ОРГАНИЗАЦИЯ ПОДПРОГРАММ. ПРОЦЕДУРЫ И ФУНКЦИИ. ПРОЦЕДУРЫ И ФУНКЦИИ

В Паскале два типа подпрограмм – процедуры и функции. Создание процедур и функций является дальнейшим развитием идеи группирования, когда несколько операторов объединяются в программный блок, который имеет свое имя и к которому можно обратиться по этому имени с помощью соответствующих средств вызова. Процедура вызывается с помощью оператора вызова процедуры с одноименным именем. Функция пользователя также имеет свое имя, и ее активизация происходит при вычислении выражения, содержащего имя функции. Возвращаемое функцией значение подставляется в это выражение.

Процедуры и их типизация

Итак, процедура – это часть программы (подпрограмма), имеющая имя и предназначенная для решения некоторой частной задачи (подзадачи). Процедуры делятся по способам описания и обращения к ним.

Процедура встроенная (*машинная*) – это процедура, описание которой считается известной транслятору, в связи с чем ее можно использовать в программе, зная только ее имя.

Процедура пользователя – процедура, которую создает (описывает) программист на основе имеющихся операторов и встроенных процедур и функций данного языка по определенным правилам данного языка.

Процедура без параметров – процедура, при обращении к которой не требуется задания начальных установок, значений и после выполнения которой в основную программу не передаются результаты работы данной процедуры.

Процедура с параметрами-значениями – процедура, при обращении к которой требуются только начальные значения. На выходе данные не передаются в основную программу.

Процедура с параметрами-переменными – процедура, не требующая начальных значений, однако передающая в основную программу результаты своей работы (передает значения некоторых переменных).

Комбинированная процедура – процедура, имеющая параметры-переменные и параметры-значения, т.е. входные и выходные данные.

Встроенные процедуры

Встроенные процедуры являются составной частью системы программирования. Среди них есть стандартные процедуры, которыми можно пользоваться в любом месте программы без какого-либо предварительного объявления. Сюда относятся уже ранее упомянутые процедуры ввода/вывода, управления работой программы, динамического распределения памяти, строковые процедуры и пр. Полный перечень встроенных процедур можно найти в справочнике для языка.

Помимо стандартных процедур в Паскале есть также стандартные модули, представленные в виде TPU – файлов, каждый из которых содержит в себе целый набор процедур и функций. Для того чтобы использовать процедуры из модулей, необходимо вызвать нужный модуль в разделе USES. Система TurboPascal имеет модули PRINTER, DOS, CRT, GRAPH и др.

CRT позволяет использовать все возможности дисплея и клавиатуры, включая управление режимом работы экрана, расширенные коды клавиатуры, цвет, окна и звуковые сигналы.

DOS поддерживает различные функции ДОС, включая установку и получение текущего значения даты и времени, поиск по каталогам файлов и выполнение программ.

PRINTER позволяет легко организовать доступ к устройству печати.

GRAPH – мощный графический пакет с набором процедур и функций обработки графических объектов (точек, отрезков, прямоугольников, окружностей и пр.).

Рассмотрим несколько примеров встроенных процедур:

- CLRSCR – процедура очистки экрана. Результатом работы является стирание всей информации с экрана. Данная процедура является примером процедур без параметров.
- GOTOXY (A, B) – процедура позиционирования курсора на экране дисплея в точку с координатами (A, B). A и B являются входными данными, следовательно, это пример процедуры с параметрами-значениями.
- WRITE (A, B, ..., Q) – процедура вывода информации на экран дисплея. Данная процедура – процедура с параметрами-значениями.
- READ (A, B, ..., Q) – процедура ввода информации в ЭВМ. При выполнении данной процедуры переменным A, B, ..., Q присваиваются конкретные значения, т.е. данные передаются в основную программу, и, значит, процедура является примером процедур с параметрами-переменными.

Процедуры пользователя

При работе с процедурами пользователя необходимо уметь производить два вида деятельности: описание процедуры и обращение к ней в основной программе. Вызов процедуры пользователя осуществляется так же, как и вызов встроенной процедуры, – с помощью оператора вызова процедуры, имя которого совпадает с именем процедуры, с указанием списка параметров, если таковые имеются. Описание же процедуры включает в себя разработку подпрограммы и правильное оформление ее заголовка. Остановимся на нем более подробно.

В основной программе все процедуры (а также и функции) пользователя должны быть объявлены. Объявление процедур и функций осуществляется после объявления переменных и перед первым словом BEGIN программы.

Процедура, как видно из ее определения, оформляется так же, как и основная программа. Вообще процедуру нужно воспринимать как программу в миниатюре. В свою очередь, основная программа может быть легко переделана в процедуру с заменой слова PROGRAM на PROCEDURE. Если

процедура объявлена, ее можно использовать в последующих частях программы, просто записывая ее имя, за которым, если необходимо, следует список параметров. Вызов процедуры для основной программы становится новым оператором. Обращение к процедуре активизирует эту процедуру, т.е. приводит к выполнению группу операторов, содержащихся в ее теле. После этого управление переходит к оператору, следующему за вызовом процедуры.

Описание процедур будем рассматривать по той же схеме, что и машинные процедуры, а именно: сначала без параметров, далее с параметрами-значениями, с параметрами-переменными и, наконец, комбинированные процедуры.

Процедуры без параметров

Заголовок процедуры без параметров выглядит как:

PROCTDURE <Имя процедуры>;

Вызываются такие процедуры путем написания в основной программе имени этой процедуры. В виде процедуры без параметров оформляются такие подзадачи, у которых нет входных и выходных данных, или же эти данные удобнее передавать с помощью операторов присваивания: READ и WRITE.

Рассмотрим несколько примеров, в которых представлены эти варианты.

П р и м е р Нарисовать три вертикальных квадрата 3 × 3 с помощью символа «*».

Очевидно, что в этой программе надо выделить рисование квадрата в виде процедуры без параметров, а затем трижды вызвать ее в основной программе.

```
program RISUNOK;
procedure KVADRAT;
begin
    writeln ('***');
    writeln ('*  *');
    writeln ('***');
end;
begin
    clrscr; KVADRAT;
    writeln; KVADRAT;
    writeln; KVADRAT;
end.
```

П р и м е р Вычислить площадь четырехугольника ABCD.

Зная длины сторон четырехугольника и длину одной из его диагоналей, например BD, можно по формуле Герона найти площади двух вспомогательных треугольников и сложить их. Отсюда следует, что в программе надо выделить процедуру вычисления площади треугольника.

```
program PLOCHAD_1;
var AB, BC, CD, AD, BD, S1, S, a, b, c, p:real;
      procedure GERON_1;
            begin
                  p := (a + b + c)/2;
                  S := sqrt (p*(p - a)*(p - b)*(p - c));
            end;
begin {*ОСНОВНАЯ ПРОГРАММА*}
            read (AB, BC, CD, AD, AC);
            a := AB; b := AD; c := BD; GERON_1; S1:= S;
            a := BC; b := CD; c := BD; GERON_1; S1:= S1 + S;
            write (S1);
end.
```

Примечание. В данной программе все вычисления проходят с помощью переменных, объявленных в разделе VAR основной программы.

Процедуры с параметрами-значениями

Как было сказано ранее, процедуры с параметрами-значениями требуют входных данных. Где они записываются и как задаются? На этот вопрос может ответить общая форма заголовка процедуры этой процедуры:

PROCEDURE <Имя процедуры> (<Параметры-аргументы>: тип);

Здесь под параметром понимают имя переменной, которая является «входной» для процедуры (формальный параметр-аргумент). Этот параметр с синтаксической точки зрения является параметром-значением, при его описании в заголовке процедуры не требуется писать слово VAR. Параметры-значения при вызове процедуры принимают из основной программы свои конкретные значения. Заметим также, что в самой процедуре значения параметров-значений не меняются в ходе работы процедуры.

При обращении к процедуре с параметрами-значениями в основной программе фактическими параметрами могут служить как имена переменных (которые описаны и определены выше), так и конкретные значения (константы) и выражения. При обращении необходимо следить за соответствием списка параметров при обращении и описании. Кроме того, следует строго соблюдать соответствие типов параметров.

Рассмотрим работу процедур такого типа на примерах.

П р и м е р Нарисовать квадрат с произвольной длиной стороны в левом верхнем углу (длина стороны задается с клавиатуры).

В этой программе также надо оформить рисование квадрата в виде процедуры, но уже с входным параметром-значением – длиной стороны квадрата.

```
program RISUNOK_2;
```

```
var I: integer;
    procedure KVADRAT (N: integer);
            var J, K: integer;
              begin
    for J := 1 to N do write ('*'); writeln;
    for J := 1 to N - 2 do
            begin
                    write ('*'); for K := 1 to N - 2 do write (' ');
                    writeln ('*');
                end;
          for J := 1 to N do write ('*');
      end;
begin { Основная программа }
          write ('Введите длину стороны – ');
          readln (I);  clrscr; KVADRAT (I);
end.
```

П р и м е р Вычислить площадь четырехугольника с применением процедуры с параметрами-значениями.

```
program PLOCHAD_2;
   var AB, BC, CD, AD, AC, S1, S: real;
       procedure GERON_2 (a, b, c: real);
            var P: real;
              begin
                    P := (a + b + c)/2;  S := sqrt (P*(P - a)*(P - b)*(P - c));
              end;
  begin {*ОСНОВНАЯ ПРОГРАММА*}
          read (AB, BC, CD, AD, AC); GERON_2(AB, BC, AC); S1:= S;
          GERON_2 (AD, AC, CD); write ('S = ', S1 + S)
  end.
```

В данной программе определена процедура GERON_2 с тремя параметрами-значениями и локальной переменной P. Значение же площади треугольника помещается в глобальную переменную S. При вызове этой процедуры формальные параметры a, b, c замещаются на фактические параметры AB, BC, AC при первом обращении и на AD, AC, CD – при втором.

Заметим также, что здесь фактические параметры представлены переменными, которые получают свое значение с помощью процедуры READ. Однако если известны длины сторон треугольника, например, 6, 7, 4, то можно вычислить площадь этого треугольника, вызвав процедуру GERON_2(6, 7, 4), и получить ответ в переменной S.

Процедуры с параметрами-переменными

В отличие от процедур с параметрами-значениями, процедуры с параметрами-переменными не имеют входных параметров, т.е. из основной программы не передаются значения переменных в процедуру, за исключением глобальных переменных. Отличие в описании и обращении к процедурам с параметрами-переменными заключается в специфическом написании заголовка процедуры. В остальном все процедуры схожи. Общий вид заголовка процедуры с параметрами-переменными выглядит так:

PROCEDURE <Имя процедуры> (VAR<Параметры-переменные>: тип);

При детальном ознакомлении с синтаксической диаграммой видно, что параметрам-переменным должно предшествовать служебное слово VAR, причем оно пишется столько раз, сколько различных типов представлено в выходных данных, например:

PROCEDURE PRIMER (VAR a, b, c : INTEGER; VAR m : CHAR; VAR i, j : REAL).

При обращении к процедурам с параметрами-переменными фактическими параметрами должны являться имена переменных, которые описаны в основной программе.

П р и м е р . Вычисление площади четырехугольника.

```
program PLOCHAD_3;
    var AB, BC, CD, AD, AC, S1, S2, a, b, c: real;
        procedure GERON_3 (var S: real);
            var P: real;
                begin
                    P := (a + b + c)/2; S := sqrt (P*(P - a)*(P - b)*(P - c));
                end;
begin { Основная программа }
        read (AB, BC, CD, AD, AC);
        a := AB; b := BC; c := AC; GERON_3 (S1);
        a := AD; b := AC; c := CD; GERON_3 (S2);
        write ('S = ', S1 + S2)
end.
```

Комбинированные процедуры

Комбинированные процедуры включают в себя входные и выходные данные. В заголовке этой процедуры, как и ранее, выходные параметры предваряются словом VAR, входные параметры – без него. Порядок следования параметров может быть произвольным, например:

PROCEDURE PRIMER (VAR a, b, c: INTEGER;
m: CHAR; VAR i, j: REAL);

Здесь a, b, c, i, j – параметры-результаты (переменные); m – параметр-аргумент (значение).

В качестве иллюстрации комбинированных процедур рассмотрим последний вариант вычисления площади четырехугольника:

```
program PLOCHAD_4;
    var AB, BC, CD, AD, AC, S1, S2: real;
        procedure GERON_4 (a, b, c : eal; var S : real);
```

```
                var P : real;
                    begin
                            P := (a + b + c)/2;
                            S := sqrt (P*(P - a)*(P - b)*(P - c));
                    end;
begin {*ОСНОВНАЯ ПРОГРАММА*}
        read (AB, BC, CD, AD, AC);
        GERON_4 (AB, BC, AC, S1);
        GERON_4 (AD, AC, CD, S2);
        write ('S = ', S1 + S2)
end.
```

Примечание. Для более полного усвоения введенных ранее терминов перечислим на базе последнего примера все виды параметров и переменных:

1) глобальные переменные AB, BC, CD, AD, AC, S1, S2;
2) локальные переменные a, b, c, S, P;
3) формальные параметры a, b, c, S:
   - параметры-значения (аргументы) a, b, c;
   - параметр-переменная (результат) S;
4) фактические параметры AB, BC, CD, AD, AC, S1, S2:
   - параметры-значения (аргументы) AB, BC, CD, AD, AC;
   - параметры-переменные (результаты) S1, S2.

Заметим также, что термины «параметр-значение» и «аргумент», как и «параметр-переменная» и «результат», не всегда идентичны. Дело в том, что характеристика «значение (переменная)» отражает синтаксическую сущность параметра, а «аргумент (результат)» – его семантику (функциональную роль в процедуре). Иногда один и тот же параметр может быть аргументом и результатом одновременно, но описан в процедуре в виде параметра-переменной.

Попытка же описать выходной параметр в виде параметра-значения (без слова VAR в заголовке процедуры) приведет к тому, что результат работы процедуры не будет возвращен в основную программу. Это происходит потому, что характер «поведения» параметров-значений и параметров-переменных в процессе работы процедуры различен. Разница эта состоит в том, что преобразования, которые претерпевают формальные параметры-значения в процедуре, не вызывают изменения соответствующих им фактических параметров, в то время как изменения параметров-переменных могут изменять значения соответствующих фактических параметров.

Причиной этого феномена является неодинаковое распределение памяти под хранение параметров процедуры. Формальному параметру-значению отводится некоторая область (ячейка) памяти, куда заносится значение соответствующего фактического параметра, вычисленного на момент обращения к процедуре. На этом связь между ними обрывается. Действительно, если фактическим параметром является константа или выражение, то как изменения в формальном параметре-значении (а это есть всегда переменная) могут повлиять, например, на выражение?

Фактическим же параметром, соответствующим формальному параметру-переменной, является всегда переменная. На время выполнения процедуры эти параметры отождествляются, им соответствует одна и та же область памяти. Вполне понятно, что в этой ситуации изменения формального параметра влекут адекватные изменения фактического параметра, и после завершения процедуры его значение может отличаться от его первоначального значения.

Именно поэтому, объявив в процедуре параметр-результат как параметр-значение, этот результат так и останется в формальном параметре-переменной без его передачи в соответствующий фактический параметр (т.е нет возврата полученного процедурой значения) в основную программу.

Функции пользователя. Рекурсивные функции

Функция как объект языка Паскаль является другой версией реализации технологии построения программ с использованием структуры группирования. Можно также сказать, что функция есть частный вид определенного типа процедур, а именно процедур с одним параметром-переменной.

Определение функции

Функция отличается от процедуры только тем, что всегда возвращает в точку вызова одно скалярное значение. При этом функция, как и процедура, может содержать параметры-значения или быть без них.

Общая форма записи заголовка функции

FUNCTION имя (список параметров: тип): тип;

или

FUNCTION имя: тип;

Тип результата есть тип значения функции. Список параметров такой же, что и для процедуры, только здесь все параметры являются аргументами. Имя переменной, которая хранит значение функции, совпадает с именем функции.

Итак, заголовок функции отличается от заголовка процедуры не только сменой слова PROCEDURE на FUNCTION, но и удалением из списка параметров параметра-результата с присвоением его типа имени функции:

PROCEDURE <имя процедуры> (аргументы;
VAR параметр-результат: тип);

FUNCTION <имя функции> (аргументы): тип;

Другой особенностью описания функции является наличие в нем хотя бы одного оператора присваивания, в левой части которого стоит имя определяемой функции, а в правой – выражение для вычисления результата функции. Очевидно, что тип этого выражения должен совпадать с указанным в заголовке типом функции.

Вызов функции также отличается от вызова процедуры. Если вызов процедуры осуществляется с помощью специального оператора вызова (оператора процедуры), то функция вызывается только внутри некоторого выражения. Для того, чтобы осуществить обращение к функции, необходимо использовать ее имя со списком фактических параметров в каком-либо выражении, тип которого совпадает с типом значения функции. Само же выражение, внутри которого вызывается функция, может быть правой частью оператора присваивания, частью логического выражения и пр.

Функции пользователя

Известно, что Паскаль имеет набор стандартных функций. Однако этот набор ограничен. Пользователь может по желанию расширить список функций, создав свои функции – функции пользователя. Так, например, в Паскале есть SQR (X) = X^2, а вот функции F (X) = X^n, где n принадлежит множеству целых чисел Z, нет. Используя определенное ранее понятие функции, можно создать для этого универсальную функцию, которая давала бы степени произвольного вещественного числа с любым целым показателем.

Определим вещественную функцию POWER, которая должна иметь два параметра-аргумента – для показателя и для основания степени:

```
function POWER (FACTOR: real; EXP: integer): real;
    var COUNT: integer; TFACTOR: real;
    begin
        if EXP = 0 then POWER := 1
                    else begin
                            TFACTOR := FACTOR;
                            for COUNT := 2 to ABS (EXP) do
                                    TFACTOR := TFACTOR*FACTOR;
                                    if EXP < 0 then POWER := 1/TFACTOR
                                                else POWER := TFACTOR
                            end
end;
```

Теперь можно эту функцию вызывать следующим образом:

а) F := POWER(5.25, 3);
б) WRITELN ("D = ", POWER (5.25, -2):5:2);
в) IF X > 2*POWER (6.2, 3) THEN WRITE ('ДА');
г) A := POWER (X, 2) + POWER (X, 3) + POWER (X, 4).

.Рекурсивные функции

К функциям можно обращаться тремя способами: из тела основной программы, из тела другой функции, из тела самой функции, т.е. функция может вызывать саму себя. Функции называются *рекурсивными*, если в описании функции происходит вызов самой себя, а процесс обращения –

рекурсией. Продемонстрируем использование рекурсии на примере вычисления значения факториала произвольного натурального числа N.

В математике известно рекурсивное определение факториала:

$$n! = 1 \quad \text{при } n = 0;$$
$$n! = (n - 1)! \cdot n \quad \text{при } n > 0.$$

Это рекурсивное определение можно реализовать с помощью соответствующей рекурсивной функции:

```
function FACTORIAL (VALUE: integer): integer;
    begin
      iF VALUE = 0 then FACTORIAL := 1
      else FACTORIAL := VALUE*FACTORIAL (VALUE - 1)
    end;
```

Теперь можно обращаться к этой функции в теле основной программы, как показано в следующем примере:

```
program FINDFACTORIAL;
   var N: integer;
       begin
          writeln ('Введите число');
          readln (N);
          if N < 0  then writeln ('Нет факториала')
                    else writeln ('Факториал ', N, ' равен ', FACTORIAL (N))
       end.
```

Характерной особенностью построенной функции является наличие в ее теле оператора присваивания:

FACTORIAL := VALUE*FACTORIAL (VALUE - 1),

где происходит вызов определяемой функции. Здесь идентификатор FACTORIAL в левой части оператора обозначает имя переменной для хранения значения функции, а в правой – имя вызываемой функции.

Важным моментом при составлении любой рекурсивной функции является организация выхода из рекурсии. В некоторых простых случаях должно существовать не рекурсивное решение. Рекурсивный процесс должен шаг за шагом так упрощать задачу, чтобы, в конце концов, для нее появилось не рекурсивное решение. В этих функциях должны проверяться значения аргумента для принятия решения о завершении. В нашем случае условием завершения рекурсии является VALUE = 0.

При описании рекурсивных функций необходимо хорошо представлять процесс вычислений. Всякая рекурсия состоит из двух этапов: углубления (погружения) внутрь рекурсии и выхода из нее. На первом этапе никаких вычислений не производится, а идет только настройка рабочей формулы на

конкретные операнды. На втором этапе происходит процесс вычислений по настроенным формулам.

В заключение покажем, что часто рекурсивные функции строятся гораздо проще, чем «обычные», хотя вполне понятно, что не всякая функция может быть переделана на рекурсивную. Сделаем это на примере уже построенной ранее функции POWER.

Данная функция явно носит рекурсивный характер, исходя из ее определения:

$$X^n = 1, \text{ если } n = 0;$$
$$X^n = X^{n-1} * X, \text{ если } n > 1.$$

Ниже следует рекурсивная функция вычисления значения степени:

```
function POWER (FACTOR: real; EXPONENT: integer): REAL;
    begin
      if EXPONENT < 0
        then POWER := 1/POWER (FACTOR, abs (EXPONENT))
        else
            if EXPONENT > 0
      then POWER := FACTOR*POWER (FACTOR, EXPONENT - 1)
      else POWER := 1
    end;
```

Замечание. Помимо рекурсивных функций, в языке Паскаль по тому же принципу можно определять и рекурсивные процедуры. Подробно о них будет сказано в следующих разделах, а пока покажем, как рекурсивная функция может быть переделана в рекурсивную процедуру на примере вычисления факториала:

```
procedure FACTORIAL (VALUE: integer; var F: integer);
    begin
       iF VALUE = 0 then F := 1
              else begin FACTORIAL (VALUE - 1, F);
                         F := F*VALUE
                   end;
    end;
```

Здесь уже, в отличие от функции FACTORIAL, для вычисления N! необходимо вызвать эту процедуру с помощью оператора процедуры FACTORIAL (N, FN), где FN – переменная для возвращения из процедуры значения N!.

Пример 1. Рассчитать значение y(x) по формуле

$$y = tg^2(x+1) + tg(x+0.5) + 7 * tg(x)$$

Для расчета значения tg(x) используется **подпрограмма-функция** tg(a) с одним параметром.

Program Project1;

```
  var x,y:real;
function tg(a:real):real;
begin
tg:=cos(a)/sin(a);
end;

begin
   write('x='); readln(x);
   y:=sqr(tg(x+1))+tg(x+0.5)+7*tg(x);
   writeln('y=',y:6:2);
   readln;
end.
```

Пример 2. Условие из предыдущего примера. В программе для расчета значения tg(x) используется **подпрограмма-процедура** tg(a,b) с двумя параметрами.

```
Program Project2;
  var x,y:real; x1,x2,x3:real;

procedure tg(a:real; var b:real);
begin
b:=cos(a)/sin(a);
end;

begin
   write('x='); readln(x);
   tg(x+1,x1);    tg(x+0.5,x2);   tg(x,x3);
   y:=sqr(x1)+x2+7*x3;
   writeln('y=',y:6:2);
   readln;
end.
```

Пример 3. Рассчитать периметр (в м) и площадь прямоугольника (в $м^2$). Длины сторон (x, y) вводятся с клавиатуры в см. В программе используется **подпрограмма-процедура** parametr(a, b, p, s).

```
Program Project3;
  var x,y:integer; s1,p1:real;

 procedure    parametr(a,b:integer; var p,s:real);
 begin
 p:=2*(a+b)/100;
 s:=a*b/10000;
```

```
 end;

begin
x:=30; y:=40;
parametr(x,y,p1,s1);
writeln('p1=',p1:6:2, '  s1=',s1:6:2);
readln;
end.
```

Пример.4. Условие из предыдущего примера. В программе используются две подпрограммы-функции.

```
Program Project4;
 var x,y:integer; s1,p1:real;

 function    p(a,b:integer):real;
 begin
 p:=2*(a+b)/100;
  end;

  function    s(a,b:integer):real;
 begin
  s:=a*b/10000;
 end;

begin
x:=30; y:=40;
p1:=p(x,y);
s1:=s(x,y);
writeln('p1=',p1:6:2, '  s1=',s1:6:2);
readln;
end.
```

Пример 5. Условие из предыдущего примера. В программе используется процедура parametr без параметров.

```
Program Project5;
  var a,b:integer; p,s:real;
  procedure parametr;
  begin
  p:=2*(a+b)/100;
  s:=(a*b)/10000;
  end;
begin
write('a='); readln(a);  write('b='); readln(b);
```

```
 parametr;
 writeln('p=',p:6:2, '  s=',s:6:2);
 a:=50; b:=80;
  parametr;
 writeln('p=',p:6:2, '  s=',s:6:2);
readln;
 end.
```

Пример 6. Условие из предыдущего примера. В программе используются две функции (s и p) без параметров.

```
Program Project6;
  var a,b:integer; p1,s1:real;
 function    p:real;
 begin
 p:=2*(a+b)/100;
  end;

  function    s:real;
 begin
  s:=a*b/10000;
 end;

begin
a:=30; b:=40;
p1:=p;
s1:=s;
writeln('p1=',p1:6:2, '  s1=',s1:6:2);
readln;
end.
```

Вопросы для самоконтроля:

1. В чем состоит принципиальное отличие процедур от функций?
2. Чем отличается вызов функции от вызова процедуры?
3. Какой переменной присваивается значение в процедуре и в функции?
4. Какие переменные в языке Паскаль называются локальными, а какие глобальными?
5. Какие процедуры и функции называют рекурсивными?
6. Как отличить «обычную» функцию от рекурсивной?

Задания

Задание 1. Написать программу с подпрограммой-функцией.

Задание 2. Написать программу с подпрограммой-процедурой (с параметром или параметрами).

1. Даны действительные s и t. Рассчитать f(t,-2s,1.17)+f(2.2,t,s-t), где $f(a,b,c)=\frac{2a-b-\sin(c)}{5+|c|}$ подпрограмма.
2. Даны действительные a, b, c. Получить $\frac{\max(a-b,a,a+b)+\max(a,b+c,a-c)}{1+\max(a+bc,1.15,a/c)}$, где max(x,y) подпрограмма.
3. Даны действительные числа S и t. Рассчитать f(t,-3*s,2.5-t)+f(5.2,3*t,s-t), где $f(a,b,c)=\frac{2a-b-\cos(c)}{3.5-|c|}$ подпрограмма.
4. Даны действительные числа S и t. Рассчитать f(1.5*t,2*s)+f(t,3-s)-f(2*s-3,t+s), где $f(a,b)=\frac{a^3-b^2+\sin(ab)}{2ab+5b^2}$ подпрограмма.
5. Даны действительные числа S и t. Рассчитать h(s,t)+h(h2 (s-t,s*t),h4 (s-t,s+t))+h(1,1), где $h(a,b)=\frac{a}{1+b^2}+\frac{b}{1+a^2}-(a-b)^2$ подпрограммы.
6. Даны действительные числа a, b. Рассчитать Y=min(a,b), Y1=min(min(a*b,a+b), min(a2 –b*a,b3-3*a)), Y2=min(Y12+Y,5.25), где min(x,x1) подпрограмма.
7. Даны действительные числа S и t. Рассчитать f(t,4*s,2.5*t)+f(5,2*t,s+t), где $f(a,b,c)=\frac{a^2+b^3+\sqrt{|c|}}{3*|c|}$ подпрограмма.
8. Даны действительные числа S и t. Рассчитать h(s,t)+h^3 (s-t,s*t)+h^2 (s-t,s+t)+h(1,1), где $h(a,b)=\frac{a^2}{1+b}+\frac{b}{1+a^3}+(a-b)^3$ подпрограмма.
9. Даны действительные числа a, b, c. Рассчитать $y=\frac{\max(a,a+b)+\max(a,b+c)}{1+\max(a+b*c,2*a,c*b)}$, где max(x,x1) описать как подпрограмму.
10. Вычислить K = (x + y + z) / 3, где x – наибольшее значение из параметров x1, x2, y - из y1, y2, z – из z1, z2. Использовать подпрограмму для нахождения наибольшего значения из двух параметров.

11. В порт в среднем приходят 3 корабля в день. Какова вероятность того, что в день придет 2 корабля, 4 корабля? Вероятность вычислять по формуле P = 3 * e - 3 / к!. Использовать подпрограмму для расчета к!.

12. Вычислить Z = (v1 + v2 + v3) / 3, где v1,v2,v3 – объемы шаров с радиусами r1, r2, r3 соответственно. Использовать подпрограмму для расчета объёмов шаров. Объем шара вычислять по формуле V = 4 / 3*Пi*R3.

13. Определить число сочетаний из n по m (n>m), по формуле C=n!/m!(n-m). Использовать одну подпрограмму для расчета n! и m!.

14. Вычислить Z = (n + m) / 2, где n – наименьшее значение из параметров n1, n2, m – из m1, m2. Использовать подпрограмму для нахождения наименьшего значения из двух параметров.

15. Составить программу вычисления значения функции S = x2 + y2 + sin(2 * x2 * y2) + x + z + sin(2 * x * z) + y2 + z2 + sin(2 * y2 * z2) , используя подпрограмму для расчета a + b + sin(2 * a * b).