Понятие процедуры и функции. Механизм параметров.

Лекция 6

Декомпозиция задачи

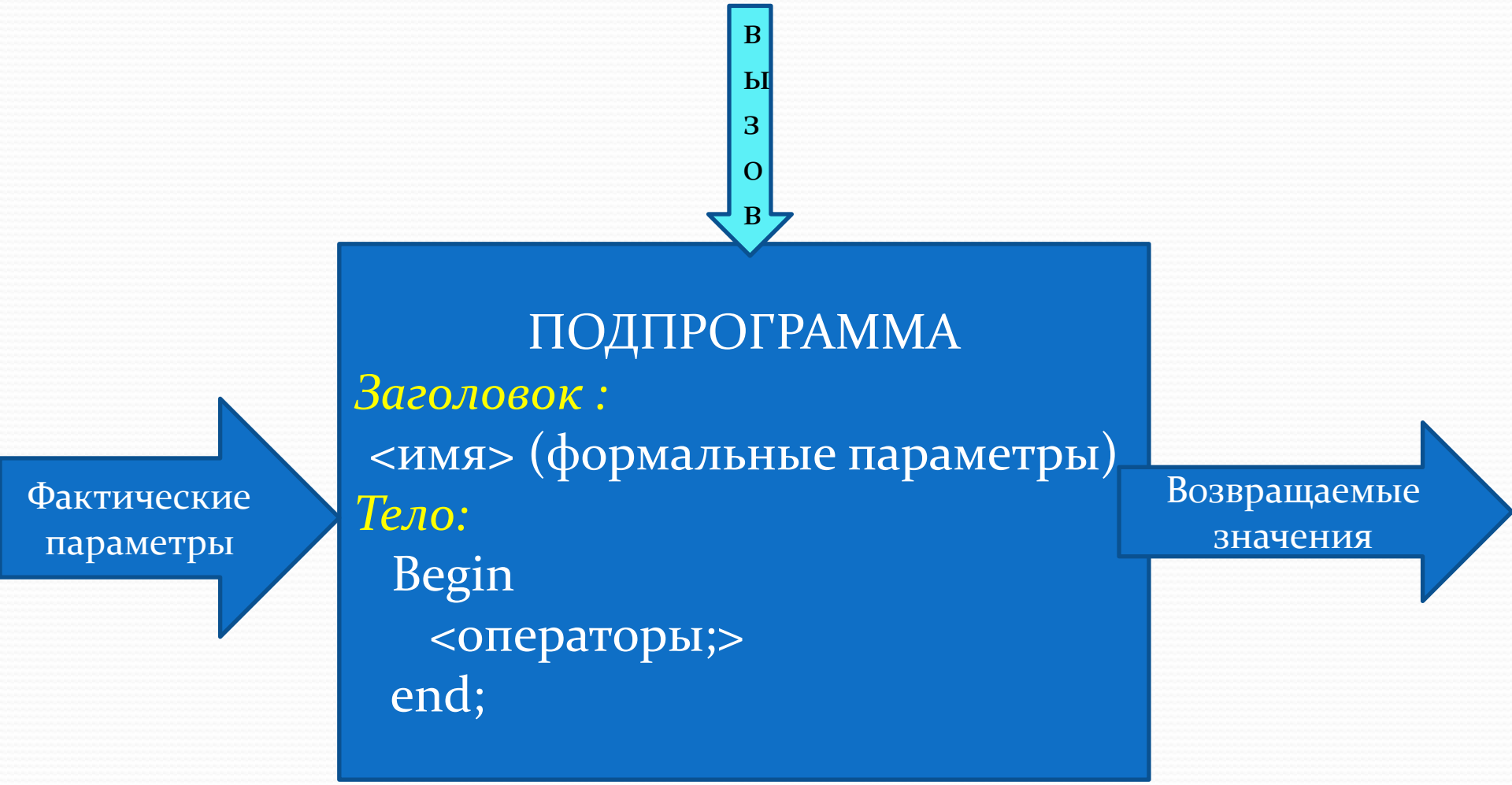
Преимущества использования подпрограмм

- появляется возможность поблочной отладки больших программ, возможно, создаваемых несколькими программистами с последующим объединением отлаженных подпрограмм в единое целое;
- экономится оперативная память, так как многократно используемые компоненты (подпрограммы) заносятся в память ЭВМ один раз;
- облегчаются изменения программы, так как изменение одной программы не вызывает корректировку других.

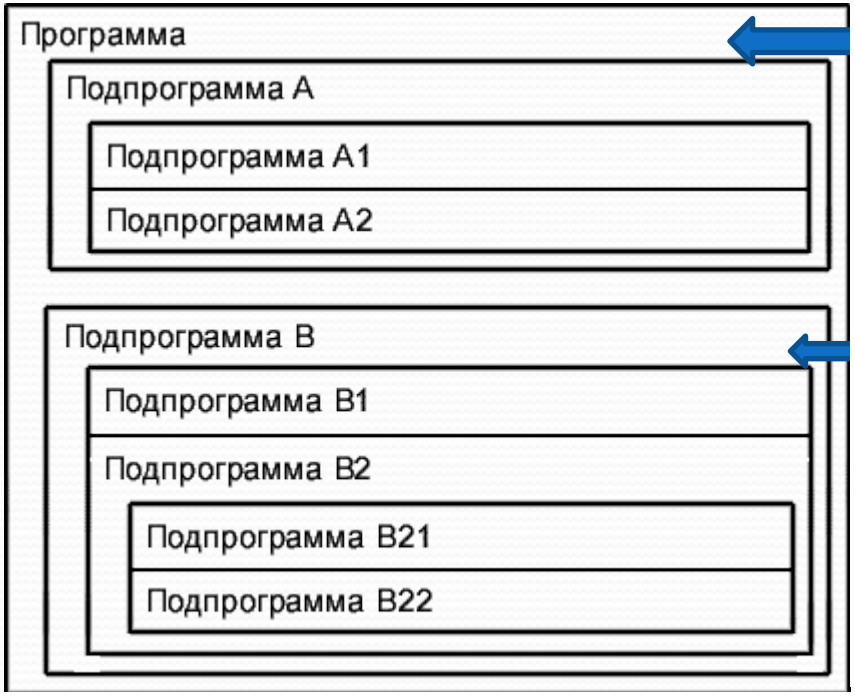
Способы использования подпрограмм:

- основная программа и подпрограммы располагаются в одном файле;
- тексты подпрограмм расположены в различных файлах и подключаются директивами компилятора;
- подпрограммы организуются как оверлейные структуры и поочередно загружаются на одно и то же место в оперативной памяти ЭВМ;
- подпрограммы пишутся на другом языке программирования и подключаются одним из вышеописанных способов;
- подпрограммы оформляются как внешние и вызываются из основной программы.

Подпрограмма



Локализация переменных




```
var i:integer;
   {глобальная переменная -
    описана вне всех подпрограмм}
Заголовок Подпрограммы;
var i:integer;
   {локальная переменная -
    описана после заголовка подпрограммы}
begin
 {Тело подпрограммы}
end;

begin
{Тело главной программы}
end.
```

- каждый идентификатор должен быть описаны перед использованием;
- областью действия идентификатора является та подпрограмма, в которой они описаны;
- все имена в пределах подпрограммы, в которой они объявлены, должны быть уникальными;
- одноименные локальные и глобальные переменные – это разные переменные;
- при обращении к подпрограмме доступны объекты, которые объявлены в ней и до ее описания.

Два типа подпрограмм:

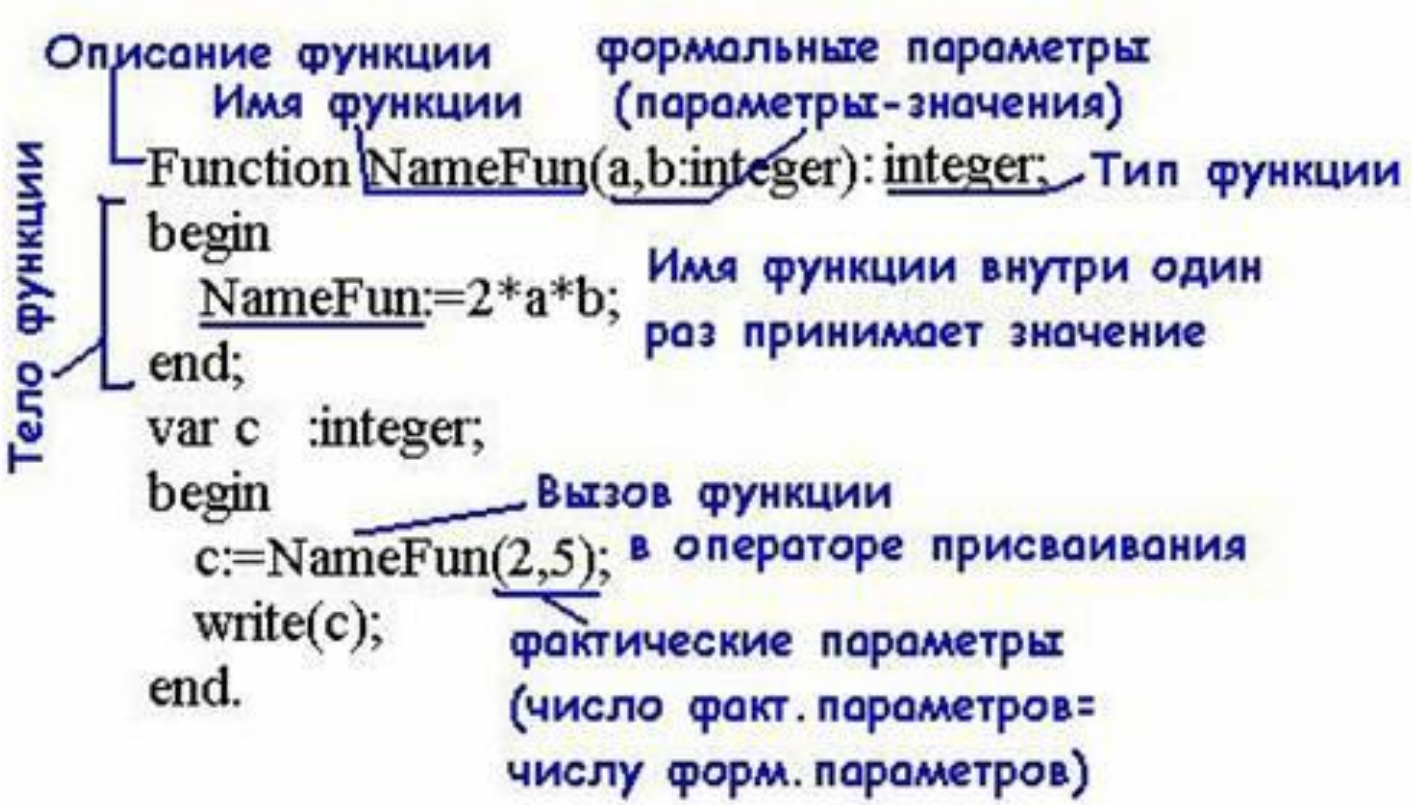
ФУНКЦИЯ — подпрограмма, в которой производятся некоторые действия, в результате которых имя функции получает одно единственное значение, присваиваемое ее имени и обязательно передаваемое в вызывающую программу, возвращая управление в точку вызова

```
Function <имя>(<список формальных параметров>):<тип результата>
 Описательная часть
 Begin
   Тело функции
  <имя>:=<значение>;
 End;
```

ПРОЦЕДУРА — подпрограмма, в которой производятся некоторые действия при этом имя процедуры не получает никакого значения, результатов у процедуры либо нет , либо результаты передаются в вызывающую программу специальным образом.

```
Procedure <имя>(<список формальных параметров>)
 Описательная часть
 Begin
   Тело функции
End;
```

Подпрограмма -функция



Вычислить значение выражения a:=(3n!+2m!)/(m+n)!

```
program fn2;
Uses crt;
Var m,n:integer; a:real;
function fact(d:integer) :longint;
  var i:integer; q:longint;
   begin
     q:=1;
     for i:=1 to d do
      q:=q*i;
     fact:=q;
    end;
Begin
clrscr;
writeln('введите значения n, m '); readln(n,m);
 a:=(3*fact(n)+2*fact(m))/fact(m+n);;
writeln('значение выражения при  m= ',m:4,' и  n= ',n:4,'равно',a:8:3);
readln; end.
```

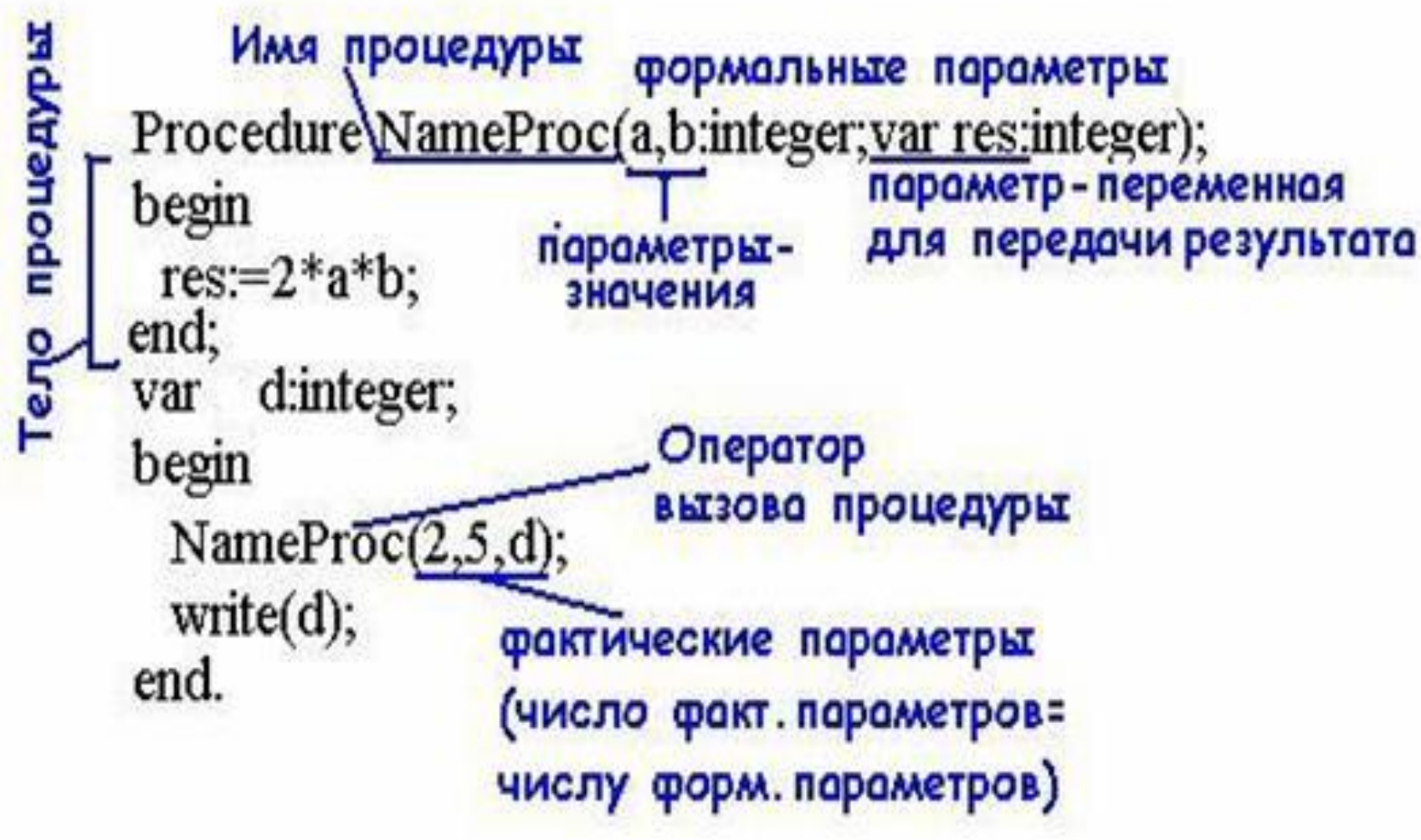
Найти разность средних арифметических значений двух вещественных массивов из 10 элементов.

```
program dif_average1;
const n = 3;
 type  mas = array[1 .. n] of real;
 var a, b : mas;  i : integer;   dif : real;
  function average(x : mas) : real;
     var i : integer;   av : real;
  begin
    av := 0;
       for i := 1 to n do
       av := av + x[i];
    average := av / n;
  end;
Begin
  for i := 1 to n do read(a[i]);
  for i := 1 to n do read(b[i]);
  dif := average(a) - average(b);
  writeln('Разность значений ', dif:6:2)
end.
```

В заголовке подпрограммы нельзя вводить новый тип

Function av(x:array[1..10] of real): real;

Подпрограмма -процедура

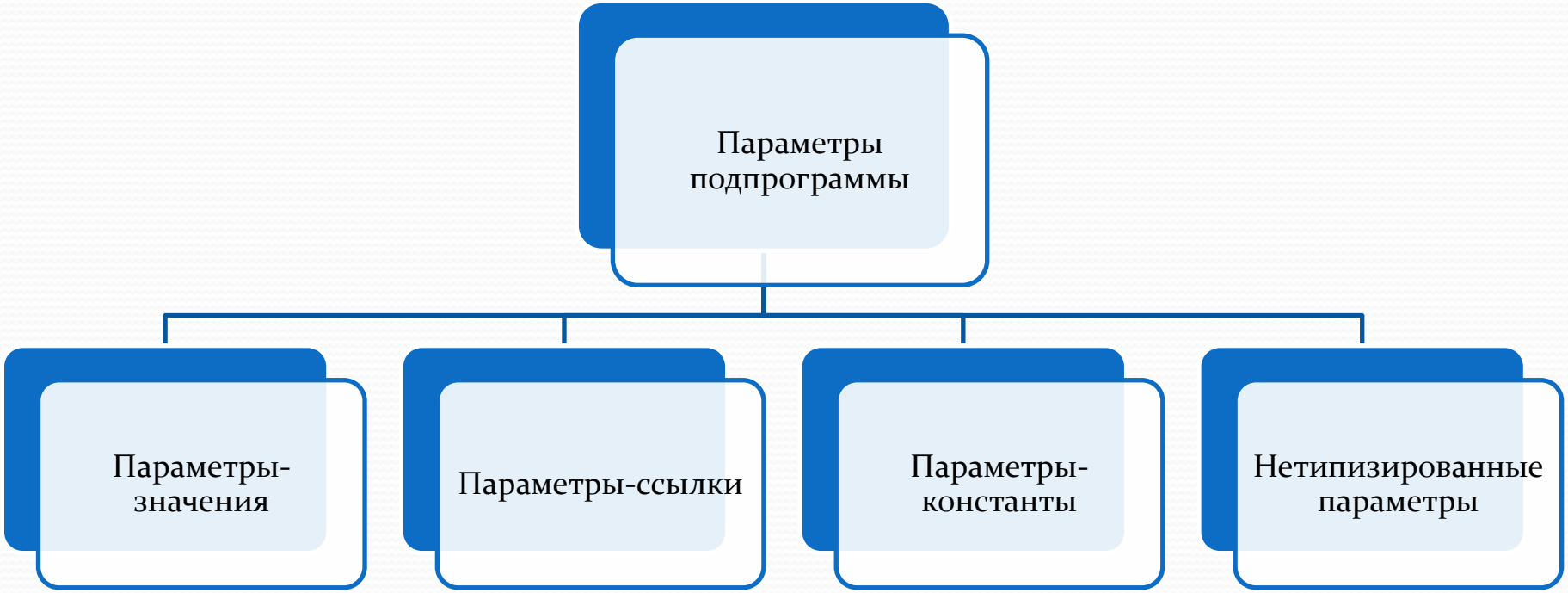


Вывести таблицу умножения

```
program pif_table;
   procedure Pifagor(n: integer);
      var i,j:integer;
    begin
       writeln;
          for i:=1 to n do
           begin
             for j:=1 to n do
              write(i*j:4);
           writeln;    writeln;
         end;
     end;
var m:integer;
      begin
           write ('введите размер таблицы Пифагора ');
       readln(m);
       Pifagor(m);
end.
```

Сравнение процедуры и функции	
Процедура	***Функция***
Имя не имеет типа	При описании указывается тип имени функции, т.е. тип возвращаемого функцией значения
Имя никакого значения не получает	Имени обязательно присваивается некоторое значение
Вызывается отдельно, оператором вызова процедуры с указанием имени и значений фактических параметров	Как и стандартные встроенные функции может быть вызвана в операторе присваивания, операторе вывода
Может не возвращать в основную программу ни одного значения	Обязательно возвращает в основную программу хотя бы одно значение (присвоено имени функции)

Механизм параметров



Карта распределения оперативной памяти

Структура страницы стека

5	Возвращаемое значение
4	Параметры
3	Адрес возврата
2	Локальные переменные
1	Указатели связи
	Глобальные переменные
	Библиотечные функции
	Код программы
	Префикс программного сегмента

Параметры значения

procedure P(x : integer);

Тип выражения должен быть совместим по присваиванию с типом параметра.

Если в вызывающей программе описаны переменные

var x : integer; c : byte; y : longint;

то следующие вызовы подпрограммы P будут синтаксически правильными:

P(x);

P(c);

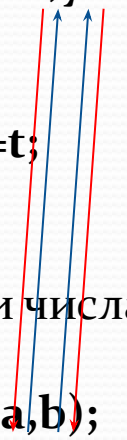
P(y);

P(200);

P(x div 4 + 1);

Параметры-переменные

```
program perest;
    var a,b,c: integer;
     procedure swap(var x,y: integer);
        var t: integer;
                begin
                t:=x; x:=y; y:=t;
               end;
begin
     writeln('Введите три числа ');
       readln(a,b,c);
        if a>b then swap(a,b);
        if b>c then swap(b,c);
       if a>c then swap(a,c);
      writeln(a,' ',b,' ',c);
   readln;  end.
```



5	2	1
2	5	1
2	1	5
1	2	5

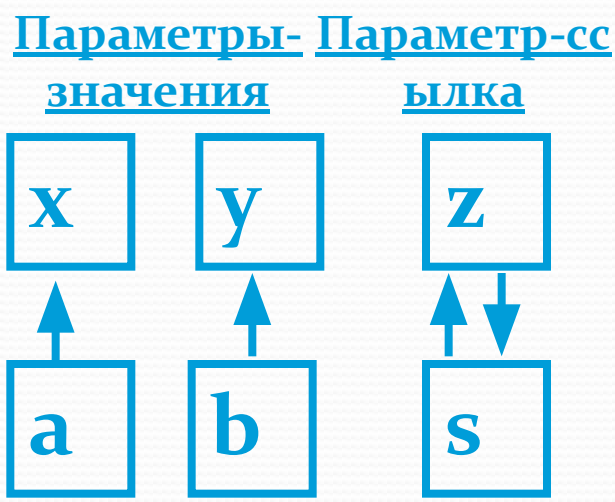
Процедура вычисления суммы двух чисел

```
program pr1;
Uses crt;
Var a,b,s:real;
```

a,b,s – глобальные переменные

x,y,z – формальные параметры, локальные переменные

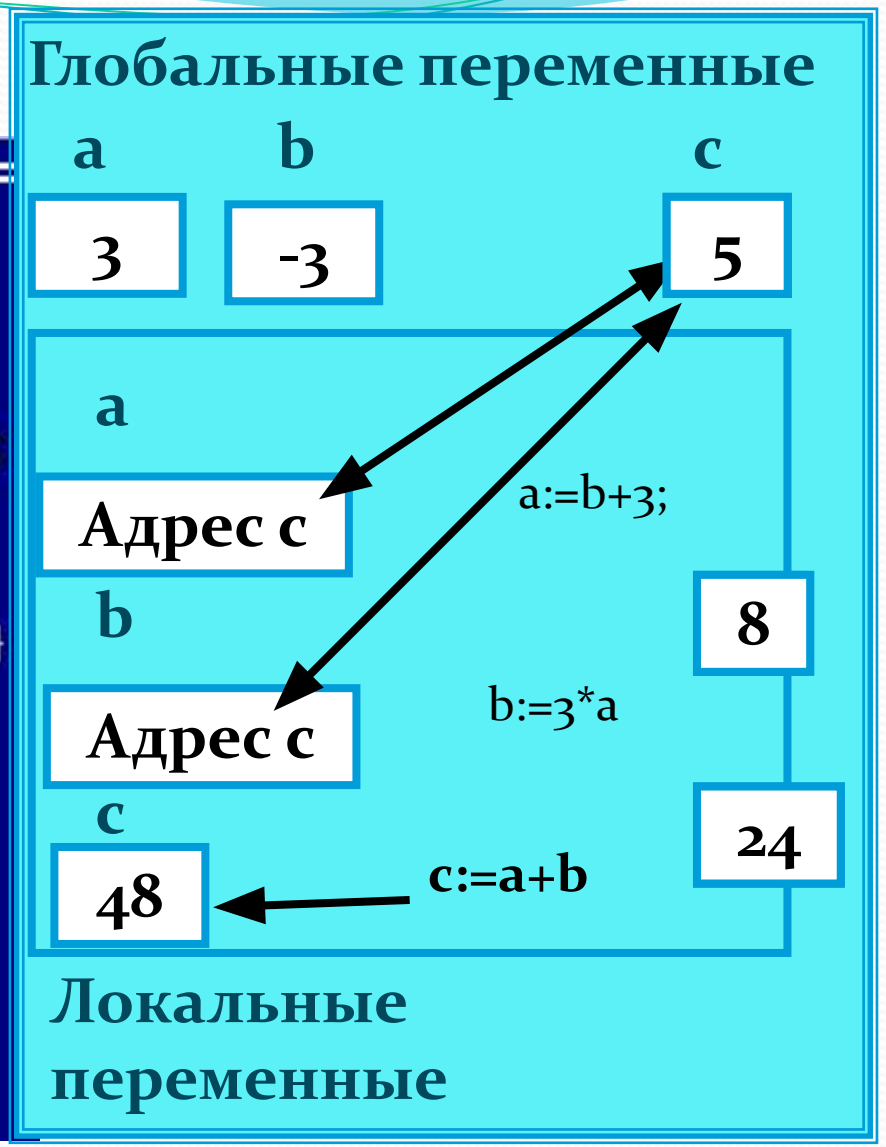
```
procedure summa(x,y:real;var z:real);
begin
z:=x+y;
end;
begin
clrscr;
writeln('введите a,b'); readln(a,b);
summa(a,b,s);
writeln(' сумма чисел  ',a:3:1,' и ',b:3:1,' = ',s:3:1);
readln; end.
```



a,b,s – фактические параметры

Параметры переменные

```
program proc4;
var
  a:integer;b,c:integer;
procedure cc(var a,b:integer);
  var c:integer;
  begin
  a:=b+3; b:=3*a; c:=a+b;   {1}
  writeln(c)
  end;
begin
  a:=3; b:=-3; c:=5;
  cc(c,c); {2}
  writeln(c)
end.
```



Ответ

```
48
24
```

Пример локализации переменных

```
program fun3;
var a,b,c,d:integer;

function f(var b:integer;c:integer): integer;
var a,d:integer;
begin
a:=2; b:=b+1; d:=3; c:=b-a;
writeln(a:3,b:3,c:3,d:3);
end;

begin
a:=0;b:=0;c:=0;d:=0;
d:=f(a,b);
writeln( a:3,b:3,c:3,d:3);
end._
```

Ответ

```
  2   1  -1   3
  1   0   0   0
```

```
PROC3.PAS
program fun3;
var
  a,b,c,d:integer;
  function f(var b:integer; c:integer):integer;
  var d:integer;
  begin
  a:=2; b:=b+1; d:=3; c:=b-a;
  writeln(a:3,b:3,c:3,d:3); f:=d
  end;
begin
  a:=0; b:=0; c:=0; d:=0;
  d:=f(a,b);
  writeln(a:3,b:3,c:3,d:3)
end.
```

Ответ

```
  3   3   0   3
  3   0   0   3
```

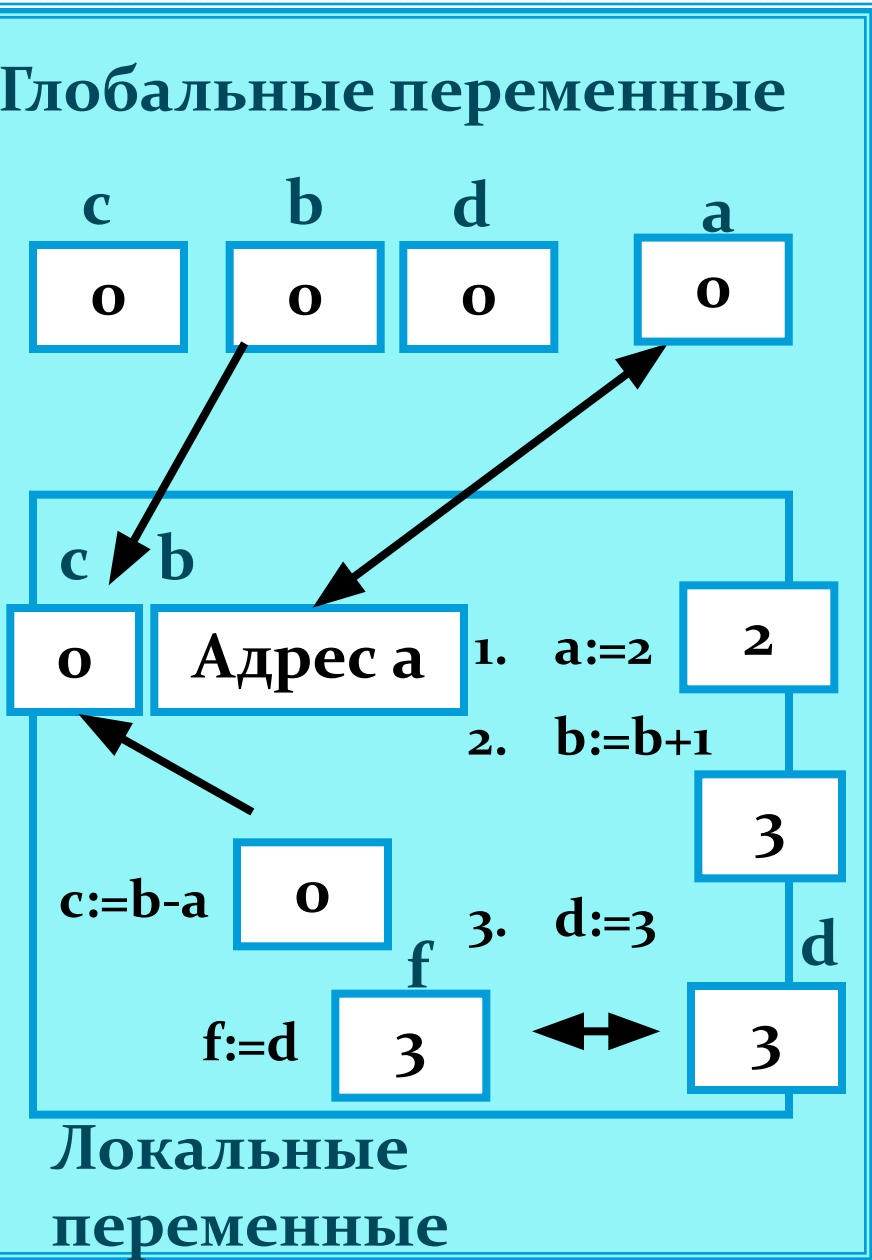
Механизм передачи параметров в функции и процедуры

PROC3.PAS

```
program fun3;
var
  a,b,c,d:integer;
  function f(var b:integer; c:integer):inte
  var d:integer;
  begin
  a:=2; b:=b+1; d:=3; c:=b-a;
  writeln(a:3,b:3,c:3,d:3); f:=d
  end;
begin
  a:=0; b:=0; c:=0; d:=0;
  d:=f(a,b);
  writeln(a:3,b:3,c:3,d:3)
end.
```

Ответ

```
  3   3   0   3
  3   0   0   3
```



Ввод и вывод элементов одномерного массива

PROC2.PAS

```
program proc2;
const n=8; l=-10; h=20;
type mas=array[1..n] of integer;
var a: mas;
procedure init(t,v,w:integer;var x:mas);
    var i: integer;
    begin
    randomize;
    for i:=1 to t do x[i]:=v+random(w-v+1);
    end;
 procedure print(t:integer;var x:mas);
    var i: integer;
    begin
    for i:=1 to t do write(x[i]:5);
    end;
begin
writeln('формирование значений
init(n,l,h,a);
writeln('Вывод ');
print(n,a);
readln;          end.
```

```
формирование значений элементов массива A
Вывод
   10   17   11   -2    2   -6   -8   -2
формирование значений элементов массива A
Вывод
   12   -6  -10   20   20   20   -1    3
формирование значений элементов массива A
Вывод
    8   -4   -9   -5    7   16   10    6
```

Разбить строку *S* на слова с учетом заданных разделителей

```
Type  TWords = Array[1 .. 100] Of String[100];
TDel = Set Of Char;
Var   i, count: BYTE;  words: TWords;
 DEL:Tdel;   S:string;
Procedure vvodvivo
  Begin
  If flag then Readln(
 End;
Procedure del32(var
Var p:byte;
begin
   Repeat
         p := Pos('  ', s);
      If p > 0 Then Delete(s, p
  Until  p = 0;

  If s[1] = ' ' Then Delete(s, 1, 1);
If s[Length(s)] = ' ' Then  Delete
end;
```

```
Function GetWords(s: String; Var mas: TWords;
                  dels: TDel): B
 Var i, p: Byte;
 Begin
For i:= 1 to Length(s) Do
If s[i] In dels Then s[i] := #32;
Del32(s);
                                    elete(s, 1, p); End
                                    Vvodvivod(s,true);
                                    Del:=[];
                                    For I:=1 to length(s) do
                                      :=del+s[i];
                                    nt := GetWords(s, words, del);
                                    or i := 1 To Count Do
                                          vvodvivod(words[i],false);
End.
```

Free Pascal

```
введите разделители
,.-
 -это предложение, которое делим на слова.
это
предложение
которое
делим
на
слова
```

Заменяем все разделители пробелами

заполняем массив словами из строки

Удаляем пробелы в начале строки

Удаляем пробелы в конце строки

формирование множества разделителей

Параметры-константы

```
Type matr=array[1..20,1..30] of byte;
  Procedure vivod( n,m: byte; const a:matr);
     Var   i,j: byte;
          Begin
              For i:=1 to n  do
                 begin
                   For j:=1 to m do
                    Write(a[i.j]:5);
                   Writeln;
                 End;
            End;
Var   i,j: byte;
M:matr;
Begin
      Randomize;
            For i:=1 to 10 do
             For j:=1 to 15 do
             m[i,j]:=random(20); // 0<=m[i,j]<20
    vivod( 10,15,m);
end.
```

Нетипизированные параметры

```
Function EQ  (const x, y; size : word) : boolean;
type mas_byte = array[0 .. MaxInt] of byte;
//mas_byte = array[0 .. MaxInt] of byte absolute x;
var   n : integer;
begin
n := 0;
 while (n < size) and (mas_byte(x)[n] = mas_byte(y)[n]) do
// while (n < size) and (mas_byte[n] = mas_byte(y)[n])
inc(n);
EQ := n = size;
End;
```

Наложение переменной

Автоопределенное преобразование

С помощью функции EQ можно сравнить две любые величины.

```
var a, b : array [1 .. 10] of integer;

EQ(a, b, sizeof(a)) ; { сравнение двух массивов }
EQ(a[2], b[5], 4)  ;  { сравнение 2-5 элементов массива "a" с 5-8
                        элементами  массива "b", соответственно }
```

Открытые массивы

```
function sum(var x : array of real) : real;
      var i:word; s:real;
  begin
    s:=0;
     for i:=Low(x) To High(x) Do s:=s+x[I];
    sum:=s;
  end;
const a:array [1..5] of real=(1,2,3,4,5.5);
begin
 writeln (sum(a):6:1);
end.
```

Найти количество элементов вектора x, попадающих в интервал [a, b] .

```
procedure Input (n:integer;
 var a:array of real);
var i:integer;
begin
 writeln ('Enter ',n,' items of array:');
 for i:=0 to n-1 do read (a[i]);
end;

function Kol (var a:array of real;
                    x1,x2:real):integer;
var i,k:integer;
begin
 k:=0;
 for i:=Low(a) to High(a) do
  if (a[i]>=x1) and (a[i]<=x2) then k:=k+1;
 Kol:=k;
End;
```

```
var x:array [1..7] of real;
    y:array [1..5] of real;
    k1,k2,i:integer;


 begin
  Input (7,x);
  Input (5,y);
  k1:=Kol(x,0,3);
  k2:=Kol(y,-1,1);
  writeln ('k1=',k1,' k2=',k2);
 end.
```

ПРОЦЕДУРНЫЕ ТИПЫ.

Для объявления процедурного типа используется заголовок процедуры (функции), в котором опускается ее имя

```
type
  Proc1 = Procedure (a, b, c: real; var d: real);
  Proc2 = Procedure (var a, b);
  Proc3 = Procedure;
  Func1 = Function: String;
  Func2 = Function (var s: String): real;
```

Вывести на экран таблицу двух функций: sin1(x) = (sin(x)+1)*exp(-x) и cos1(x) = (cos(x)+1)*exp(-x).

```
Uses CRT;
type
  Func = Function (x: real): real;
Procedure PrintFunc(XPos: byte; F: Func);
const
  np = 20;
var
  x: real;
  i: integer;
begin
  for i := 1 to np do
    begin
      x := i * (2 * pi / np);
      GotoXY (XPos, WhereY)
      WriteLn (x:5:3, F(x):18:5
  end;
end;
```

```
Function Sin1(x: real): real; far;
begin
  sin1 := (sin(x) + 1) * exp(-x)
```

Free Pascal

0.314	0.95611	0.314	1.42506
0.628	0.84706	0.628	0.96509
0.942	0.70490	0.942	0.61870
1.257	0.55529	1.257	0.37256
1.571	0.41576	1.571	0.20788
1.885	0.29624	1.885	0.10492
2.199	0.20062	2.199	0.04572
2.513	0.12861	2.513	0.01547
2.827	0.07745	2.827	0.00290
3.142	0.04321	3.142	0.00000
3.456	0.02181	3.456	0.00154
3.770	0.00950	3.770	0.00440
4.084	0.00322	4.084	0.00694
4.398	0.00060	4.398	0.00850
4.712	0.00000	4.712	0.00898
5.027	0.00032	5.027	0.00859
5.341	0.00092	5.341	0.00761
5.655	0.00144	5.655	0.00633
5.969	0.00177	5.969	0.00499
6.283	0.00187	6.283	0.00373

Переменным процедурных типов допускается присваивать в качестве значений имена соответствующих подпрограмм

```
type
  Proc = Procedure (n: word; var a: byte);
var
  ProcVar: Proc;
  x, y: byte;
Procedure Proc1(x: word; var y: byte); far;
begin
  if x > 255 then
    y := x mod 255
  else
    y := byte(x)
  end;
begin    {Главная программа}
  ProcVar := @Proc1;
  for x := 150 to 180 do
    begin
      ProcVar (x + 100, y);
      Write (y:8)
    end
end.
```

```
type
  FuncType = Function (i: integer): integer;
var
  VarFunc: FuncType;
  i: integer;
Function MyFunc (count: integer): integer; far;
begin
  . . . . .
end;    {MyFunc}
begin    {Основная программа}
  . . . . .
  i := MyFunc(1);
{использование результата функции}
  . . . . .
  VarFunc := @MyFunc;
  {Присваивание переменной процедурного типа имени функции MyFunc}
end.
```

Предварительные и внешние описания подпрограмм

```
Procedure A(X,Y: Real): Forward;

Procedure B(A,B: Integer): Forward;
    . . . . . .

Procedure A;
    Begin
    . . . . . .
       B(3,5);
 . . . . . .
       End;

Procedure B;
    Begin
       . . . . . .
       A(1,0);
    . . . . . .
    End;
```

Внешнее описание

```
{$L ABC.OBJ}
   Procedure A(C,D,E: Real); External;
   Procedure B(I,F,J: Integer); External
```