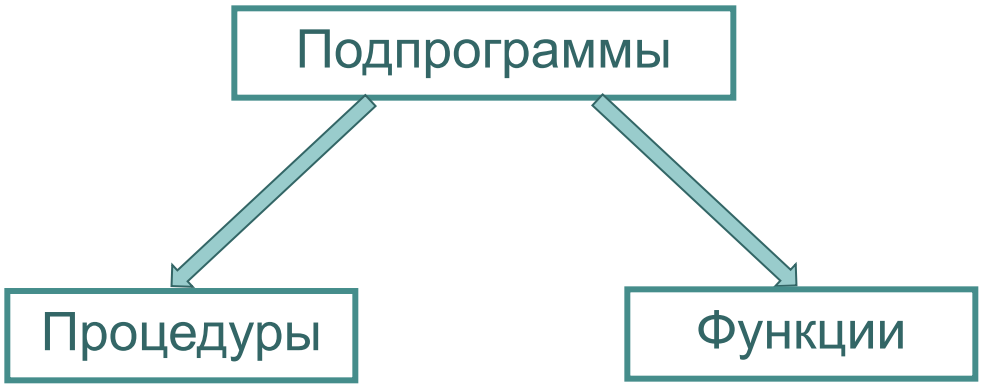
ПОДПРОГРАММЫ

Процедуры
и функции

Общие положения

Подпрограмма - именованная логически законченная группа операторов, которую можно вызвать по имени (выполнить) любое количество раз из различных мест программы.



Процедура – действие, которое вы просите компьютер выполнить.

Функция – некоторое вычисление, возвращающее значение.

Объявление процедуры

Формат объявления:

procedure <имя процедуры> (<список формальных параметров>); <директивы>;

<локальные объявления>
const ...;
type ...;
var ...;
<вложенные подпрограммы>
begin
<операторы>
end;

Объявление процедуры

Заголовок процедуры – имя подпрограммы, список формальных параметров, директивы.

<имя процедуры> - любой корректный идентификатор

<операторы> (**тело подпрограммы**) - операторы, которые будут выполнены при вызове подпрограммы

<список формальных параметров>, **<директивы>**, **<локальные объявления>** могут отсутствовать

Объявление функции

Формат объявления:

function <имя функции> (<список формальных параметров>) : <тип>; <директивы>;

<локальные объявления>
const ...;
type ...;
var ...;
<вложенные подпрограммы>
begin
<операторы>
Result := <значение>;
// или <имя функции> := <значение>;
end;

Объявление функции

Заголовок функции – имя подпрограммы, список формальных параметров, тип результата, директивы.

<имя функции> - любой корректный идентификатор

<операторы> (**тело подпрограммы**) - операторы, которые будут выполнены при обращении к функции

<список формальных параметров>, **<директивы>**, **<локальные объявления>** могут отсутствовать

Примеры

Процедура:

```
procedure Max2 (a,b: integer; var c: integer);
begin
    if a>b then c:=a else c:=b;
end;
```

Обращение в программе:

```
x:=32; y:=45; . . . ; Max2(x,y,z);
```

Параметры, передаваемые в подпрограмму при ее вызове, называются **фактическими**

В примерах **x, y, z – фактические параметры**

Примеры

Функция:

```
function Max2 (a,b: integer) : integer;
begin
      if a>b then Result:=a else Result:=b;
end;
```

Обращение в программе:

```
x:=32; y:=45; . . . ; z:=Max2(x,y);
```

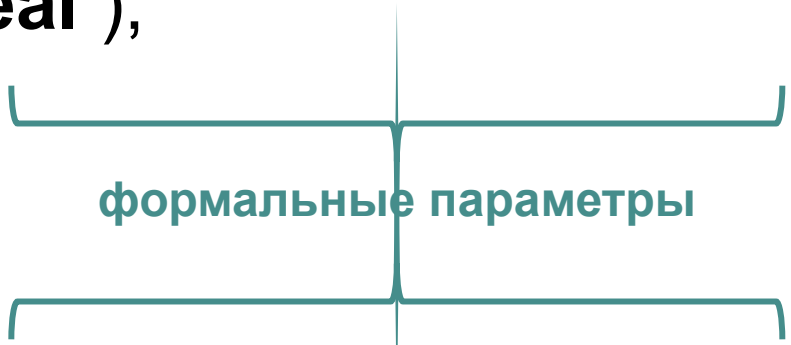
Формальные параметры

- **Список формальных параметров** – последовательность объявлений однотипных параметров
- Объявления **отделяются** друг от друга **точкой с запятой**.
- Каждое объявление состоит из **списка имен параметров**, **символа ‘:’** и указания их **типа**.
- Внутри объявления **имена параметров** разделяются **запятой**.

Формальные параметры

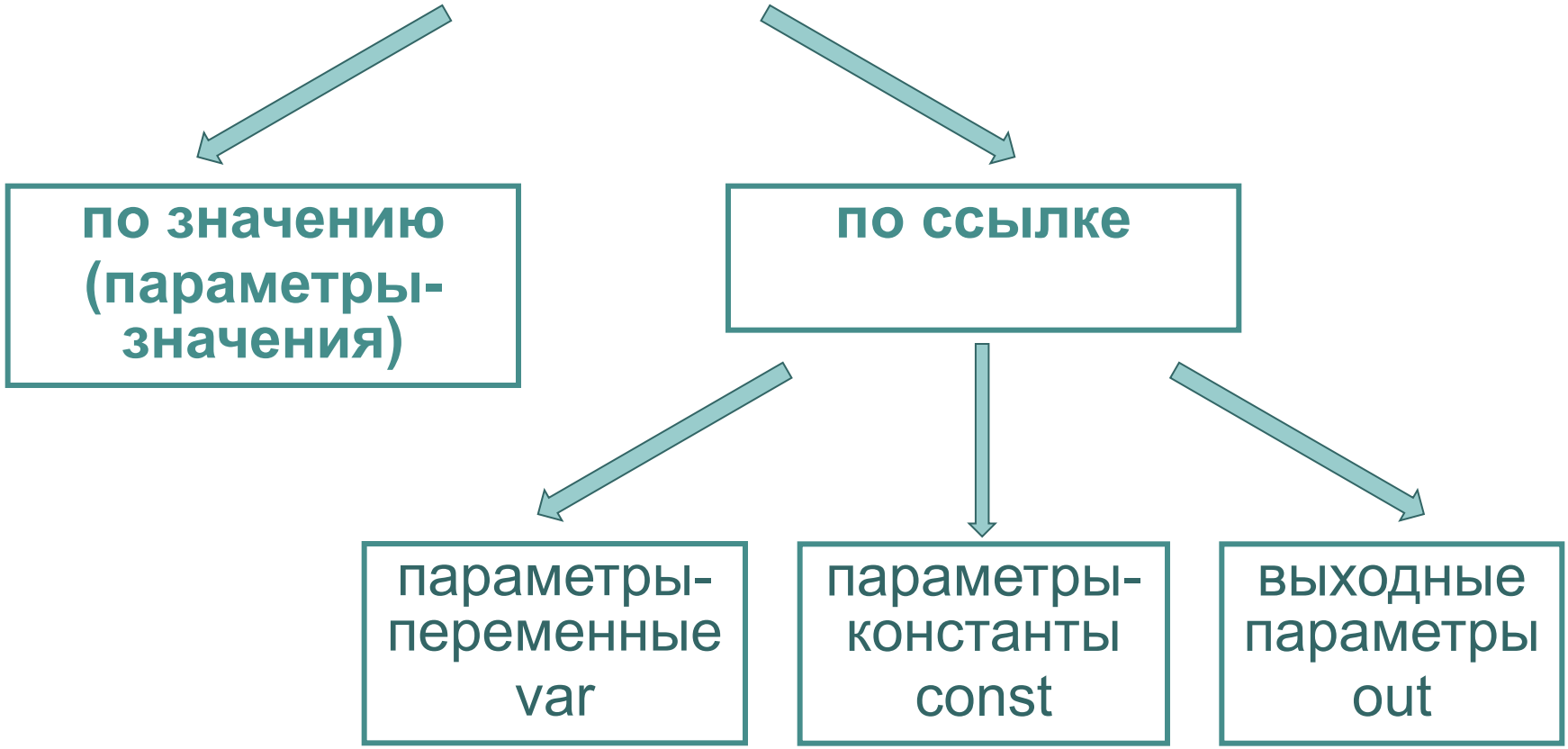
ПР:

procedure Primer1 (**x, y : integer ; c, ch : char ; var s : real**);



function Primer2 (**x : integer; c, ch : real ; var k : integer**) : real;

Способы передачи параметров в подпрограмму



Способы передачи параметров в подпрограмму

параметры-переменные var	параметры-константы const	выходные параметры out
Для передачи данных в подпрограмму и для получения данных из подпрограммы	Нельзя изменять значение в теле подпрограммы (только для чтения)	Используется только для получения данных из подпрограммы

Параметры-значения

Параметры-значения - локальные переменные, которые получили начальное значение при вызове подпрограммы.

Если переменная передана как параметр-значение, то подпрограмма создает копию переменной.

Изменения этой копии не приводят к изменению исходной переменной и утериваются после завершения работы подпрограммы.

Параметры-значения

ПР: function **DoubleByValue** (X: Integer): Integer;

```
    // X – параметр-значение
    begin  X := X*2;  Result :=X; end;
```

Вызовем функцию:

```
Var  K, J, V: Integer;
begin
     K := 4;  V := 4;
     J := DoubleByValue(K);   // J = 8, K = 4
end;
```

Параметры-переменные

Параметры-переменные похожи скорее на указатели.

Изменения параметра-переменной внутри тела подпрограммы сохраняются после завершения работы подпрограммы и возвращаются в вызывающую программу.

Параметры-переменные

ПР: function **DoubleByRef** (var X: Integer): Integer;
// X – параметр-переменная
begin X := X * 2; Result := X; end;

Вызовем функцию:

```
Var  K, V, W: Integer;
begin
     K := 4;  V := 4;
     W := DoubleByRef(V);     // W = 8, V = 8
end;
```

Особенности

Для **параметров-значений** и **параметров-констант** возможно указание в заголовке процедуры или функции значения по умолчанию.

ПР:

(var S: string; **X: Integer = 5**)

(const P, I: Integer; **const M: real = 4.5**)

Особенности

Для **параметров, передающихся по ссылке,** можно не указывать тип параметра. Такие параметры называют **нетипизированными**.

ПР: (**var S, X; out Z**)
(**const P, I; var M**)

Внутри подпрограммы нетипизированные параметры несовместимы ни с какими типами.

Необходимо выполнить приведение типов.

Локальные объявления

Локальные переменные – переменные, которые описываются и используются внутри подпрограммы

В теле подпрограммы можно описать **константы**, **типы**, **другие подпрограммы** (вложенные функции и процедуры).

Область видимости локальных идентификаторов **ограничена подпрограммой, где они объявлены**.

Локальные объявления

```
procedure DeleteRandomSymbol (var S : string);
    function RandomNumber (S : string) : integer;
    var LenString : Integer;
    begin
        LenString:=Length(s);
        Result:=Random(LenString)+1;
    end;
begin
     Delete(S, RandomNumber(s), 1);
end;
```

Функцию **RandomNumber** можно вызывать **только внутри** процедуры **DeleteRandomSymbol**.

Замечания

- В теле функции должен быть, по крайней мере, один оператор, присваивающий значение **имени функции** или неявной локальной переменной **Result**
- Переменная **Result** может участвовать в выражениях **как операнд** внутри тела функции
- Переменная **Result** и **имя функции** всегда представляют одно и то же значение

Замечания

ПР:

```
function MyFunction : Integer;
begin
    MyFunction := 5;
    Result := Result * 2;
    MyFunction := Result + 1;
end;
//  Эта функция вернет значение 11.
```

- Использование **имени функции** в правой части оператора присваивания приведет **к рекурсивному вызову этой функции** в отличие от переменной Result

Соглашения о вызове подпрограмм

Директивы, определяющие правила вызова подпрограммы (в заголовке подпрограммы):

register, stdcall, pascal, cdecl

ПР: procedure Proc; register;
function Func(X: Integer): Boolean; stdcall;

Соглашения о вызове подпрограмм определяют **порядок передачи параметров**

Соглашения о вызове подпрограмм

- **register** задействует регистры процессора для передачи параметров и обеспечивает наиболее эффективный способ вызова подпрограмм. Эта директива применяется по умолчанию.
- Директива **stdcall** используется для вызова стандартных подпрограмм операционной системы.
- Директивы **pascal** и **cdecl** используются для вызова подпрограмм, написанных на языках **Delphi** и **C/C++** соответственно.

Директиву **safecall** изучим позже

Вызов процедур и функций

При вызове подпрограмм необходимо помнить:

- Выражения, используемые для передачи в типизированные параметры-константы и в параметры-значения должны быть совместимыми по присваиванию с соответствующими формальными параметрами.
- Выражения, используемые для передачи в параметры-переменные и в выходные параметры должны быть одинаковых с соответствующими формальными параметрами типов, исключая нетипизированные параметры.

Вызов процедур и функций

При вызове подпрограмм необходимо помнить:

- На место параметров, передаваемых по ссылке можно подставлять только переменные, нельзя – числа и константы.
- Если список формальных параметров отсутствует, то список фактических параметров (в том числе круглые скобки) не указывается.

Перегрузка процедур и функций

Рассмотрим подпрограммы, выполняющие одинаковые действия, но над переменными разных типов данных

- Дадим этим процедурам (функциям) одинаковые имена
- Отличаться они будут списком параметров (количеством параметров или типом параметров)

Такая возможность называется **перегрузкой**.

Для этого применяется директива **overload**.

Перегрузка процедур и функций

ПР: procedure Increment (var Value: **Integer**); **overload**;
// процедура 1
procedure Increment (var Value: **Real**); **overload**;
// процедура 2

Какую именно процедуру использовать в том или ином случае компилятор будет определять **по типам фактических аргументов**, передаваемых при вызове.

```
Var  X: Integer;  Y: Real;
Begin   X:=1;  Y:=2.0;
        Increment(X); // Вызывается процедура 1
        Increment(Y); // Вызывается процедура 2
end.
```

Рекурсивные подпрограммы

Если в теле подпрограммы происходит вызов этой же подпрограммы, такая подпрограмма называется **рекурсивной**.

Рекурсия полезна, например, когда основную задачу можно разделить на подзадачи, имеющие ту же структуру, что и первоначальная задача.

Рекурсивные подпрограммы

ПР: Функция **Factorial** для вычисления факториала

X! = 1 * 2 * ... * (X – 1) * X

Заметим, что **X! = (X – 1)! * X, где 0! = 1**

```
function Factorial (X: Integer) : Longint;
begin
    if X = 0  then
    // Условие завершения рекурсии
        Factorial := 1
    else Factorial := Factorial(X - 1) * X;
end;
```

Упреждающее объявление подпрограмм

Косвенная рекурсия: когда первая подпрограмма вызывает вторую, а вторая - первую.

Записанная первой подпрограмма будет содержать еще **неизвестный компилятору** идентификатор второй подпрограммы, т.е. возникнет ошибка.

Эта проблема решается с помощью **упреждающего описания подпрограмм** с помощью директивы предварительного описания подпрограмм **forward**.

ПР: procedure Proc; **forward**;
function Func (X: Integer): Boolean; **forward**;