ПРОЦЕДУРЫ И ФУНКЦИИ

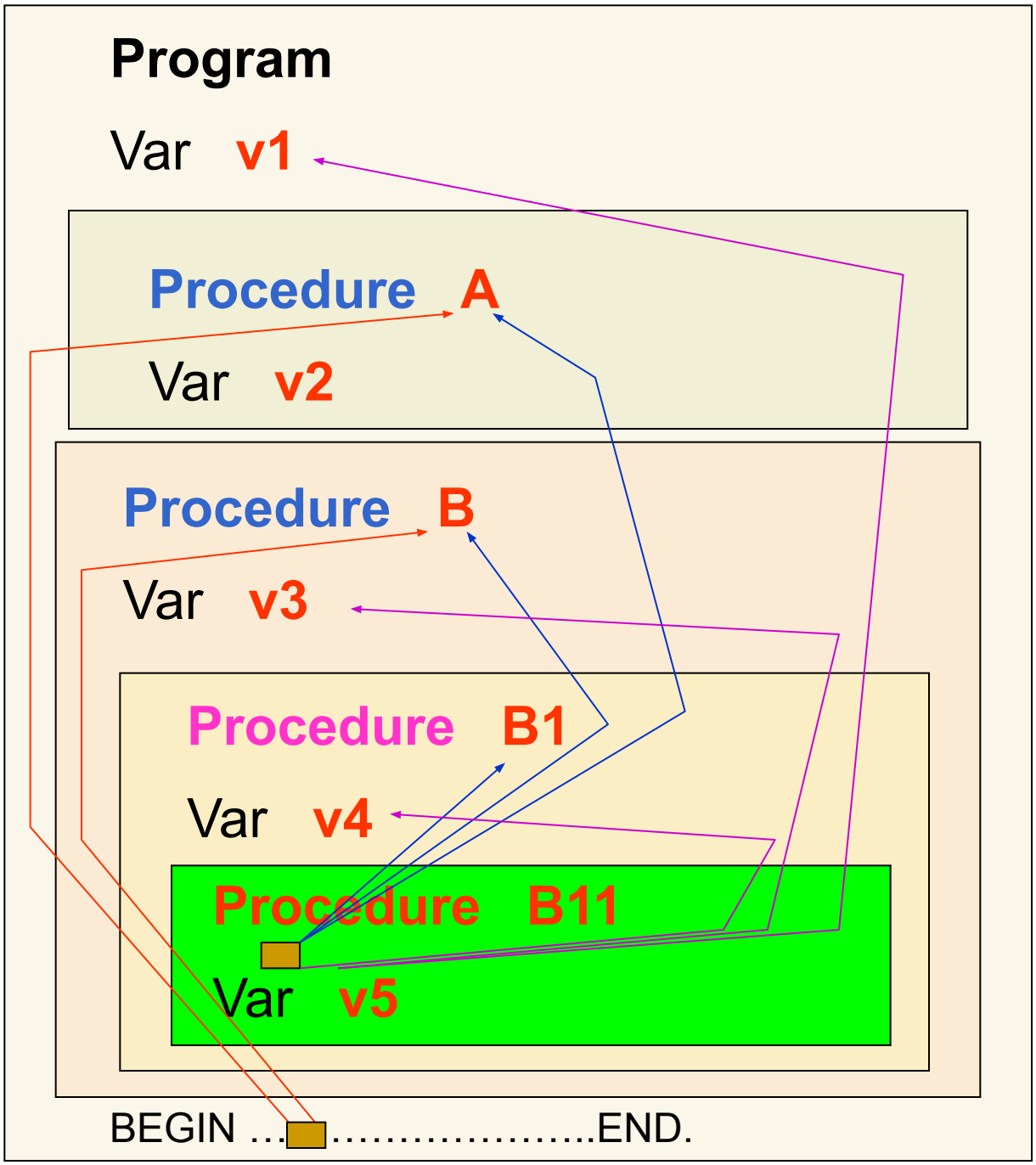
- При разработке алгоритма программы часто возникает ситуация, когда отдельные ее фрагменты неоднократно повторяются в различных ее частях. Эти фрагменты полностью идентичны с точки зрения написания операторов и различаются лишь фактическими значениями участвующих в вычислениях данных.
- Следовательно, эти фрагменты программы являются логически обособленными и их можно оформить в виде особых относительно самостоятельных программных единиц, обладающих собственным именем, – процедур или функций

- Использование процедур и функций позволяет сделать программу более компактной и понятной с точки зрения визуального восприятия текста и уменьшить область памяти, занимаемою программой. Упоминание в тексте программы имени процедуры или функции называется вызовом процедуры (функции).

Локализация имен

- Вызов блока происходит при обращении к нему по имени.
- Любое имя в Паскале должно быть предварительно описано. Следовательно, все, используемые в программе, процедуры и функции должны быть описаны в разделе описаний.
- Описать блок – это значит указать его имя и тело. В заголовке указывается имя блока и приводится список его формальных параметров. Для функции еще указывается ее тип, т.е. тип возвращаемого параметра.

Локальные и глобальные переменные



- Все имена, описанные внутри блока, локализуются в нем, т.е. невидимы снаружи. В приведенном примере из основной программы можно обратиться к блокам ***A, B,*** но нельзя вызвать какую либо из процедур, находящихся внутри них (***B1***). Это относится не только к именам блоков, но и к любым именам, объявленным в них.
- При входе в блок нижнего уровня становятся доступными не только имена, локализованные в нем, и имена верхних уровней. Из блока ***B11*** можно обратиться к блокам ***A, B, B1*** и использовать имена, объявленные в них и в основной программе – ***V1,V3,V4***.

- Program.....
- Var V1 .....
- Procedure A;
- Var V2 .....
- .....
- End {A}
- Procedure B;
- Var V3 .....
- Procedure B1;
- Var V4 .....
- Procedure B11;
- Var V5 .....
- .....
- Из блока ***B11*** доступны четыре переменные V1, V3, V4, V5.
- Из блока ***B1*** доступны - V1, V3, V4.
- Из блока ***B*** доступны - V1, V3.
- Из блока ***A*** доступны - V1, V2.
- Из основной программы – только V1.

- Имена, локализованные в блоке, могут совпадать с глобальными именами. В этом случае, **локальное имя «накрывает» глобальное имя внутри блока и делает его недоступным.**

```
Var
      i:integer;
      Procedure  P;
        Var
          i:integer;
          begin
            writeln(i);
          end {P};
    BEGIN
       i:=1;
       P;
    END.
```

- Результат действия этой программы непредсказуем, т.к. локальное имя ***«i»*** накрывает глобальное ***«i»*** и в самой процедуре ***P*** оно только объявлено, но ему не присвоено никакого значения.
- Если убрать из процедуры ***P*** описание

  i:integer;
- то программа выведет на экран значение глобальной переменной ***i=1***.
- Одноименные локальные и глобальные переменные – это разные переменные.

Синтаксис написания заголовка блока следующий –

- ***Procedure <имя> [(<список формальных параметров>)]***
- ***Function <имя> [(<список формальных параметров>)]: <тип>***

Список формальных параметров может быть опущен. Если же формальные параметры присутствуют, то должен быть указан их тип.

- ***Function ABC(a:integer; b:real): real;***

или

- ***Function ABC1(a, b:real): real;***

- В пределах блока список формальных параметров является как бы расширением раздела описаний данного блока.

- Допустим, что в основной программе мы дважды обращаемся из разных мест к функции

  ABC(a:integer; b:real): real.

  Вызовы имеют вид

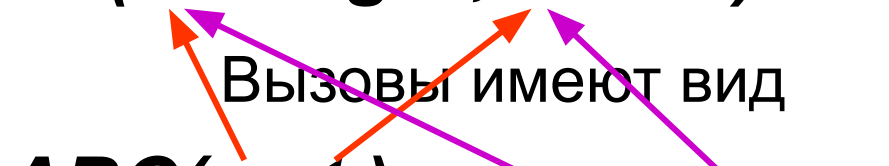
  ABC(s,t) и

  ABC(u,r) .

  Это означает, что при первом обращении формальные параметры ***a, b*** замещаются фактическими переменными ***s, t***, и при втором обращении - ***u, r.***

  Это замещение происходит в заголовке функции момент обращения к ней.

ABC(a:integer; b:real): real.



Вызовы имеют вид

ABC(s, t) и

ABC(u, r)

- Рассмотрим операцию возведения в степень

$$x^y$$

$$c = x^y \quad \ln(c) = y * \ln(x)$$

$$c = \exp(y * \ln(x))$$

Function Power(a,b:real):real

```
Program PowerDen
  Var
    x,y:real;
  {----------------}
  Function Power(a,b:real):real;
     Begin {Power}
                Power:= exp(b*ln(a));
     End {Power};
   {----------------}
BEGIN {main}
   readln(x,y);
   writelh('Power=', Power(x,y));
END.
```

Procedure Power1(var a,b,c:real);

```
Program PowerDen1;
  Var
    x,y,s:real;
  {----------------}
  Procedure Power1(var  a,b,c:real);
     Begin {Power1}
              c:= exp(b*ln(a));
     End {Power1};
   {----------------}
BEGIN {main}
   readln(x,y);
   Power1(x,y,s);
   Writeln('Power=', s);
END.
```

Procedure Power2(var a,b,c:real);

```
Program PowerDen2;
  Var
    x,y,c:real; {c – глобальная переменная}
  {----------------}
  Procedure Power2( a,b:real);
      Begin {Power1}
                c:= exp(b*ln(a));
      End {Power1};
   {----------------}
BEGIN {main}
   readln(x,y);
   Power2(x,y);
   Writeln('Power=', c);
END.
```

- При работе с функциями результат вычислений внутри самой функции присваивается идентификатору функции. В данном примере - это ***Power.***
- В варианте с использованием процедуры результат передается в основную программу через список переменных.
- Третий вариант вычислений – переменную ***«c»*** объявить в основной программе и использовать в блоке как глобальную переменную.
- Списки формальных (в блоке) и фактических (в вызове) параметров должны строго соответствовать друг другу по своему типу и порядку следования.

Параметр-переменная и параметр-значение.

- Существует два способа задания параметров в блоке: параметр-переменная и параметр-значение.
- ***Procedure P1(a,b,c : real);***
- ***Procedure P2(var a,b : real; c : real);***
- В первом случае все параметры объявлены как параметры-значения, а во втором - ***a,b*** объявлены как параметры-переменные.
- Вариант объявления переменной существенен только для вызывающей программы.

- Если формальный параметр объявлен как параметр-переменная, то при вызове ему должен соответствовать фактический параметр в виде переменой того же типа.
- Если формальный параметр объявлен как параметр-значение, то при вызове ему может соответствовать произвольное выражение.
- Разница в этих двух вариантах объявления параметров заключается в механизме замещения формальных параметров фактическими параметрами.

- Если формальный параметр объявлен как параметр-значение, то перед вызовом блока это значение вычисляется, полученный результат помещается во временную память и передается блоку. Даже если в качестве фактического параметра указана просто переменная или константа, в блок передается только ее копия.
- Если формальный параметр объявлен как параметр-переменная, то при вызове блока передается сама переменная, а не копия.
- Любые возможные изменения параметра-значения в блоке никак не скажутся на вызывающей программе, а при использовании параметра-переменной эти изменения передадутся в вызывающую программу, т.к. приводят к изменению фактической переменной вызывающей программы.

Function Power(a,b:real):real

```
Program PowerDen
  Var
    x,y,z:real;
  {----------------}
 Function Power(var a,b,c:real):real;
     Begin {Power}
                Power:= exp(b*ln(a));
                c:=a+10;
     End {Power};
   {----------------}
BEGIN {main}
   readln(x,y);
   writelh('Power=', Power(x,y,z));
END.
```

```
Program Proba;
……….
    a:=5;
    b:=7;
 {-----------}
 Procedure Pr1(var a : integer; b : integer)
     begin{Pr1}
      a:=a+a;
      b:=b+b;
      writeln('удвоение', a:5, b:5);
    end{Pr1};
  {-----------}
BEGIN
  Writeln('исходное', a:5, b:5);
  Pr1(a,b);
  writeln('результат', a:5, b:5);
END.
```

Исходное 5 7

Удвоение 10 14

Результат 10 7

- Параметры-переменные используются для передачи данных из блока в вызывающую программу.
- Другой способ передачи данных – через глобальные переменные. Но при этом нужно быть очень внимательным, чтобы случайно не испортить фактические переменные в вызывающей программе. Некоторым недостатком является невозможность использовать в вызове блока в качестве фактического параметра какое-либо выражение.
- При использовании параметров-значений следует избегать в объявлении больших массивов данных, т.к. в этом случае нерационально используется память и время на создание их копии.

Параметры-массивы и параметры-строки.

- Существует различие в объявлении формальных параметров блока в заголовочном списке и в разделе описаний блока. Через список можно объявлять только стандартный или ранее объявленный тип. Объявление

Procedure SS(a:array [1..10] of real);

неверно!

- Следует сначала объявить переменную ***Mas*** как массив

```
Type
  Mas=array [1..10] of real;
```

- затем в списке формальных переменных объявить параметр ***a*** как тип ***Mas.***
- ***Procedure SS(a : Mas);***

- При использовании в качестве формального параметра символьной строки следует сначала объявить тип строки, (поскольку строка является своего рода массивом символов).
- ***Type***
- ***intype = string [15];***
- ***outtype = string [30];***
- ***Function PP(s1 : intype) : outtype;***