Подпрограммы

В языке Pascal

При создании сложной программы возникает необходимость декомпозиции (разделении) ее на подзадачи.

Pascal имеет различные средства для деления программы на части:

- на верхнем уровне (больших задач) – это модули;
- на нижнем уровне (элементарных подзадач) – это процедуры и функции.

Все процедуры и функции языка Pascal делятся на две группы:

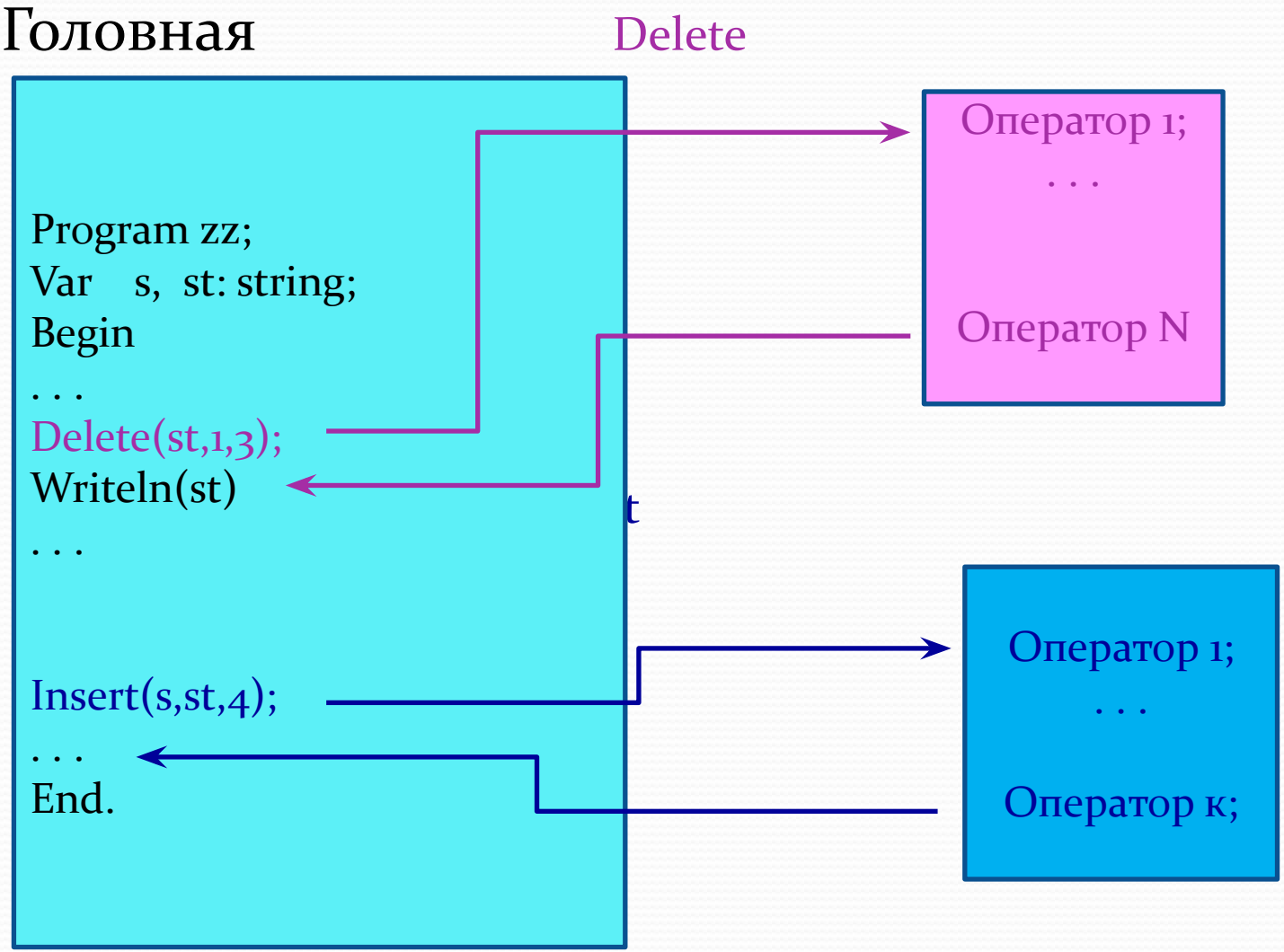
- **встроенные** (стандартные) - хранятся в стандартных библиотечных модулях ;
- **пользовательские** - описываются в разделе описаний головной программы.

Подпрограмма-процедура – независимая именованная часть программы, которую можно вызвать по имени для выполнения определенных действий.

Процедура не может выступать как операнд (данное) в выражении. Упоминание имени процедуры в тексте головной программы приводит к активизации процедуры и называется ее вызовом.

Например: Readln(x), Delete(St, 5, 2)

Вызов подпрограмм



Подпрограмма-функция

предназначена для вычисления какой-либо скалярной (простой) величины.

Имя функции может входить в выражение как операнд. В теле функции имени функции хотя бы раз должно быть присвоено значение, того же типа, что и результат функции.

Например: Ord('F'), Sqrt(a+b)

Описание подпрограммы-процедуры

```
1   Procedure Имя (Список формальных параметров);
                    {Заголовок процедуры}
2       Label
        Const       {раздел описаний локальных
                    (внутренних) данных}
        Type
        Var
        Procedure
        Function

3       Begin       {операторы процедуры}
        . . .
    End;
```

Описание подпрограммы-функции

{Заголовок функции}

1 **Function Имя** (Список формальных параметров) **: тип результата**;

2
Label
Const {раздел описаний локальных
Type (внутренних) данных}
Var
Procedure
Function

3
Begin {операторы функции}
…
Имя:= выражение;
End;

где

Procedure **,** ***Function*** – служебные слова;

Имя – имя пользовательской процедуры/функции;

Список формальных параметров – состоит из имен параметров с указанием типов параметров, которые перечисляются через «;». Если несколько параметров имеют один тип, то их можно сгруппировать, разделив имена запятой. Если в качестве параметра используется структурированный тип данных (массив, множество, запись или файл), то он должен быть описан в разделе описаний типов Type головной программы. Список формальных параметров может отсутствовать.

Например: type ff=array[1..5,1..10] of real;

Procedure Max(s:ff; k,n:integer);

Тип результата функции – любой простой (вещественный, целочисленный, логический, символьный) , строка символов или ранее определенный пользовательский.

Тип результата **не может быть :** массивом, множеством, записью или файлом.

Например: Function Factorial (x:byte) :real;

Function Perevod (x: longint) : string;

Результатом вычисления ***процедуры*** могут быть несколько величин, в том числе и структурированных типов (массив, множество, запись, файл или строка).

Результат присваивается параметрам-переменным. Перед такими параметрами в списке формальных параметров ставится служебное слово Var, действие которого распространяется до ближайшей точки с запятой.

Например:

```
Procedure Kol ( st:string;  var k1,k2:integer;  var  L:Boolean);
```

Выходные параметры

Область видимости переменных

Область действия переменной определяется местом ее объявления.

Если переменная используется только в рамках одной процедуры/функции, то она называется ***локальной***.

Если действие распространяется на несколько вложенных процедур/функций, то такая переменная называется ***глобальной***.

Локальные данные во внешнем окружении не действуют.

Выделение памяти под локальные данные происходит автоматически в начале выполнения подпрограммы, а освобождение – после выполнения последнего оператора подпрограммы.

Если имена глобальных и локальных идентификаторов совпадают, то действует только внутренний локальный идентификатор.

```
Program  primer;
Var  a,b,c: real;                    {глобальные}
Procedure A1  ();
Var  a1, b1, c1 : real;
  Procedure  A2 ();
  var  a2, b2, c2 : real;      {локальные}
  begin                        {операторы п /п A2}
  a, b, c, a1,b1, c1, a2,b2, c2
  end;
Begin                          {операторы п/п A1}
a, b, c, a1,b1, c1
End;
Begin    {операторы основной программы}
a, b, c
End.
```

Различают ***формальные параметры*** – параметры, определенные в заголовке подпрограммы, и ***фактические*** – выражения, задающие конкретные значения при обращении к подпрограмме.

В момент замены формальных параметров фактическими должны выполняться следующие условия:

1) количество формальных и фактических параметров должно быть одинаковым;
2) должен совпадать порядок следования параметров;
3) должны совпадать их типы.

Классификация способов передачи параметров (формальных)

1. по механизму передачи:

- передача по значению – value;
- передача по адресу (ссылке) – adr.

2. по взаимодействию вызываемой и вызывающей подпрограмм:

- только как входной параметр- in (input);
- только как выходной параметр – out (output);
- как входной и как выходной параметр – in/out (input|output).

Возможные способы передачи формальных параметров

- Value – in (параметры-значения)
- Value - out
- Value – in/out
- Adr – in (параметры-константы)
- Adr – out
- Adr – in/out (параметры-переменные)

Параметры-значения

- Перед параметрами-значениями (value-in) в списке формальных параметров дополнительных служебных слов не ставится.
- В качестве фактических параметров нельзя использовать файловый тип

Например:

```
Function  S (  a,b,c:real):real;
```

нет служебных слов

Параметры-значения

- В ячейки памяти формального параметра, выделенные при вызове подпрограммы, передается копия значения фактического параметра и обратно не возвращается.

Параметры-константы

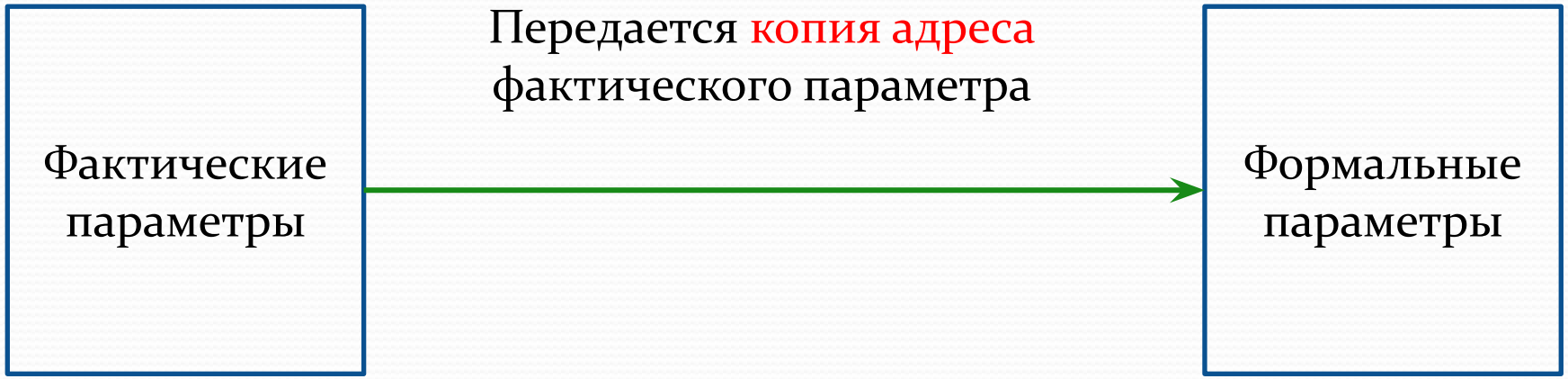
- Перед параметрами-константами (adr-in) в списке формальных параметров ставится служебное слово Const

Например:

```
Type  mas=array[1..100] of real;
Procedure  Pl(Const  a:mas;   n:integer);
```

- В ячейки памяти формального параметра, выделенные при вызове подпрограммы, передается копия адреса фактического параметра.
- В качестве фактических параметров нельзя использовать файловый тип данных.

Параметры-константы



По имеющемуся адресу разрешено только считывать значение фактического параметра, а изменять запрещено.

Параметры-переменные

- Перед параметрами-переменными (adr-inout) в списке формальных параметров ставится служебное слово Var.

Например:

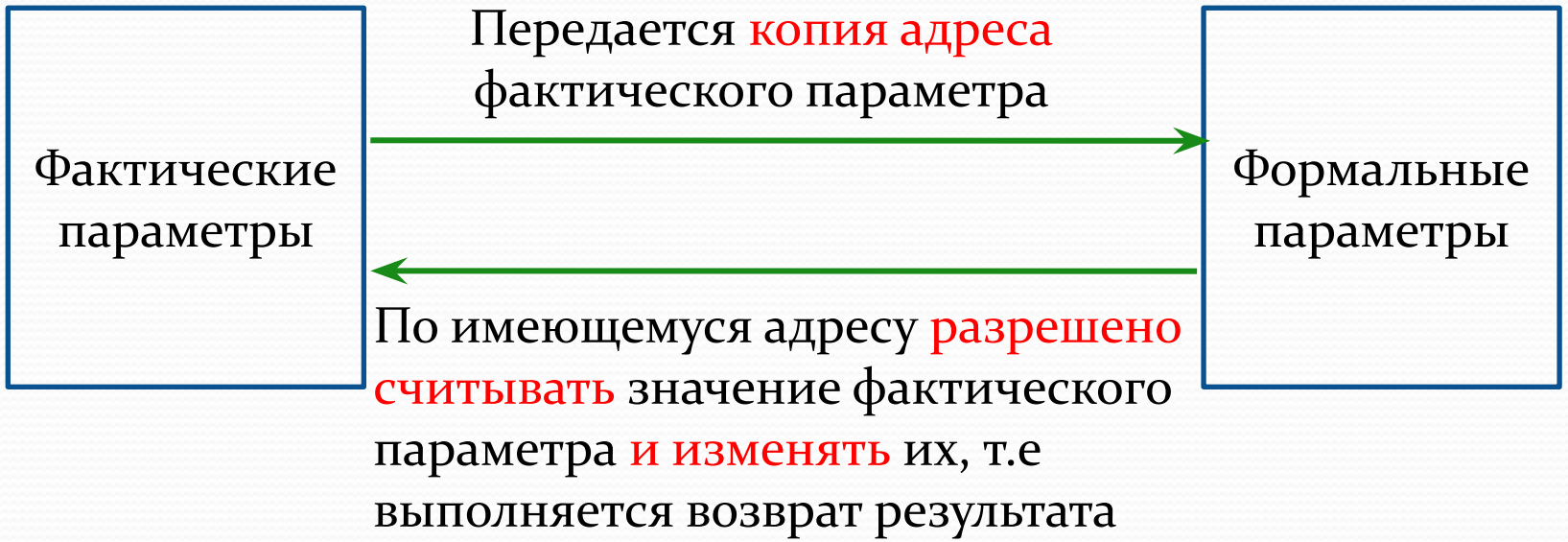
```
Type ff=file of real;
     massiv= array[1..20,1..30] of char;
 . . .
Procedure poisk ( Const   a:massiv;  n,k:integer;  Var  x:ff  );
```

параметры-константы — параметры-значения — параметры-переменные

Параметры-переменные

- В ячейки памяти формального параметра, выделенные при вызове подпрограммы, передается копия адреса фактического параметра
- По имеющемуся адресу разрешено считывать значение фактического параметра и изменять его. Изменение значений в ячейках памяти фактических параметров происходит во время выполнения операторов подпрограммы.
- В качестве фактических параметров можно использовать любой тип данных.

Параметры-переменные



Процедура Exit( ) – используется для досрочного выхода из подпрограммы

Например:Описать функцию, определяющую первое отрицательное число в массиве.

```
Type mas=array[1..100]  of real;
Function  minus ( Const b:mas;  n:integer) : real;
Var    i:integer;                    {описание локальных данных}
Begin
    minus:=0;                  {функции присваивается значение}
    For  i:=1 to n do
    If   b[i]<0  then  begin
                minus:=b[i];   {функции присваивается значение}
                Exit           {досрочное завершение функции}
                 end;
end;
```

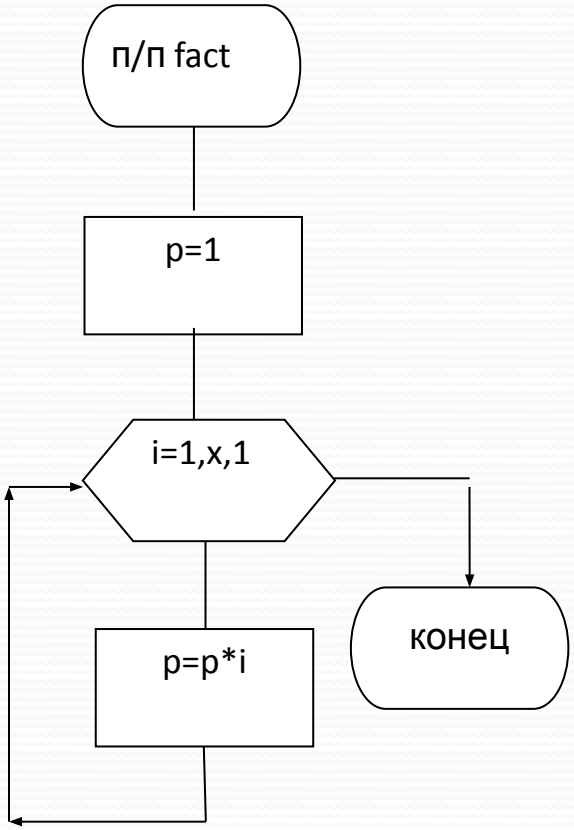
Задача:

Используя подпрограмму вычисления факториала вычислить биноминальный коэффициент для натуральных чисел n и m.

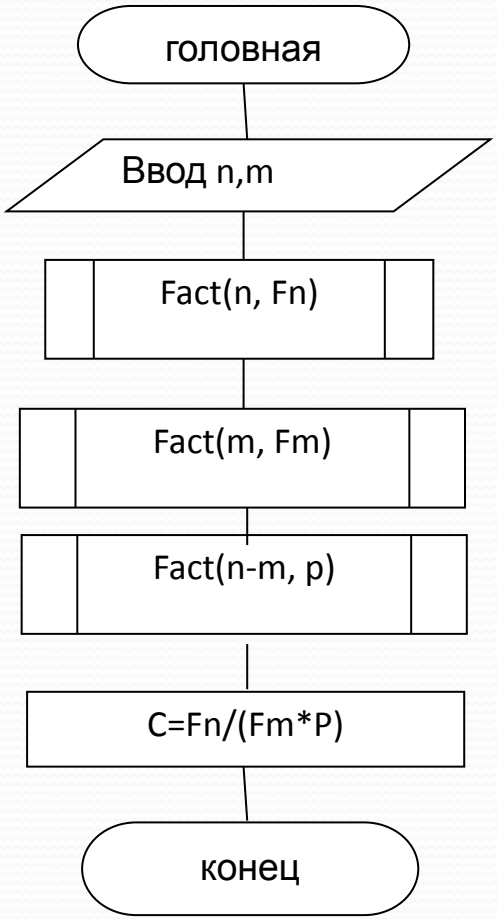
X!=1*2*. . .*X

С использованием подпрограммы-процедуры Fact

Алгоритм подпрограммы-процедуры



Алгоритм головной программы

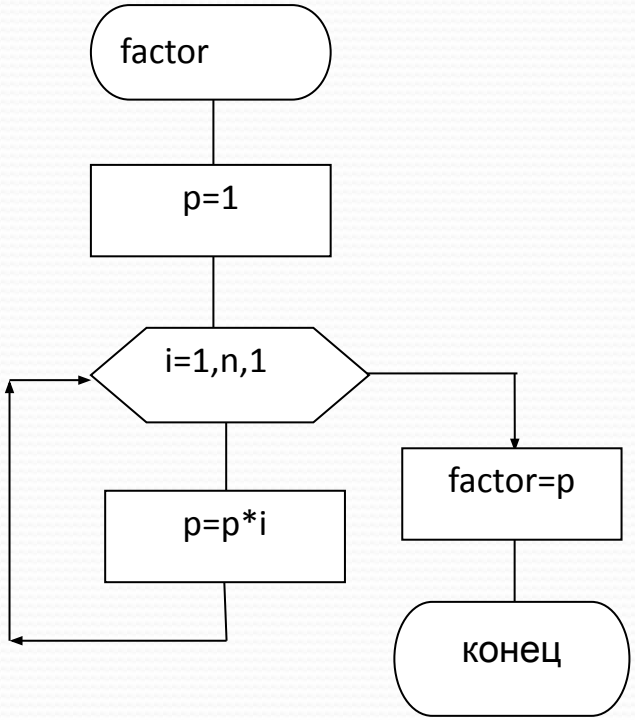


```
Program z1;
Var n,m:integer;
    C,P,Fn,Fm: real;
{процедура нахождения факториала числа x}
Procedure fact(x:integer; var p:real);
Var i:integer;              {локальные данные}
Begin
    P:=1;
    For i:=1 to x do
    P:=P*i                  {p-результат выполнения процедуры}
End;

{операторы головной программы}
Begin
Writeln('введите n, m');
Readln(n,m);
Fact(n,Fn);                 {Fn -факториал числа n}
Fact(m,Fm);                     {Fm- факториал числа m}
Fact(n-m,P);                { p- факториал числа n-m}
C:=Fn/(Fm*P);
Writeln('биноминальный коэффициент =', C:8:1)
End.
```

С использованием подпрограммы-функции Factor

Алгоритм
Подпрограммы-функции



Алгоритм
головной программы