第三章

流程控制

- 请用户输入两个浮点数，求和后保留两位有效数字输出。
- 如果任一个浮点数大于1e5，就直接退出；

本章重难点

重点

1. 掌握顺序结构程序设计
2. 理解五种类型的 C 语言语句
3. 学会使用 if ~else 结构实现条件分支
4. 学会使用 switch ~case 结构实现等值分支
5. 理解选择结构的嵌套
6. 学会使用 while、 do ~while、 for 语句实现循环
7. 学会使用循环嵌套和程序转移

难点

1. 理解用嵌套 if ~else 结构实现的多分支选择结构
2. 学会使用 while、 do ~ while、 for 语句嵌套实现多重循环
3. 理解程序转移对程序执行顺序的影响
4. 掌握分支、循环结构设计常见算法
5. 学会绘制 N-S 流程图描述算法

3.1 顺 序

- 顺序结构是一种最简单、最常用的结构，程序执行是完全按照语句出现的先后顺序依次执行。
- 图 3-1 表示的程序执行顺序为：先执行 A语句，再执行 B 语句。

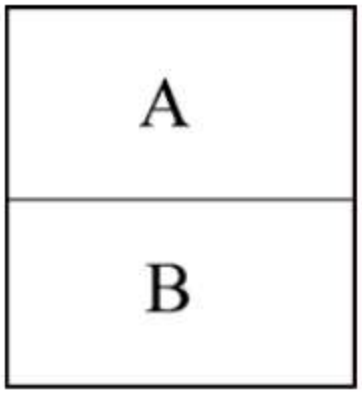


图 3-1 顺序结构的 N-S 图

【例 3-2】 用 C 语言编程实现输入一个五位正整数，要求顺序打印出各位的数字，具体格式为：假设输入的数是51268，则打印 5，1，2，6，8。

解题思路：要求设计从一个五位正整数分离出它的个位、十位、百位、千位、万位数字的算法。

试着写一下？

还能简短些吗？

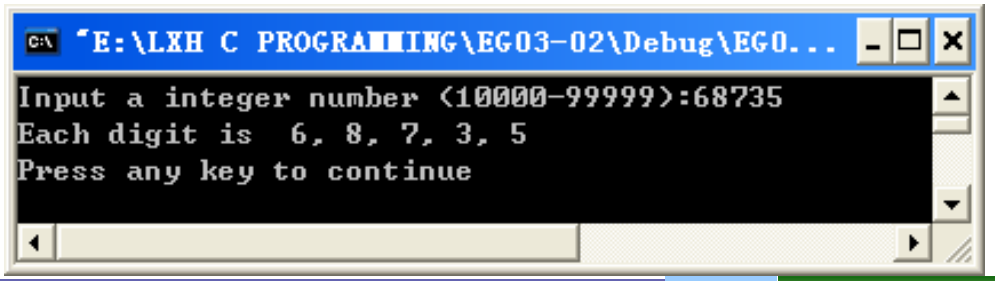
可以逆序打印吗？

输入一个五位正整数num
e = num % 10;
num /= 10;
d = num % 10;
num /= 10;
c = num % 10;
num /= 10;
b = num % 10;
num /= 10;
a = num
顺序打印出各位数字

图 3-4 ［例 3-2］的 N-S 图

```
/* Program: EG03-02.C                          */
/* Description: 输入一个五位正整数，要求顺序打印出各位数字。 */

#include <stdio.h>
void main( void )
{
    int num;
    int a, b, c, d, e;
    printf("Input a integer number (10000-99999):");
    scanf("%d", &num);
    e = num % 10;  num /= 10;
    d = num % 10;  num /= 10;
    c = num % 10;  num /= 10;
    b = num % 10;  num /= 10;
    a = num;
    printf("Each digit is %2d,%2d,%2d,%2d,%2d\n", a, b, c, d, e);
}
```

运行结果：



【想一想】

- （1）也可以将五位数分解为五个数字：

e=num%10; d=num/10%10; b=num/100%10; c=num/1000%10; a=num/10000%10;

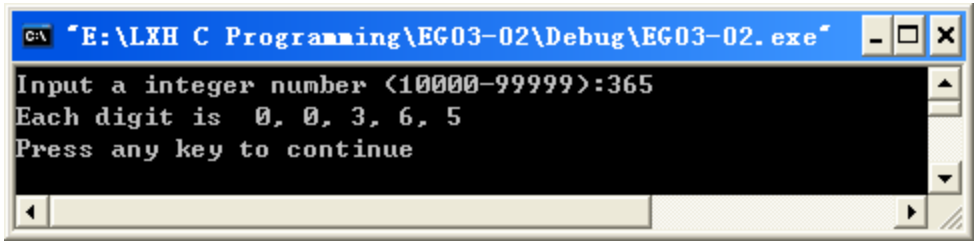
这种解题思路没有前面的清晰。前面的解题思路不仅容易理解，还很容易拓展到求 N位正整数的各位数字。

- （2）如果要求逆序打印各位数字，程序应该怎样修改？

由于已经求出了各位数字，要逆序打印各位数字，只需要修改输出的顺序即可：

printf("Inverse numberis %d%d%d%d%d\n", e, d, c, b, a);

- （3）如果不小心输入了三位正整数，程序运行结果会怎么样呢？



【例 3-3】 用 C 语言编程实现:从键盘输入一个一元二次方程$ax^2+bx+c=0$ 的三个系数 a、b、c，要求计算并打印出方程的两个实数根。

解题思路：我们选择公式法来求解本题。
公式法求解一元二次方程，
首先要求解判断式 b2-4ac，
然后计算判断式的平方根，计算（−b+平方根）/2a、（−b−平方根）/2a 分别存入两个浮点型变量，
打印输出两个解。

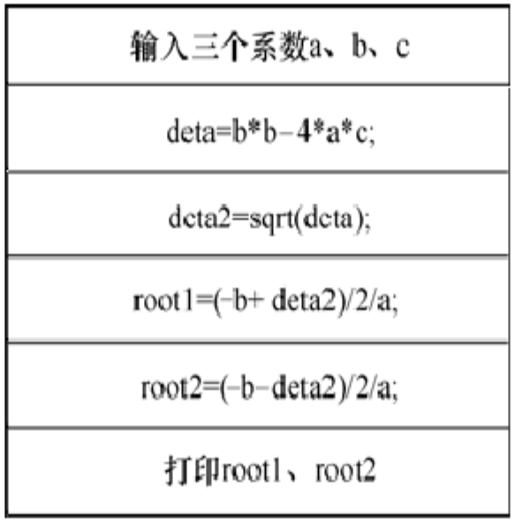
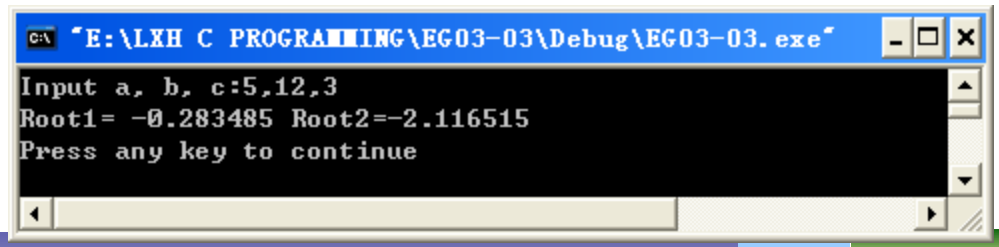


图 3-7 ［例 3-3］的 N-S 图

```
/* Program: EG03-03.C                                    */
/*  Description: 输入一元二次方程的三个系数a、 b、 c,要求计算并打印出方程
的两个实数根。  */

#include <stdio.h>
#include <math.h>  /*程序中调用了求平方根的标准库函数 sqrt()，所以必须嵌
入相应的包含头文件：math.h。*/
void main( void )
{
    double a, b, c, deta, deta2, root1, root2;
    printf("Input a, b, c:");
    scanf("%lf,%lf,%lf", &a, &b, &c );
    deta=b*b-4*a*c;
    deta2=sqrt(deta);
    root1=(-b+deta2)/2/a;
    root2=(-b-deta2)/2/a;
    printf("Root1= %lf Root2=%lf\n", root1, root2 );
}
```

运行结果：



【想一想】

- 求解一元二次方程判断式时，deta>0，方程有两个不相等的实数根；deta=0，方程有两个相等的实数根；deta <0，方程没有实数根。本题没有对 deta 作判断就计算判断式的平方根，然后将（-b+平方根）/2a、（-b-平方根）/2a 分别存入两个浮点型变量，这样做是否妥当？

- 不妥！ 因为无法保证运行程序时输入的 a、b、c 使 deta>0。一旦 deta<0，程序便会出错。因此，必须引入选择结构来处理 deta=0、deta<0 这两种情况。

- 实际上，C 语言的语句除了表达式语句、函数调用语句、空语句、复合语句这四种语句以外，还包括第五种语句——控制语句：if～else 语句、switch～case 语句、while、do～while和 for语句。

3.2 选　　择

- 设计选择结构程序时，首先要对给定的条件进行判断，根据判断的结果决定执行哪一种操作。

3.2.1 if～else 结构

C 语言提供的选择结构是 if～else 结构,语法是：

```
if(条件表达式)
    语句1;
else
    语句2;
```

- **if～else 结构的执行过程：**首先要对给定的条件表达式进行计算，如果计算结果是 1，则执行语句 1，语句 1 通常被称为 if分支（真分支）；否则执行语句2，语句2 通常被称为 else 分支（假分支）。
- 其中，条件表达式既可以是逻辑表达式、关系表达式，也可以是任何合法的 C 语言表达式。
- 需要说明，设计程序时，书写的条件表达式值非 0 时即为逻辑真（C 语言将所有的非 0 值均视为逻辑真处理）。而逻辑运算的结果只有 0（假）和 1（真）两种可能。

```
if(条件表达式)
    语句1;
else
    语句2;
```

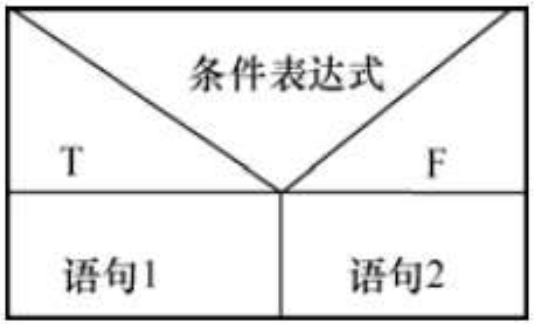


图3-9 选择结构的N-S图

【例 3-4】 用选择结构改进［例 3-2］：输入一个五位正整数，要求顺序打印出各位数字。如果输入的数字不是五位正整数，则给出出错提示。

解题思路：当程序接收到键盘输入的整数后，需要判断它是否是五位正整数，如果不是五位正整数，则给出相应的输入错误提示：**Error input!**

翻译成 C 语言的逻辑表达式即为：

num > 99999 || num < 10000

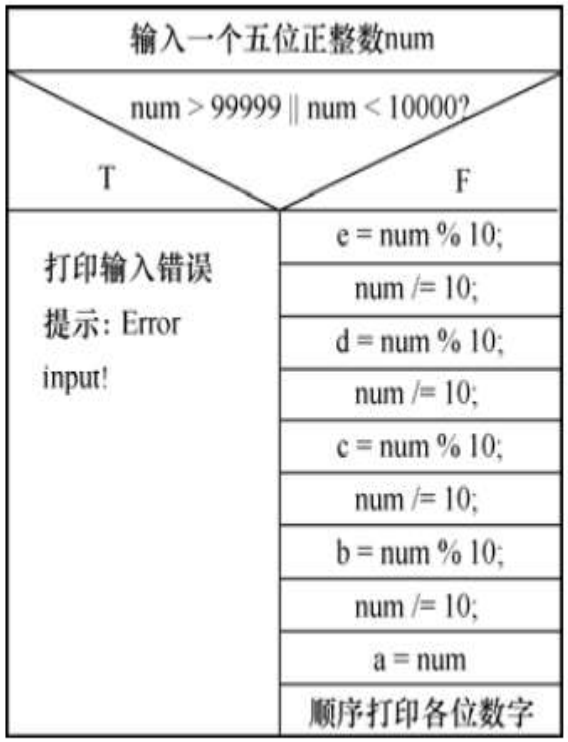
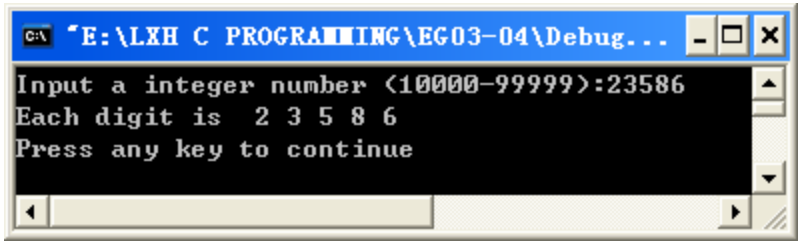
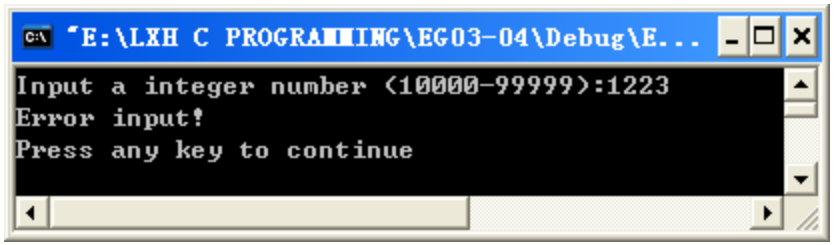


图 3-10 ［例 3-2］的 N-S 图

```
/* Program: EG03-04.C                                    */
/* Description: 输入一个五位正整数, 要求顺序打印出各位数字。 */
#include <stdio.h>
void main( void )
{
    int num;
    int a, b, c, d, e;
    printf("Input a integer number (10000-99999):");
    scanf("%d", &num);
    if( num > 99999 || num < 10000 )
        printf("Error input!\n");
    else
    {
        e = num % 10;  num /= 10;
        d = num % 10;  num /= 10;
        c = num % 10;  num /= 10;
        b = num % 10;  num /= 10;
        a = num;
        printf("Each digit is %2d%2d%2d%2d%2d\n",a, b, c, d, e);
    }
}
```

运行结果





【例 3-5】 用 C 语言的选择结构实现下述功能：从键盘输入一个年份，要求判断出它是否是平年。

解题思路：

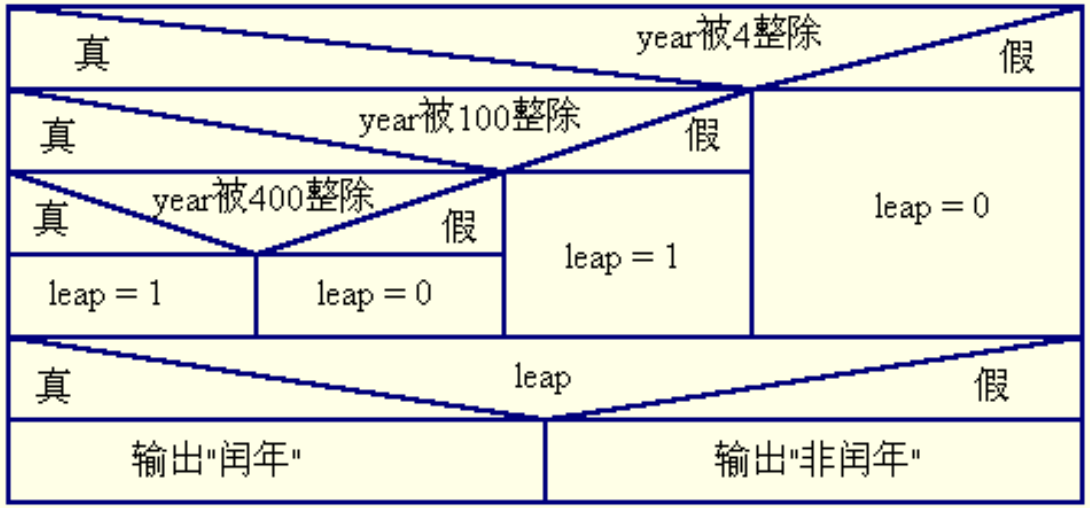
◆判断闰年：年份能被4整除但不能被100整除；或者能被100整除且能被400整除。

```
((year % 4 = = 0 )&& (year % 100 != 0)) ||
((year % 100 = =0) && (year % 400==0))
```

◆判断平年：年份不能被4整除且不能被100整除；或者能被100整除但不能被400整除。

```
((year % 4 != 0 )&& (year % 100 != 0)) ||
(( year % 100 == 0 ) && ( year % 400 !=0))
```

```
/* Program: EG03-05.C */
/* Description: 从键盘输入一个年份, 要求判断出它是否是平年 */

#include <stdio.h>

void main( void )
{
    int year;

    printf("请输入一个年份:");
    scanf("%d", &year );

    if( ( ( year % 100 != 0 ) && ( year % 4 != 0 ) ) ||
      ( ( year % 100 == 0 ) && ( year % 400 != 0 ) ) )
            printf("%d 年是平年。\n", year );
    else
            printf("%d 年是闰年。\n", year );

}
```



【想一想】

- （1）本题设计的关于年份 year是否是平年的条件表达式太长了，能否简捷点？
  - 判断平年：年份不能被4整除且不能被100整除；或者能被100整除但不能被400整除。

```
((year % 4 != 0 )&& (year % 100 != 0)) ||
(( year % 100 == 0 ) && ( year % 400 !=0))
```

- 考虑到不能被4 整除的年份肯定是不能被100整除，表达式可以简化为:

```
(year % 4 != 0 ) || (( year % 100 == 0 ) && ( year % 400 !=0))
```

- 再考虑非 0 值均视为逻辑真处理，条件表达式可以进一步简化为：

```
(year%4) || !( year%100) && (year%400)
```

那如何判断闰年呢？

记住，对于表达式a而言：

if(a)等价于 if(a!=0)　　if(!a)等价于if(a==0)

几点注意：

（1）else 以及 else 分支可以缺省。

```
min = first;
if( second < min )          /*求first、second中的小数*/
      min = second; ……
```

但绘制缺省了 else 以及 else 分支的 N-S 图时必须画出空白 else分支。

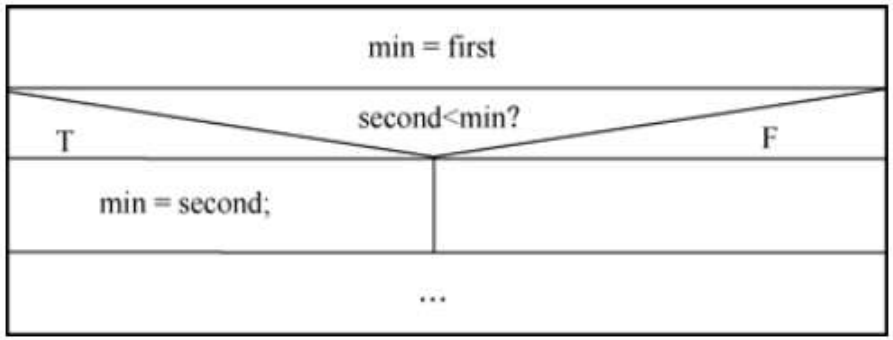


图 3-14　缺了 else 分支的 N-S 图

（2）包含 if 分支和 else 分支的 if～else 结构是一条选择语句。缺省了 else 以及 else 分支的 if结构仍然是一条选择语句。

（3）else 以及 else 分支不能作为语句单独使用。

```
if( a > b )
      c=a;
      a=b;
else
      b=c;
```

在 if和 else之间出现了两条语句，这种程序设计错误多半是因为试图将多于一条的语句充当 if 分支或试图将 else 以及 else 分支作为语句单独使用。

if～else 结构是一条完整的选择语句。虽然 if～else 结构允许省略 else 以及 else 分支，但 else 必须和 if 配对出现，不能作为语句单独使用。

（4）if 分支和 else 分支都只能是一条语句。可以将多个语句用“{}”括起来成为一条复合语句充当 if分支或 else分支：

```
float a, b, tmp;     /* 对a,b按从大到小排序 */
if( a < b )          /* 若a < b, 交换 a, b的值 */
{
      tmp=a;
      a=b;
      b=tmp;
}
```

3.2.2 if嵌套

if～else 结构的嵌套：

```
if(条件表达式1)
    if(条件表达式2)
        语句1;
    else
        语句2;
else
    if(条件表达式3)
        语句3;
    else
        语句4;
```

执行过程：

先判断条件表达式 1是否为真？若条件表达式 1为真，再判断条件表达式 2 是否为真？若条件表达式 2 也为真，执行语句 1，否则，执行语句 2；

若条件表达式 1 为假，则判断条件表达式 3 是否为真？若条件表达式 3 为真，执行语句 3，否则，执行语句 4。

当语句 1、语句 2、语句 3、语句 4 也是 if～else 结构时，这就构成了 if～else 结构的多重嵌套。某些 if～else结构还可能省略了 else以及 else 分支。要注意 if和 else 的配对问题。

例如：

```
if(条件表达式1)
if(条件表达式2)
 语句1;
else
 语句2;
```

其中的 else究竟和第一个 if配对？

```
if(表达式1)
    if(表达式2)
        语句1;
else
    语句2;
```

还是和第二个 if配对？

```
if(表达式1)
    if(表达式2)
        语句1;
    else
    语句2;
```

C 语言规定：else 总是与它前面最近的且无 else 配对的 if 配对。

```
if(条件表达式1)
if(条件表达式2)
    语句1;
else
    语句2;
else
    语句3;
```

相当于

```
if(条件表达式1)
  if(条件表达式2)
     语句1;
  else
     语句2;
else
  语句3;
```

C 语言还允许使用“{}”来限定内嵌 if的使用范围：

```
if(条件表达式1)
{
  if(条件表达式2)
     语句1;
}
else
   语句2;
```

由于使用“{}”来限定第二个 if的使用范围，所以这里的 else 与第一个 if配对。

【例 3-6】 用 C 语言的选择结构实现符号函数 sign(x)：从键盘输入一个浮点数 x，要求输出它的符号。 sign(x)是一个分段函数：

$$\text{sin}(x)\begin{cases} -1 & x<0 \\ 0 & x=0 \\ 1 & x>0 \end{cases}$$

解题思路：定义了一个 double 型变量 x 接受键盘输入的浮点数，定义了一个 short 型变量 sign 存放求出的符号，运用嵌套选择结构编写的程序如下：

```
#include <stdio.h>
void main( void )
{
    double x;
    short sign;
    printf("Input x:");
    scanf("%lf", &x);
    if( x>=0  )
        if( x>0  )
            sign =1;
        else
            sign =0;
    else
        sign =-1;
    printf("sign(%lf)=%d\n", x, sign);
}
```

可以使用三条缺省else 分支的if结构求x 的符号：

```
if( x>=0  )
      if( x>0  )
           sign =1;
      else
           sign =0;
else
      sign =-1;
```

相当于

```
if( x<0 )  sign =-1;
if( x==0 )  sign =0;
if( x>0 )   sign =1;
```

多分支通常使用else 分支嵌套if-else 结构求x 的符号：

```
if( x<0 )              sign = -1;
else if( x==0 )    sign = 0;
else if( x>0 )      sign = 1;
```

相当于

```
if( x<0 )
     sign =-1;
else
     if( x==0 )
          sign =0;
     else
     if( x>0 )
          sign =1;
```

【例 3-7】 用 else 分支嵌套 if-else 结构实现下述功能，从键盘输入一个一元二次方程ax2+bx+c=0的三个系数a、b、c，要求计算并打印出方程的实数根。

解题思路：本题将输入的三个系数存放在浮点型变量a、b、c中，首先要求解公式法的判断式：deta = b2 − 4ac；

当deta=0 时， 方程有两个相等的实数根：−b/2a；

当deta <0 时，方程没有实数根；

当deta>0时，方程有两个不相等的实数根：
root1 = (−b+deta)/2a、
root2 = (−b−deta) /2a。

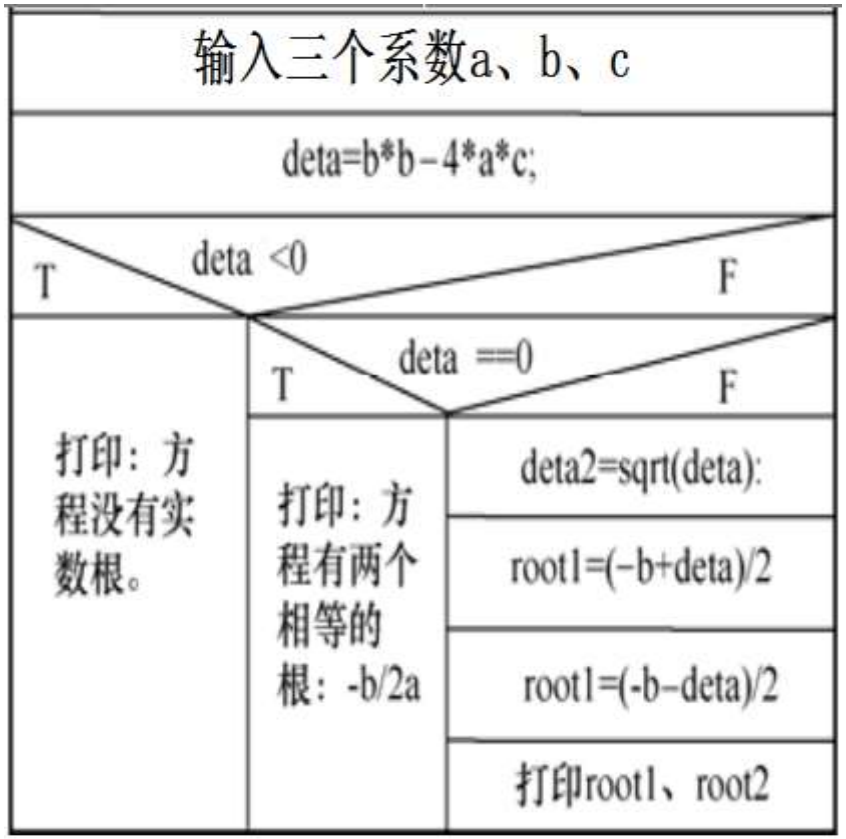


图 3-15 ［例 3-7］的 N-S 图

```
#include <stdio.h>
#include <math.h>  //需要调用到sqrt( )函数
void main( void )
{
    double a, b, c, deta, deta2, root1, root2;
    printf("请输入一元二次方程的三个系数a, b, c:");
    scanf("%lf,%lf,%lf", &a, &b, &c );
    deta = b*b-4*a*c;
    if(deta<0)
        printf("方程没有实数根。\n");
    else if(deta==0)
        printf("方程有两个相等的根： %lf\n", -b/2/a );
    else
    {
        deta2 = sqrt( deta );
        root1 = ( - b + deta2 ) / 2 / a;
        root2 = ( - b - deta2 ) / 2 / a;
        printf("Root1= %lf Root2=%lf\n", root1, root2 );
    }
}
```

3.2.3 switch～case 结构

- if～else 结构只能处理从两者之间选择其一，当要实现更多可能之间选择其一时，就要用嵌套 if～else 结构来实现。

- 可是，当供选择的分支较多时，程序会变得逻辑复杂冗长，难以理解。为此，C 语言专门提供了switch～case 结构处理多路等值选择分支的情形。

switch～case结构的一般格式

```
switch(整型表达式)
{
    case 常量表达式 1:
            [语句组 1;
            [break;]]
    case 常量表达式 2:
            [语句组 2;
            [break;]]
    …
    case 常量表达式 n:
            [语句组 n;
            [break;]]
    default:
            [语句组 n;]
}
```

switch～case 结构的执行过程

➢计算整型数值表达式的值，并逐个与其后的各常量表达式的值进行比较，当表达式的值与某个常量表达式的值相等时，先执行其后的语句组，若该 case 分支最后有 break 语句，则中止 switch～case 结构，转到switch～case 结构后的程序顺序执行；

➢若该case 分支没有break语句，则继续执行下一个case 分支。若整型数值表达式的值与所有case 后的常量表达式的值均不相等，则执行default分支。

➢其中各常量表达式的值必须是整型，字符型或者枚举类型。

➢各语句组允许有多条语句，不需要加“{}”。若语句组 i为空，该case 分支又没有break 语句，则直接执行下一个case 分支。

【例 3-8】 用 switch～case 结构编写一个能进行四则运算的计算器程序：用户输入运算数和四则运算符，程序计算并输出结果。

解题思路：

本题需要定义两个用于存放运算数的单精度浮点型变量num1、num2，一个存放四则运算符的字符型变量 op，利用格式化输入函数 scanf 接受一个四则运算表达式后，需要用switch～case 结构根据不同的运算符，编写相应的 case 分支，进行不同的运算，打印表达式及其计算结果。

对于不能识别的运算要通过 default 分支给出“错误的运算符！ ”提示。对于除法运算符，还要增加对除数是否为 0 的判断和处理。

每个 case 分支的最后都要有 break 语句，中止switch～case 结构，结束程序的执行。

```
#include <stdio.h>
void main( void )
{
    float a, b;  char op;
    printf("请输入一个四则运算表达式(eg:1.5/2.5):");
    scanf("%f%c%f", &a, &op, &b );
   switch(op)
   {
     case '+':
           printf("%f + %f =%f\n", a, b, a + b); break;
     case '-':
           printf("%f - %f =%f\n", a, b, a - b); break;
     case '*':
           printf("%f * %f =%f\n", a, b, a * b); break;
     case '/':
           if(b)   //相当于if(b!=0)
                printf("%f / %f =%f\n", a, b, a / b);
           else
                printf("除数不能为0\n"); break;
     default:
            printf("错误的运算符！ \n");
   }
}
```

【例 3-9】 用 switch～case 结构编程实现输入一个百分制成绩，将其转换成五级记分制成绩并输出结果。具体转换标准为：100～90 分→等级A，80～89 分→等级B，70～79 分→等级C，60～69 分→等级D，60 分以下→等级E。

解题思路：

本题需要定义一个用于存放百分制成绩的短整型变量score， 在利用格式化输入函数scanf接受一个百分制成绩后，需要借助于整数的整除性质将诸如80 <= score && score <= 89的条件判断转化成score / 10 = = 8的等值判断， 然后用switch～case结构根据不同的等值， 编写case分支打印相应的五级记分制等级。

需要注意的是，10、9分支相同，5、4、3、2、1、0分支相同。对于不正确的成绩要通过default分支给出“输入的成绩错误！ ”提示。

每个case分支的最后都要有break语句，中止switch～case结构，结束程序的执行。

```
#include <stdio.h>
void main( void )
{
    int score;
    printf("请输入一个百分制成绩:");
    scanf("%d", &score );

    switch(score / 10 )
    {
        case 10:
        case 9:
            printf("A\n");
            break;
        case 8:
            printf("B\n");
            break;
        case 7:
            printf("C\n");
            break;
        case 6:
            printf("D\n");
            break;
        case 5:
        case 4:
        case 3:
        case 2:
        case 1:
        case 0:
            printf("E\n");
            break;
        default:
            printf("输入的成绩错误！\n");
    }
}
```

【想一想】

（1）如果共同的分支过多，可以在 switch～case 结构之前利用单分支 if 结构将其转换成特定数值：

```
if( 0 <= score && score < 60)
        score = 0;
…
            case 6:
                printf("D\n");
                break;
            case 0:
                printf("E\n");
                break;
…
```

（2）如果要求输入一个带小数的百分制成绩，要求将其转换成五级记分制成绩，应该怎样写？看起来似乎不难，将score 定义成单精度浮点型变量，对程序做如下修改：

```
float score;
printf("请输入一个百分制成绩:");
scanf("%f ", &score );

switch( (int)score / 10 )
…
```

结果运行程序，显示出错信息。

这实际上是 VC 编译器的一个 bug。如果在 VC 中编写普通 C 语言程序，输入一个浮点数而没有其他浮点操作，VC 编译器根本就不连接浮点库。如果换个编译器，就没问题。

当然，也可以给程序增加一点浮点运算：

```
switch( (int)(score*1.0) / 10 )
```

（3）如果要求输入一个五级记分制成绩，将其转换成百分制成绩，应该怎样做？

例：
```
switch(grade)
{
    case 'A' : printf("85~100   ");break;
    case 'B' : printf("70~84    ");break;
    case 'C': printf("60~69    ");break;
    case 'D' :printf("<60     "); break;
    default : printf("error ");
}
```

若grede= 'A'，输出结果是什么？

85~100　70~84　60~69　<60　error

3.3 循 环

3.3.1 while语句

while 语句先判断给定的条件是否成立，在条件成立的前提下重复执行循环体，所以被称为“当型”循环控制语句。

while 语句的一般形式为：

```
while( 条件表达式 )
    语句;
```

其中被称为循环体的语句部分是一条语句。如果需要在循环体内执行多条语句，可使用复合语句：

```
while( 条件表达式 )
{
    语句;
    …
}
```

图 3-20 while 循环结构的 N-S 图

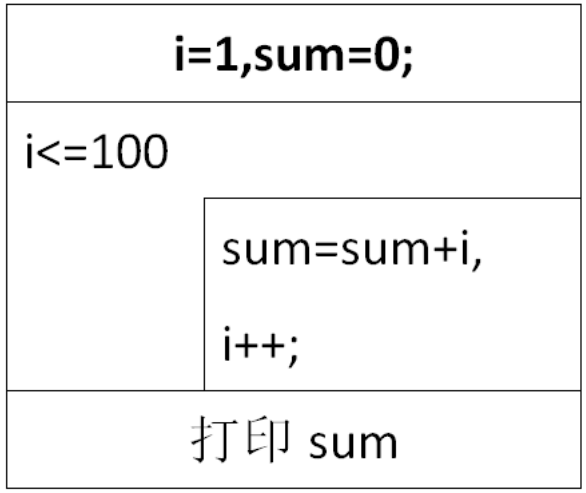
while 语句的执行过程：先计算**while** 后面的条件表达式，如果其值为真，则执行一次循环体；再次计算 **while** 后面的条件表达式，如果其值仍然为真，则再执行一次循环体；如此反复，直到**while** 后面的条件表达式的值为假，结束**while** 语句，程序开始顺序执行循环结构后面的语句。

while 语句的特点是：先判断，后执行，若一开始条件表达式就不成立，则循环体一次也不执行。

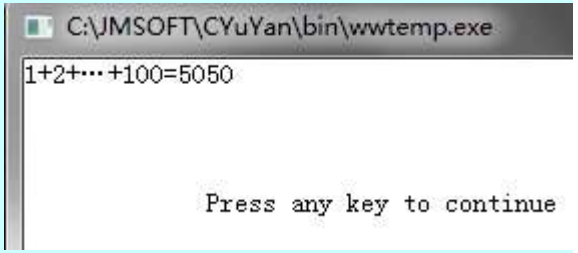
【例 】 用while 循环结构编程求 1+2+...+100 的和。

解题思路：先定义一个整型的累加器 sum，将第 i 项的值加到前 i-1 项的累加和里。累加器在累加之前必须清零：sum=0;

我们还需要定义一个整型计数器i，i 的初值为1，i 不断增1，从1变化到100，i每加一次1，sum =sum+ i；




```
#include <stdio.h>
void main( void )
{
    int i, sum;
    i=1, sum=0;
while( i <= 100 )
     sum = sum+i, i++;
    printf("1+2+…+100=%d\n", sum);
}
```



【例 3-10】 用while 循环结构编程求1+2+...+n 的和，n 从键盘输入。

解题思路：本题是一个累加求和的问题，我们需要先定义一个短整型的累加器sum，我们要做的工作就是不停地将第 i 项的值加到前 i-1 项的累加和里。累加器在累加之前必须清零：sum=0;

我们还需要定义一个短整型的计数器i，i 的初值为0，i 不断增1，从1变化到n，i 每加一次1，sum += i;

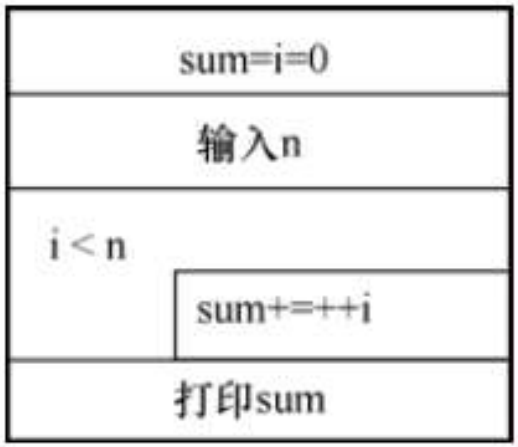
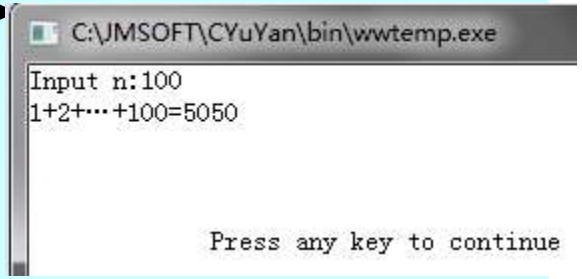
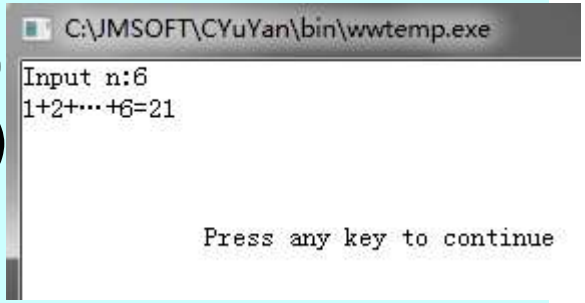


图 3-21 ［例 3.10］的 N-S 图

```
#include <stdio.h>
void main( void )
{
    short n, i, sum;
    i=sum=0;
    printf("Input n:")
    scanf("%d", &n )
    while( i < n )
     sum += ++ i;
    printf("1+2+…+%d=%d\n", n,
sum);
}
```





【想一想】

（1）为了使得循环体清晰、易懂，能否将循环体改为：

```
++ i;
sum += i;
```

如果不行，到底应该如何改？

```
while( i <n )
{
++ i;
sum += i;
}
```

【想一想】

（2）如果计数器i 的初值为1，后续的程序应该如何写？

```
#include <stdio.h>
void main( void )
{
   short n, i, sum;
   i=1,sum=0;
   printf("Input n:");
   scanf("%d", &n );
   while( i <=n )
   {sum += i; i++;}      // 等同sum += i++;
   printf("1+2+…+%d=%d\n", n, sum);
}
```

【想一想】

（3）下面程序段为什么不能得出正确结果，应该如何修改程序段？

```
i=sum=0;
printf("Input n:");
scanf("%d", &n );
while( ++ i < n )      // 条件表达式改为：++ i <= n
sum += i;
printf("1+2+…+%d=%d\n", n, sum);
```

【想一想】

（4）在求解累加和问题时，必须留意累加和的取值范围，以便为累加器选择合适的数据类型，如在求 1!+2!+…+n!=? 时，定义一个短整型的累加器 sum 便会造成数据溢出，我们应该为累加器选择一个数值范围更大的数据类型：float 或double。

在求解累加和问题时，在累加之前，累加器必须清零、计数器必须清零或赋 1；在累加过程当中，循环变量——控制循环结束的变量（此处为计数器 i）必须沿着某一趋势变化，直到循环条件不对应循环结束，否则，循环将无限进行下去，成为死循环。

【例3-11】 利用格里高利公式编程求π的近似值，直到最后一项的绝对值小于10^{-6}为止。

$$\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$$

解题思路：通过对奇数项和偶数项分别分析归纳，就会发现：
奇数项均为奇数的倒数，偶数项均为负的奇数倒数，这样，就用归纳法（递推法）找出了从第 i−1 项出发求第 i 项的规律：

$$pi_i \begin{cases} 0 & (\text{初值}) \\ pi_{i-1} + \dfrac{1}{2i-1} & (i = 1, 3, 5) \\ pi_{i-1} + \dfrac{-1}{2i-1} & (i = 2, 4, 6) \end{cases}$$

格里高利公式仍然是一个累加求和问题。因此，需要先定义一个双精度型的累加器pi，并在累加之
数器 i，i 的初值为 1，对应
变量 t存放奇数的倒数 1/(2*i
根据 i 的奇偶情况，将奇数
小于 10^{-6}为止；最后打印出

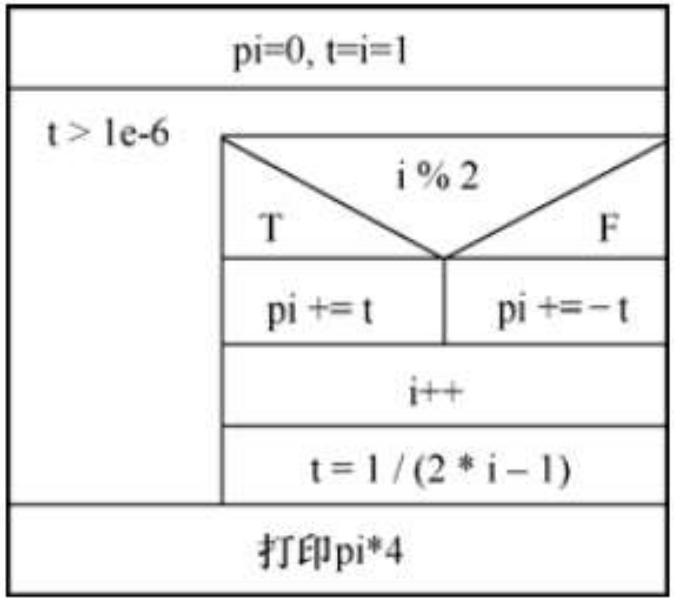


图 3-23 ［例 3.11］的 N-S 图

```
#include <stdio.h>
void main( void )
{
  double pi = 0, t = 1;
  short i = 1;
  while( t >1e-6)
  {
     if( i%2) // 等价if( i%2!=0) ,i为奇数！
        pi += t;
     else
        pi += - t;
     i ++ ;
     t=1.0 / (2 * i - 1);   //为什么是1.0
  }
printf(" pi = %lf\n", pi * 4);
}
```

【例3-12】 已知求平方根的迭代公式为：

$$x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n})$$

编程从键盘输入a，利用迭代法求$x=\sqrt{a}$，直到前后两次求出的x的差的绝对值小于10^{-6}为止。

很多实际问题不能用直接法（又称为一次解法）一次性解决问题。迭代法，也称辗转法，是用计算机解决问题的一种基本方法。它利用计算机运算速度快、适合做重复性操作的特点，让计算机按一定步骤重复执行一段程序，不断从变量的旧值递推出它的一个新值，直到新值精确或近似等于旧值为止。“二分法”和“牛顿迭代法”属于近似迭代法。

解题思路：

S1: 设定一个x 的初值x0，如x0＝a/2（也可以是其他值）；

S2: 用迭代公式求出下一个x 的初值x1；

S3: 检测前后两次求出的x 差值的绝对值是否小于10^{-6}；如果精度不够，执行S4；如果精度已经达到要求，转到S6 执行；

S4: 将xl 的值赋给x0，

S5: 将x0 代入迭代公式，求出新的x1；转到S3 执行；

S6: 打印求出的平方根。

```
/* Program: EG03-12.C */
/* Description: 利用迭代法求a 的平方根。 */
#include <stdio.h>
#include <math.h>    //程序中调用了求绝对值函数fabs();
void main( void )
{
double a, x0, x1;
printf("Enter a positive number:");
scanf("%lf", &a );

x0=a/2;   x1=(x0+a/x0)/2;

while( fabs(x0-x1)>=1e-6 )
{
x0=x1;    x1=(x0+a/x0)/2;
}
printf("The square root of %8.3lf is%6.3f\n", a, x1);
}
```

3.3.2 do～while 语句

do～while 语句也是 C 语言的“当型”循环控制结构，不过它和 while 语句有所不同，do～while 语句先执行一次循环体，然后才判断给定的条件是否成立，在条件成立的前提下重复执行循环体。

do～while 语句的一般形式为：

```
do
   语句;
while( 条件表达式 )
```

循环体只能是一条语句。如果需要在循环体内执行多条语句，可使用复合语句：

```
do
{
   语句;
   ……
} while( 条件表达式 )
```

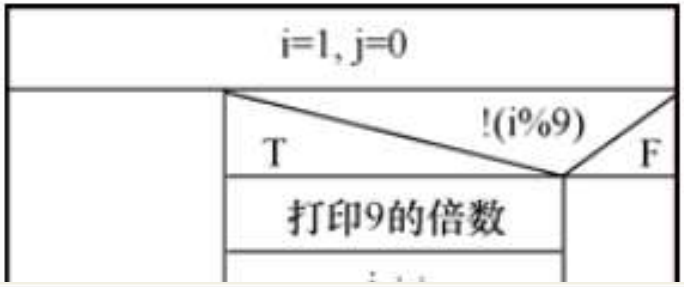
执行过程：先执行一次循环体，再计算 while后面的条件表达式，如果其值为真，则重复执行一次循环体；然后再次计算 while 后面的条件表达式，如此反复，直到条件表达式不再成立，结束 do～while 语句，程序开始顺序执行循环结构后面的语句。

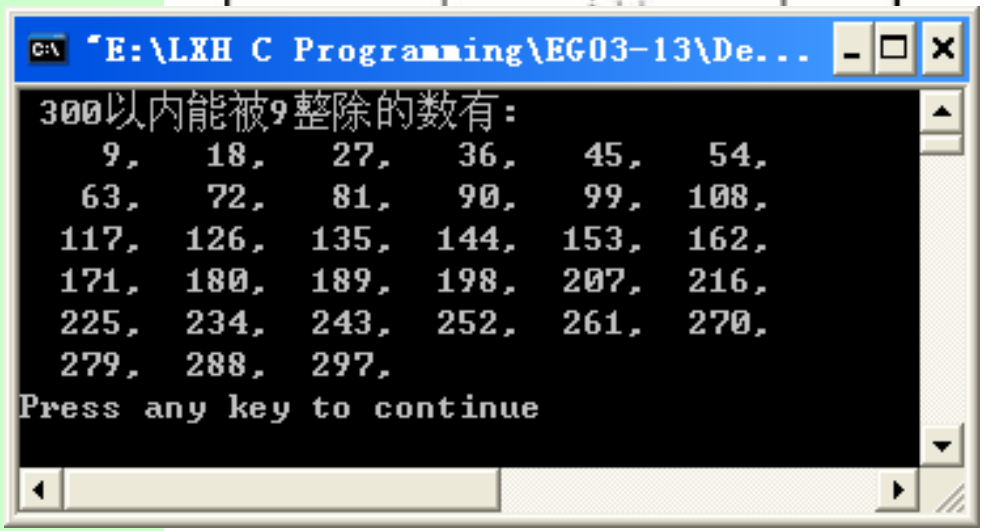
do～while 语句的特点是先执行，后判断，若一开始条件表达式就不成立，则循环体也要执行一次。

图 3-26 do～while 循环结构的 N-S 图

【例 3-13】 用 do～while循环结构编程求 300 以内所有能被 9 整除的数。程序的输出格式如下：

```
#include <stdio.h>
void main( void )
{
    int i=1, j=0;
    printf(" 300以内能被9整除的数有:\n");
    do
    {
        if( !(i%9) )   // 等价 if( i%9==0 )
        {
            printf("%5d,", i);
            j++;
            if( !(j%6) )
                printf("\n");
        }
    }while( ++i<300 );
    printf("\n");
}
```





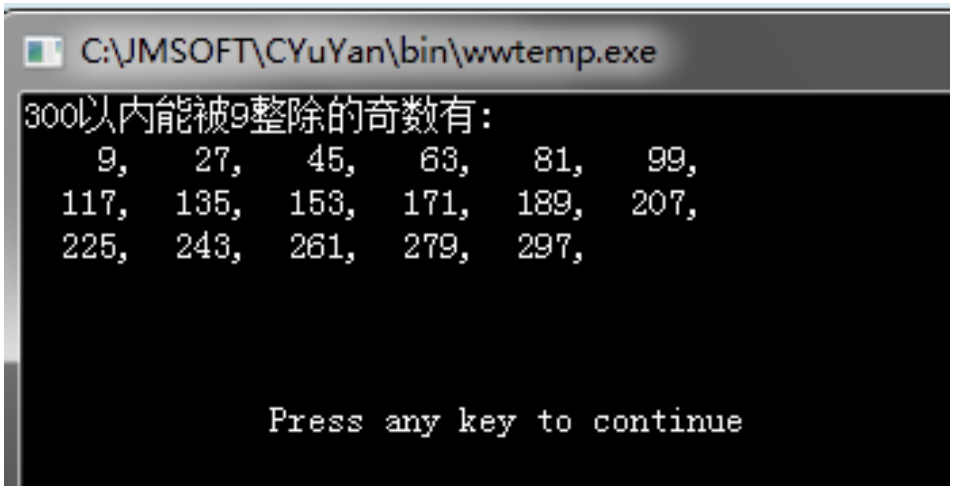
想一想：如果循环条件 while(++i<300);中的++i写到循环体内，该怎么改写？

```
#include <stdio.h>
void main( void )
{
    int i=1, j=0;
    printf(" 300以内能被9整除的数有:\n");
    do
    {
        if( !(i%9) )   // 等价 if( i%9==0 )
        {
            printf("%5d,", i);
            j++;
            if( !(j%6) )
                printf("\n");
        }
    } while( ++i<300 );
    printf("\n");
}
```

```
#include <stdio.h>
void main( void )
{
    int i=1, j=0;
    printf("300以内能被9整除的数有:\n");
    do
    {
    if( !(i%9) )
{
            printf("%5d,", i);
            j++;
            if( !(j%6) )
                printf("\n");
}
++i;
    } while( i<=300 );
    printf("\n");
}
```

想一想： 如果要求300以内能被9整除的奇数呢，程序如何修改？

```
#include <stdio.h>
void main( void )
{
    int i=1, j=0;
    printf(“ 300以内能被9整除的奇数有:\n");
    do
    {
    if( !(i%9)&&(i%2)) // 能被9整除，不能被2整除
{
            printf("%5d,", i);
            j++;
            if( !(j%6) )
                printf("\n");
        }
    }while( ++i<300 );
    printf("\n");
}
```



【例 3-15】用二分法求一元二次方程 $3x^3-2x^2+5x-16=0$ 在(−2,4)之间的根，要求精度为 10^{-6}。

用二分法求一元二次方程 f(x)=0 的根，先在根的左右区间各选一个点 x1、x2，对应的函数值 f(x1)、f(x2)肯定异号，将区间(x1,x2)二分得其中点 x0=(x1+x2)/2，求出f(x0)， 若精度不够， 选择 f(x1)、 f(x2)中和 f(x0)异号的那一个和 x0 组成新的区间继续搜索；就这样不断缩小解所在的区间范围，x0 逐步逼近方程的根 x*，直到 f(x0)达到精度要求为止。

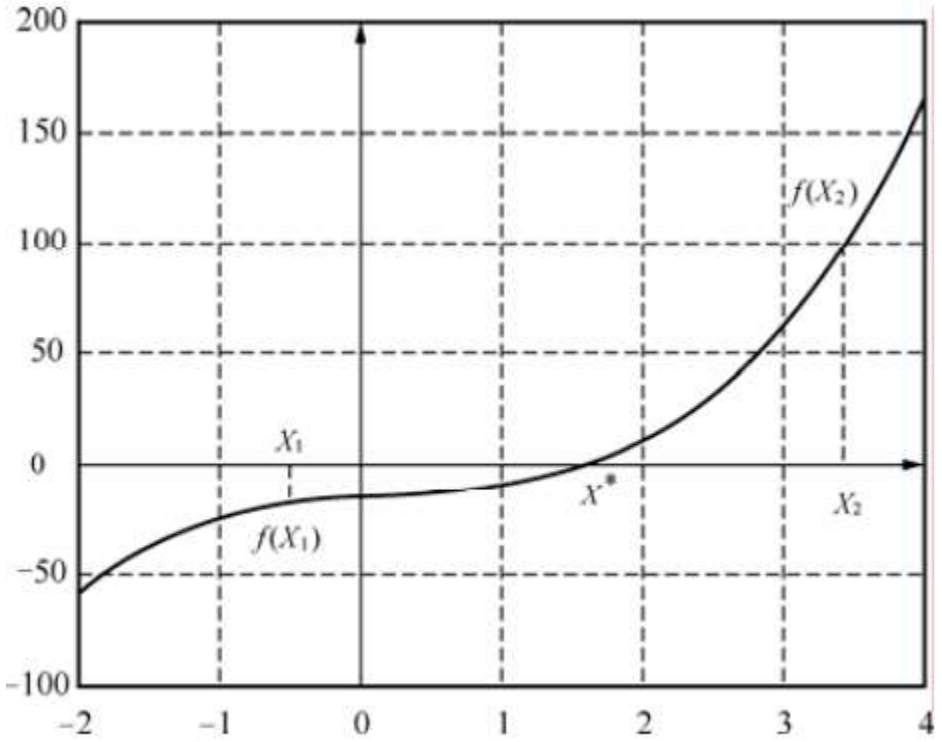


图 3-31 用二分法求 $3x^3-2x^2+5x-16=0$ 的根

具体算法步骤如下：

S1:
输入x1、x2 初值；
S2:
计算对应函数值 y1=f(x1)、y2=f(x2)；
S3:
若 y1,y2异号，转 S4；否则转 S1；
S4:
x0=(x1+x2)/2
S5:
计算x0对应函数值 y0=f(x0)；
S6:
若 y0,y1异号，x2=x0；否则 x1=x0；
S7:
若 $y0 \le 10^{-6}$转 S8，否则转 S4；
S8:
打印方程的根 x0。

```
#include  <stdio.h>
#include  <math.h>

void main( void )
{
   double x0, x1, x2, y0, y1, y2;
   do
   {
     printf("please input x1,x2 :");
     scanf("%lf,%lf", &x1, &x2);
     y1=3*x1*x1*x1-2*x1*x1 +5*x1-16;
     y2=3*x2*x2*x2-2*x2*x2 +5*x2-16;
   }while(y1*y2>0);
   do
   {
     x0=(x1+x2)/2 ;
     y0=3*x0*x0*x0-2*x0*x0 +5*x0-16;
     if( y0 *y1 <0 )
         x2 = x0;
     else
         x1=x0;
   } while ( fabs(y0) >= 1e-6 );
   printf("The root is %lf\n", x0 );
}
```

3.3.3 for 语句

for 语句是循环控制结构中使用最为灵活、最为广泛的一种循环控制语句。for 语句既适用于已知循环次数的情况，也可以用于只知循环结束条件不知道循环次数的情况。for语句可以完全取代前两种循环控制语句。

for语句的一般形式为：

```
for(初始化表达式; 条件表达式; 增量表达式)
    语句;
```

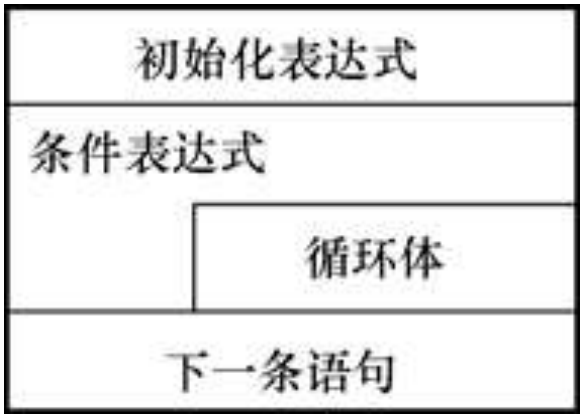
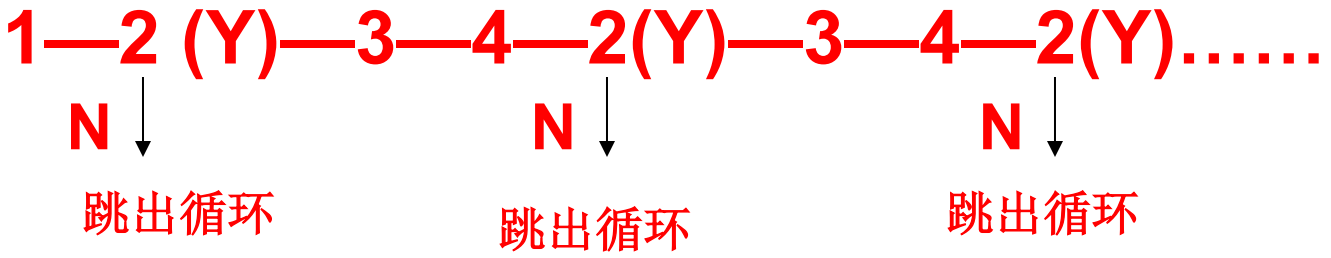


图3-33 for 语句的 N-S 流程图

for(初始化表达式1; 条件表达式2; 增量表达式4)
    语句3;

for语句其执行过程如下：
（1）首先计算初始化表达式1；
（2）然后计算条件表达式2，若条件表达式为真，则转到第（3）步执行；否则转到第（5）步执行；
（3）执行循环体3；
（4）计算增量表达式4，然后转到第（2）步，进行循环；
（5）退出循环，顺序执行 for循环后的下一条语句。



例题：for循环实现求 1+2+……+100的和！

```
#include <stdio.h>
void main( void )
{
 int i, sum=0;
 for(i=1;i<=100;i++)
    sum=sum+i;

printf("%d\n", sum);
}
```

```
i=1;
while(i<=100)
{
sum=sum+i;
i++;
}
```

- 用 for循环结构编程时，需要留意 for语句的三个表达式可以是任何类型：

```
for(i =1, gcd = 1; i <= num1; i ++ ) …
for(i =1;(ch=getchar())!='\n '; i ++ ) …
for(i =1, gcd = 1; i <= num1; gcd=i, i ++ ) …
```

- for语句的三个表达式还可以省略，不过分号“;”绝对不能省略。

```
gcd= num1<num2 ? num1: num2;
    for(; num1%gcd || num2%gcd; )
        gcd--;
```

- 省略 for语句的条件表达式要特别小心，因为 for语句将认为循环条件始终成立，循环将无休止地进行下去，构成“死循环” 。一般要在循环体内适当位置利用条件表达式加 break 语句在条件满足时，跳出循环。

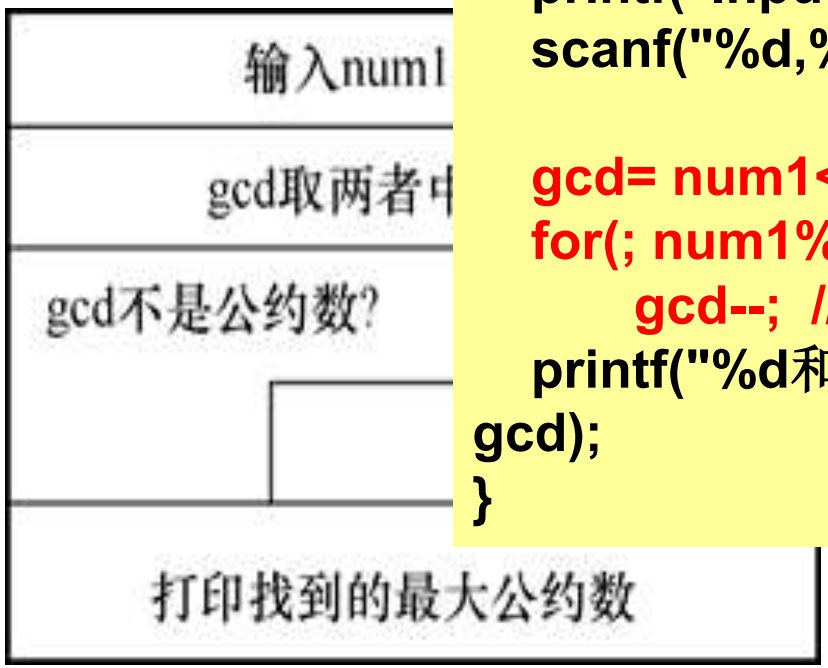
```
for(lcm= num2; ; lcm +=num2)
    if（lcm %num1==0）
        break;
```

- for语句还可以省略循环体。这时，for语句起延时的作用。

```
for(i=0; i<10000 ; i ++);
```

【例 3-16】 用 for循

自然数是指非 0 的正
自然数的公因子中的
然数中的小值着手，
两个 short型变量 nu
short 型变量 gcd 存




```
/* Program: EG03-16.c */
/* Description: 编程求输入的两个自然数的最大公约数 */

#include <stdio.h>

void main( void )
{
    short num1, num2, gcd;
    printf("Input num1,num2:");
    scanf("%d,%d",&num1,&num2);

    gcd= num1<num2 ? num1: num2;
    for(; num1%gcd || num2%gcd; )
        gcd--;  //for语句省略了初始化表达式和增量表达式
    printf("%d和%d的最大公约数是%d\n", num1, num2,
gcd);
}
```

```
#include <stdio.h>
void main( void )
{
    short num1, num2, gcd;
    printf("Input num1,num2:");
    scanf("%d,%d",&num1,&num2);
     for(gcd= num1<num2 ? num1: num2; num1%gcd!=0 || num2%gcd!=0; )
          gcd--;
printf("%d和%d的最大公约数是%d\n", num1, num2, gcd);
}
```

能否将循环体写到for语句里面，如果可以，怎么写？

```
for(gcd= num1<num2 ? num1: num2;num1%gcd!=0 || num2%gcd!=0; gcd--);
```

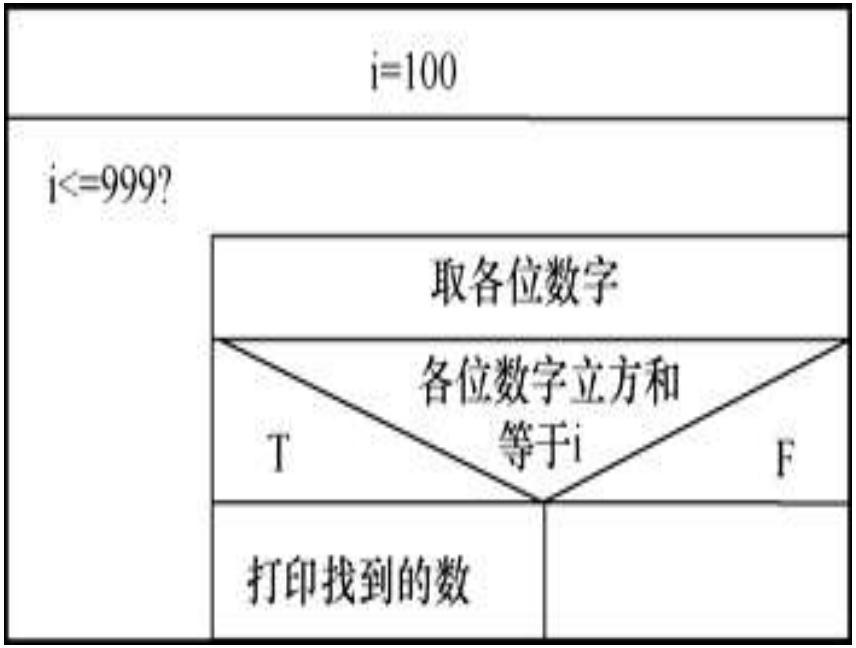
求两个自然数最大公约数常常采用“辗转相除法”。“辗转相除法”又叫做“欧几里得算法”，是欧几里得在他的著作《几何原本》第七卷中提出的。利用这个方法，可以较快地求出两个自然数的最大公约数。

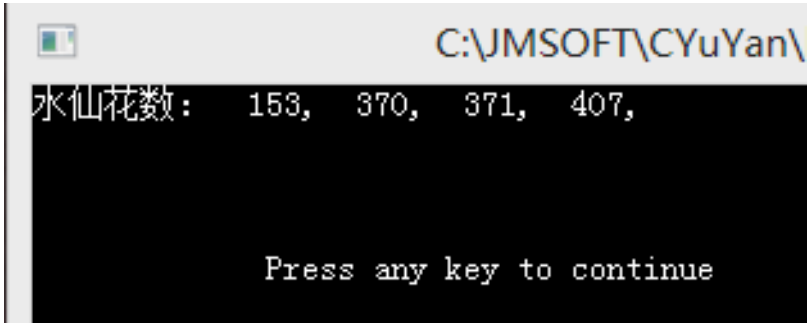
```
# include<stdio.h>
void main(void)
{
   int r,a,b;
   printf("请输入两个自然数:");
   scanf("%d,%d",&a,&b);
   r=a%b;
   while(r!=0)          //如何改成for循环？
{
       a=b;
       b=r;
       r=a%b;
   }
  printf("\n最大公约数是: %d",b);
}
```

```
for(r=a%b;r!=0; r=a%b)
  {
     a=b;
     b=r;
  }
```

a	b	r
36	28	8
28	8	4
8	4	0
…	…	…
…	…	…

【例 3-17】 用 for循环结构编程打印出所有的“水仙花数”。






```
#include <stdio.h>
void main( void )
{
    short a, b, c, i;
    printf("水仙花数： ");
    for(i=100; i<=999; i++)
    {
        a=i%10;       //个位
        b=i%100/10;   //十位
        c=i/100;      //百位
        if(i==a*a*a+b*b*b+c*c*c)
            printf("%5d,",i);
    }
   printf("\n");
}
```

C 语言中三种循环控制语句 while、do～while 和 for。三者基本上可以互相替代，但在实际应用中还是存在一些细微的差别。

（1）循环变量赋初值：使用 while、do～while 语句编程，必须保证在 while、do～while语句之前完成循环变量赋初值；for语句可以在初始化表达式中给循环变量赋初值。

（2）循环条件：while、do～while 语句均在 while 后面指定循环条件；for 语句在条件表达式中指定。

（3）循环体的执行：while 和 for语句都是先判断后执行，在条件不成立的情况下，循环体一次也不执行；do～while 语句是先执行后判断，循环体至少执行一次。

（4）循环变量递增：while、do～while 语句一般在循环体内增减循环变量，最终导致循环结束；for语句在增量表达式中改变循环变量的值。

（5）退出循环，顺序执行循环语句后的下一条语句。

在实际应用中，可根据具体情况选用不同的语句：

- （1）如果是计数型循环或循环次数在执行循环之前就已确定，一般选用 for 语句；如果循环次数不确定，或是由循环体的执行情况方能确定的，一般选用 while 语句或者do～while语句，此时的循环条件表达式往往比较复杂，使用 for语句会导致程序晦涩难懂。

- （2）当循环体至少要执行一次时，选用 do～while 语句；反之，如果循环体可能一次也不执行，选用 while 语句或者 for语句。

3.3.4 循环嵌套

当一个循环的循环体中出现了另一个循环结构，称其为循环嵌套。内嵌的循环还可以再次嵌套循环结构，这种嵌套过程被称为多重循环。一个循环仅内嵌一层循环，称为二重循环或双重循环；一个循环内嵌两层循环，称为三重循环……处于外面的循环称为外循环，处于内部的循环称为内循环。

```
while(…)
{
  while(…)
    …
}
```

```
for(…,…, …)
{
   while(…)
     …
}
```

```
for(…,…, …)
{
   for(…,…, …)
     …
}
```

三种循环结构 语句可以互相嵌套自由组合。但不能违反以下原则：

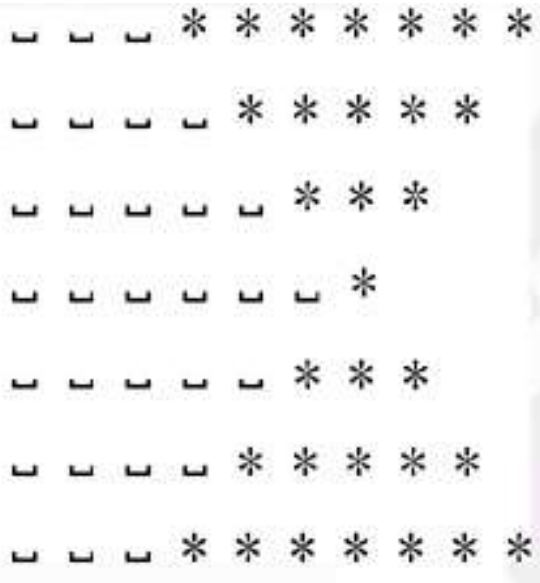
（1）内循环必须完整地嵌套在外循环内，相互之间不能出现交叉：

```
do
{
  …
   for(…, …, …)
   {
        …
}while( );
   }
```

（2）内外循环各自使用自己的循环变量，不能共用同一个循环变量。

（3）多重循环的执行顺序：外层循环每执行一次循环体，内层循环要完整地遍历一次，再开始下一轮外循环。

【例3-18】 用双重循环结构编程实现利用规律编写算法打印出星形图案。
注意：首行的最左边有3个空格。
打印星形图案是双重循环结构程序的典型应用。在求解这类题目的时候一定要注意到星形图案不仅由“*”组成，还包含了大量的空格。因此，在进一步分析题目之前，要先把题目中包含的空格标出来。



分析：

行号i	每行空格数	每行星号数
第0行	3个空格= 3+0= 3+i	7个*=7-2*0=7-2*i
第1行	4个空格= 3+1= 3+i	5个*=7-2*1=7-2*i
第2行	5个空格= 3+2= 3+i	3个*=7-2*2=7-2*i
第3行	6个空格= 3+3= 3+i	1个*=7-2*3=7-2*i
第4行	5个空格= 9-4= 9-i	3个*=2*4-5=2*i-5
第5行	4个空格= 9-5= 9-i	5个*=2*5-5=2*i-5
第6行	3个空格= 9-6= 9-i	7个*=2*6-5=2*i-5

由此归纳出：
当行号i < 4时，每行空格数为3+i个，每行星号数为7–2*i个；
当行号i>=4时，每行空格数为9–i个，每行星号数为2*i–5个；

由此归纳出：

当行号i < 4 时，每行空格数为3+i个，每行星号数为7-2*i个；

当行号i>=4 时，每行空格数为9-i个，每行星号数为2*i-5个；一共需要打印7行。

本题需要使用双重循环结构来打印星形图案。外循环控制行数，外循环体包含求每行空格数、求每行星号数、打印空格的内循环1、打印星号的内循环2、打印回车换行。内循环1负责按每行空格数规律打印当前行的空格；内循环2负责按每行星号数规律打印当前行的星号。这里需要定义两个char型变量star、space代表星号和空格，两个 short 型变量 i、j，i 代表控制行数的外循环变量， j代表控制空格、 星号的内循环变量，两个short型变量num1、num2 分别表示每行空格数规律、每行星号数规律。

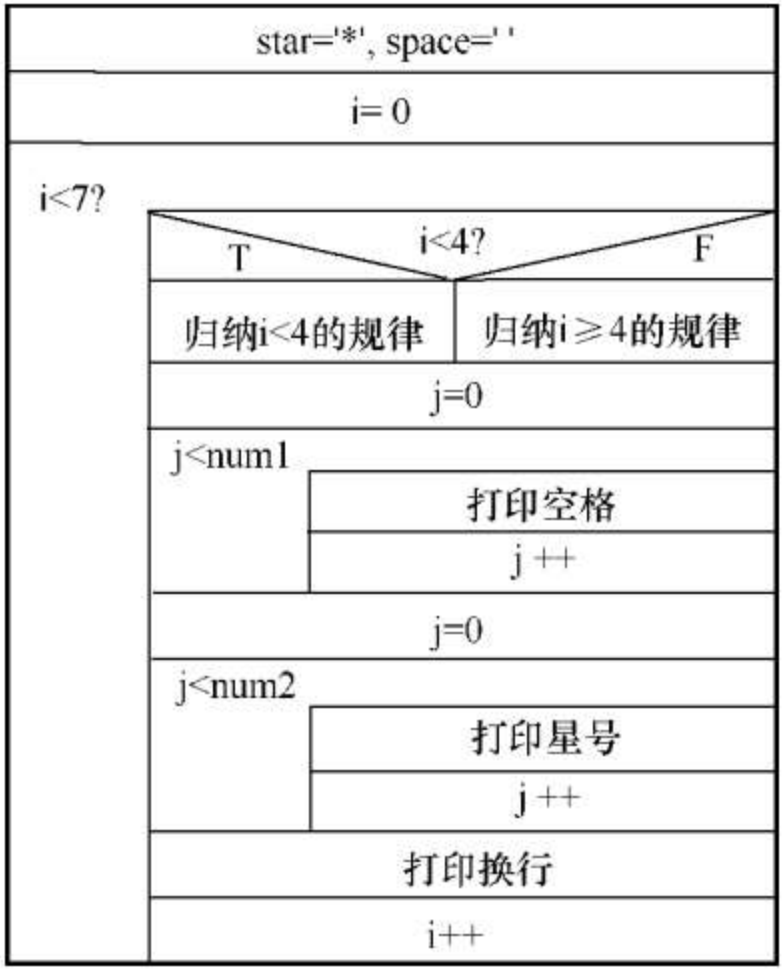


图 3-40 ［例 3-18］的 N-S 图

```
/* Description: 编程打印指定的星形图案 */
#include <stdio.h>
void main( void )
{
   char star='*', space=' ';                    //定义打印符号
   int i, j, num1, num2;                        //定义各控制变量
   for(i=0;i<7;i++)                             //行控制
   {
      if(i<4)  num1=3+i,  num2=7-i*2;           // i<4的规律
      else   num1=9-i,   num2=i*2-5;            // i>=4的规律
      for(j=0; j< num1; j++)
          putchar(space);                       //输出空格
      for(j=0; j< num2; j++)
          putchar(star);                        //输出星号
      printf("\n");                             //换行
   }
}
```

【例 3-20】 百元买百鸡：用一百元钱买一百只鸡。已知公鸡 5 元/只,母鸡 3 元/只,小鸡 1元/3 只。请设计程序求买公鸡、母鸡、小鸡各多少只？

先假定买了公鸡cock只，母鸡hen只，小鸡chicken只，根据已知条件可以得到两个方程：

cock + hen + chicken = 100

5 * cock + 3 * hen +chicken / 3 = 100

穷举法来求解。穷举法，又称枚举法，它的基本思想是把所有可能的解答情况一一测试，从而筛选出符合条件的解。由于要测试所有可能的状态，穷举法的效率比较低。可以根据具体问题对算法进行优化设计。

一百元买一百只鸡。很显然，cock、hen、chicken的取值范围都是 0～100。借助穷举法的思想，使用三重循环，让 cock、hen、chicken 分别从 0 循环到 100，测试每一种组合是否满足上述方程。

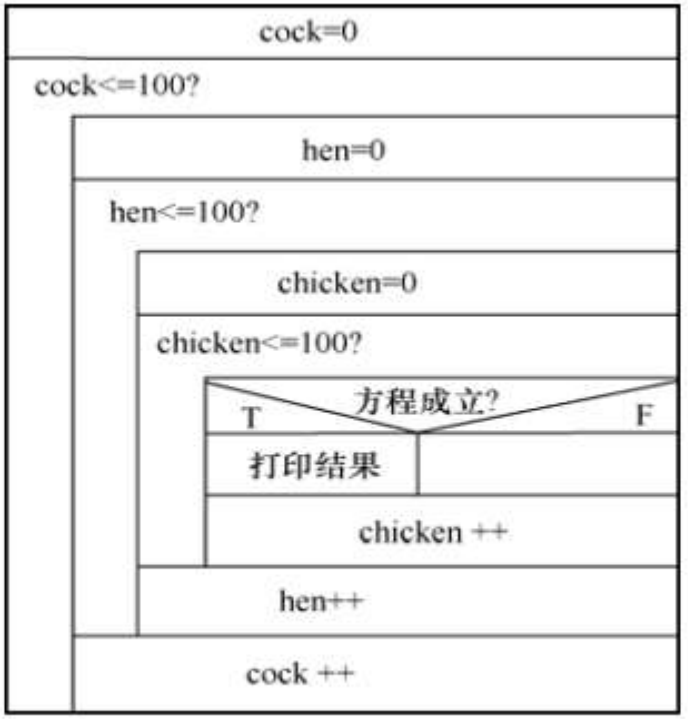


图 3-44 ［例 3-20］的 N-S 图

将图 3-44 所示 N-S流程图转换成的C 语言源程序如下：

```
/* Program: EG03-20.c */
/* Description: 百元买百鸡。*/
#include <stdio.h>

void main( void )
{
   short cock,hen,chicken;
   for (cock=0; cock<=100; cock++)
     for (hen=0; hen<=100; hen++)
       for (chicken=0; chicken<=100; chicken++)
       {
         if(cock+hen+chicken==100&&5*cock+ 3*hen + chicken/3.0 ==
100 )
         printf("cocks=%d, hens=%d,chickens=%d\n", cock,  hen,
chicken);
       }
}
```

【想一想】

上述穷举法算法共测试不定方程 101×101×101=1030301 次≈103 万次，算法效率实在太低，算法能否优化、改进呢？

我们注意到公鸡5元/只， 母鸡3元/只， 也就是说cock最多100/5=20只， hen最多100/3=33只，而 cock、hen 确定后，小鸡肯定是 100-cock-hen 只。 修改上述穷举法算法，我们得到以下程序：

```
for (cock=0; cock<=20; cock++)
  for (hen=0; hen<=33; hen++)
   {
      chicken= 100 – cock – hen ;
      if( 5*cock+ 3*hen + chicken/3.0 == 100 )
         printf("cocks=%d, hens=%d,chickens=%d\n", cock,  hen, chicken);
   }
```

优化后的算法执行效率大大提高。

诸如鸡兔同笼问题、排列组合问题、选手分组比赛名单问题、客人握手问题等都可以通过穷举法算法得到解决。

3.3.5 转移控制

C 语言提供了 4 种用于控制流程转移的语句：break、continue、goto 和 return。标准库函数 exit()也可以用于控制程序的流程。

1. break语句

在前面学习 switch 语句时，我们已经接触了 break 语句，通过 break 语句可以使程序在执行完该 case 分支后立即跳出 switch 结构。break 语句还可以用在循环语句中，作用是在循环体的执行过程中，在遇到应立即结束循环的条件后，程序流程立即终止整个循环的执行，跳出循环结构，转去执行循环语句后的语句。

【例 3-21】 用 C 语言编写一个程序，将输入的一个正整数分解成质因子的连乘积。例如：输入 88，打印出 88=2*2*2*11。

我们需要定义一个 int 型变量 n 接收输入的正整数，让 int 型计数器 i 从最小的质数 2 开始进行质因子分解：

（1）如果找到的质数 i 恰好等于 n，则说明分解质因数的过程已经结束，打印即可。

（2）如果 n!=i，但 n 能被 i 整除，则应打印出 i 的值，并用 n 除以 i 的商作为新的正整数 n，重复试探 i是否是 n 的质因子。

（3）如果n!=i，但 n 不能被 i 整除，则结束内循环， i 增1后继续执行第一步。

/* Program: EG03-21.c 编程将输入的正整数分解成质因子的连乘积 */

```
#include <stdio.h>
void main( void )
{
   int n, i;
   printf("Please input a number:\n");
   scanf("%d",&n);
   printf("%d=",n);
   for(i=2; i<=n; i++)
     while(n!=i)
     {
       if(n%i==0)
       {
          printf("%d*",i);
          n=n/i;
       }
       else
       break;
     }
   printf("%d",n);
}
```

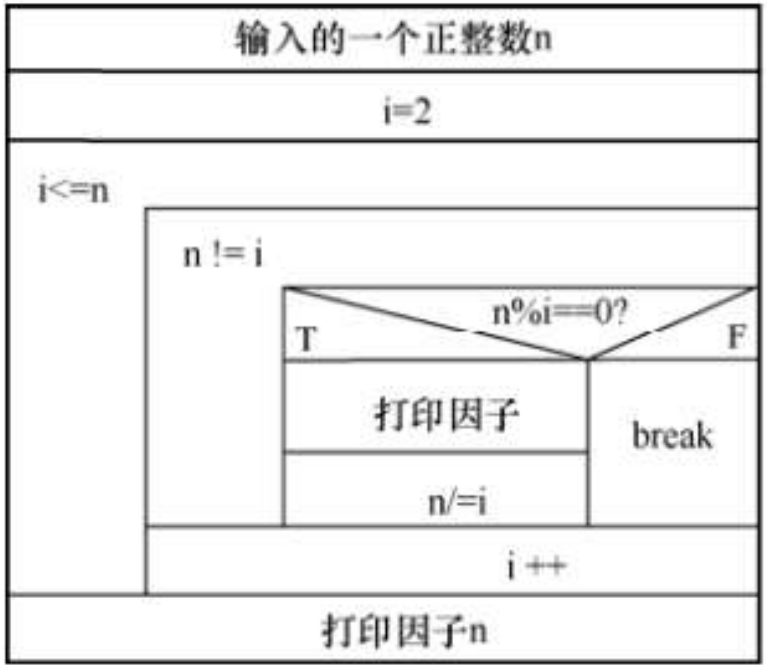


图 3-46 [例 3-21] 的 N-S 图

```
E:\LXH C PROGRAMMING\EG03-21\D
Please input a number:
88
88=2*2*2*11
Press any key to continue
```

2. continue语句

continue 语句只能在循环体中使用：在循环体的执行过程中，在满足某种条件下，跳过循环体剩下的语句，提前结束本次循环周期并开始下一轮循环。

goto 语句：它的使用会破坏程序的逻辑结构，因此，在结构化程序设计中不提倡使用 **goto**语句。

return 语句将函数的返回值返回给函数的调用者。在执行函数的过程中，无论 **return** 语句在函数的什么位置，一旦执行到它，就立即返回到函数的调用者，不再继续执行函数剩下的语句。 类型为**void**的函数没有返回值， 所以在函数体内可以有**return**语句， 也可以没有**return**语句。其他类型的函数有返回值，因此在函数体内必须有 **return** 语句。
使用标准库函数 **exit()**可以强制终止整个程序的执行，直接返回操作系统。调用**exit()**函数需要嵌入头文件：**stdlib.h**。

函数 **exit()**的一般调用形式为：
exit(数字);

其中，数字 **0** 表示程序正常退出，数字非 **0** 表示程序出现某种错误后退出。

习题3-1-15. 执行下面程序后，j的值是?

```
int i, j;
for(i=1, j=1; j<=50; j++)
{
   if(i>10)
      break;
   if(i%2)
   {
      i+=5;    continue;
   }
   i-=3;
}
```

A）6 B）2 C）8 D）4

C

验证

```
#include <stdio.h>
void main( void )
{
int i, j;
for(i=1, j=1; j<=50; j++)
{
   if(i>10)
      break;
   if(i%2)
   {
      i+=5;    continue;
   }
   i-=3;
}
printf("i=%d,j=%d",i,j);
}
```