递归 — 基本思想

郭 炜　刘家瑛

北京大学

递归的基本思想

- 什么是递归
  - 递归 — 某个函数直接或间接的调用自身
  - 问题的求解过程

  →划分成许多相同性质的子问题的求解

  →而小问题的求解过程可以很容易的求出
  - 这些子问题的解就构成里原问题的解

递归的基本思想

- 总体思想
  - 待求解问题的解 → 输入变量x的函数f(x)
  - 通过寻找函数g( ), 使得f(x) = g(f(x-1))
  - 且已知f(0)的值, 就可以通过f(0)和g( )求出f(x)的值
- 推广
  - 扩展到多个输入变量*x*, *y*, *z*等, *x*-1也可以推广到 $x - x_1$,

只要递归朝着 “出口” 的方向即可

递归与枚举的区别

- 枚举:

  把一个问题划分成一组子问题, 依次对这些子问题求解

  - 子问题之间是横向的, 同类的关系

- 递归:

  把一个问题逐级分解成子问题

  - 子问题与原问题之间是纵向的, 同类的关系
  - 语法形式上: 在一个函数的运行过程中, 调用这个函数自己
    - 直接调用: 在fun()中直接执行fun()
    - 间接调用: 在fun1()中执行fun2(); 在fun2()中又执行fun1()

递归的三个要点

- 递归式:

  如何将原问题划分成子问题

- 递归出口:

  递归终止的条件, 即最小子问题的求解,可以允许多个出口

- 界函数:

  问题规模变化的函数, 它保证递归的规模向出口条件靠拢

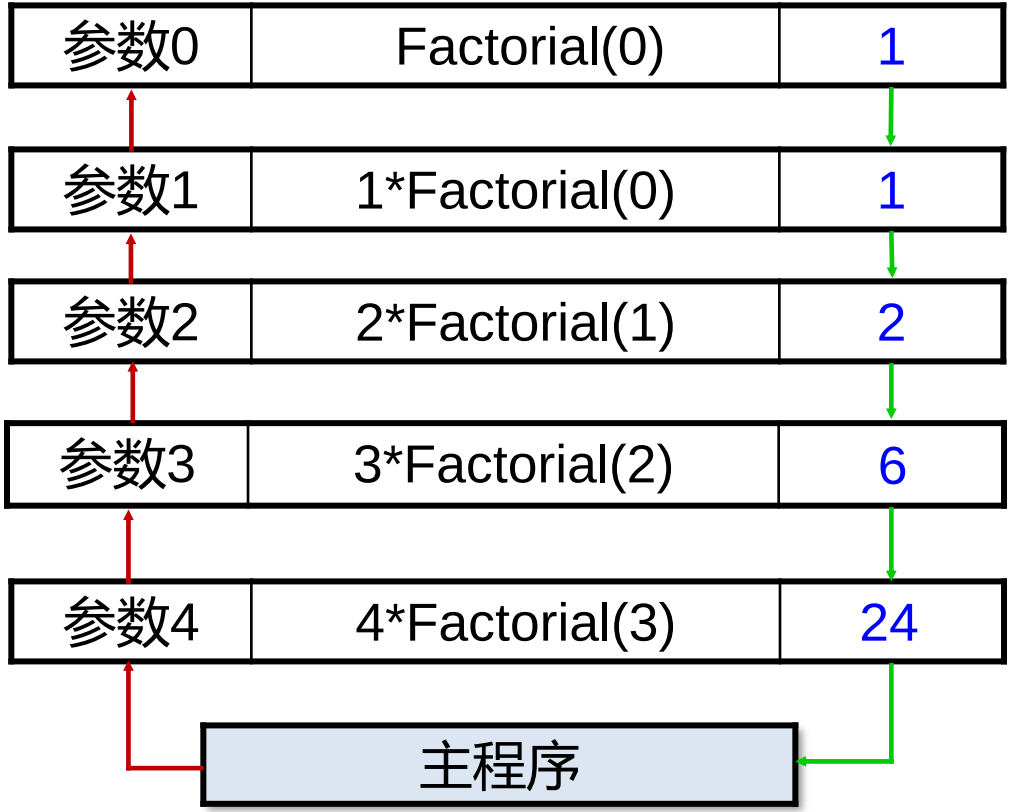
求阶乘的递归程序

- 给定n, 求阶乘n!

```
int n, m=1;
for (int i=2; i<=n; i++)
    m *= i;
printf(“%d的阶乘是%d\n”,
n, m);
```

```
int Factorial(int n){
    if (n == 0)
        return 1;
    else
        return  n * Factorial(n - 1);
}
```

阶乘的栈


参数0
Factorial(0)
1
参数1
1*Factorial(0)
1
参数2
2*Factorial(1)
2
参数3
3*Factorial(2)
6
参数4
4*Factorial(3)
24
主程序

递归解决问题的关键

1) 找出递推公式

2) 找到递归终止条件

注意事项: 由于函数的局部变量是存在栈上的
如果有体积大的局部变量, 比如数组,
而递归层次可能很深的情况下, 也许会导致栈溢出
→可以考虑使用全局数组或动态分配数组