типа>;]
[ var <имя_переменной> : <тип_переменной>;]

[ procedure <имя_процедуры>
  <описание_процедуры>;]
[ function <имя_функции> <описание_функции>;]

begin      {начало тела подпрограммы}
           <операторы>
end;      (* конец тела подпрограммы *)
```

Если использовалась директива **forward**

Описание

- `function <имя_подпрограммы> [<список входных параметров>];`

- `<тело подпрограммы>;`

или

- `procedure <имя_подпрограммы> [<список входных параметров>];`

- `<тело подпрограммы>;`

Пример подпрограммы-процедуры:

```
procedure err(c:byte; s:string);  
  var zz: byte;  
  begin if c = 0  
        then writeln(s)  
        else writeln('Ошибка!')  
  end;
```

Список параметров отсутствует:

- ❑ `procedure proc1;`
- ❑ `function func1: boolean;`

Описание параметров

- [`<способ_подстановки>`]`<имя_параметра>`:`<тип>`;
- [`<способ_подстановки>`]`<имя1 >`,...,`<имяN>`:
`<тип>`;
- `function func2(a,b:byte; var x,y,z:real; const c:char) : integer;`

Ошибка:

```
❑ procedure proc2(a: array[1..100]of char);
```

Правильно:

```
type arr = array[1..100] of char;
```

```
procedure proc2(a: arr);
```

```
function func2(var x: string; a1:arr): arr;
```

Возвращаемые значения

```
function min(a,b: integer): integer;  
  begin if a>b  
        then min:= b  
        else min:= a  
  end;
```

Вызов подпрограмм

<имя_подпрограммы>[(<список_аргументов>)]

пример

- `c := min(a, a*2);`
- `if min(z, min(x, y)) = 0 then...;`
- `err(res, 'Привет!');`

Способы подстановки аргументов

- <пустой>;
- var;
- const.

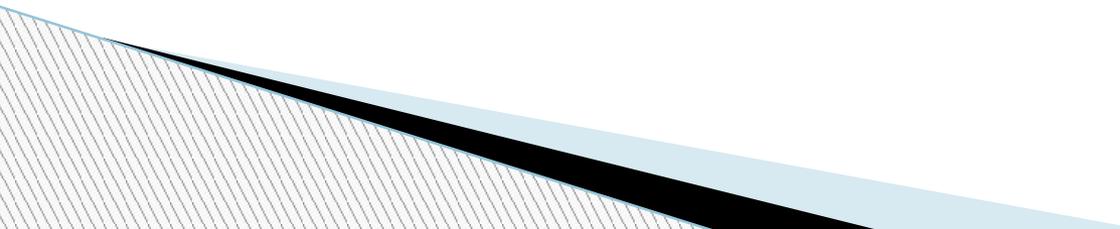
Параметр-значение

```
function func3(x:real; k:integer; flag:boolean):real;
```

```
dlina:= func3(shirina+2, min(a shl 1,ord('y')), true)+0.5
```

Параметр-переменная

```
procedure proc3(var x,y:real; var k:integer;  
  flag:boolean);
```



Параметр-константа

```
procedure proc4(var k:integer; const  
  flag:boolean);
```

