# Рекурсия в Pascal.

***Рекурсия*** **—** это такой способ организации вспомогательного алгоритма (подпрограммы), при котором эта подпрограмма (процедура или функция) в ходе выполнения ее операторов обращается сама к себе.

В рекурсивном определении должно присутствовать ограничение, граничное условие, при выходе на которое дальнейшая инициация рекурсивных обращений прекращается.

Для того чтобы такое обращение не было бесконечным, в тексте подпрограммы должно быть условие, по достижению которого дальнейшее обращение к подпрограмме не происходит.

```
procedure Rec(n: integer; var y: integer);
begin
  if <условие> then y:= <значение>
    else
Begin
Rec(n-1, y); <операторы>;
end;
end;
```

**Пример 1. Определение факториала**

Факториал определяется так:

$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$

Рекурсивное определение:

$$n! = \begin{cases} 1, & \text{если } n = 0; \\ n \cdot (n-1)!, & \text{если } n > 0. \end{cases}$$

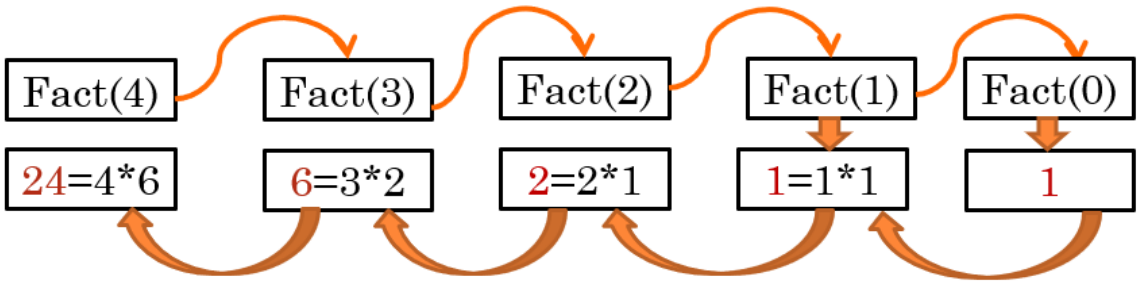
Граничным условием в данном случае является n=0.

**Функция на Pascal**

```
Function Fact (N:integer):longint;
Begin
     If N=0
    Then Fact:=1
    Else Fact:=Fact (N-1)*N;
 End;
```

**Процедура на Pascal**

```
Procedure Factorial(N:integer; Var F:longint);
Begin
      If N=0  Then F:=1
      Else
Begin
 Factorial(N-1, F); F:=F*N;
End;
End;
```




**Пример 2. Числа Фибоначчи**

***Числа Фибоначчи*** *– это элементы числовой последовательности 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …, в которой каждое последующее число равно сумме двух предыдущих.*

Вывести на экран ряд чисел Фибоначчи, состоящий из n элементов.

Рекурсивное определение для вычисления чисел Фибоначчи выглядит следующим образом:

$$f(n) = \begin{cases} 1, \text{если } n = 1 \\ 1, \text{если } n = 2 \\ f(n-1) + f(n-2), \text{ если } n > 2 \end{cases}$$

```
function fib(n: Integer) :  longint;
begin
      if n <= 2 Then
            fib:= 1
      else
            fib:= fib(n-1) + fib(n-2);
      end;
```

Вызов функции в программе:

```
for i:=1 to n do begin
        A:= fib (i);
        write (' ',a );
end;
```

**Пример 3. Нахождение НОД двух натуральных чисел**

Даны два натуральных числа *a* и *b*.

Если *a* = *b*, то нод (*a ,b)=a.*

Если *a* >*b*, то нод (*a ,b)*= нод (*a -b ,b).*

Если *a* < *b*, то нод (*a ,b)*= нод (*a ,b-a).*

**Пример 4.** Написать программу вычисления степени с натуральным показателем.

Дано: основание степени **x**

Показатель степени **k**

Если k=0, тогда **степень(k,x)**=1,

Иначе

**степень (k,x)**= x· **степень (k-1,x)**

**Пример 5.** Не используя операции умножения и деления, рекурсивно описать функцию M(a,b) от целых чисел a и b (a≥0, b>0), которая вычисляет остаток от деления a на b, т.е. M(a,b)=a mod b.

**Пример 6.** Описать рекурсивную функцию degree5(N), которая вычисляет, какой степенью числа 5 является натуральное число N. Если N не степень пяти, функция должна вернуть число -1. Например, degree5(50) = -1, degree5(125) =3, degree5(5) =1, degree5(1) =0.

**Пример 7.** Рекурсивно описать функцию *maxdig*(*N*), которая находит наибольшую цифру в десятичной записи неотрицательного целого числа *N*. Например, *maxdig*(27306) = 7.