

程序设计方法

——以循环结构为例

Wang Houfeng
EECS, PKU

内容

➤ **程序设计方法**

§ 循环程序设计例子

程序设计的基本要求

§ 数据加工，得到期望的结果，要求：

- ★ 结果正确（**作文扣题：最基本的要求**）
- ★ 结构清晰（**作文书写与版面的美观**）
- ★ 速度快+占用存储空间少（**作文的深度：技能**）

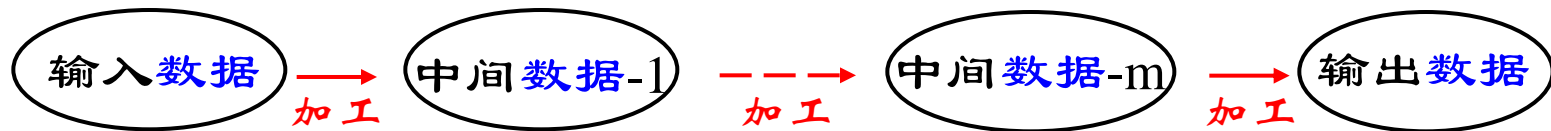
提升程序设计技能的两大要点

§ 数据的表示 — 数据结构

★ 输入/输出/中间数据的表示

§ 数据的加工方法 — 算法

★ 如何将输入数据有效变换成输出数据



N.Wirth 的观点：程序= 数据结构+算法



结构化程序设计语言 —Pascal 语言提出者
图灵奖获得者

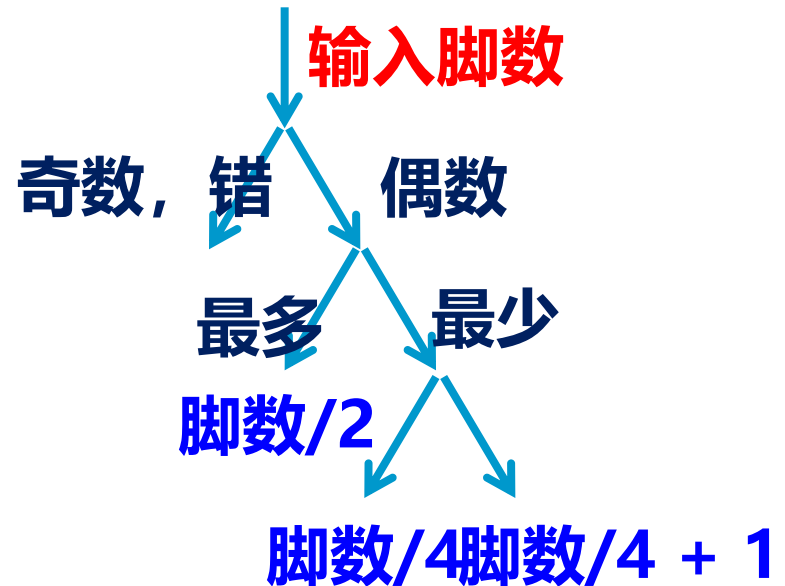
数据结构：关于数据的表示，数据元素间的关系

算法：数据加工的基本步骤（用五大特征描述）：

- ★ 有穷性：描述的操作可以在有穷的时间内完成（****）
- ★ 确定性：描述的每一种操作都是确定的，没有任何歧义
- ★ 有0个或多个输入
- ★ 至少有一个输出
- ★ 有效性：操作的每一步都是可行的，例如，不能有 $n/0$ 的问题。

举例-3：鸡兔同笼

- ❖ 一个笼子里面关了鸡和兔子（鸡2只脚，兔4只脚，没有例外）。已经知道了笼子里面脚的总数 a ，问笼子里面至少有多少只动物，至多有多少只动物？
- ❖ 分析（算法）：



鸡兔同笼

```
#include <iostream>
using namespace std;
int main()
{
    int i, nFeet; //nFeet 表示输入的脚步数。
    cin >> nFeet;
    if (nFeet % 2 != 0) // 如果有奇数只脚，则输入不正确，
                        // 因为不论2只还是4只，都是偶数
        cout << "0 0" << endl;
    else if (nFeet % 4 != 0)
        //若要动物数目最少，使动物尽量有4只脚
        //若要动物数目最多，使动物尽量有2只脚
        cout << nFeet / 4 + 1 << " " << nFeet / 2 << endl;
    else
        cout << nFeet / 4 << " " << nFeet / 2 << endl;
    return 0;
}
```

最少动物数 **最多动物数**

算法的表示

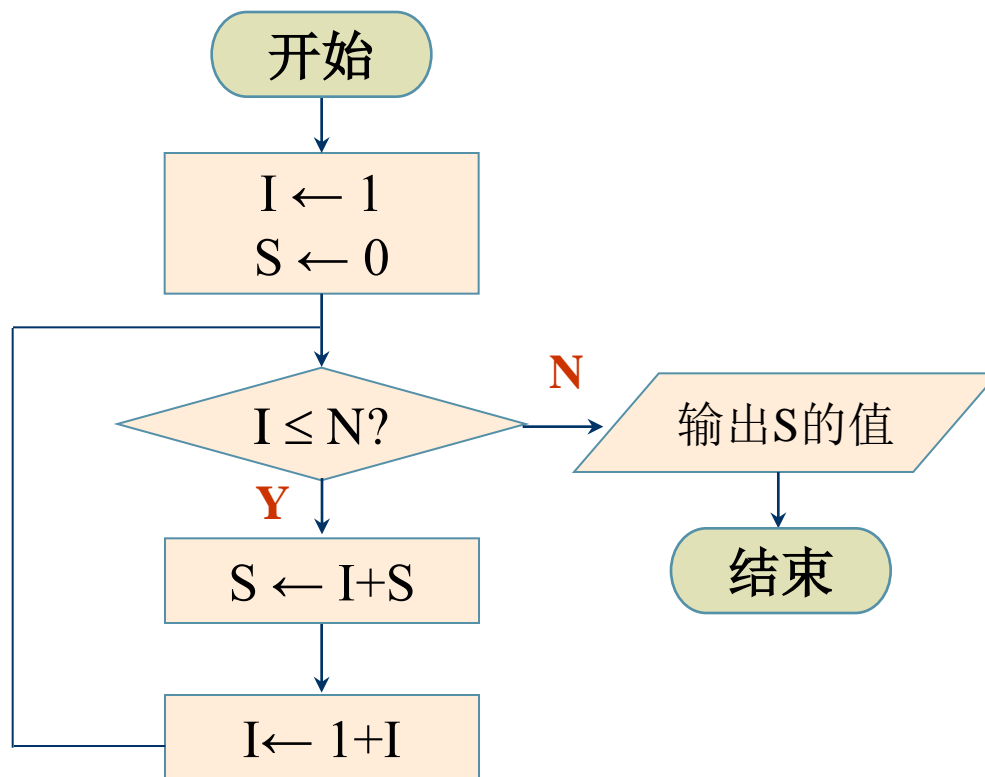
§ 图表示：流程图 vs. N-S 图表示

§ 伪码表示

§ 程序设计语言（如 C /C++）表示

§ ...

流程图举例：计算 $S=1+2+\dots+N$



素数判别

给定一个正整数，判定其是否为素数

- 如何判定给定正整数 n 是否为素数呢？
- 基本思想：从2开始，找 n 的因子，即，依次检验 2, 3, ..., $n-1$ 各个整数中，是否有某个数是 n 的因子，如果有，说明 n 不是素数；否则，说明 n 是素数。

素数判别的算法

§ 下面的算法中 n 表示待判断的正整数， k 用于检验是否 n 的因素为 k

否定与逐一试探肯定

★ 步骤1: $k \leftarrow 2$

★ 步骤2: 若 n 能被 k 整除，则转向步骤5；否则，做步骤3；

★ 步骤3: 准备检验下一个值: $k \leftarrow k + 1$;

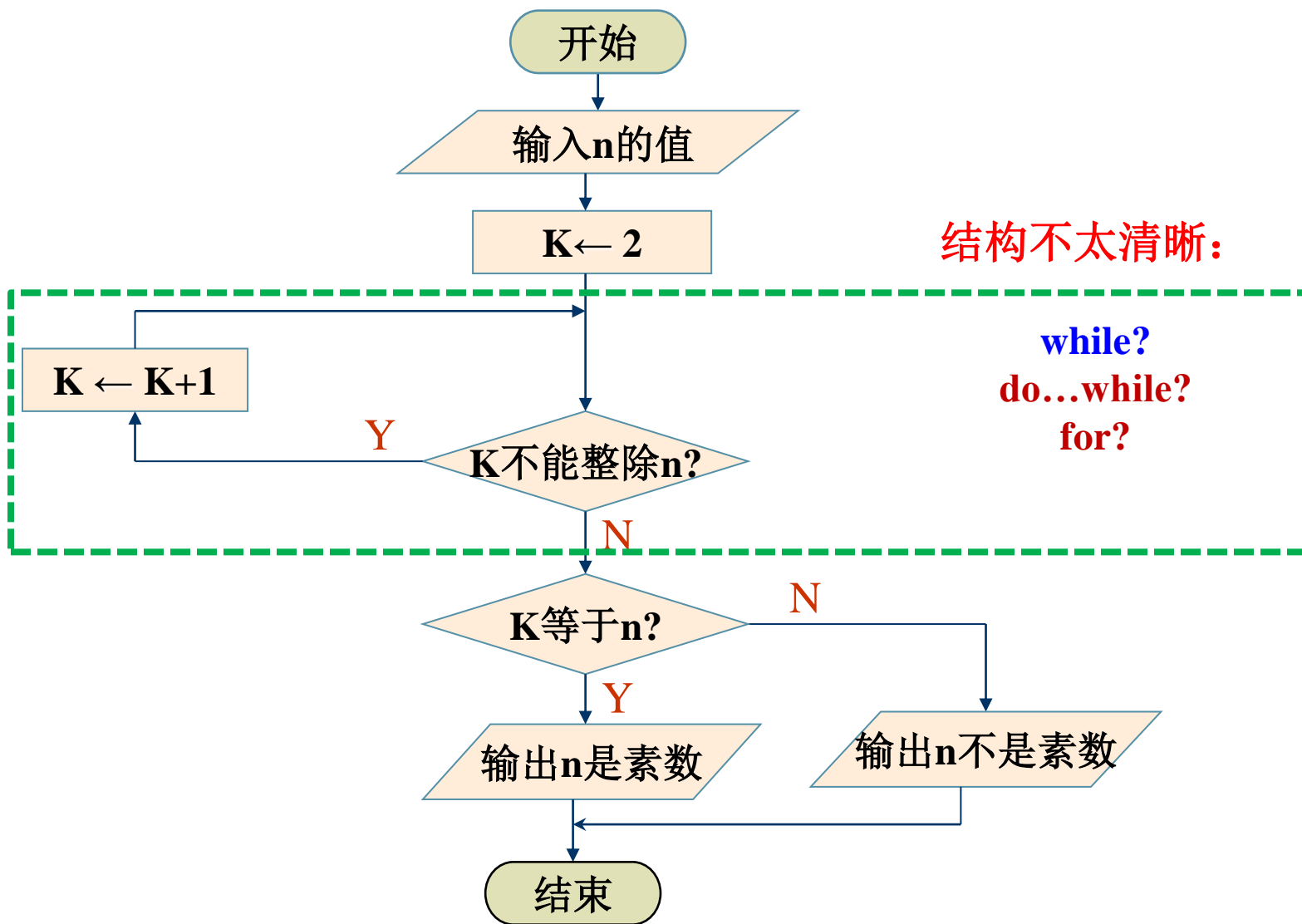
★ 步骤4: 转向步骤2; (进入下一轮检测)

★ 步骤5: 若 n 等于 k ; 则转向步骤6; 否则，转向步骤7;

★ 步骤6: 输出 n 是素数，算法结束。

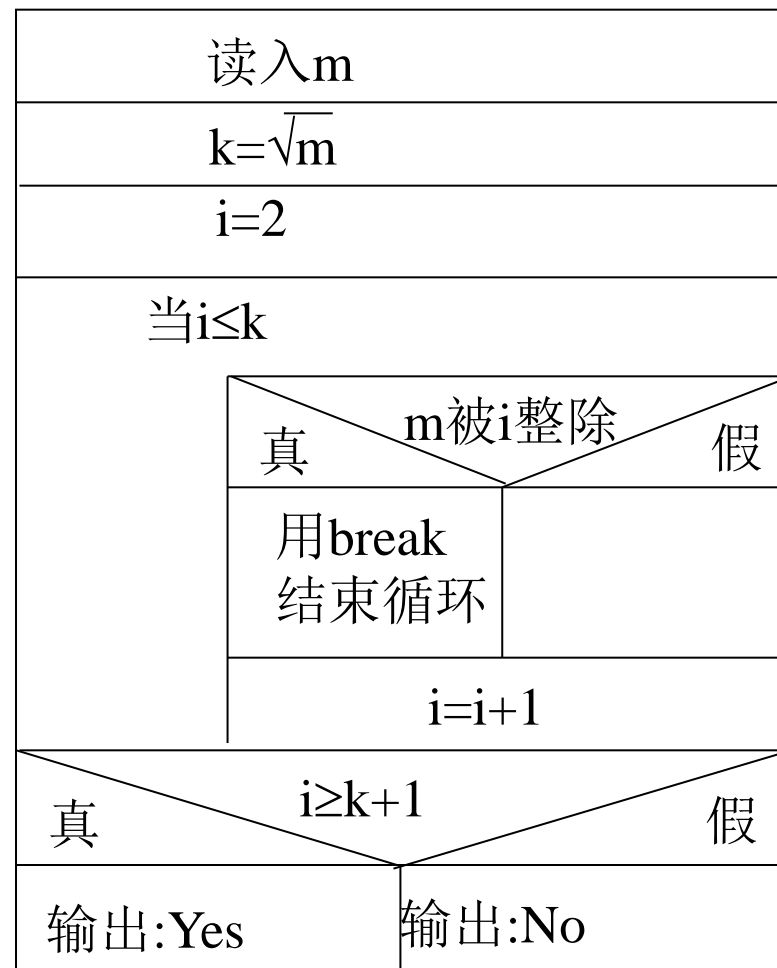
★ 步骤7: 输出 n 不是素数，算法结束

素数判别流程图



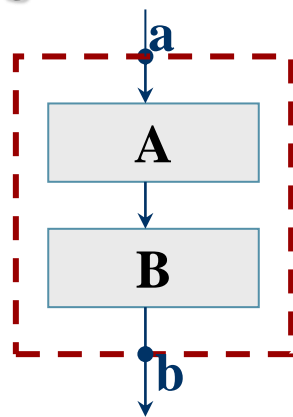
优化

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{  int m,i,k;
    cin>>m;
    k=sqrt(m);
    for(i=2;i<=k;i++)
        if(m%i==0) break;
    if(i>=k+1)
        cout<<c<< "Yes" <<endl;
    else
        cout<<m<< "No"<<endl;
    return 0;
}
```

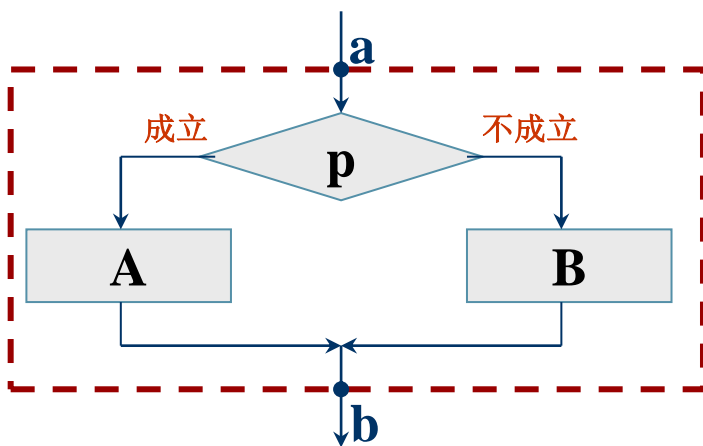


用N-S图描述算法

❖ 流程图符号

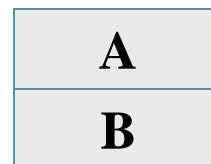


顺序结构

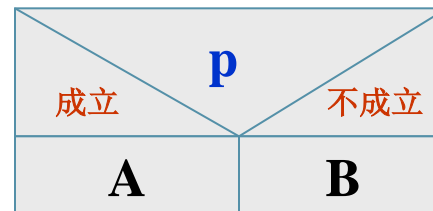


选择结构

❖ N-S盒图的基本符号



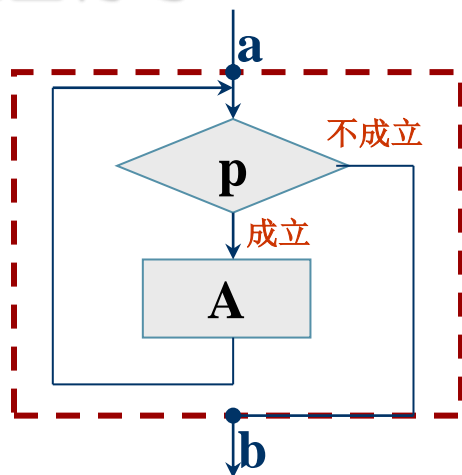
顺序结构



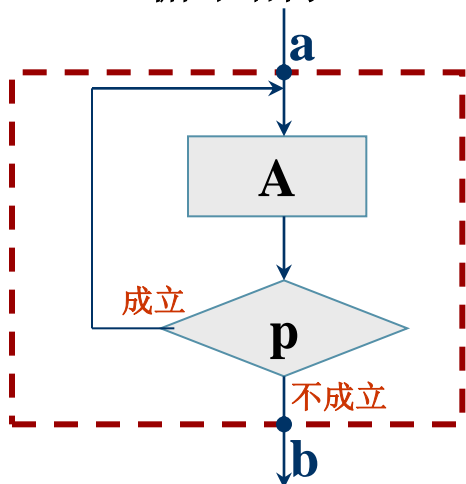
选择结构

用N-S盒图描述算法

❖ 流程图符号

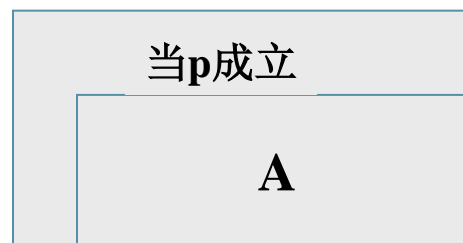


循环结构1

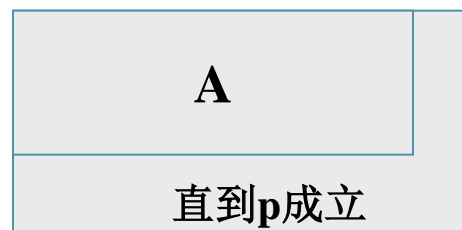


循环结构2

❖ N-S盒图的基本符号



while



do-while

最大公约数问题

设有两个正整数 m 和 n ，如何求其最大公约数？

- 有多种方法
- 一种经典方法：辗转相除（除余法——反复除余）

辗转相除法（欧几里得算法）：

给定两个正整数 m 和 n ($m \geq n$)，求它们的最大公约数(公因子)

步骤1: 【求余数】以 n 除 m 并令 r 为所得余数 ($0 \leq r < n$)

步骤2: 【余数为0?】若 $r=0$ ，算法结束； n 即为答案

步骤3: 【互换】置 $m \leftarrow n$ ， $n \leftarrow r$ ，转向步骤1。

基于上述方法，请以 15 和 9 为例，写一写运算步骤

§ 定理：设有不全为0的正整数 m 、 n 和 r ，且

$$m = n \times t + r \quad (0 \leq r < n, \quad t \text{是整数})$$

§ 那么， m 与 n 的最大公因子等于 n 和 r 的最大公因子

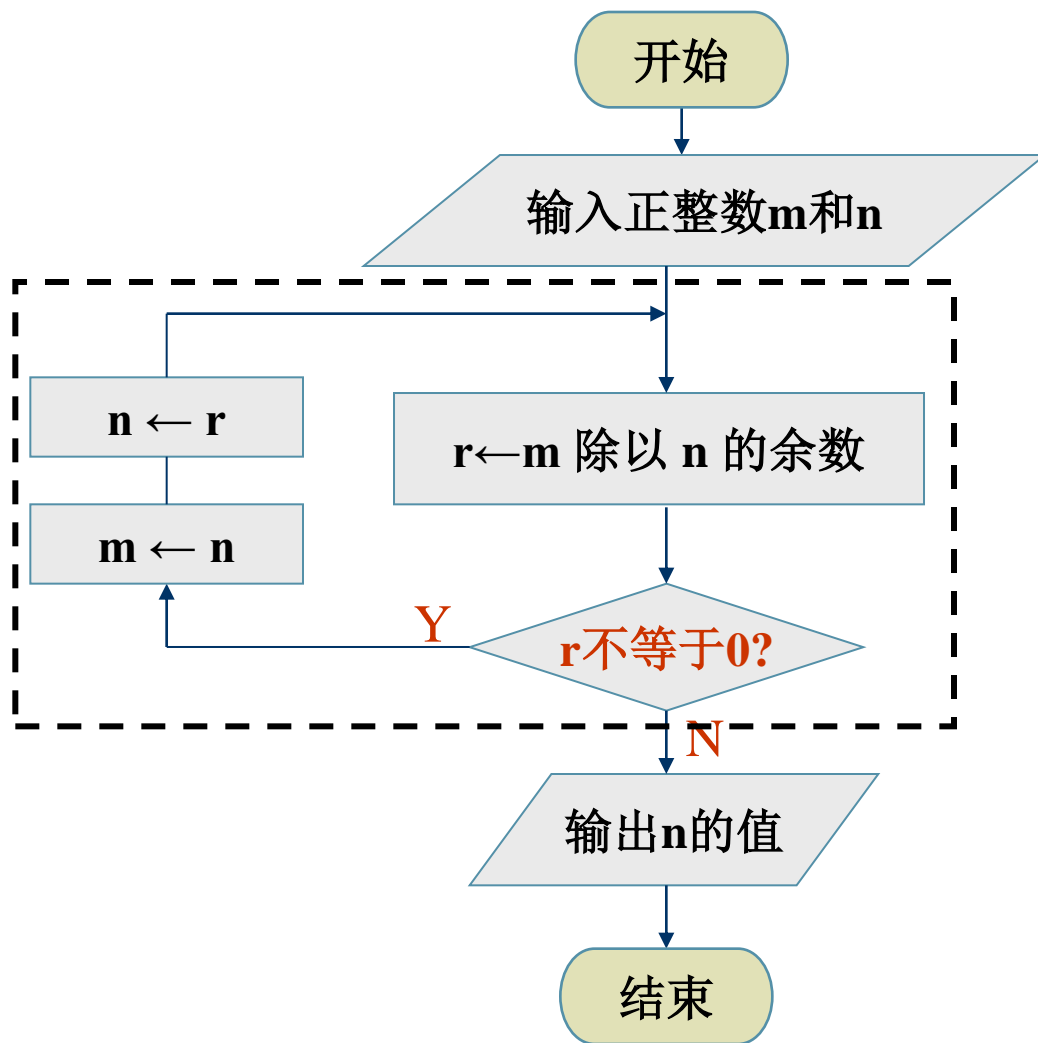
证明：设 $x = m$ 与 n 的最大公因子， $y = n$ 与 r 的最大公因子

1. x 是 m 和 n 的因子，因此 x 能整除 $n \times t + r$ ，即， x 也是 r 的因子，而 y 是 n 和 r 的最大公因子，因此 $x \leq y$

2. 同理， y 是 r 的因子，同时， y 也是 n 的因子，因此， y 一定是 m 的因子，即， y 是 m 和 n 的因子，于是 $y \leq x$

综合1、2， $x=y$

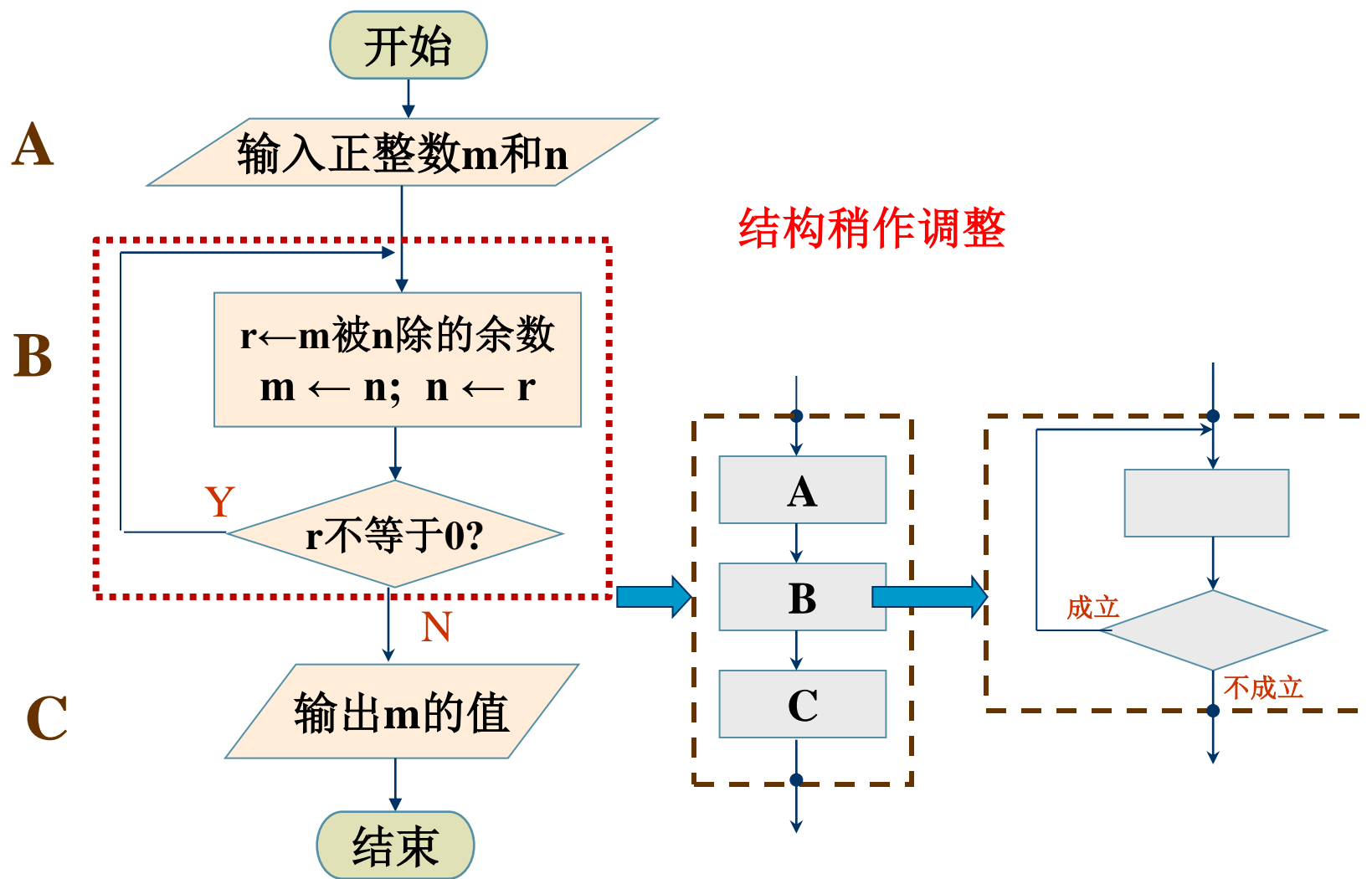
求最大公约数流程图



结构很不清晰
(三种循环似乎都不适合):

while?
do...while?
for?

求最大公约数流程图



流程图的优缺点

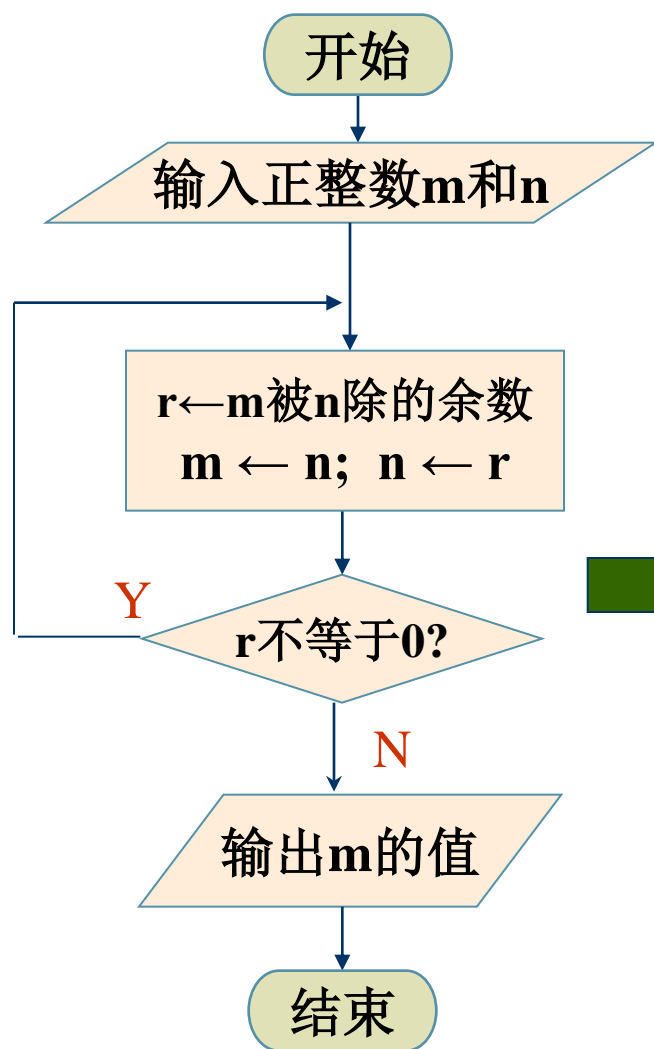
§ 优点

- ★直观形象，各个框图的逻辑关系比较清晰

§ 缺点

- ★占用篇幅较多
- ★作图的随意性较强，结构不明显

伪代码算法：求最大公约数



算法：辗转相除法求最大公约数

BEGIN

input m,n; /*输入正整数m和n*/

do

{

r ← m mod n;

m ← n; n ← r;

} while r ≠ 0;

print m; /*输出最大公约数*/

END

内容

➤ 程序设计方法

➤ 循环程序设计例子

判断回文数

§ 什么是回文数：正向看和反向看相等的整数

§ 特点：正序=逆序

§ 例子：

★ 1, 22, 212, 4444, 12321

§ 一种直观的方法：

★ 计算正整数 m 的逆序表示 n

★ 判断是否 $(m = n)$

§ 如何计算？举例说明（设 $m=12345$ ）


§ 令逆序 n 的初始值 $n=0$;

- {
 - $k = m \% 10$, 得 $k=5$; $n = n * 10 + k = 0 * 10 + 5 = 5$
 - $m = m / 10$, 得 $m=1234$
- {
 - $k = m \% 10$, 得 $k=4$; $n = n * 10 + k = 5 * 10 + 4 = 54$
 - $m = m / 10$, 得 $m=123$
- {
 - $k = m \% 10$, 得 $k=3$; $n = n * 10 + k = 54 * 10 + 3 = 543$
 - $m = m / 10$, 得 $m=12$
- {
 - $k = m \% 10$, 得 $k=2$; $n = n * 10 + k = 543 * 10 + 2 = 5432$
 - $m = m / 10$, 得 $m=1$
- {
 - $k = m \% 10$, 得 $k=1$; $n = n * 10 + k = 5432 * 10 + 1 = 54321$
 - $m = m / 10$, 得 $m=0$
- $m=0$ (结束条件)

你能找出公共操作部分吗？

程序

```
#include <iostream>
using namespace std;
int main()
{   int m,k,temp ,n=0;
    cin>>m;
    temp=m;
    while(temp>0)
    {   k=temp%10;
        n=n*10+k;
        temp=temp/10;
    }
    if(m==n)
        cout<<m<<"is palindrome number"<<endl;
    else   cout<<m<<"is NOT palindrome number"<<endl;
    return 0;
}
```



求m的逆序

汽车的速度

§ 汽车里程表上的读数是一个回文数95859，7小时之后里程表的读数依然是一个5位的回文数，问汽车的速度（只考虑整数）

§ 分析

★ 条件1: $95859 + 7 * \text{speed} = \text{回文数 } (x)$

★ 条件2: $x \in (95859, 100000)$

★ 条件3: 整数，即 $(x - 95859)$ 为 7 的倍数

❖ $\text{speed} = (x - 95859) / 7$

第一种方法

```
int main()
{ int s, a, b, c, e, d; double f;
  s = 95959;
  while (s <= 99999 )
  {
    a = s / 10000;
    b = (s - a * 10000) / 1000;
    c = (s - a * 10000 - b * 1000) / 100;
    d = (s - a * 10000 - b * 1000 - c * 100) / 10;
    e = s - a * 10000 - b * 1000 - c * 100 - d * 10;
    f = s - 95859;
    if ( a == e && b == d && f % 7 == 0 )
      cout << "The speed is " << f / 7 << end;
    s++;
  }
  return 0;
}
```

循环(99999-95959)次

分离出每一位

已知位数的回文判断方法

第一种方法：如何提高效率

```
int main()
{ int s, a, b, c, e, d; double f;
  s = 95959;
  while (s <= 99999 )
  {   f = s - 95859;
      if (f % 7 == 0 ) // 6/7的情况不需下面运算
      {   a = s / 10000;
          b = (s - a * 10000) / 1000;
          c = (s - a * 10000 - b * 1000) / 100;
          d = (s - a * 10000 - b * 1000 - c * 100) / 10;
          e = s - a * 10000 - b * 1000 - c * 100 - d * 10;
          if ( a == e && b == d )
              cout << "The speed is " << f / 7 << end;
      }
      s++;
  }
  return 0; }
```

分开判断、尽快分流

第二种方法

```
int main()
{
    int s, a, b, c, d, i;
    for ( i = 1; i <= 591; i++)
    {
        s = 95859 + i * 7;
        a = s / 10000
        b = (s - a * 10000) / 1000;
        c = (s - a * 10000 - b * 1000) / 100;
        d = (s - a * 10000 - b * 1000 - c * 100) / 10;
        e = s % 10;
        if (a == e && b == d)
            cout << "The speed is " << i << endl;
    }
    return 0;
}
```

只需要循环591次

为什么是591?

只考虑7的倍数!

与前一页方法比较, 减少了哪些运算?

第三种方法

```
int main()
```

```
{
```

```
    int s, i, j;
```

```
    s = 95959 - 95859;
```

```
    if ( s % 7 == 0)
```

```
        cout << "The speed is " << s / 7 << endl;
```

```
    for (i = 6; i <= 9; i++)
```

```
    {
```

```
        for (j = 0; j <= 9; j++)
```

```
        {
```

```
            s =(90000 + i * 1000 + j * 100 + i * 10 + 9) - 95859;
```

```
            if (s % 7 == 0 )
```

```
                cout << "The speed is " << s / 7 << endl;
```

```
        }
```

```
    }
```

```
    return 0;
```

```
}
```

万位只能是9，个位也必为9
千位从6开始，十位同样取值
百位从0到9

回文的基本要求!

95859

95959

96069

。 。 。

99999

需要循环

$(9 - 5) * 10 + 1 \text{次} = 41$

省略掉了回文数的计算!

寻找不同的方法，选择最优解法

典型的算法设计：穷举法

§ **基本思想**：首先根据问题的部分条件预估答案的范围，然后在此范围内对所有可能的情况进行逐一验证，符合所举条件的情况均为答案，直到将所有情况验证完毕。

§ **上述例子的求解方法即为穷举法**

- ★ 穷举法也需要考虑缩小范围

- ★ 充分挖掘问题中所蕴含的各种条件来缩小范围

穷举法解百钱百鸡问题

§ 百钱百鸡问题

★其题目如下：鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一；百钱买百鸡，翁、母、雏各几何？

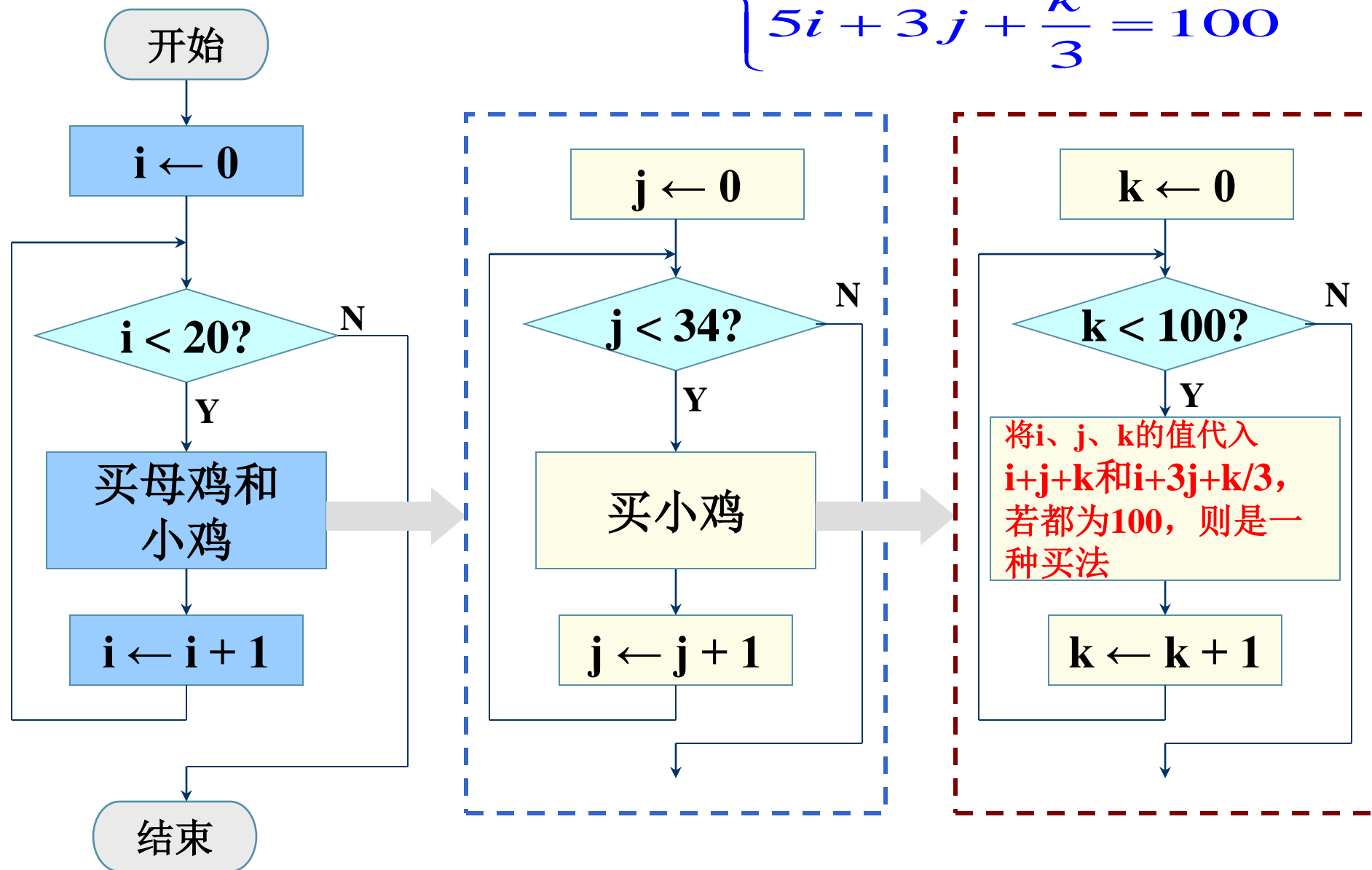
➤ 解：设*i*、*j*、*k*分别代表公鸡、母鸡、小鸡的数量，根据题意列方程：

$$\begin{cases} i + j + k = 100 \\ 5i + 3j + \frac{k}{3} = 100 \end{cases}$$

根据题意可知，*i*、*j*、*k*的范围一定是0到100的正整数，那么，最简单的解题方法是：穷举*i*、*j*、*k*每一种可能的取值组合，直接代入方程组，若满足该方程组则是解。这样即可得到问题的全部解（*i*、*j*、*k*的范围可否缩小？）。

百钱百鸡问题

$$\begin{cases} i + j + k = 100 \\ 5i + 3j + \frac{k}{3} = 100 \end{cases}$$



```
{ int i, j, k;
```

```
  i = 0;
```

```
  while (i < 20 ) {
```

```
    j = 0;
```

```
    while (j < 34) {
```

```
      k = 0;
```

```
      while ( k < 100) {
```

```
        if ( i + j + k == 100 && i*5 + j*3 + k/3 == 100)
```

```
          cout<<i<<j<<k<<endl;
```

```
          k++;}
```

```
    j++; }
```

```
  i++;
```

```
}
```

$$\begin{cases} i + j + k = 100 \\ 5i + 3j + \frac{k}{3} = 100 \end{cases}$$

```
}
```

百钱百鸡问题 (for)

main()

{

int i, j, k;

$$\begin{cases} i + j + k = 100 \\ 15i + 9j + k = 300 \end{cases}$$

```
for(i = 0; i < 20; i++ ) {
```

```
    for(j = 0; j < 34; j++) {
```

```
        for(k = 0; k < 100; k++) {
```

```
            if ( i + j + k == 100 && i*15 + j*9 + k == 300)
```

```
                cout<<i<<j<<k<<endl;
```

```
        }
```

```
    }
```

```
}
```

```
}
```

优化：减少循环层次与循环次数

main()

{

int i, j, k;

$$\begin{cases} i + j + k = 100 \\ 15i + 9j + k = 300 \end{cases}$$

```
for(i = 0; i < 20; i++ ) {
```

```
    for(j = 0; j < 34; j++) {
```

```
        k = 100 - i - j ;
```

```
        if (i*15 + j*9 + k == 300)
```

```
            cout<<i<<j<<k<<endl;
```

```
    }
```

```
}
```

```
}
```

亲和数

§ 遥远的古代，人们发现某些自然数之间有特殊的关系：

★ 两个数 a 和 b ， a 的所有除本身以外的因数之和等于 b ， b 的所有除本身以外的因数之和等于 a ，则称 a, b 是一对亲和数

例如：220和284就是一对亲和数

220的真因子包括：1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110.

$1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284$

284的真因数： $1 + 2 + 4 + 71 + 142 = 220$

§ 任意给定一个正整数 n ($1 \leq n \leq 100000$)，计算小于等于 n 的所有亲和数对 a, b

• 例子输入

1500

• 例子输出

220,284

1184,1210

亲和数求解分析

§ 计算 a 的因子之和 b (如何计算?)

§ 判断 b 是否在 $[1, n]$ 范围内, 若是, a 和 b 即为一对亲和数

★ 注意, 不要重复找, (a, b) 和 (b, a) , 可以限定 $a \leq b$

§ 自己完成并优化

另一个例子：谁做了好事？

§ 北大附小有四位同学中的一位做了好事，不留名，表扬信来了之后，校长问这四位是谁做的好事。

❖ A说：不是我。

❖ B说：是C。

❖ C说：是D。

❖ D说：他胡说（C胡说，不是D）。

§ 已知**只有三个人说的是真话**，一个人说的是假话。
现在要根据这些信息，找出做了好事的人。

如何求解？穷举！

§ 思路：

- ★ 首先**假定**做好事的是某个人；
- ★ 然后去**推理**，推理方法是检测是否所有人的话正确；
- ★ 如果能够保证**有三句**是正确的，则假设正确；
- ★ 否则**更换**一个人，再做同样的尝试；

面临的问题

1. 如何表示数据？

- 如何表示做好事的人？
- 如何表示A/B/C/D所说的话

2. 如何实现算法？

- 先假设A为“好人”，然后分别假设B、C、D为“好人”；用循环依次控制（如何转换控制？）
- 如何检测恰好3个人的话正确；

四句话的描述

§ A说：不是我。写成关系表达式为

$(\text{thisman} \neq 'A')$

§ B说：是C。 写成关系表达式为

$(\text{thisman} == 'C')$

§ C说：是D。 写成关系表达式为

$(\text{thisman} == 'D')$

§ D说：他胡说。写成关系表达式为

$(\text{thisman} \neq 'D')$

四句话的进一步分析

§ 先假定是A同学，让thisman='A';

§ 代入到四句话中

- | | |
|------------------------------|---------|
| ■ A说: thisman!='A'; 'A'!='A' | 假, 值为0。 |
| ■ B说: thisman=='C'; 'A'=='C' | 假, 值为0。 |
| ■ C说: thisman=='D'; 'A'=='D' | 假, 值为0。 |
| ■ D说: thisman!='D'; 'A'!='D' | 真, 值为1。 |

对应的程序

```
#include <iostream>
using namespace std;
int main()
{
    thisman = 'A';
    sum = (thisman != 'A')
        + (thisman == 'C')
        + (thisman == 'D')
        + (thisman != 'D');
    if (sum==3)
    {
        cout << “做好事的是： ” <<thisman<<endl;
    }

    return 0;
}
```

四句话的进一步分析

§ 依次再假定是B/C/D同学; 并代入到四句话中

- A说: ...
- B说: ...
- C说: ...
- D说: ...

如何寻找一致的控制

- ★ 解决方案中先假设A为“好人”，然后分别假设B、C、D为“好人”；
- ★ 每次循环的标志是A、B、C、D，如何转换为循环？
 - A、B、C、D之间的共同变化点：

■ 字符	A	B	C	D
■ ASCII码值	65	66	67	68
<ul style="list-style-type: none">• $B = 'A' + 1;$• $C = 'A' + 2;$• $D = 'A' + 3;$				

完整程序

```
#include <iostream>
using namespace std;
int main()
{
    int k,sum = 0,g = 0;
    char thisman;
    for(k = 0; k < 4; k++)
    {
        thisman = 'A' + k;
        sum = (thisman != 'A')+(thisman == 'C') + (thisman == 'D') + (thisman != 'D');
        if (sum==3)
        {
            cout << "做好事的是: " <<thisman<<endl;
            g = 1;
        }
    }
    if( g == 0)
        cout << "此题无解" << endl;
    return 0;
}
```

例子：破案

•帮助某地刑侦大队利用以下已经掌握的确切线索，从6个嫌疑人中找出作案人：

- A，B至少有一人作案；
- A，E，F 3人中至少有2人参与作案；
- A，D不可能是同案犯；
- B，C或同时作案，或与本案无关；
- C，D中有且仅有1人作案；
- 如果D没有参与作案，则E也不可能参与作案

分析与思路

- § 采取枚举方法，枚举什么呢？
- § 6个人每个人都有作案或不作案两种可能，因此有 2^6 种组合，从中挑出符合6个条件的作案者
- § 定义6个整数变量，分别表示6个人A, B, C, D, E, F。
- § 枚举每个人的可能性
- § 让0表示不是罪犯；
- § 让1表示就是罪犯。

6个条件对应的表达

§ A, B至少有一人作案;

★ $cc1 = A \vee B$;

§ A, E, F 3人中至少有2人参与作案;

★ $cc2 = (A \wedge E) \vee (A \wedge F) \vee (E \wedge F)$

§ A, D不可能是同案犯;

★ $cc3 = \neg(A \wedge D)$

§ B, C或同时作案, 或与本案无关;

★ $cc4 = (B \wedge C) \vee (\neg B \wedge \neg C)$

§ C, D中有且仅有1人作案;

★ $cc5 = (C \wedge \neg D) \vee (\neg C \wedge D)$

§ 如果D没有参与作案, 则E也不可能参与作案

★ $cc6 = D \vee \neg E$

综合破案条件

$CC = CC1 \ \&\& \ CC2 \ \&\& \ CC3 \ \&\& \ CC4 \ \&\& \ CC5 \ \&\& \ CC6$

每个人有2种取值的可能， $A=0$ (清白), $A=1$ (作案)
将6个人的情况组合起来，共有64种可能，对每种可能用CC加以判断。

```
for (A = 0; A <= 1; A++)  
  for (B = 0; B <= 1; B++)  
    for (C = 0; C <= 1; C++)  
      for (D = 0; D <= 1; D++)  
        for (E = 0; E <= 1; E++)  
          for (F = 0; F <= 1; F++)  
            { 求出CC1,CC2,CC3,CC4,CC5,CC6;  
              if ((CC1+CC2+CC3+CC4+CC5+CC6)==6)  
                { 根据A,B,C,D,E,F输出每个人的状态; }
```

```

int main()
{
    int A,B,C,D,E,F,CC1,CC2,CC3,CC4,CC5,CC6;
    for ( A = 0; A <= 1; A++)
        for ( B = 0; B <= 1; B++)
            for ( C = 0; C <= 1; C++)
                for ( D = 0; D <= 1; D++)
                    for ( E = 0; E <= 1; E++)
                        for ( F = 0; F <= 1; F++)
                            {
                                CC1 = A || B;
                                CC2 = (A && E) || (A && F) || (E && F);
                                CC3 = !(A && D);
                                CC4 = (B && C) || (!B && !C);
                                CC5 = (C && !D) || (D && !C);
                                CC6 = D || !E;
                                if ( (CC1+CC2+CC3+CC4+CC5+CC6)== 6)
                                {
                                    cout << "A: " << (A==0? "innocent": "guilty") << endl;
                                    cout << "B: " << (B==0? "innocent": "guilty") << endl;
                                    cout << "C: " << (C==0? "innocent": "guilty") << endl;
                                    cout << "D: " << (D==0? "innocent": "guilty") << endl;
                                    cout << "E: " << (E==0? "innocent": "guilty") << endl;
                                    cout << "F: " << (F==0? "innocent": "guilty") << endl;
                                } //if
                            } //forF

    return 0;
}

```