

第6章 函数

——函数的嵌套调用与递归调用

本节要讨论的主要问题

- 什么是嵌套调用和递归调用？
- 递归函数的两个基本要素是什么？
- 递归调用可以终止的条件是什么？
- 什么情况下考虑使用递归？

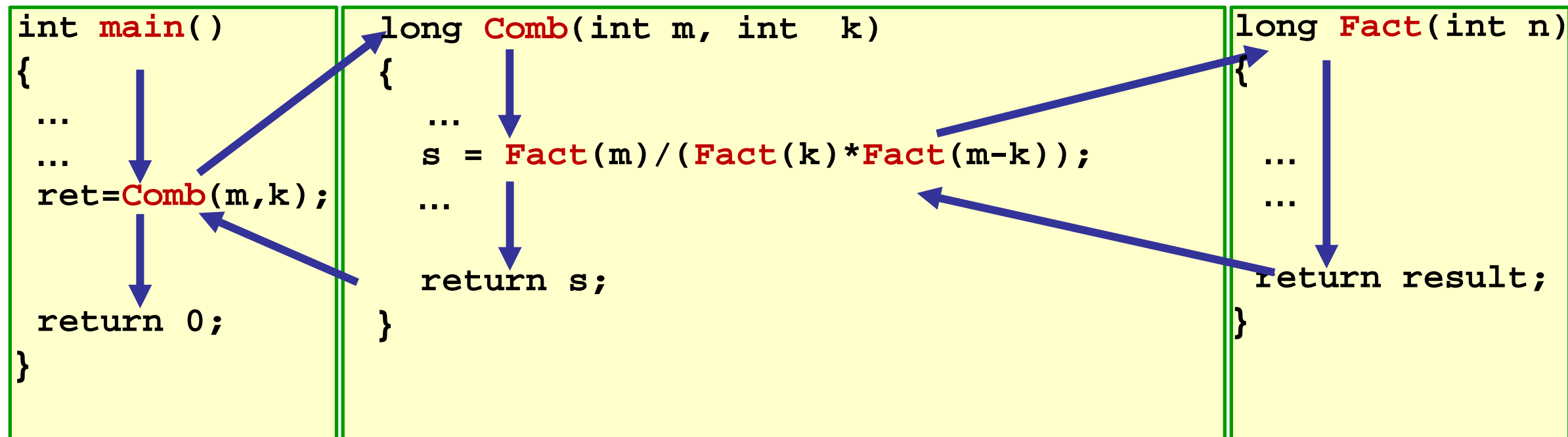


函数的嵌套调用

- 函数不能嵌套定义
- 但可以嵌套调用

— 在调用一个函数的过程中又调用另一个函数

$$C_m^k = \frac{m!}{k!(m-k)!}$$

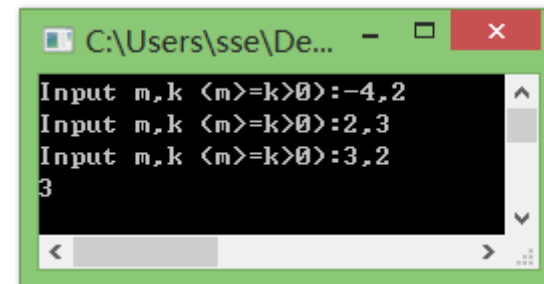


■ 计算组合数 $C_m^k = \frac{m!}{k!(m-k)!}$

```
#include <stdio.h>
long Fact(int n);
long Comb(int m, int k);
int main()
{
    int m, k;
    do{
        printf("Input m,k (m>=k>0):");
        scanf("%d,%d", &m, &k);
    }while (m<k || m<0 || k<0);
    printf("%ld\n", Comb(m, k));
    return 0;
}
```

```
long Comb(int m, int k)
{
    return Fact(m)/(Fact(k)*Fact(m-k));
}
```

函数的嵌套调用



```
long Fact(int n)
{
    int i;
    long result = 1;
    for (i=2; i<=n; i++)
    {
        result *= i;
    }
    return result;
}
```

复用 (Reuse)

函数的递归调用

- 函数直接或间接调用自己，称为递归调用（Recursive Call）
- 这样的函数，称为递归函数（Recursive Function）

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

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

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

```
long Fact(int n)
{
    int i;
    long result = 1;
    for (i=2; i<=n; i++)
    {
        result *= i;
    }
    return result;
}
```

递归函数

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

基本条件(base case)

一般条件(general case)

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

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

递归函数

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

基本条件控制递归调用结束

一般条件控制递归调用向基本条件转化

```
.....
if (基本条件)
    return 递归公式的初值;
else
    return 递归函数调用的返回值;
```

条件递归

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

问题：如果没有基本条件或者一般条件不能最终转化为基本条件会怎样？

生活中的递归实例

■ 老和尚讲故事——无穷递归

- * 从前有座山，山上有座庙，庙里有个老和尚，老和尚给小和尚讲故事，故事说
 - * “从前有座山，山上有座庙，庙里有个老和尚，老和尚给小和尚讲故事，故事说.....”
 - * “从前有座山，山上有座庙，庙里有个老和尚，老和尚给小和尚讲故事，故事说.....”

■ 字典

- * 使用者读懂字典中全部词汇的释义的前提是必须掌握一些少量的、非常基础的词汇的含义，否则...

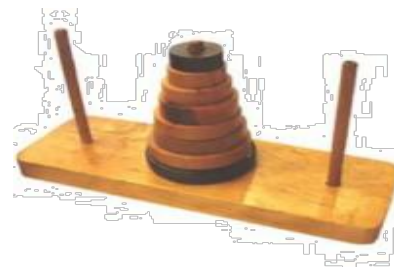
什么情况下考虑使用递归？

- 通常下面三种情况需要使用递归：
 - * **数学定义**是递归的
 - * 如计算阶乘，最大公约数和Fibonacci数列
 - * **数据结构**是递归的
 - * 如队列、链表、树和图
 - * **问题的解法**是递归的
 - * 如Hanoi塔，骑士游历、八皇后问题（回溯法）

一个经典的递归问题

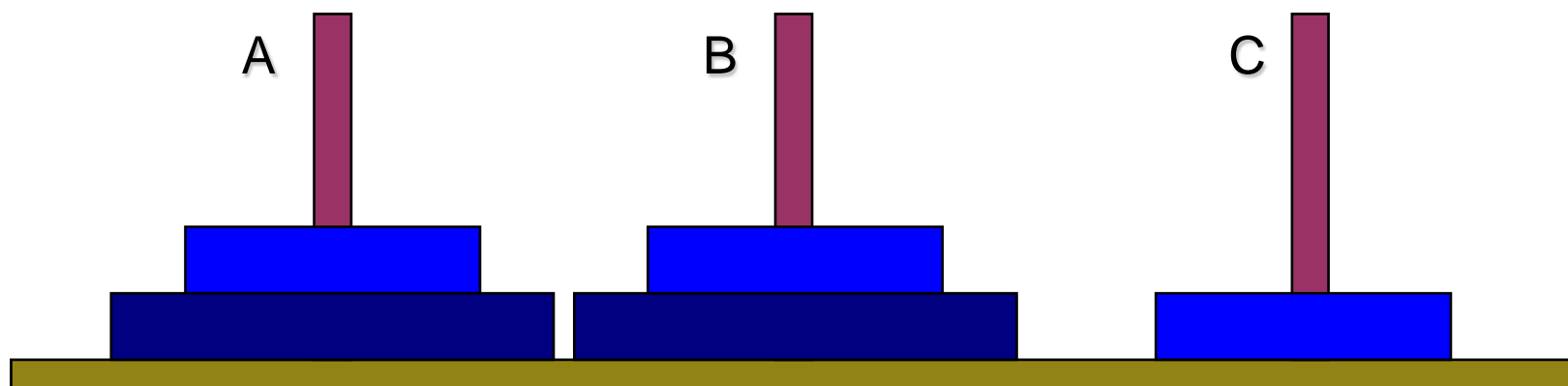
■ 汉诺塔（Hanoi）问题

- * 印度神话，上帝创造世界时作了三根金刚石柱子，第一根上从下往上按大小顺序摞着64片黄金圆盘，上帝命令婆罗门把圆盘从下开始按大小顺序重新摆放到第二根上，规定每次只能移动一个圆盘，在小圆盘上不能放大圆盘
- * 当 $n=64$ 时，需移动多少次呢？
 - * 18446744073709551615，即1844亿亿次
 - * 若按每次耗时1微秒计算，则64个圆盘的移动需60万年



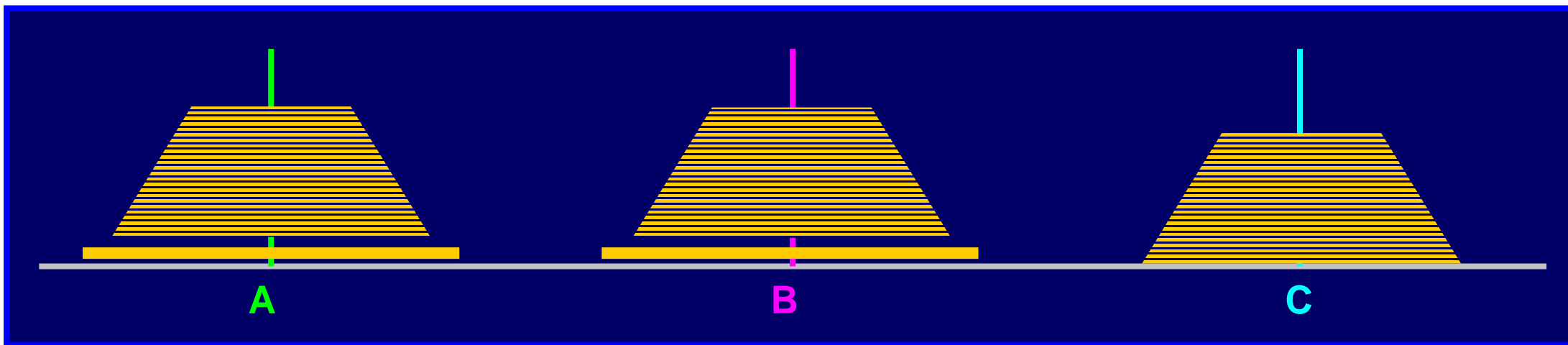
汉诺塔（Hanoi）问题

■ 较为简单的情形—— $n=2$



- * 将1号圆盘从A移到C
- * 将2号圆盘从A移到B
- * 将1号圆盘从C移到B

汉诺塔问题的递归求解



■ 数学归纳法

- 假设 $n-1$ 个圆盘的汉诺塔问题已解决
- 将“上面 $n-1$ 个圆盘”看成一个整体

移动 n 个圆盘



移动 $n-1$ 个圆盘

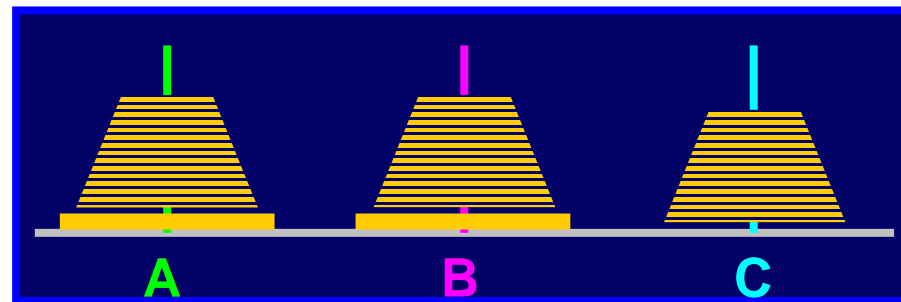
- ✓ 将“上面 $n-1$ 个圆盘”从A移到C
- ✓ 将第 n 号圆盘从A移到B
- ✓ 将“上面 $n-1$ 个圆盘”从C移到B



汉诺塔问题的递归函数实现

✓ 将“n个圆盘”借助于C从A移到B

```
void Hanoi(int n, char a, char b, char c)
{
    if (n == 1)
    {
        Move(n, a, b);
    }
    else
    {
        Hanoi(n-1, a, c, b);
        Move(n, a, b);
        Hanoi(n-1, c, b, a);
    }
}
```




将“上面n-1个圆盘”从A移到C
将第n号圆盘从A移到B
将“上面n-1个圆盘”从C移到B

汉诺塔问题的递归函数实现

✓ 将“n个圆盘”借助于C从A移到B

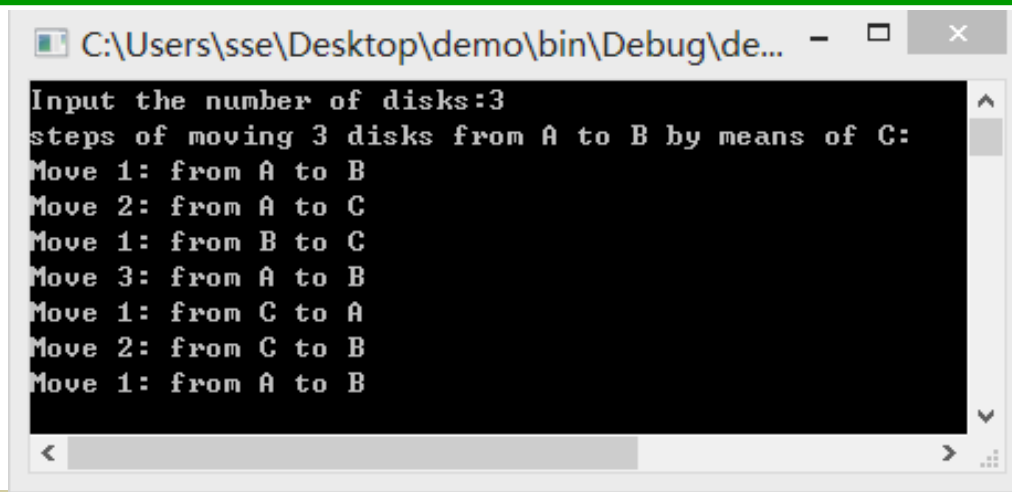
```
void Hanoi(int n, char a, char b, char c)
{
    if (n == 1)
    {
        Move(n, a, b);
    }
    else
    {
        Hanoi(n-1, a, c, b);
        Move(n, a, b);
        Hanoi(n-1, c, b, a);
    }
}
```

```
void Move(int n, char a, char b)
{
    printf("Move %d: from %c to %c\n", n, a, b);
}
```



将“上面n-1个圆盘”从A移到C
将第n号圆盘从A移到B
将“上面n-1个圆盘”从C移到B

```
#include <stdio.h>
void Hanoi(int n, char a, char b, char c);
void Move(int n, char a, char b);
int main()
{
    int n;
    printf("Input the number of disks:");
    scanf("%d", &n);
    printf("steps of moving %d disks from A to B by means of C:\n", n);
    Hanoi(n, 'A', 'B', 'C');
    return 0;
}
```



```
C:\Users\sse\Desktop\demo\bin\Debug\de...
Input the number of disks:3
steps of moving 3 disks from A to B by means of C:
Move 1: from A to B
Move 2: from A to C
Move 1: from B to C
Move 3: from A to B
Move 1: from C to A
Move 2: from C to B
Move 1: from A to B
```

讨论

- 汉诺塔问题可以用非递归的方法求解吗？
- 你能举出生活中递归的例子吗？

