**Пример описания процедуры.**

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

const ...;

type ...;

var ...;

begin

<операторы>

end;

**Вызов процедуры** для выполнения осуществляется по ее имени, за которым в круглых скобках следует список *фактических параметров*, т.е. передаваемых в процедуру данных:

<имя процедуры> ( <список фактических параметров> );

Если процедура не принимает данных, то список фактических параметров (в том числе круглые скобки) не указываются.

**Пример** программы, использующей процедуру Power для вычисления числа X в степени Y. Результат вычисления процедура Power заносит в глобальную переменную Z.

program Console;

{$APPTYPE CONSOLE}

uses

SysUtils;

var

Z: Double;

procedure Power(X, Y: Double); // X и Y - формальные параметры

begin

Z := Exp(Y \* Ln(X));

end;

begin

Power(2, 3); // 2 и 3 - фактические параметры

Writeln('2 в степени 3 = ', Z);

Writeln('Press Enter to exit...');

Readln;

end.

**Пример описания функции.**

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

const ...;

type ...;

var ...;

begin

<операторы>

end;

В теле функции должен находиться по крайней мере один оператор, присваивающий значение имени функции или неявной локальной переменной Result. Если таких присваиваний несколько, то результатом функции будет значение последнего из этих операторов.

В качестве примера заменим явно неуклюжую процедуру Power (см. выше) на функцию с таким же именем:

program Console;

{$APPTYPE CONSOLE}

uses

SysUtils;

function Power(X, Y: Double): Double; // X и Y - формальные параметры

begin

Result := Exp(Y \* Ln(X));

end;

begin

Writeln('2 в степени 3 = ', Power(2, 3)); // 2 и 3 - фактические параметры

Writeln('Press Enter to exit...');

Readln;

end.

**Параметры процедур и функций.**

***Входные*** параметры объявляются с помощью ключевого слова **const**; их значения не могут быть изменены внутри подпрограммы:

function Min(const A, B: Integer): Integer;

begin

if A < B then Result := A

else Result := B;

end;

Для объявления ***выходных*** параметров служит ключевое слово **out**:

procedure GetScreenResolution(out Width, Height: Integer);

begin

Width := GetScreenWidth;

Height := GetScreenHeight;

end;

Установка значений выходных параметров внутри подпрограммы приводит к установке значений переменных, переданных в качестве аргументов:

var

W, H: Integer;

begin

GetScreenResolution(W, H);

...

end;

После вызова процедуры GetScreenResolution переменные W и H будут содержать значения, которые были присвоены формальным параметрам Width и Height соответственно.

Если параметр является одновременно ***и входным, и выходным***, то он описывается с ключевым словом **var**:

procedure Exchange(var A, B: Integer);

var

C: Integer;

begin

C := A;

A := B;

B := C;

end;

Изменение значений **var**-параметров внутри подпрограммы приводит к изменению значений переменных, переданных в качестве аргументов:

var

X, Y: Integer;

begin

X := 5;

Y := 10;

...

Exchange(X, Y);

// Теперь X = 10, Y = 5

...

end;

При вызове подпрограмм на место **out**- и **var**-параметров можно подставлять только переменные, но не константы и не выражения.

Если при описании параметра не указано ни одно из ключевых слов **const**, **out**, или **var**, то параметр считается входным, его можно изменять, но все изменения не влияют на фактический аргумент, поскольку они выполняются с копией аргумента, создаваемой на время работы подпрограммы. При вызове подпрограммы на месте такого параметра можно использовать константы и выражения. Пример подпрограммы:

function NumberOfSetBits(A: Cardinal): Byte;

begin

Result := 0;

while A <> 0 do

begin

Result := Result + (A mod 2);

A := A div 2;

end;

end;

Разные способы передачи параметров (**const**, **out**, **var** и без них) можно совмещать в одной подпрограмме.

program Console;

{$APPTYPE CONSOLE}

uses

SysUtils;

procedure Average(const X, Y: Double; out M, P: Double);

begin

M := (X + Y) / 2;

P := Sqrt(X \* Y);

end;

var

M, P: Double;

begin

Average(10, 20, M, P);

Writeln('Среднее арифметическое = ', M);

Writeln('Среднее геометрическое = ', P);

Writeln('Press Enter to exit...');

Readln;

end.

Передача фактических аргументов в подпрограмму осуществляется через специальную область памяти — *стек*. В стек помещается либо значение передаваемого аргумента (*передача значения*), либо адрес аргумента (*передача ссылки на значение*). Конкретный способ передачи выбирается компилятором в зависимости от того, как объявлен параметр в заголовке подпрограммы. Связь между объявлением параметра и способом его передачи поясняет таблица 2.10:

|  |  |  |
| --- | --- | --- |
| Ключевое слово | Назначение | Способ передачи |
| <отсутствует> | Входной | Передается копия значения |
| const | Входной | Передается копия значения либо ссылка на значение в зависимости от типа данных |
| out | Выходной | Передается ссылка на значение |
| var | Входной и выходной | Передается ссылка на значение |

Таблица 2.10. Способы передачи параметров

Если передается значение, то подпрограмма манипулирует копией аргумента. Если передается ссылка на значение, то подпрограмма манипулирует непосредственно аргументом, обращаясь к нему через переданный адрес.