实验舱蛟龙三班

递归（2）

zlj

2022

程序中解决**重复**的问题：

1、循环语句

2、递归调用
在函数中自己调用自己

1、二分查找
2、快速幂
3、分解质因子
4、汉诺塔

作业：《第K个数字》

```cpp
int int_dig(long long n,int m){
    // int_dig(数,第几位);
    int r=m;
    while (r >= 1){
        m=n%10;
        n/=10;
        r--;
    }
    return m;
}
int main(){
    cin >> n >> k;
    cout << int_dig(n,k);
    return 0;
}
```

```cpp
using namespace std;
int kgs(int n,int k) {
    if(k==1)return n%10;
        //else 可省略
    return kgs(n/10,k-1);
}
int main() {
    int n,k;
    cin>>n>>k;
    cout<<kgs(n,k);
    return 0;
}
```

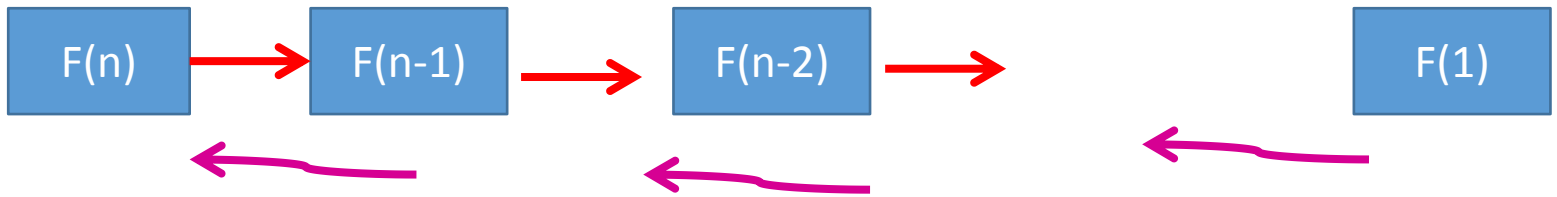
比较递归 与递推 《路径计算》

```cpp
//20 以内可递归
typedef long long ll;
ll sum(int n,int m) {
    if(n==1 || m==1) return 1;
        return sum(n,m-1)+sum(n-1,m);
}
int main() {
    int n,m;
    cin>>n>>m;
    cout<<sum(n,m);
    return 0;
```

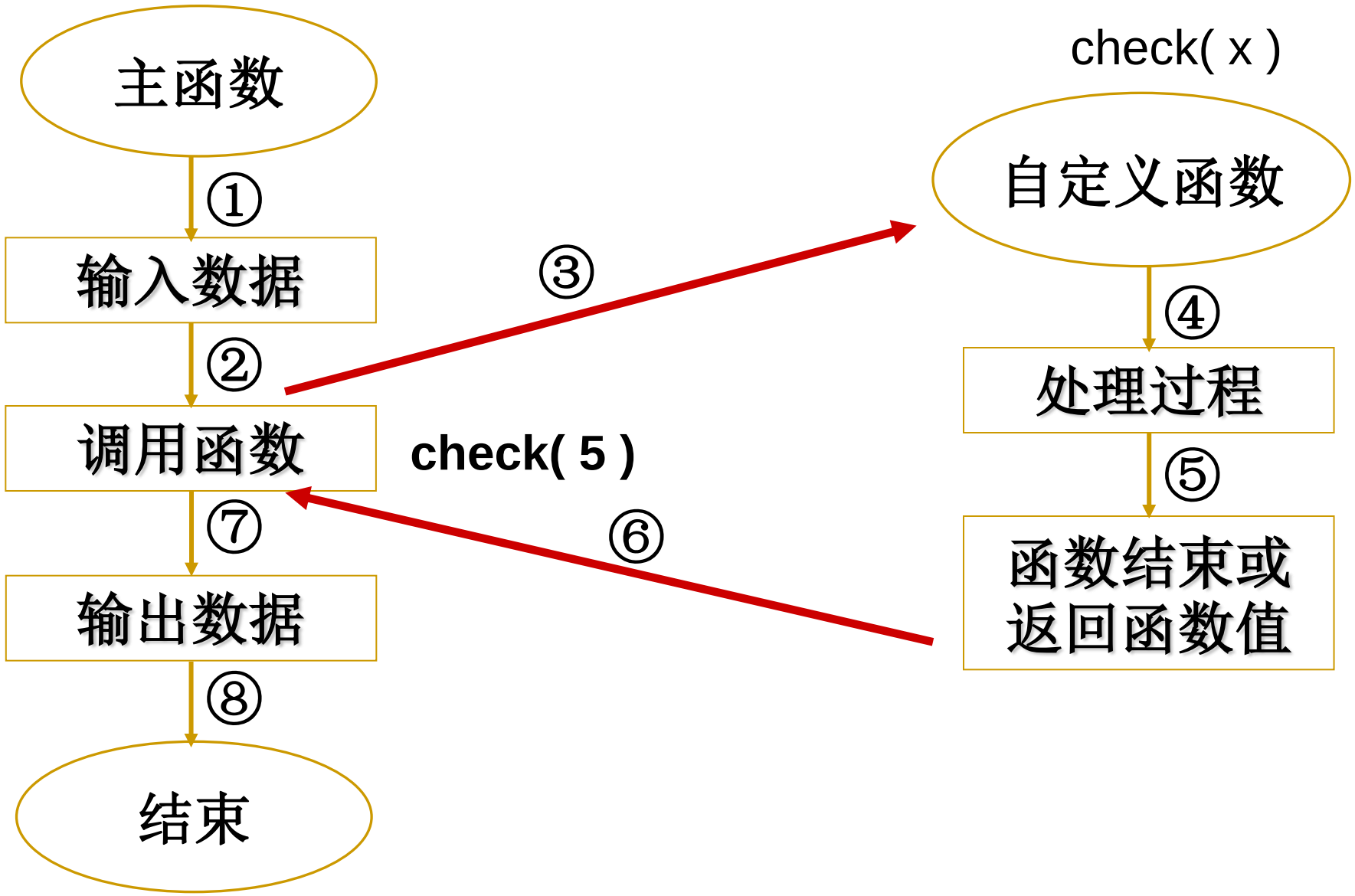
```cpp
long long a[105][105]={0},n,m;
int main(){
    cin>>n>>m;
    for(int i=2;i<=m;i++){
        a[1][i]=1;//第1行第1列都是1条路
    }
    for(int i=2;i<=n;i++){
        a[i][1]=1;
    }
    for(int i=2;i<=n;i++){ //递推每一个位置
        for(int j=2;j<=m;j++){
            a[i][j]=a[i-1][j]+a[i][j-1];
        }
    }
    cout<<a[n][m];
```

复习：递归解决问题

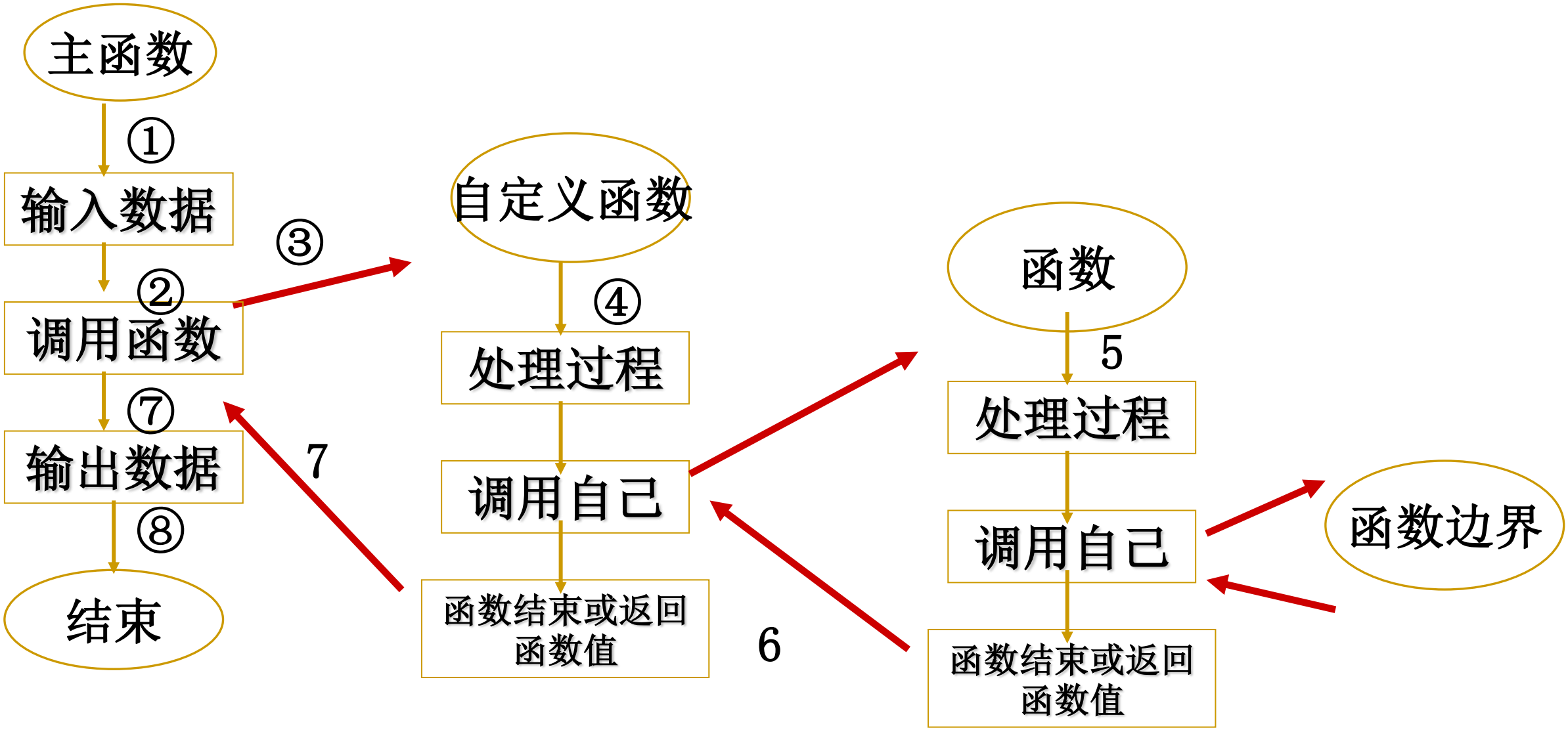
- 递归函数调用
  - 在函数中自己调用自己
- 递归算法（用递归的思想解决问题）
  - 解决用递归形式定义的问题
  - 将问题分解为规模更小的子问题进行求解
  - 替代循环，简化代码



函数调用和返回的过程


主函数
①
输入数据
②
调用函数
⑦
输出数据
⑧
结束
③
check(5)
⑥
check(x)
自定义函数
④
处理过程
⑤
函数结束或
返回函数值

函数递归调用和返回的过程



练习2、写出下列程序的运行结果。

```
20  char a[10];int n;
21  void B(int i){
22      if(i>=n)return;
23      B(i+1);
24      cout<<a[i];
25  }
26  int main(){
27      gets(a);  // 12345
28      n=strlen(a);
29      B(0);
30      return 0;
31  }
```

阅读程序2 看递归执行过程

```
4  void A(int deep) {
5      if(deep>n) return;
6      A(deep+1);
7      for(int i=1; i<=deep; i++) {
8          cout << i;
9      }
10     cout<<endl;
11 }
12 int main() {
13     cin>>n; //5
14     A(1);
15     return 0;
16 }
```

用递归打印如下图形

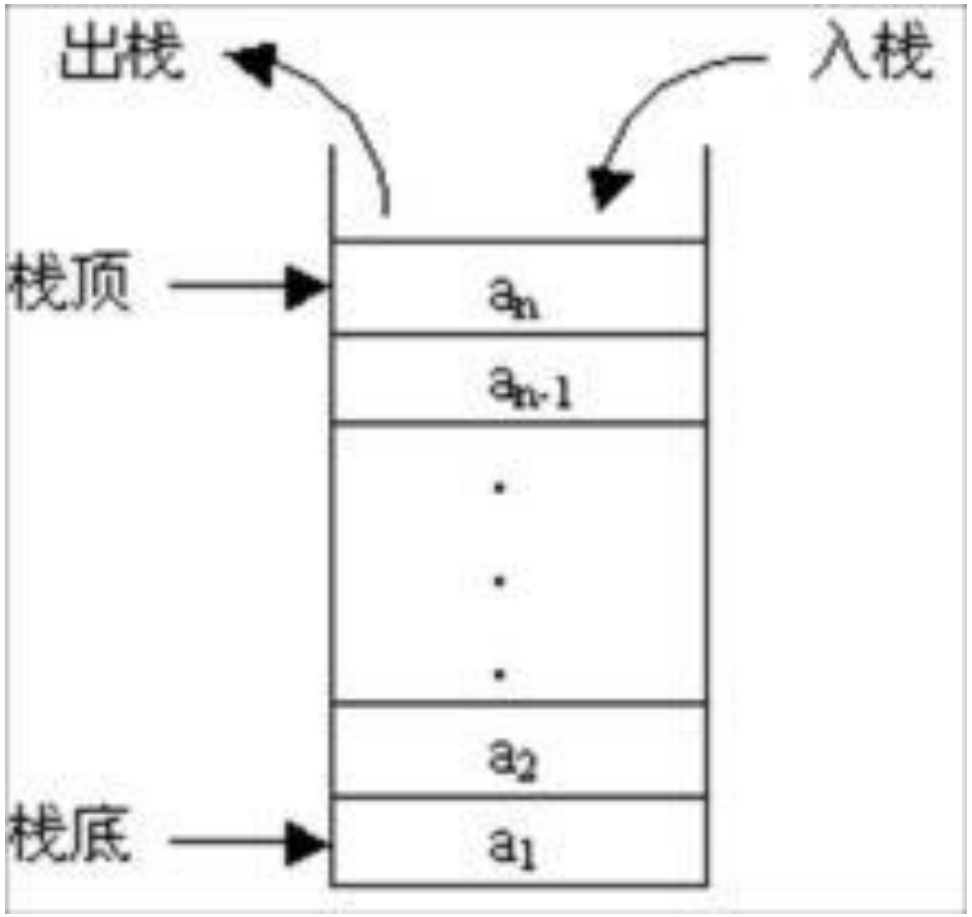
输入N，打印N行如下图形：
3

1
12
123

```
2  using namespace std;//打图形
3  int x;
4  void p(int n){
5      if(n<=x){
6          for(int i=1;i<=n;i++)cout<<i;
7          cout<<endl;
8          p(n+1);
9      }
10 }
11 int main(){
12 cin>>x;
13 p(1);//调用P()
14 return 0;
15 }
```

递归如何做到有序地回归，正确传递回答案呢？

系统栈
后进先出



递归解题：由 一路 到 多路（简单到复杂）：

简单的： s(n)=s(n-1)+n （求和）

s(n)=s(n-1)*n （阶乘）

gcd(x,y)=gcd(y,x%y)

2路的：

s(n)=s(n-1)+s(n-2) （上台阶，非波数列）

阿克曼函数ACK（） 稍复杂的

在数学上有一个著名的“阿克曼函数”，它是二元函数，其定义式为：

（1）$ACK(0,n)=1+n$

（2）$ACK(m,0)=ACK(m-1,1)\ (n>0)$

（3）$ACK(m,n)=ACK(m-1,\ ACK(m,n-1))\ (m>0,n>0)$

```
 2  using namespace std;
 3  int ack(int m,int n) {
 4      if(m==0&&n>0) return n+1;
 5      if(n==0&&m>0) return ack(m-1,1);
 6      if(m>0&&n>0) return ack(m-1,ack(m,n-1));
 7  }
 8  int main() {
 9      int x,y;
10      cin>>x>>y;
11      cout<<ack(x,y)<<endl;
12      return 0;
13  }
```

探索递归函数解决问题的规律

- 1、简单递归函数
  - 在函数中自己调用自己 （存在明显先后处理顺序及相同子问题）
- 2、存在递归模型

  满足递归的概念的定义，存在多个子问题及多个边界，可以分开解决再合并（有点分治的意思）

例2 ：放苹果

把M个同样的苹果放在N个同样的盘子里，允许有盘子空着不放，问共有多少种不同的分法？（用K表示）5，1，1和1，5，1 是同一种分法。
输入:第一行是测试数据的数目t（0 <= t <= 20）。以下每行均包含二个整数M和N，以空格分开。1<=M，N<=10。
输出:对输入的每组数据M和N，用一行输出相应的K。
样例输入
1
7 3
样例输出
8

分析：

f(m,n)表示m个苹果放到n个盘子中的方法数 。

1、当n>m时，必然有盘子是空的，因为最多也就用到m个盘子，因此f(m,n)=f(m,m);

2、当n<=m时，就是有空和没空两种情况 的和,

当盘子不空的时候：全部n个盘子都有装苹果，那所有的都可以拿掉一个苹果，也就是 f(m,n)=f(m-n,n) 方法数是一样的，只不过所有盘子都用上的时候每个盘子装的数量可能不一样 ，

有盘子为空时： f(m,n)=f(m,n-1)因为至少有一个为空，那去掉一个完全不影响已有的方法数（反正这个空盘子不会放苹果）

f(n,m)=f(m,n-1)+f(m-n,n) 两者递归时，n和m都会逐渐减小 ，出口为n==1||m==0的时候，都只有一种方法

```cpp
int f(int m,int n)
{
    if( n > m ) return f(m,m);
    if( m == 0) return 1;
    if( n <= 0 ) return 0;
    return f(m,n-1) + f(m-n,n);
}
int main()
{
    int t,m,n;
    cin >> t;
    while( t--)
    {
        cin >> m >> n;
        cout << f(m,n) << endl;
    }
    return 0;
}
```

例5 ：又是杨辉三角

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
以上图形为杨辉三角，请输入两个数n和m，输出杨辉三角的第n行的第m个数,
(1<n,m<100000)

输入

两个整数n和m

输出

杨辉三角的第n行的第m个数

输入样例

3 2

样例输出

2

分析：

本题的递归关系明显：

递归表达式为： f(n,m)=

递归的出口是：

参考代码，　理解后可再做〈路径计算〉

```
using namespace std;
long long a[10001][10001],n,m;
long long f(int x,int y) {
    if(x==0||y==0)return 0;  //防止越界
    if((x==1&&y==1)||(y==1)||(y==x))return 1;
    if(a[x][y]!=0) return a[x][y];
    a[x][y]=f(x-1,y-1)+f(x-1,y);
    return a[x][y];
}

int main() {
    cin>>n>>m;
    cout<<f(n,m)<<endl;
    return 0;
}
```

例6 ：判断元素是否存在

有一个集合M是这样生成的： (1) 已知 k 是集合 M 的元素； (2) 如果 k 是 M 的元素，那么， 2k+1 和 3k+1 都是 M 的元素； (3) 除了上述二种情况外，没有别的数能够成为 M 的一个元素。
问题：任意给定 k 和 x，请判断 x 是否是 M 的元素。这里的 k是无符号整数，x 不大于 100000， 如果是，则输出YES，否则，输出 NO

输入

输入整数 k 和 x, 逗号间隔。

输出

如果是，则输出 YES，否则，输出NO

样例输入

0,22

样例输出

YES

分析：递归关系式不明显

递归关系式为：

递归的出口是：

2、注意输入的格式；

0，22　中间有个逗号，　cin.get(); 的用法。

Scanf(“%d,%d,&n,&m)

填空：

```
2  using namespace std;//判断是否存在
3  int k,x;
4  bool check(int y){
5      if(y>x)return false;
6      if( (1) )return true;
7      if(check(2*y+1)|| (2) )return true;
8      return  (3) ;
9  }
10 int main(){
11     cin>>k;
12     cin.get();
13     cin>>x;
14     int a=k;
15     bool b=check(a);
16     if(b)cout<<"YES"<<endl;
17     else cout<<"NO"<<endl;
18 }
```

例：角谷猜想

给定一个整数N，若n为1，结束，若n为偶数，n=n/2,若n为奇数，n=n*3+1,如此循环，最终能得到1，用递归描述此过程。

填空：

```
2  using namespace std;//角谷猜想
3  void gu(int n) {
4      if( (1) )return;
5      else if(n%2==0) {
6          cout<<n<<'/'<<2<<"="<<n/2<<endl;
7          gu( (2) );
8      } else {
9          cout<<n<<'*'<<3<<"+"<<1<<"="<<n*3+1<<endl;
10         gu( (3) );
11     }
12 }
13 int main() {
14     int x;
15     cin>>x;
16     gu(x);
17     return 0;
18 }
```

五笔字型

递归应用：

进制转换、二分查找、快速幂 。。。

1、递归转进制

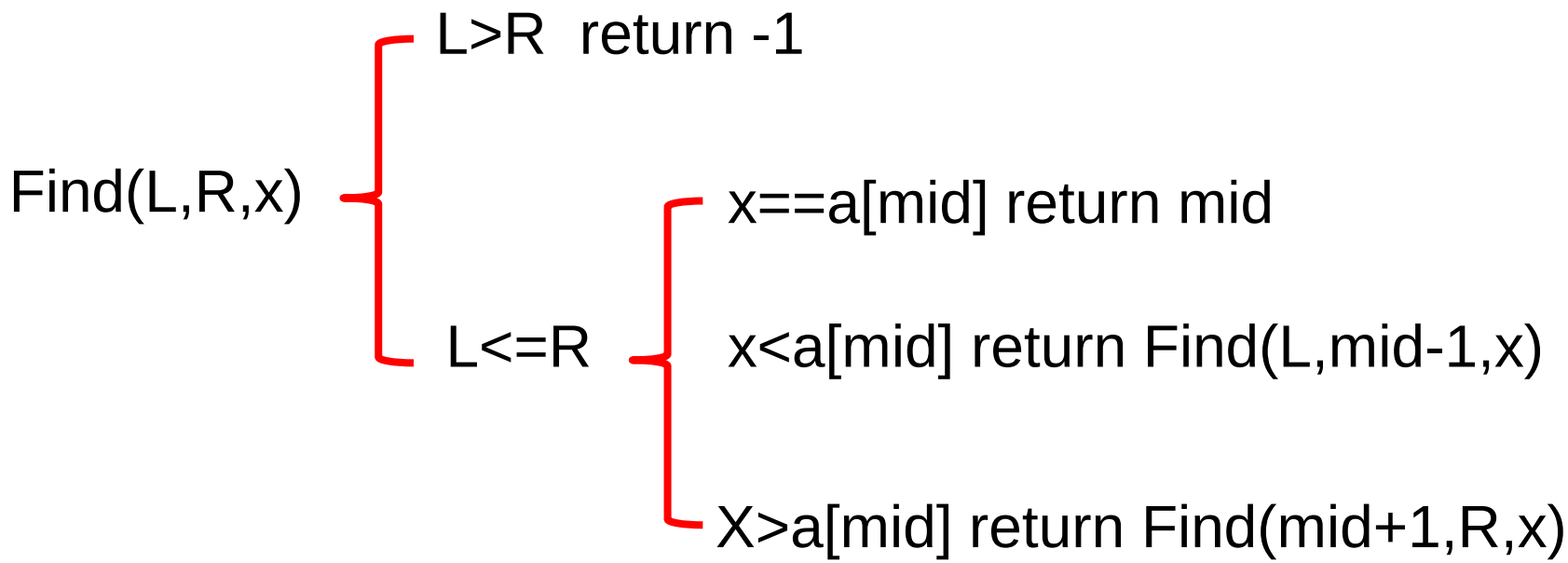
- 将123转换成等值的二进制数：

除以2的商（取整）	余数
123/2 = 61	1
61/2 = 30	1
30/2 = 15	0
15/2 = 7	1
7/2 = 3	1
3/2 = 1	1
1/2 = 0	1

- 自下而上收集余数：1111011

```
void o2(int x) {
        if(x==0){return;}
        To2(_____);
        cout<<_____;
        return;
}
int main() {
        int n;cin>>n;
        To2(n);
        return 0;
}
```

2、二分查找


Find(L,R,x)
L>R return -1
L<=R
x==a[mid] return mid
x<a[mid] return Find(L,mid-1,x)
X>a[mid] return Find(mid+1,R,x)

```
4  int bserch(int L,int R,int x) {
5      while(L<=R) {
6          int m=(L+R)/2;
7          if(a[m]==x)return m;
8          else if(a[m]>x)R=m-1;
9          else L=m+1;
10     }
11     return -1;
12 }
```

填空：

```
2  using namespace std;//二分查找（递规）
3  int a[1001],n,k;
4  int bfind(int L,int R,int x) {
5      int mid=(L+R)/2;
6      if( (1) )return -1;
7      if(a[mid]==x)return mid;
8      if(a[mid]>x)return bfind( (2) );
9      else return bfind( (3) );
10 }
11 int main() {
12     cin>>n>>k;
13     for(int i=1; i<=n; i++)cin>>a[i];
14     sort(a+1,a+n+1);
15     if(bfind(1,n,k)== (4) )cout<<"No"<<endl;
16     else cout<<"yes"<<endl;
17     return 0;
```

3、质因子分解

输入一个正整数N，用递归的方法从小到到输出它的所有质因子（因子是质数）2<=N<=10000

输入样例
18
输出样例
2 3 3

分解质因子

分析：如果是1，就不能分解了，否则，从2开始试除，if(x%2==0),得到一个因子，问题转化成 x/2

比较递归和 非递归写法

```
using namespace std;
int main() {
    int n;cin>>n;
    for(int i=2;i*i<=n;i++){
        while(n%i==0){
            cout<<i<<" ";
            n/=i;
        }
    }
    if(n>1)cout<<n<<endl;
    return 0;
```

```
void prime(int n,int i){
    if(i>n)return;
    if(n%i==0){
        cout<<i<<" ";
        prime(n/i,i);
    }else{
        prime(n,i+1);
    }
}
int main() {
    int n;
    cin>>n;
    prime(n,2);
    return 0;
}
```

4、快速幂

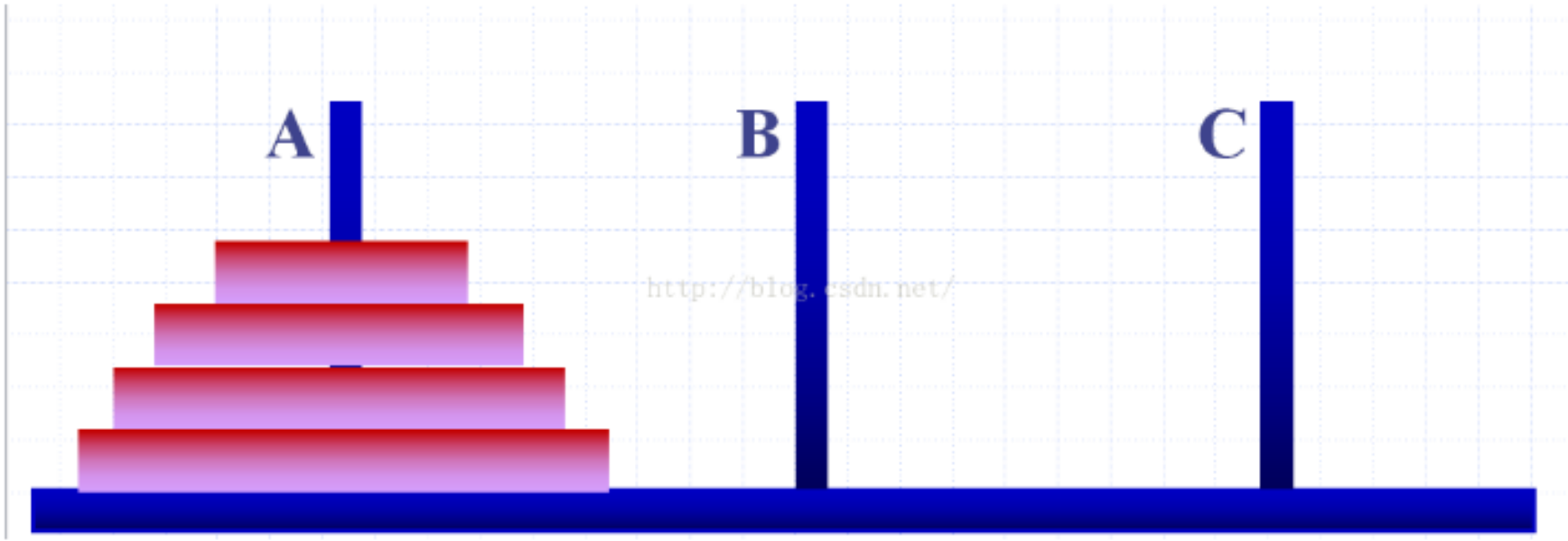
A 的 B 次方（a^b）%k　　k=1007

Fast(A,B)

- B==0 return 1
- B>0
  - B%2==1:
    A*Fast(A,B-1)%k== A*Fast(A,B/2)%k *Fast(A,B/2)%k
  - B%2==0:
    Fast(A,B/2)%k *Fast(A,B/2)%k

参考代码：

5、《汉诺塔》



分析：

1、什么情况下结束？

2、如何进行第一步？将n-1个盘子看做一个整体。

显然是先将n -1 个在a柱子上的盘子通过b柱移动到c柱上，

再将在a柱子上的编号为n的盘子移动到b柱上，

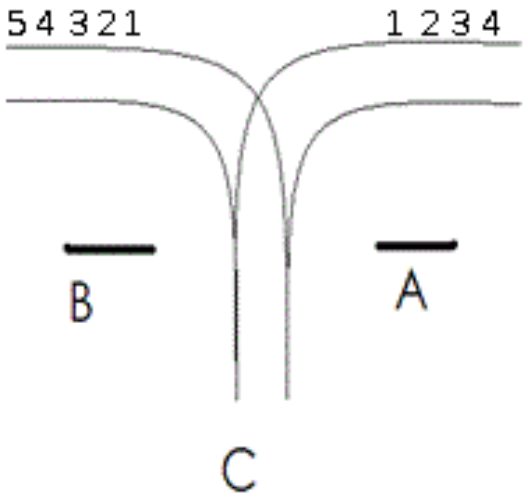
再将c柱子上的n-1个盘子通过a柱移动到b柱上，over

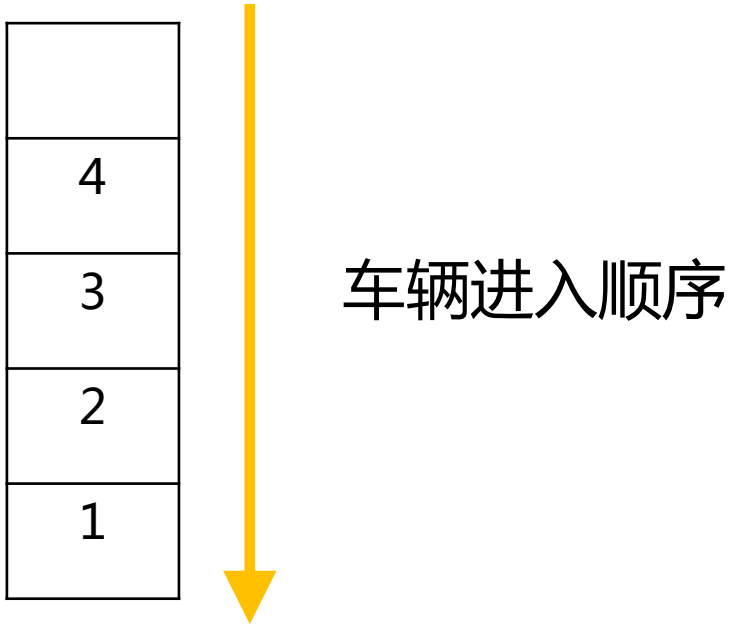
```
2  using namespace std;
3  int cnt=1;
4  void out(int k,int n,char a,char b) {
5      printf("第%d步，将%d从%c杆移动到%c杆",k,n,a,b);
6      cout<<endl;
7  }
8  void hnt(int n,char a,char b,char c) {
9      if(n==1) out(cnt++,n,a,c);
10     else {
11         hnt(n-1,a,c,b); //分为3步，2个子问题
12         out(cnt++,n,a,c);
13         hnt(n-1,b,a,c);
14     }
15 }
16 int main() {
17     int n;
18     cin>>n;
19     hnt(n,'X','Z','Y') ;
20     return 0;
```

- 栈
  - 计算机中栈的概念有两种：
    - 数据结构：后进先出
    - 程序运行时的一种内存：
  - 内存中的栈（一种不严谨的理解）：
    - 函数调用占用的空间

- 栈

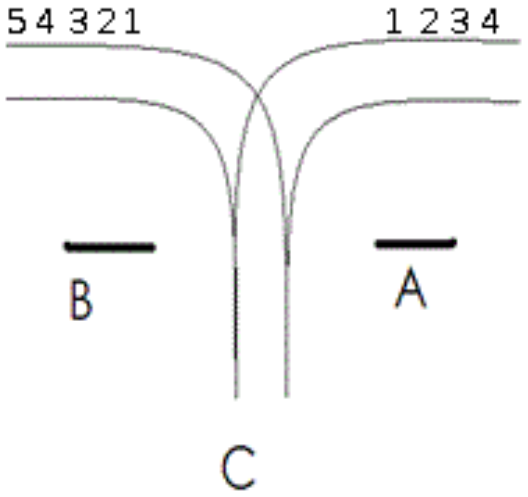
- C是一个停车通道，车辆从A通道进入C，从B通道驶出C，C通道非常狭窄，宽度只够停一辆车，这就导致一辆车要出去，它后面进入C的车辆必须先出去





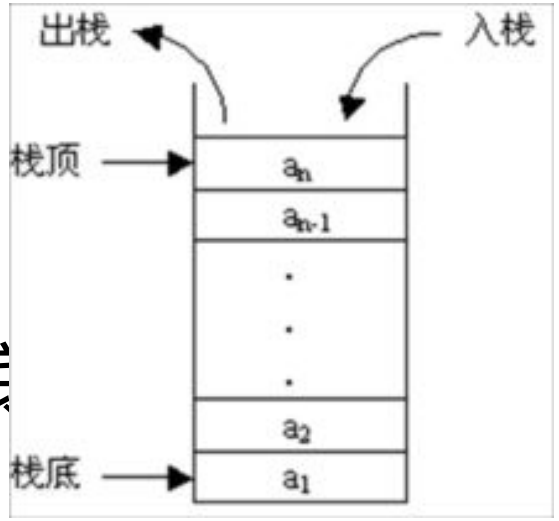
- 栈

- 现在有人在入口观察到车辆进入通道的顺序是1,2,3,4，离开的顺序的4,2,3,1这可能吗？如果是2,1,4,3，可能吗？



- “后
后进

戋底和**栈顶**，进入栈的元素从底
出栈



- stl中的栈

- stack<int> st; 声明栈
- st.empty() 判断栈是否为空
- st.push(x); 栈中加入一个元素x
- st.pop(); 栈顶元素出栈
- x = st.top(); 取出栈顶元素（栈顶元素不出栈）

- Stack 的应用　《括号匹配 基础》

- 从栈的角度思考括号匹配，左括号相当于一个进栈操作，右括号相当于一个出栈操作

- 括号不匹配的情况：
  - 1. 一个右括号找不到对应的左括号：
  - 2. 一个左括号找不到对应的右括号：

- 例1 表达式括号匹配

- 从栈的角度思考括号匹配，左括号相当于一个进栈操作，右括号相当于一个出栈操作

- 括号不匹配的情况：
  - 1. 一个右括号找不到对应的左括号：
  - 2. 一个左括号找不到对应的右括号：

```cpp
using namespace std;
string s;
stack<int>a;
int main() {
    cin>>s;
    int len=s.size();
    for(int i=0; i<len; i++) {
        if( s[i]=='(' ) a.push(1);
        if( s[i]==')' ) {
            if(!a.empty()) a.pop();
            else {
                cout<<"NO"<<endl;
                return 0;
            }
        }
    }
    if(a.empty())
        cout<<"YES"<<endl;
    else cout<<"NO"<<endl;
    return 0;
}
```