Процедуры и функции

- При решении разных задач часто возникает необходимость проводить вычисления по одним и тем же алгоритмам, например, вычислять корень уравнения f(x)=0.

- В Паскале предусмотрена возможность объединения любой последовательности операторов в самостоятельную подпрограмму, называемую **процедурой**.

- Те алгоритмы, которые программистом оформляются как процедуры в его собственной программе, должны *начинаться с заголовка* и *заканчиваться зарезервированным словом* ***end***;

Подпрограмма

- *Подпрограммой* называют независимую часть программы, предназначенную для решения некой подзадачи.
-
- Подпрограмма взаимодействует с основной программой через механизм *параметров* -- так называют входные и выходные данные, с которыми работает подпрограмма.

- Однажды написанная подпрограмма, выполненная с теми или иными значениями параметров, может решать некоторый класс задач.

Зачем нужны подпрограммы

- использование подпрограмм позволяет решить следующие задачи:

- уменьшение размеров кода и экономия памяти за счет возможности неоднократного вызова одной и той же подпрограммы в рамках одной программы;
- лучшее структурирование программы за счет разбиения задачи на более простые подзадачи;
- эффективное повторное использование однажды написанного кода.

Процедуры

Общий вид подпрограммы-процедуры

- **procedure Имя** (Список формальных параметров);
- **var** описания локальных переменных;
- **begin**
- {Тело процедуры}
- **end;**

- *Другие подразделы раздела описаний, такие как label, const также могут присутствовать между заголовком и телом процедуры и действие их также будет локально -- то есть, определено лишь в рамках данной процедуры*

Структура процедуры

- Поскольку подпрограмма -- отдельная и, в идеале, независимая часть программы, она может содержать собственный раздел описания *локальных* переменных, предназначенных лишь для ее нужд и невидимых из других частей программы.

Структура процедуры

- Общий вид заголовка:
- **PROCEDURE** N(P1:TYPE1; P2:TYPE2; VAR P3:TYPE3 ...); *{где N - имя процедуры, PI - формальные параметры, TYPEI - их типы.}*

- Процедура имеет ту же структуру, что и главная программа (**PROGRAM**):
- разделы **label**, **const**, **type**, **var**
- и выполняемую часть (от **begin** до **end**).
-
- Процедура помещается в главной программе после раздела **var** и перед **begin** программы.

Пример

- **PROGRAM** MA;
- **VAR** A:**INTEGER**; B: **REAL**; C: **CHAR**;
- ***PROCEDURE*** *N (P1:* ***REAL****; P2:* ***CHAR****);*
- **VAR**
- ***BEGIN*** *(*Начало работы процедуры *)*

  ..... ***END****;* (*Конец процедуры*)

- **BEGIN** (*Начало работы PROGRAM*)
- **END**.

- Данные, описанные в первом разделе -- *глобальные*, они доступны всем частям программы, расположенным ниже по ее тексту.
- Данные второго раздела описаний также глобальны, но доступны лишь главной программе, так как описаны непосредственно перед ней.
-
- **Общее правило очень простое:**
- подпрограммы "видят" все глобальные переменные, описанные выше их тела. Подпрограмма "видит" и может вызвать любую другую подпрограмму, расположенную выше нее по тексту программы.

- Вторая и следующие подпрограммы могут вызвать первую, но не наоборот. Главная программа, как правило, расположенная последней, может вызвать все подпрограммы.

Пример

```
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.
```

Глобальные переменные видимы (доступны) от точки их определения до конца файла.

Локальные переменные доступны только в том блоке операторных скобок, в котором они описаны и во вложенных в него блоках.

Связь процедуры с "внешним миром"

- Единственная правильная связь процедуры с "внешним миром", то есть, другими подпрограммами и главной программой -- указанный после имени ***список формальных параметров.***
- В этом списке через запятую указываются ***входные и выходные параметры процедуры*** с указанием их типов данных.
- ***Входные параметры служат исходными данными для процедуры***, а ***выходные определяют результаты ее вычислений***, передаваемые в главную программу или другую подпрограмму.
- Перед *выходными* параметрами, измененные значения которых должны сохраняться после завершения процедуры, следует указывать ключевое слово **var.**

Чтобы процедура сработала, ее нужно *вызвать*, записав в нужной точке программы имя процедуры со *списком фактических параметров*

```
procedure Equation (a,b,c:real;var x1,x2:real);
var d:real;
begin
 d:=sqr(b)-4*a*c;
 if d>=0 then
    begin
     x1:=(-b+sqrt(d))/(2*a);
     x2:=(-b-sqrt(d))/(2*a);
    end;
end;
```

```
var a,b,c,d,x1,x2,x3,x4:real;
...
write ('Введите значения a,b,c:');
read (a,b,c);
Equation (a,b,c,x1,x2);
```

Согласование параметров

- При каждом вызове подпрограммы значения фактических параметров подставляются на место формальных и с ними производятся вычисления, предусмотренные операторами подпрограммы.
- Указанные требования называют *согласованием параметров* и описывают следующим образом: формальные и фактические параметры должны быть согласованы между собой по количеству, типу и порядку следования.
- Это означает, что количество формальных и фактических параметров должно быть одинаковым, при этом, при вызове процедуры каждый фактический параметр должен иметь тот же тип и занимать в списке то же место, что и соответствующий ему формальный параметр.

Пример

- Процедуру Equation (см. выше) можно вызвать только с пятью параметрами (а не тремя, семью или нулём), причем все эти параметры должны быть вещественными.

- Если формальный параметр является выходным (перед ним в заголовке процедуры указано ключевое слово **var**), то соответствующий фактический параметр не может быть константой (ведь значение константы нельзя изменить).

Соответствие формальных и фактических параметров

Формальный параметр в заголовке процедуры	Соответствующий фактический параметр при вызове процедуры
Переменная некоторого типа данных без ключевого слова var	Переменная, константа, элемент массива или значение выражения того же типа данных
Переменная некоторого типа данных с ключевым словом var	Переменная или элемент массива того же типа данных
Массив	Массив

Формальные параметры

- это наименования переменных, через которые передается информация из программы в процедуру либо из процедуры в программу.
- Вызов **процедуры N** производится оператором вида
- N(P1,P2,P3,...);
- Здесь N - имя процедуры,
- P1,P2,P3 - фактические параметры.

Вызов процедуры

- При вызове процедуры машина производит **следующие действия**:
- Устанавливает взаимно однозначное соответствие между фактическими и формальными параметрами, затем управление передает процедуре.
- После того, как процедура проработает, управление передается вызывающей программе на оператор, следующий за вызовом процедуры.

- **Соответствие между фактическими и формальными параметрами**

- должно быть следующим:
- число фактических параметров должно быть равно числу формальных параметров;
- соответствующие фактические и формальные параметры должны совпадать по порядку следования и по типу.
- соответствующие параметры не обязательно должны быть одинаково обозначены.

Пример

- Вызвать процедуру SQ можно так:
- **SQ(P,Q,R,Y,Z);** P, Q, R - коэффициенты кв. ур-я, а Y и Z - корни.
- Если вызвать SQ оператором
- **SQ(X1,X2,A,B,C)**;то машина воспримет X1, X2,A как коэффициенты уравнения, а корни зашлет в переменные B и C.

Параметры-значения

- Примером таких параметров служат параметры A,B и C в процедуре SQ.
- **PROCEDURE** SQ (A,B,C:**REAL**; **VAR** X1,X2: **REAL**);
- В этом случае фактическим параметром, соответствующим A, либо B, либо C, может быть выражение соответственного типа, в частности, константа.
- Например, обратиться к SQ можно так:
- SQ((25./3+2)*2, - 1.5, (8.2-3.1)/3, X1, X2);

Параметры-значения

- Для **параметров-значений** машина при вызове процедур производит следующие действия:
- выделяется место в памяти для каждого формального параметра,
- вычисляется значение фактического параметра и засылает его в ячейку, соответствующую формальному параметру.
- Если фактический параметр есть имя переменной, например, К, то значение этой переменной пересылается в соответствующий формальный параметр, например, А.

!!!!!

- На этом всякая связь между А и К обрывается.
- Если даже фактический и формальный параметры одинаково обозначены, в памяти компьютера эти параметры занимают разные ячейки.
- Это надо знать, чтобы не допустить распространенной ошибки, а именно: пытаться передать информацию из процедуры в вызывающую программу через параметр-значение

Передача по значению

- procedure p1 (x:integer);
- {для x создастся локальная копия}
- begin
- x:=x+1; {значение копии увеличилось на 1}
- writeln ('x=',x); {и было выведено}
- end;
-
- var x:integer;
- begin
- x:=3;
- p1(x);
- writeln ('x=',x);
- *{после вызова с передачей параметра по значению, x, по-прежнему, равно 3}*
- end.

Параметры-переменные

- Если перед именем формального параметра стоит ключевое слово **var**, то такой параметр и есть **параметр-переменная**. Пример: X1 и X2 в процедуре SQ.

- Результат выполнения может быть передан только через параметр- переменную

- Использование var в заголовке процедуры подчеркивает, что параметр является переменной и может изменяться этой процедурой

Параметры-переменные

- Если формальный параметр процедуры помечен ключевым словом var как параметр -переменная, это означает, что передача фактического параметра осуществляется *по адресу*, то есть, в процедуру передается адрес того места в памяти, где находится фактический параметр.
- Таким образом, все выполняемые в процедуре изменения значения параметра-переменной будут выполнены непосредственно над значением фактического параметра.

Параметры-переменные

```
procedure p1 (var x:integer);
 {получаем адрес переменной,
  переданной в качестве x}
begin
 x:=x+1;   {значение x увеличилось на 1}
 writeln ('x=',x);     {и было выведено}
end;

var x:integer;
begin
 x:=3;
 p1(x);
 writeln ('x=',x);
 {после вызова с передачей параметра
  по ссылке, x стало равно 4 -
  оно было изменено процедурой p1}
end.
```

Функции

- Другой вид подпрограммы- **функция**- оформляется аналогично процедуре.

- Отличительные особенности функции: **она имеет только один результат выполнения** (но может иметь несколько входных параметров);

- Результат обозначается именем функции и передаётся в основную программу.

Функция оформляется в следующем виде:

- **Function** <имя функции>(формальные параметры: тип): тип значения функции;
- **Var**
  . . .
  Begin
  . . .
  End ;

- Вызывается функция по её имени с указанием фактических параметров.
- Вызов функции можно делать непосредственно внутри выражения.
- При вызове функции тип не указывается.
- В теле функции обязательно должен быть хотя бы один оператор присваивания, где в левой части стоит имя функции, а в правой - ее значение. Иначе, значение не будет определено.

- Вызывается функция по её имени с указанием фактических параметров.
- Вызов функции можно делать непосредственно внутри выражения.
- При вызове функции тип не указывается.
- В теле функции обязательно должен быть хотя бы один оператор присваивания, где в левой части стоит имя функции, а в правой - ее значение. Иначе, значение не будет определено.

Различие между процедурами и функциями.

- Функции - это процедуры особого характера, результатом работы которых является некоторое значение, подобное переменной.
- Функция, как и процедура, может иметь список параметров, следующих за именем функции в круглых скобках.
- Но если *имя процедуры используется только для ее вызова*, то *с именем функции связывается ее значение*.

Примеры

Пример

```
Program ProcedureAndFunction;

Var
   a, b, SumNumbers : integer;

Procedure Summa1(Var Sum: integer; a, b : integer);
Begin
  Sum:= a+b;
End;

Function Sum(a, b : integer) : integer;
Begin
  Sum:= a+b;
End;
```

```
Begin
      ClrScr;
      a := 12;
      b := 15;
      Summa1(SumNumbers, a, b);
      writeln ('С помощью процедуры сумма чисел равна ',SumNumbers);
      writeln ('С помощью функции сумма чисел равна ',Sum(a, b));
End.
```

Пример процедуры для обмена значений между целыми переменными m и n

- **program** имя-программы;
- **procedure** имя(**параметры**);
- **описания локальных величин**
- **Begin**
- операторы процедуры
- **end;**
- **BEGIN**
- операторы программы
- **END.**

- **program** primer;
- **var** x,y: **integer**;
- **procedure** swap(**var** m,n:**integer**);
- **var** s: **integer**;
- **Begin**
- s:=m; m:=n; n:=s
- **end;**
- **BEGIN**
- read(x,y); swap(x,y); writeln(x:10,y:10)
- **END.**

Пример функции для вычисления натуральной степени (n) целого числа (m)

- **program** имя-программы;
- **function** имя(параметры): тип-результата;
- описания локальных величин
- **Begin**
- операторы функции
- **end;**
- **BEGIN**
- операторы программы
- **END.**

- **program** primer;
- **var** x,y: **integer**;
- **function** power(m,n:**integer**):**integer**;
- **var** s,i: **integer**;
- **Begin**
- s:=1;
- for i:=1 to n do s:=s*m;
- power:=s
- **end;**
- **BEGIN**
- read(x,y); writeln(power(x,y))
- **END.**