



程序设计与算法(二)

算法基础

郭 炜

微信公众号



微博: <http://weibo.com/guoweiofpku>

学会程序和算法，走遍天下都不怕!

讲义照片均为郭炜拍摄



北京大学
PEKING UNIVERSITY

信息科学技术学院

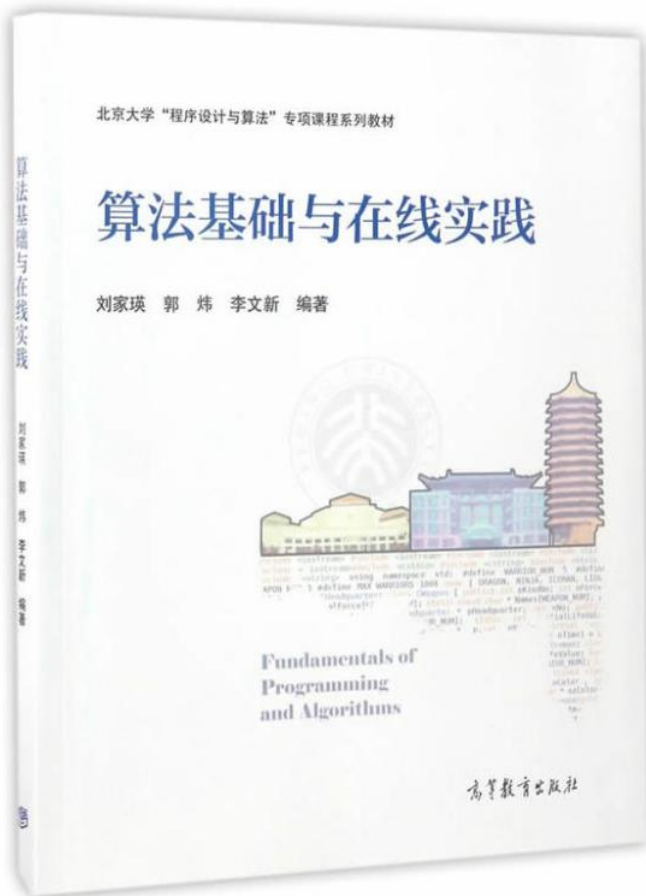
配套教材：

高等教育出版社

《算法基础与在线实践》

刘家瑛 郭炜 李文新 编著

本讲义中所有例题，根据题目名称在
<http://openjudge.cn>
“百练”组进行搜索即可提交





递 归(一)

递归的基本概念

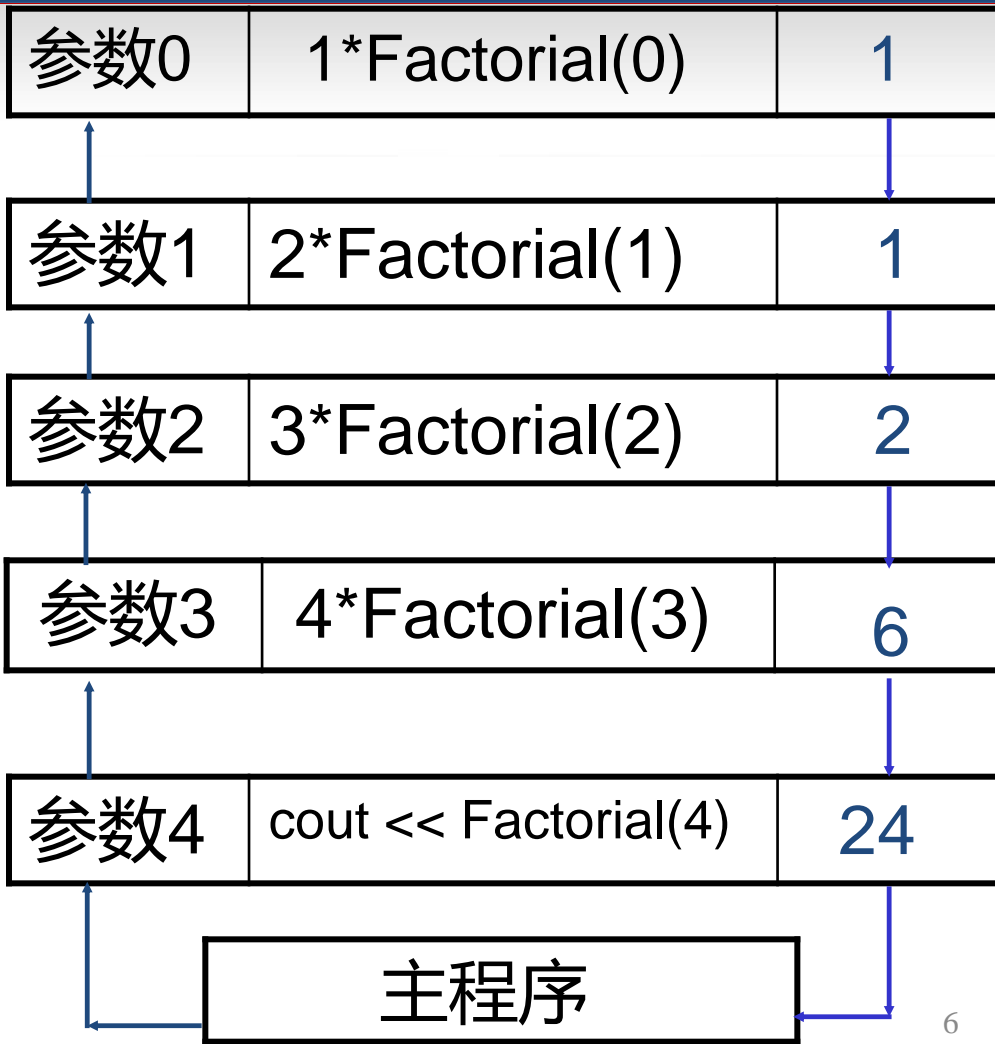
- 一个函数调用其自身，就是递归
- 求 $n!$ 的递归函数

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

```
1. int Factorial(int n)
2. {
3.     if (n == 0)
4.         return 1;
5.     return n * Factorial(n - 1);
6. }
```

F(3)3->F(3)5->F(2)3->F(2)5->F(1)3->F(1)5-> F(0)3->F(0)4:返回1->
F(1)5:返回1*1->F(2)5:返回2*1-> F(3)5:返回3*2-> 函数执行结束

递归和普通函数
调用一样是通过
栈实现的





北京大学
PEKING UNIVERSITY

信息科学技术学院

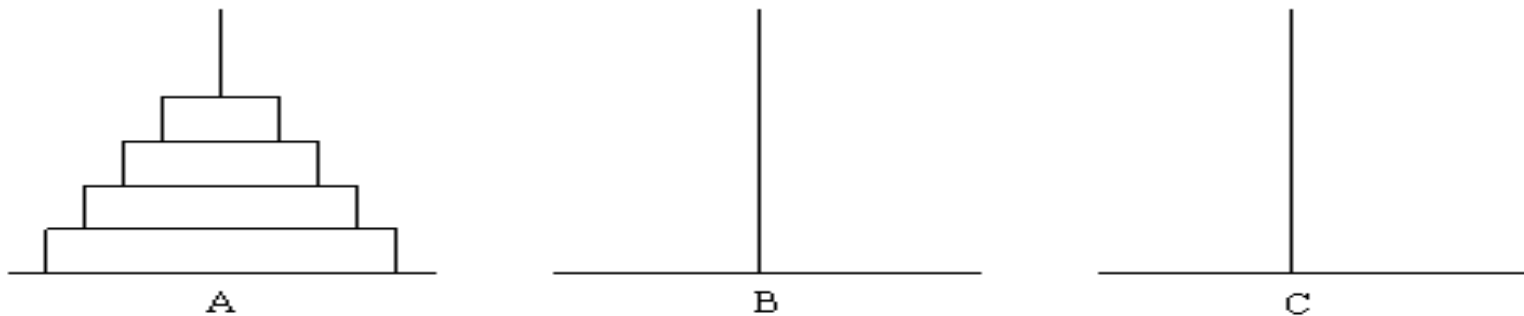
例题: 汉诺塔问题 (Hanoi)



河北草原天路

汉诺塔问题(Hanoi)

古代有一个梵塔，塔内有三个座A、B、C，A座上有64个盘子，盘子大小不等，大的在下，小的在上（如图）。有一个和尚想把这64个盘子从A座移到C座，但每次只能允许移动一个盘子，并且在移动过程中，3个座上的盘子始终保持大盘在下，小盘在上。在移动过程中可以利用B座，要求输出移动的步骤。



简单的版本:

```
#include <iostream>
using namespace std;
void Hanoi(int n, char src, char mid, char dest)
//将src座上的n个盘子, 以mid座为中转, 移动到dest座
{
    if( n == 1) { //只需移动一个盘子
        cout << src << "->" << dest << endl;
        //直接将盘子从src移动到dest即可
        return ; //递归终止
    }
    Hanoi(n-1, src, dest, mid); //先将n-1个盘子从src移动到mid
    cout << src << "->" << dest << endl;
    //再将一个盘子从src移动到dest
    Hanoi(n-1, mid, src, dest); //最后将n-1个盘子从mid移动到dest
    return ;
}
```

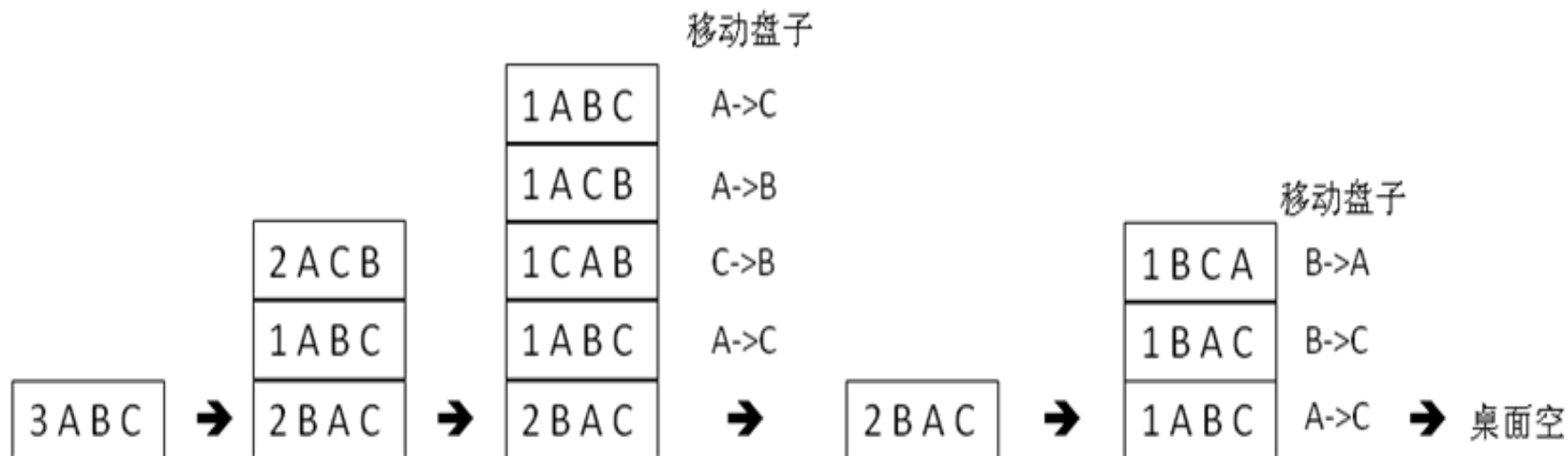
```
int main()
{
    int n;
    cin >> n; //输入盘子数目
    Hanoi(n, 'A', 'B', 'C');
    return 0;
}
```

提交能通过的版本:

```
#include <iostream>
using namespace std;
void Hanoi(int n, char src,char mid,char dest,int src_n)
//将src座上的n个盘子, 以mid座为中转, 移动到dest座
//src座最上方盘子编号是 src_n
{
    if( n == 1) { //只需移动一个盘子
        cout << src_n << ":" << src << "->" << dest << endl;
        //直接将盘子从src移动到dest即可
        return ;
    }
    Hanoi(n-1,src,dest,mid,src_n); //先将n-1个盘子从src移动到mid
    cout << src_n + n - 1 << ":" << src << "->" << dest << endl;
    //再将一个盘子从src移动到dest
    Hanoi(n-1,mid,src,dest,src_n); //最后将n-1个盘子从mid移动到dest
    return ;
}
```

```
int main()
{
    char a ,b,c ;
    int n;
    cin >> n >> a >> b >> c; //输入盘子数目
    Hanoi(n,a,b,c,1);
    return 0;
}
```

汉诺塔问题手工解法(三个盘子)





北京大学
PEKING UNIVERSITY

信息科学技术学院

递归的作用



银川沙湖(航拍)

递归的作用

- 1) 替代多重循环
 - 2) 解决本来就是用递归形式定义的问题
 - 3) 将问题分解为规模更小的子问题进行求解
-



北京大学
PEKING UNIVERSITY

信息科学技术学院

N皇后问题



宁夏中卫沙坡头

用递归替代多重循环

n皇后问题：输入整数n, 要求n个国际象棋的皇后，摆在 $n \times n$ 的棋盘上，互相不能攻击，输出全部方案。

用递归替代多重循环

n皇后问题：输入整数n, 要求n个国际象棋的皇后，摆在 $n \times n$ 的棋盘上，互相不能攻击，输出全部方案。

八皇后问题：八重循环。n皇后，n重循环？

用递归替代多重循环

n皇后问题：输入整数n, 要求n个国际象棋的皇后，摆在 $n \times n$ 的棋盘上，互相不能攻击，输出全部方案。

八皇后问题：八重循环。n皇后，n重循环？

递归解决！

N皇后问题

输入一个正整数N，则程序输出N皇后问题的全部摆法。

输出结果里的每一行都代表一种摆法。行里的第i个数字如果是n，就代表第i行的皇后应该放在第n列。

皇后的行、列编号都是从1开始算。

样例输入：

4

样例输出：

2 4 1 3

3 1 4 2

```
#include <iostream>
#include <cmath>
using namespace std;
int N;
int queenPos[100];
//用来存放算好的皇后位置。最左上角是(0,0)
void NQueen( int k);
int main()
{
    cin >> N;
    NQueen(0); //从第0行开始摆皇后
    return 0;
}
```

```

void NQueen( int k) { //在0~k-1行皇后已经摆好的情况下，摆第k行及其后的皇后
    int i;
    if( k == N ) { // N 个皇后已经摆好
        for( i = 0; i < N;i ++ )
            cout << queenPos[i] + 1 << " ";
        cout << endl;
        return ;
    }
    for( i = 0;i < N;i ++ ) { //逐尝试第k个皇后的位置
        int j;
        for( j = 0; j < k; j ++ ) {
            //和已经摆好的 k 个皇后的位置比较，看是否冲突
            if( queenPos[j] == i ||
                abs(queenPos[j] - i) == abs(k-j)) {
                break; //冲突，则试下一个位置
            }
        }
    }
}

```

```
if( j == k ) { //当前选的位置 i 不冲突
    queenPos[k] = i; //将第k个皇后摆放在位置 i
    NQueen(k+1);
}
} //for( i = 0;i < N;i ++ )
}
```



北京大学
PEKING UNIVERSITY

信息科学技术学院

逆波兰表达式



甘肃张掖七彩丹霞（航拍）

用递归解决递归形式的问题

例题：逆波兰表达式

逆波兰表达式是一种把运算符前置的算术表达式(其实一般教科书上称这种表达式为波兰表达式)，例如普通的表达式 $2 + 3$ 的逆波兰表示法为 $+ 2 3$ 。逆波兰表达式的优点是运算符之间不必有优先级关系，也不必用括号改变运算次序，例如 $(2 + 3) * 4$ 的逆波兰表示法为 $* + 2 3 4$ 。本题求解逆波兰表达式的值，其中运算符包括 $+ - * /$ 四个。

输入

输入为一行，其中运算符和运算数之间都用空格分隔，运算数是浮点数

输出

输出为一行，表达式的值。

用递归解决递归形式的问题

样例输入

* + 11.0 12.0 + 24.0 35.0

样例输出

1357.000000

提示： $(11.0+12.0)*(24.0+35.0)$

用递归解决递归形式的问题

本题中“逆波兰表达式”的定义：

- 1) 一个数是一个逆波兰表达式，值为该数
- 2) "运算符 逆波兰表达式 逆波兰表达式" 是逆波兰表达式，值为两个逆波兰表达式的值运算的结果

一般教科书将本题中的“逆波兰表达式”称为“波兰表达式”，而将运算符后置的表达式成为“逆波兰表达式”

```
#include <iostream>
#include <stdio>
#include <stdlib>
using namespace std;
```

```
double exp() {
```

```
    //读入一个逆波兰表达式，并计算其值
```

```
    char s[20];
```

```
    cin >> s;
```

```
    switch(s[0]) {
```

```
        case '+': return exp()+exp();
```

```
        case '-': return exp()-exp();
```

```
        case '*': return exp()*exp();
```

```
        case '/': return exp()/exp();
```

```
        default: return atof(s);
```

```
        break;
```

```
    }
```

```
}
```

```
int main()
```

```
{
```

```
    printf("%lf",exp());
```

```
    return 0;
```

```
}
```