

ПРОЦЕДУРЫ И ФУНКЦИИ В ПАСКАЛЕ

Подпрограмма – это часть программы, оформленная в виде отдельной синтаксической конструкции и снабженная именем.

Процедуры

Довольно часто уже на этапе разработки алгоритма программы можно обнаружить, что некоторые одинаковые или очень похожие действия в программе должны выполняться несколько раз. Текст программы, реализующей такой алгоритм, будет содержать последовательности одинаковых инструкций.

Избежать дублирования инструкций можно, если повторяющиеся инструкции удалить из текста программы и оформить их как *процедуру*, а в то место программы, где раньше были удаленные и теперь оформленные как процедура инструкции, поместить инструкцию *вызова* процедуры.

Процедурой называется имеющая имя последовательность инструкций, предназначенная для решения определенной задачи.

Объявление процедуры программиста в общем виде :

```
procedure Имя(var параметр1:тип1;var параметр2:тип2;...;  
var параметрK.:типK);      (формальные параметры)  
var  
{здесь объявления внутренних переменных процедуры}  
begin  
{здесь инструкции процедуры}  
end;
```

Указанные в объявлении процедуры параметры являются **формальными параметрами**.

Слово **var**, стоящее перед именем параметра, не является обязательным. *Однако если оно стоит, то это означает, что при вызове процедуры параметром должна быть переменная основной программы, при этом изменение параметра внутри процедуры приведет к изменению значения переменной основной программы, указанной в качестве фактического параметра при вызове процедуры.*

Использование параметров-переменных позволяет использовать процедуры для изменения значений переменных основной программы.

Инструкция вызова процедуры в общем виде:

Имя(СписокПараметров);

где Имя — имя вызываемой процедуры;

СписокПараметров — разделенные запятыми фактические параметры, в качестве которых, в зависимости от описания параметров в объявлении процедуры, могут быть использованы константа, переменная или выражение.

Прочтите

Программа на языке Pascal состоит из основной программы и, возможно, процедур и функций программиста. Каждая из них содержит раздел объявления переменных.

Переменные, объявленные в основной программе, доступны всем инструкциям программы, в том числе и инструкциям процедур и функций программиста. Такие переменные называются *глобальными*.

Переменные, объявленные в процедуре или функции программиста, называются *локальными*. Локальные переменные доступны только инструкциям той подпрограммы (процедуры или функции), в которой они объявлены.

ПРОЦЕДУРЫ И ФУНКЦИИ В ПАСКАЛЕ

Запишите

Глобальные переменные – это переменные, объявленные в описании основной части программы и действующие в любой ее части.

Локальные переменные – те, которые объявлены в подпрограмме (процедуре или функции) и действующие лишь в ней.

Фактические параметры – это переменные, которые передаются процедуре при обращении к ней.

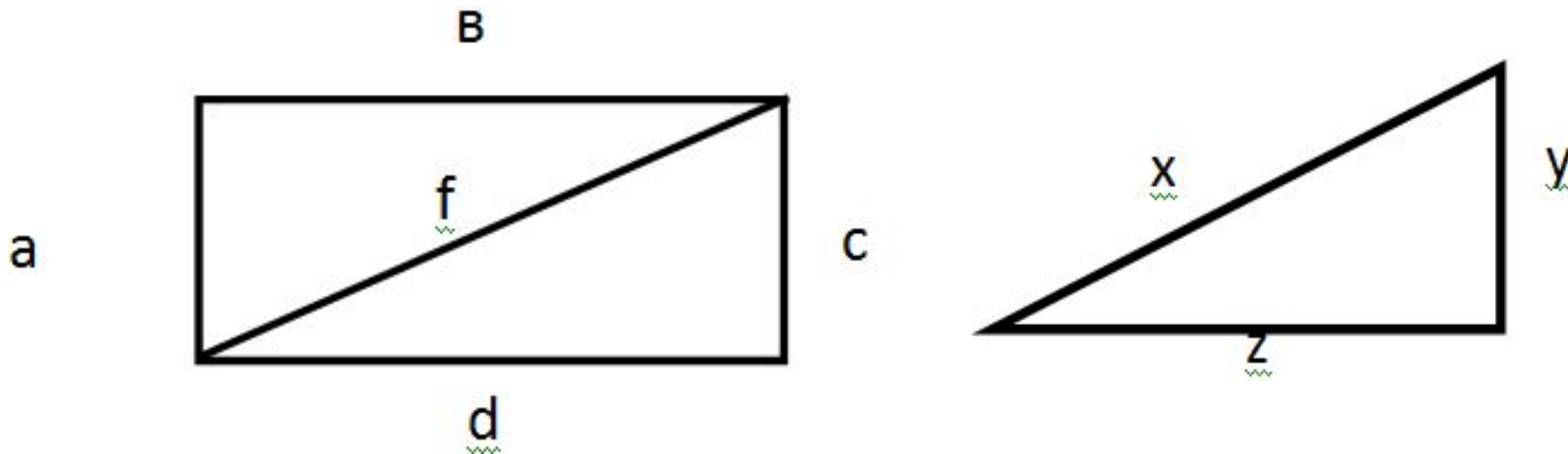
Формальные параметры – это переменные фиктивно присутствующие в процедуре и определяющие тип и место подстановки фактических параметров, над которыми производятся действия.

Число и тип формальных и фактический параметров должны совпадать с точностью до их следования.

Пример. Вычисление площади круга и длины окружности по значению радиуса.
Используется процедура программиста Окр.

```
program prozedura1;  
Var                {переменные основной процедуры}  
    r,l,s:real;     { радиус, длина и площадь окружности }  
{Процедура программиста}  
procedure Окр(r:real; var s:real; var l:real);    {r – радиус (задаем, поэтому var не пишем), s – пл.  
круга, l - длина окр. Вычисляем, пишем var}  
    begin  
        s:=pi*r*r;  
        l:=2*pi*r;  
    end;  
{ Основная программа }  
begin  
    writeln('Вычисление длины окружности и площади круга:');  
    write('Задайте радиус и нажмите <Enter> ');  
    readln(r);  
    l:=0;  
    s:=0;  
    Окр(r,s,l); {Вызов процедуры, переменные программы – фактические параметры процедуры}  
    writeln('Радиус окружности: ',r:6:3);  
    writeln('Длина: ',l:6:3,', площадь: ',s:7:3);  
end.
```

Пример. Четырехугольник задан четырьмя своими сторонами a , b , c , d , и диагональю f . С помощью процедуры вычисления площади треугольника по трем сторонам, вычислить площадь заданного четырехугольника.



Площадь треугольника определим по формуле Герона:

$$S = \sqrt{p(p-x)(p-y)(p-z)} \quad \text{х, у, z — стороны треугольника}$$

p — полупериметр, который вычисляется по формуле:

$$p = \frac{x+y+z}{2}$$

Program pl;

Var a, b, c, d, f, s1, s2, s: real; (фактические параметры)

Procedure treug(x, y, z: real; var s: real); (формальные параметры)

Var p: real;

Begin

P := (x + y + z)/2;

S := sqrt(p*(p - x)*(p - y)*(p - z));

End;

Begin

Writeln('Введите стороны четырехугольника и диагональ: ');

Readln(a, b, c, d, f); (фактические параметры)

Treug(a, b, f, s1); {2 стороны треугольника a и b, диагональ f и площадь s}

Treug(c, d, f, s2); {2 стороны треугольника c и d, диагональ f и площадь s}

s := s1 + s2;

Write('Площадь четырехугольника = ', s:5:2);

End.

Введите стороны 4-х
угольника и диагональ:
4 2.5 6 3.5 4.5
Площадь 4-хугольника =
12.80

Функция – это подпрограмма, предназначенная для того, чтобы вычислять только одно значение.

Также функции отличаются от процедур:

- Заголовком;
- В теле функции обязательно должен присутствовать оператор присваивания, где в левой части стоит имя функции, а в правой – ее значение. Иначе, значение не будет определено;
- Обращением к функции не оператор, а выражение.

```
Function <имя >( <список формальных параметров> ): <тип результата>;  
    <раздел описаний>  
begin  
    <тело функции>  
    <имя>:=<значение>;  
end;
```

Пример. Четырехугольник задан четырьмя своими сторонами a , b , c , d , и диагональю f . С помощью функции вычисления площади треугольника по трем сторонам, вычислить площадь заданного четырехугольника.

Пример. Четырехугольник задан четырьмя своими сторонами a, b, c, d , и диагональю f . С помощью функции вычисления площади треугольника по трем сторонам, вычислить площадь заданного четырехугольника.

Program pl2;

Var a, b, c, d, f, s: real;

Function PL_t(x, y, z: real):real; входные параметры

Var p: real;

Begin

P := (x + y + z)/2;

Pl_t := sqrt(p*(p - x)*(p - y)*(p - z));

End;

Begin

Writeln('Введите стороны четырехугольника и диагональ: ');

Readln(a, b, c, d, f);

S := PL_t(a,b,f)+PL_t(c,d,f);

Write('Площадь четырехугольника = ', s:5:2);

End.

Пример: Написать программу для нахождения значения функции $y = \sin x - 1$, где x изменяется от $\pi/2$ до 2π с шагом $\pi/12$. Вычисление y оформить в виде функции.

```
program f_2;      { Вычисление значений y }
var x,y:real;
function f(x:real):real;
begin
  f:=sin(x)-1;
end;
begin
writeln(' x      y');
writeln('-----');
x := pi/2;»
repeat          { повторять до тех пор, пока }
  y:=f(x);
  writeln(x:6:2,y:15:6);
  x := x+pi/12;
until x>2*pi;  { не станет истинным условие }
writeln('-----');
readln;
end.
```

| x | y |
|------|-----------|
| 1.57 | 0.000000 |
| 1.83 | -0.034074 |
| 2.09 | -0.133975 |
| 2.36 | -0.292893 |
| 2.62 | -0.500000 |
| 2.88 | -0.741181 |
| 3.14 | -1.000000 |
| 3.40 | -1.258819 |
| 3.67 | -1.500000 |
| 3.93 | -1.707107 |

Рекурсия в Паскале

Алгоритм называется **рекурсивным**, если в качестве вспомогательного алгоритма (подпрограммы) он использует самого себя.

$$n! = 1 * 2 * 3 * \dots * n$$

$$n! \begin{cases} 1, & \text{если } n=1 \\ n*(n-1)!, & \text{если } n>1 \end{cases}$$

Процедуры и функции, использующие вызовы самих себя, называют рекурсивными (прямая рекурсия).

Пример. Рассмотрим пример рекурсивной функции вычисления x^n , где n - натуральное число.

$$\text{Вспользуемся известным фактом: } x^n = \begin{cases} 1, & \text{при } n = 0 \\ x^{n-1} * x, & \text{при } n \geq 1 \end{cases}$$

```
function deg(x, n: integer): longint;  
begin  
  if n = 0 then deg:=1  
  else deg:=deg(x, n-1)*x;  
end;
```