第6章 函数

——递归函数与函数调用栈

本节要讨论的主要问题

- 递归函数的调用过程是如何执行的？
- 递归方法编写程序的优缺点是什么？

递归的基本思想

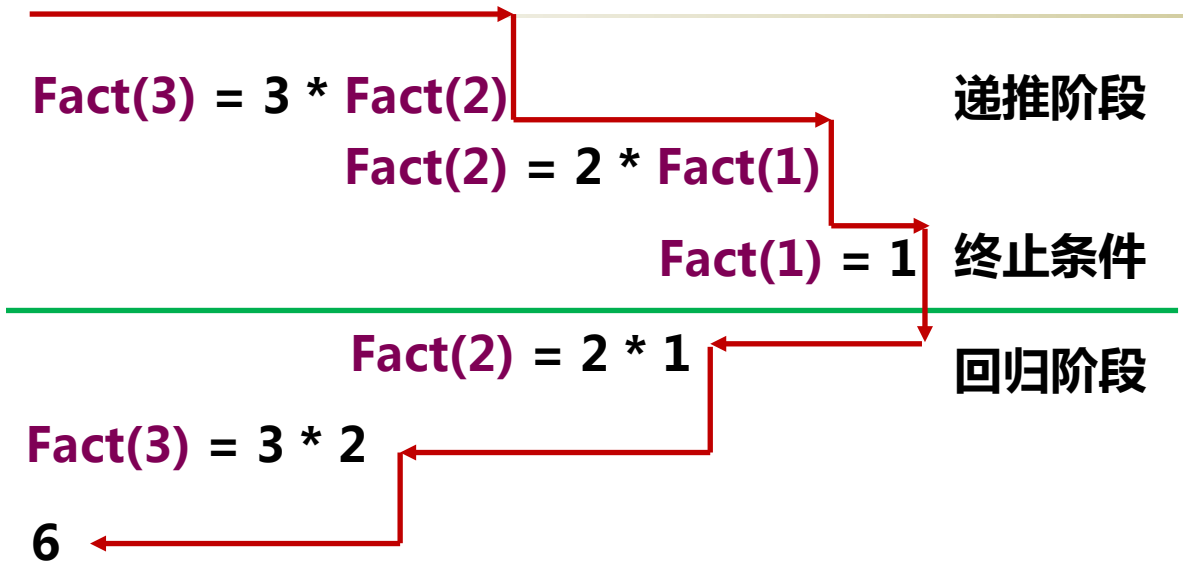
递归的一般条件

把规模较大的，较难解决的问题转化成规模较小的、易于解决的同类子问题。

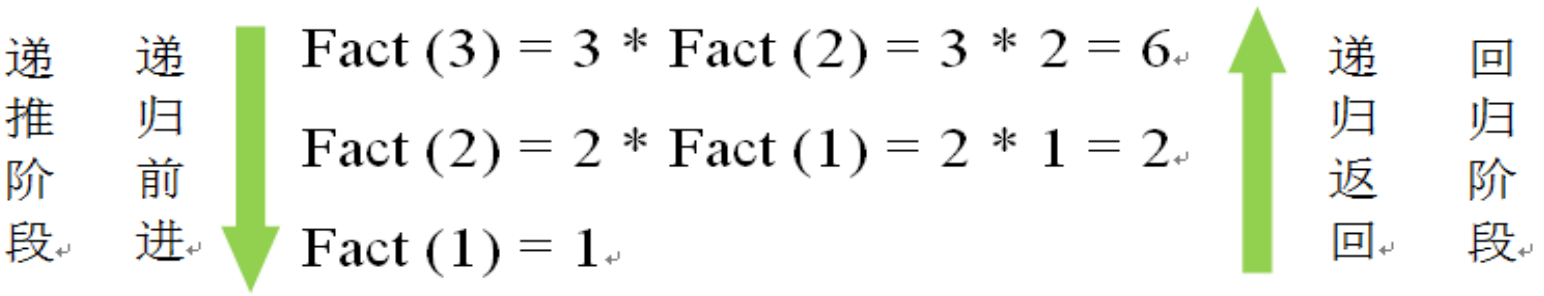
规模较小的子问题又转化为规模更小的子问题，且小到一定程度可以直接得出它的解，从而得到原始问题的解

递归的基本条件（终止条件，出口）

递归执行的两个阶段




```
unsigned long Fact(unsigned int n)
{
  if (n==0 || n==1)
      return 1;
  else
      return  n * Fact(n-1);
}
```



递归函数的调用过程

main()

```
Fact(3)
{
  if  (3==0 || 3==1)
      return 1;
  else
      return 3 * Fact(3-1);
}
```

递推阶段（递归前进阶段）

```
Fact(2)
{
  if (2==0 || 2==1)
      return 1;
  else
      return 2 * Fact(2-1);
}
```

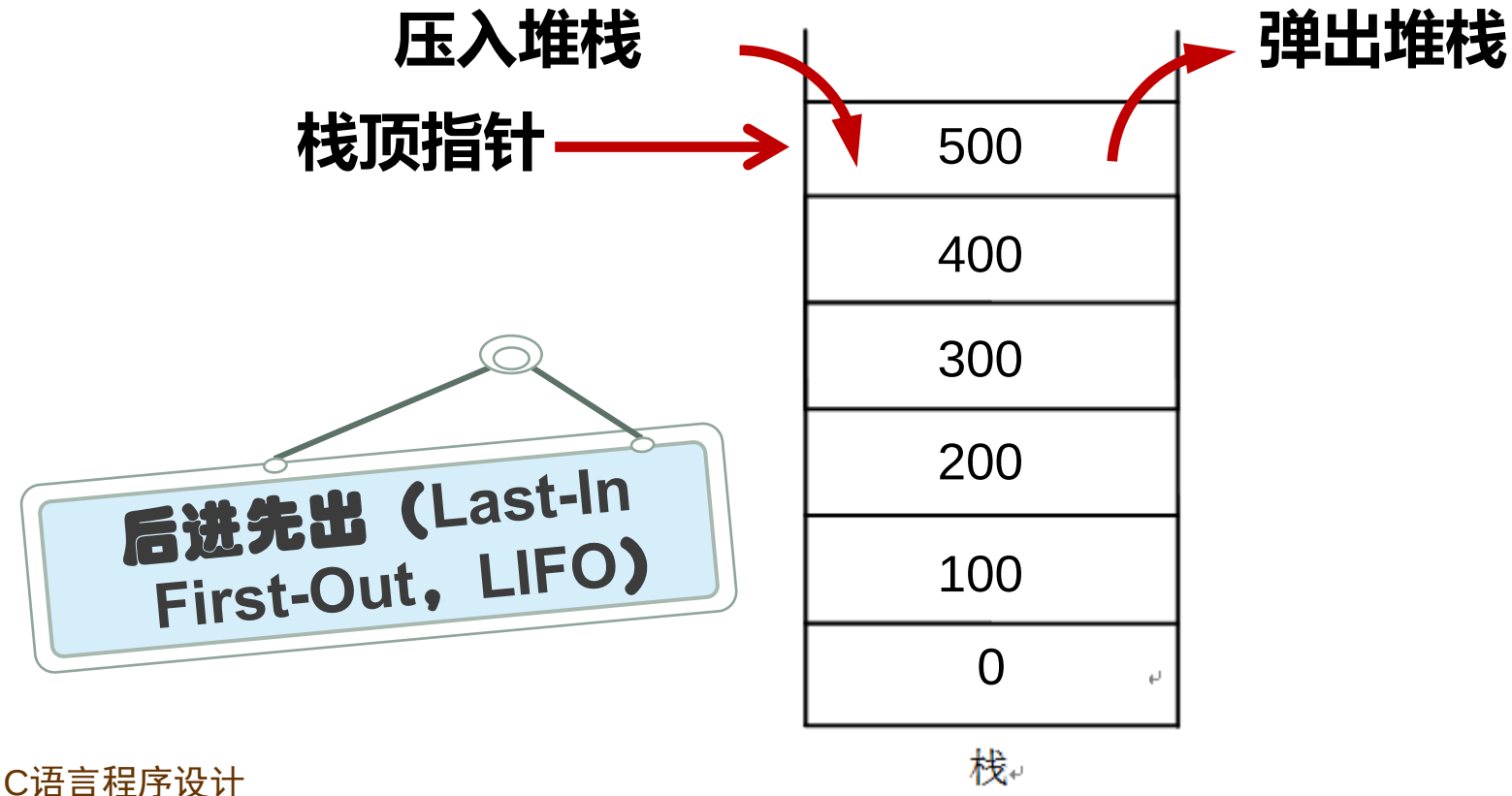
```
Fact(1)
{
  if (1==0 || 1==1)
      return 1;
  else
      return 1 * Fact(1-1);
}
```

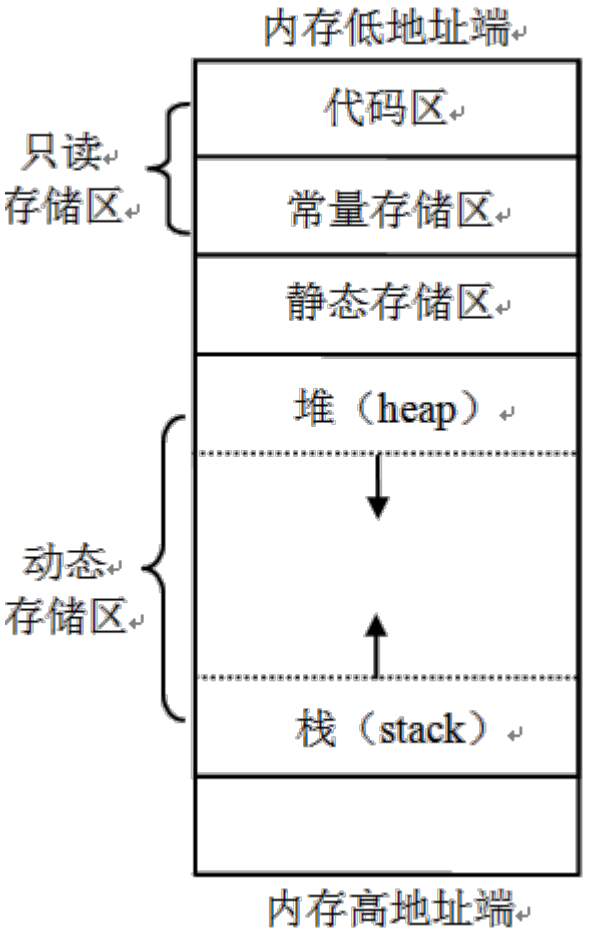
```
unsigned long Fact(unsigned int n)
{
  if (n==0 || n==1)
      return 1;
  else
      return  n * Fact(n-1);
}
```

回归阶段（递归返回阶段）

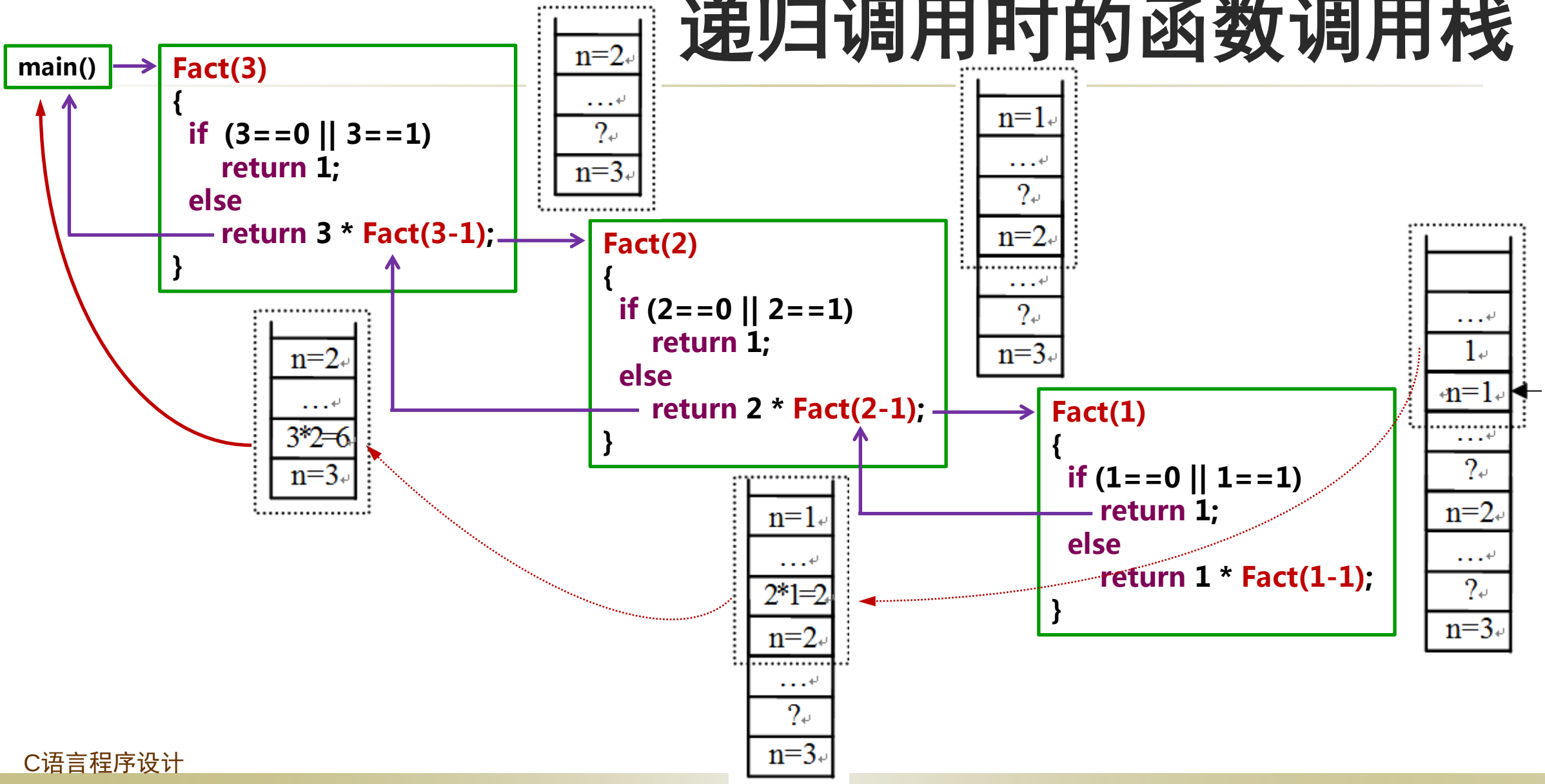
C程序的内存映像

- **堆栈溢出（Stack Overflow）**
  - 往堆栈中存入的数据超出预先给堆栈分配的容量





递归调用时的函数调用栈

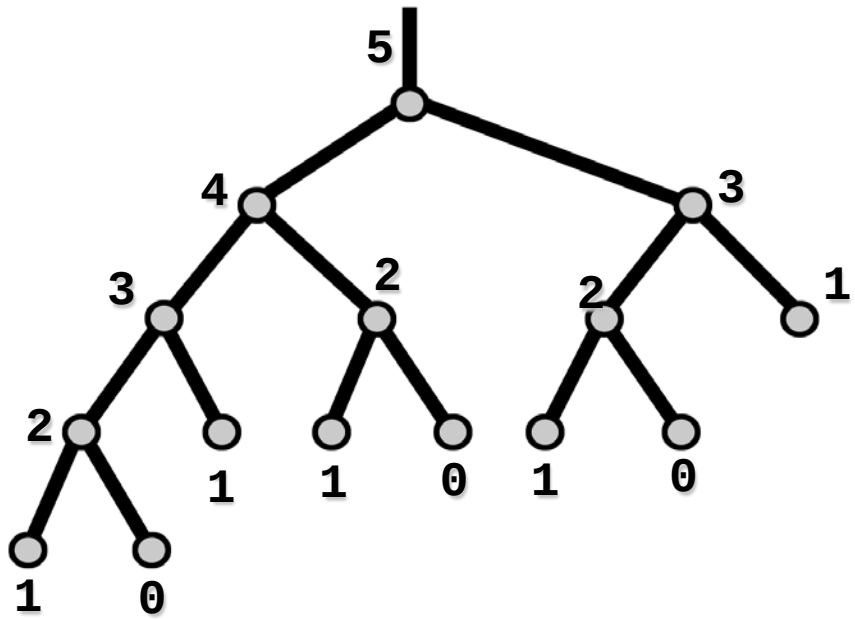


用递归法计算Fibonacci数列的第n项

$$fib(n)=\begin{cases}0 & n=0\\ 1 & n=1\\ fib(n-1)+fib(n-2) & n>1\end{cases}$$

0，1，1，2，3，5，8，.....

```
long Fib(int n)
{
  long f;
  if (n == 0)        f = 0;
  else if (n == 1)   f = 1;
  else     f = Fib(n-1) + Fib(n-2);
  return f;
}
```



计算Fib(5)需15次Fib调用

递归方法的优缺点

- 优点
  - 简洁、直观、精炼，易编、易懂、逻辑清楚，结构清晰、可读性好，更符合人的思维习惯，逼近数学公式的表示
- 缺点
  - 函数调用开销大，耗费更多的时间和栈空间，时空效率偏低
  - 易产生大量的重复计算

讨论

- 编程计算Fibonacci数列各项的平方和，即

$$F_1^2 + F_2^2 + \ldots + F_n^2$$

- 从程序的运行结果中，你发现了Fibonacci数列具有的一个什么性质？你能不用编程的方法直接得到这个答案吗？