C 语言函数的递归调用

一个函数在它的函数体内调用它自身称为递归调用，这种函数称为递归函数。执行递归函数将反复调用其自身，每调用一次就进入新的一层。

【示例】用递归计算 n!。阶乘 n! 的计算公式如下：

$$n! = \begin{cases} 1 & (n = 0, 1) \\ n * (n-1)! & (n > 1) \end{cases}$$

根据公式编程：

```
long factorial(int n){
    long result;

    if(n==0 || n==1){
        result = 1;
    }else{
        result = factorial(n-1) * n;  // 递归调用
    }

    return result;
}
```

这是一个典型的递归函数。调用 factorial 后即进入函数体，只有当 n==0 或 n==1 时函数才会执行结束，否则就一直调用它自身。

由于每次调用的实参为 n-1，即把 n-1 的值赋给形参 n，所以每次递归实参的值都减 1，直到最后 n-1 的值为 1 时再作递归调用，形参 n 的值也为 1，递归就终止了，会逐层退出。

例如求 5!，即调用 factorial(5)。当进入 factorial 函数体后，由于 n=5，不等于 0 或 1，所以执行 result = factorial(n-1) * n;，即 result = factorial(5-1) * 5;，接下来也就是调用 factorial(4)。这是第一次递归。

进行四次递归调用后，实参的值为 1，也就是调用 factorial(1)。这时递归就结束了，开始逐层返回。factorial(1) 的值为 1，factorial(2) 的值为 1*2=2，factorial(3) 的值为 2*3=6，factorial(4) 的值为 6*4=24，最后返回值 factorial(5) 为 24*5=120。

注意：为了防止递归调用无终止地进行，必须在函数内有终止递归调用的手段。常用的办法是加条件判断，满足某种条件后就不再作递归调用，然后逐层返回。

递归调用不但难于理解，而且开销很大，如非必要，不推荐使用递归。很多递归调用可以用迭代（循环）来代替。

【示例】用迭代法求 n!。

```
long factorial(int n){
    int i;
    long result=1;

    if(n==0 || n==1){
        return 1;
    }

    for(i=1; i<=n; i++){
        result *= i;
    }
    return result;
}
```