第5章 循环程序设计

循环结构是一种重复执行的程序结构。它判断给定的条件，如果条件成立，则重复执行某一些语句（称为循环体），否则结束循环。通常，循环结构有“当型循环”（先判断条件，后执行循环）和“直到型循环”（先执行循环，再判断条件）。在C语言中，实现循环结构的语句主要有以下3种：

（1）for语句。

（2）while语句。

（3）do-while语句。

5.1 问题的提出

实际应用中的许多问题，都会涉及重复执行的操作步骤和相应的算法，如级数求和、方程的迭代求解、统计报表打印，等等。有时重复处理的次数是已知的，有时重复处理的次数是未知的。不管怎样，程序设计中都要用到循环结构。比如以下问题的求解过程都要用到循环结构。

（1）计算1+2+3+…+n，这是一个循环累加的问题，每次循环累加一个自然数，总共需要n次加法运算，从而得到这个自然数数列之和。

（2）计算n！=1×2×3×…×n，这是一个循环累乘的问题，每次循环乘一个自然数，总共需要n次循环，即累乘n个数，从而得到n的阶乘。

（3）利用公式：$\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots$，计算π的近似值，直到最后一项的绝对值小于$10^{-6}$时认为满足精度要求。这是一个事先未知循环次数的问题，需要根据给定的条件来判断循环是否终止。

类似以上需要重复处理的问题，必须用循环语句来编写程序解决。其实循环语句不只在数学问题中发挥重要作用，联合使用选择语句和循环语句还可以设计出许多实用的程序。

循环结构和顺序结构、选择结构一起被称为结构化程序设计的3种基本结构。按照结构化程序设计的观点，任何可计算问题都可以用这3种基本结构来解决。

5.2 while 语句

while 语句用来实现“当型”循环结构。

while 语句的一般形式为：

```
while(表达式) 循环体语句;
```

while 语句的执行过程可以用如图 5-1 的传统流程图(a)与 N-S 流程图(b)来表示。

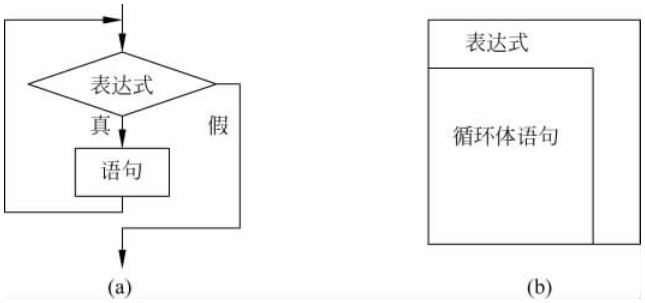

图 5-1 while 循环的流程图

其执行过程为：先判断表达式，为真(非 0)则执行语句，然后再判断。如果表达式为假(值为 0)则跳过循环体而直接执行 while 语句的下一语句。因此循环体可能一次也没有执行。当初始条件为假时，是不会执行循环的。

注意：

(1) 如果循环体包含两条及以上的语句，应用{}括起来，构成一个复合语句，否则系统只把第一个语句当成循环体部分加以重复执行，余者作为 while 循环的后续语句。

(2) 循环体中应包含改变循环条件的语句，否则可能导致死循环。

【例 5.1】 求 1＋3＋…＋99 的值。

首先画出流程图如图 5-2 所示。

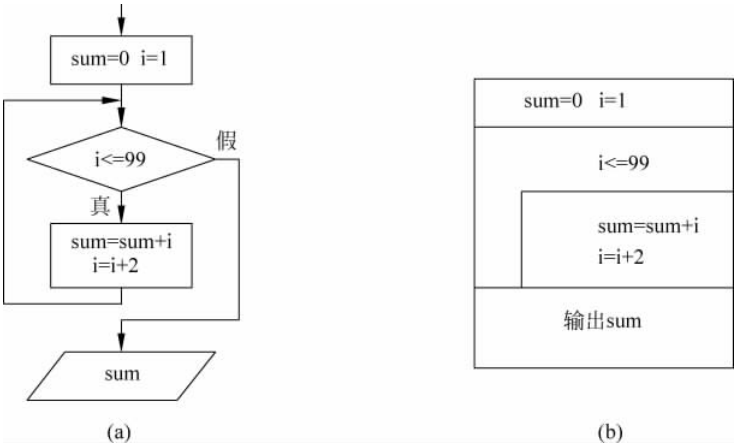

图 5-2 1＋3…＋99 的 while 循环流程图

```
#include <stdio.h>
void main( )
{   int i,sum;
    i = 1; sum = 0;
    while(i <= 99)
    {   sum = sum + i;
        i = i + 2;
        }
    printf("sum = %d\n",sum);
}
```

当然，本题也可用其他算法完成，如转换成 $\sum_{i=1}^{50} 2i-1$ 的形式。

【例 5.2】 从键盘读入一系列字符，以#结束，统计字符的个数。

程序分析：

这是典型的标志法。以#作为标志，当此标志出现就结束循环。由于有可能第一个字符就是#，因此适合用当型循环完成。此外，还需要一个计数器，用来统计实际字符个数。

程序流程图如图 5-3 所示。

```
#include <stdio.h>
void main( )
{   int count;
    char ch;
    count = 0;
    scanf("%c",&ch);
    while(ch!= '#')
    {     count++;
          scanf("%c",&ch);
    }
    printf("total = %d\n",count);
}
```

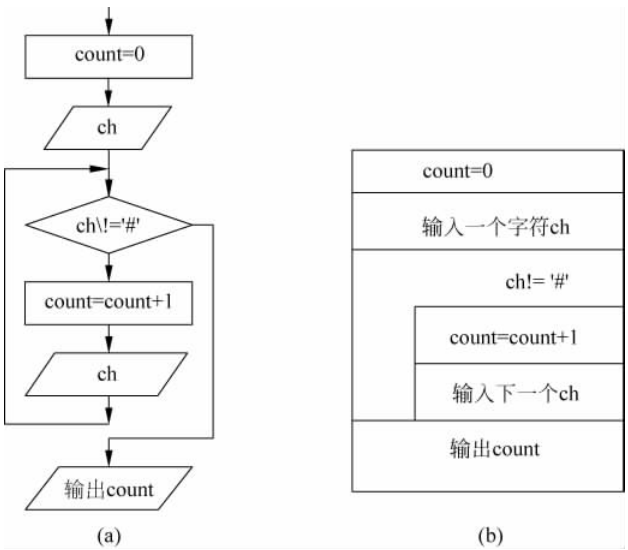

图 5-3　例 5.2 的两种流程图表示

5.3 do-while 语句

do-while 语句是另一种用来实现“当型”循环的结构。与 while 循环不一样，它是先执行，后判断。

它的一般形式为：

```
do
循环体语句
while (表达式);
```

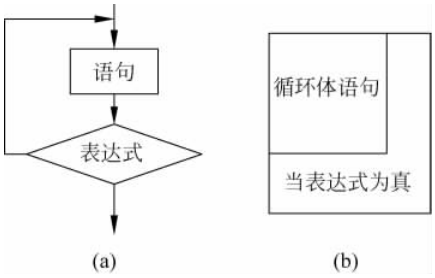

图 5-4 do-while 循环的流程图

do-while 语句的执行过程可以用 5-4 的流程图表示。

其执行过程为：先执行循环体语句，然后判断表达式，如果表达式值为真，则重复执行循环体，如表达式值为假，则结束循环。因此 do-while 的循环体语句至少会执行一次。

【例 5.3】 用 do-while 循环求 1＋3＋…＋99 的值。

其流程图与例 5.1 类似，只是条件判断放在循环体之后，如图 5-5 所示。

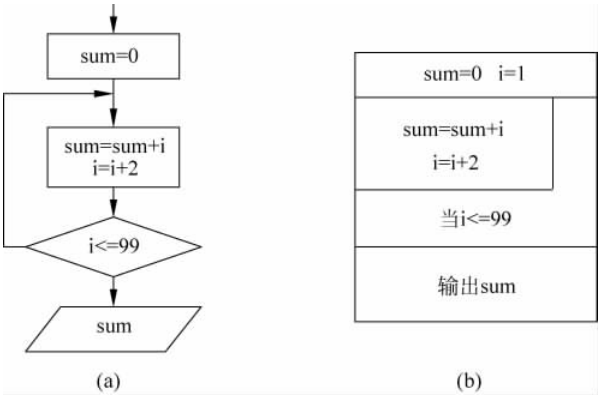

图 5-5 用 do-while 流程图表示 1＋3＋…＋99

```
#include <stdio.h>
void main( )
{  int sum,i = 1;
       sum = 0;
       do
           sum = sum + i;
           i = i + 2;
       while (i <= 99);
printf("sum = %d\n",sum);
     }
```

说明：当 do-while 之间的循环体由多个语句构成时，可以用{}括起来，也可不用，系统

会自动把它们都当作循环体语句处理。

【例 5.4】 从键盘输入某班级学生的英语考试成绩，编程计算总分和平均分。

程序分析：

(1)由于学生人数未知，也只有采用标志法。由此设定：在最后一个学生成绩的后面添加一个标志－1，因为学生成绩一般情况下是≥0 的。

(2) 由于班级中学生人数＞0，故可使用 do-while 循环。

程序流程图如图 5-6 所示。

```
#include <stdio.h>
void main( )
{  int count,score;
    float total,aver;
    total = 0;count = 0;
    printf("input scores:\n");
    scanf("%d",&score);
    do
       total = total + score;
       count++;
          scanf("%d",&score);
    while(score!= -1);
    aver = total/count;
    printf("count = %d total = %7.2f aver = %5.2f\n",count,total,aver);
}
```

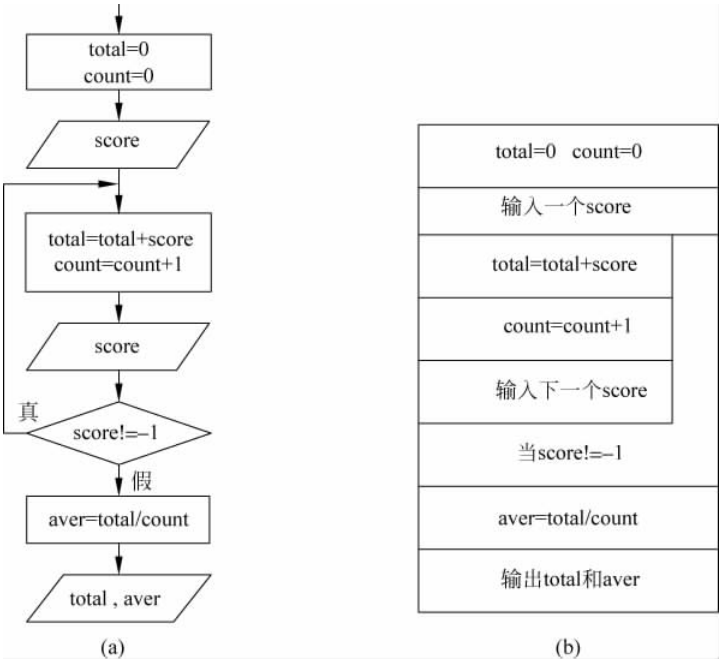

图 5-6　例 5.4 的流程图

5.4 for 语句

for 循环是 C 语言中使用最频繁也是最灵活的一种语句，它主要适用于循环次数已知的情况，但也可用于循环次数未知的情况。

for 语句的一般形式为：

```
for(表达式 1;表达式 2;表达式 3)
循环体语句;
```

它的执行过程可以用如图 5-7 所示的流程图表示。

for 循环的执行过程为：

(1) 求表达式 1 的值；

(2) 判断表达式 2；

(3) 若值为真，则执行循环体语句，并执行表达式 3，重复步骤(2)；

(4) 若表达式 2 的值为假，则结束循环，执行 for 语句的后续语句。

for 循环的执行可以理解为如下的形式：

```
for(循环变量赋初值;循环条件;循环变量自增值)
循环体语句;
```

【例 5.5】 用 for 循环求 1＋3＋…＋99 的值。

```
#include <stdio.h>
void main( )
{   int i,sum = 0;
   for(i = 1;i <= 99;i = i + 2)
         sum = sum + i;
   printf("sum = %d\n",sum);
}
```

【例 5.6】 从键盘输入 10 个数，找出其中的最小值。流程图如图 5-8 所示。

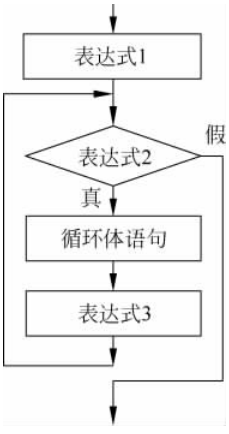

图 5-7 for 循环的流程图表示

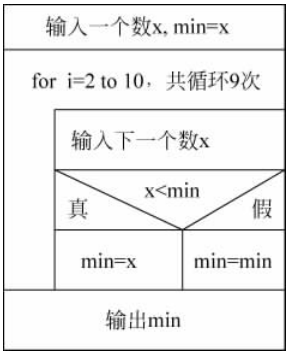

图 5-8 例 5.6 的 N-S 流程图

```
#include <stdio.h>
void main( )
{ int i,x,min;
    printf("input 10 datas:\n");
    scanf("%d",&x);
    min = x;
    for(i = 2;i <= 10;i++)
     {  scanf("%d",&x);
        if(x < min) min = x;
     }
    pritnf("min = %d\n",min);
}
```

关于 for 循环使用的具体过程中，有几点值得说明的地方：

(1) 表达式 1、表达式 3 可以是简单的表达式或逗号表达式，表达式 2 一般是关系表达式或逻辑表达式(也可以是其他表达式，值为 0 表示假，非 0 表示真)。

上述例 5.5 可以表示为如下形式：

```
for(sum = 0,i = 1;i <= 99;i = i + 2 )
      sum = sum + i;
```

或

```
for(sum = 0,i = 1;i <= 99;i++,i++)
      sum = sum + i;
```

例如，统计键盘输入字符个数可以用以下 for 语句表示：

```
for(i = 0;(ch = getchar())!= '\n';i++);
```

(2) 表达式可以省略，但分号不能省。

① 表达式 1 可以省略，但应在 for 语句之前给循环变量赋初值，如上例可以改为：

```
sum = 0;i = 1;
for(;i <= 99;i = i + 2)
     sum = sum + i;
```

② 表达式 2 可以省略，执行时就没有循环条件可判断，循环无休止地执行下去，如：

```
for(i = 1;;i = i + 2)
    sum = sum + i;
```

就构成死循环。但可以采取其他办法避免出现这种情况，如在循环体中使用 goto 语句或 break 语句等。

③ 表达式 3 可以省略，但应在循环体中添加循环变量自增的语句，否则也会造成死循环。如上例也可表示为：

```
  for(i = 1;i <= 99;)
{   sum = sum + i;
    i = i + 2;
    }
```

④ 可以同时省略表达式 1 和表达式 3，只有表达式 2，例：

```
i = 1;
for(;i <= 99;)
{ sum = sum + i;
i = i + 2;
}
```

此时相当于 while 语句。实际上，for 循环完全可以替代 while 循环，但其用法远比 while 灵活。

⑤ 3 个表达式均可同时省略，此时情况与②类似，也要采取其他办法来控制循环结束。

5.5 goto、break、continue 语句

1. goto 语句

有时需要从程序中的某个语句转移到另一个语句，这时可以使用 goto 语句。goto 语句是无条件转移语句，它的一般形式为：

```
goto    语句标号;
```

其中语句标号为标识符，它的命名规则与变量名一样，只能由字母、数字、下画线组成，且只能由字母或下画线开头。例如：

```
goto loop;
```

表示将流程无条件地转移到 loop 所标识的语句去继续执行。

goto 语句一般与 if 语句配套使用，用来构成循环，或者从循环体内跳转到循环体外。

【例 5.7】 用 goto 语句求 1＋3＋…＋99。

```
#include <stdio.h>
void main( )
{    int i,sum = 0;
     i = 1;
next:
     sum = sum + i;
     i = i + 2;
     if (i <= 99) goto next;
     printf"sum = %d\n",sum);
}
```

结构化程序不提倡使用 goto 语句，因为频繁地使用 goto 语句使得程序结构毫无规律可言，如同一团乱麻。

2. break 语句

break 语句不仅能跳出 switch 语句，而且能跳转出任何一种循环语句的循环体，进而执行循环语句的下一个语句。

break 语句的一般形式为：

```
break;
```

【例 5.8】 用 break 语句完成例 5.7。

```
#include <stdio.h>
void main( )
{   int i,sum;
    for(sum = 0,i = 1;;i = i + 2)
      {  sum = sum + i;
         if(i >= 99) break;
      }
    printf("sum = %d\n",sum);
}
```

注意：

(1) break 语句一般与 if 语句配套使用，用来控制是否继续循环。

(2) break 语句只能用于 switch 语句和循环语句中，不能用于任何其他语句。

3. continue 语句

continue 语句用于结束本次循环，即跳过循环体中尚未执行的语句，流程转移到判断循环条件处，准备下一次循环。

continue 语句的一般形式为：

```
continue;
```

【例 5.9】 从键盘输入 10 个整数，打印所有的负数。

```
#include <stdio.h>
void main( )
{   int x,i;
    printf("input 10 datas:\n");
    for(i = 1;i <= 10;i++)
      {  scanf("%d",&x);
         if(x >= 0) continue;                    /* 非负数就跳过 */
         printf("%8d",x);
      }
}
```

下面举例来形象地区分二者。假设有以下两种结构的循环：

```
(1) while(表达式 1)
    {  ⋮
       if(表达式 2) break;
       ⋮
    }
(2) while(表达式 1)
    {  ⋮
       if(表达式 2) continue;
       ⋮
    }
```

图 5-9 是它们的流程图，请注意 break 与 continue 的区别。

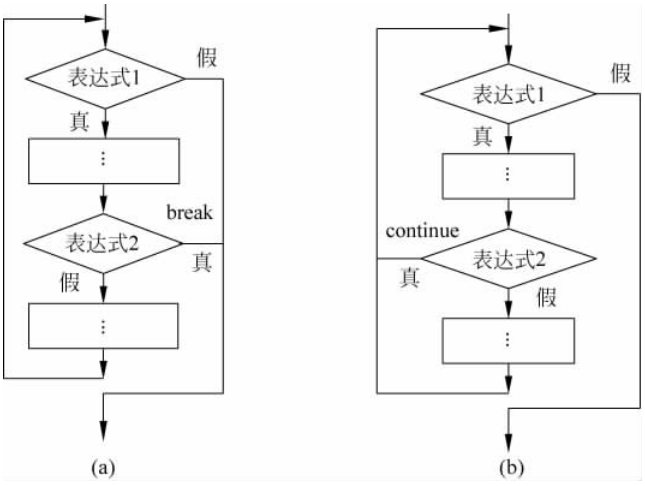

图 5-9　break 与 continue 的区别

5.6 循环的嵌套

如果在一个循环内完整地包含另一个循环结构，则称为多重循环或循环嵌套。嵌套的层数可以根据需要而定，嵌套一层称为二重循环，嵌套二层称为三重循环，以此类推。

3 种循环语句（while 循环、do-while 循环和 for 循环）可以相互嵌套，下面是几种常见的二重嵌套形式。

① for(…)
{…
for(…)
{
…
}
}

② for(…)
{…
while(…)
{
…
}
}

③ while(…)
{ …
for(…)
{
…
}
}

④ while(…)
{ …
while(…)
{
…
}
}

⑤ do
{…
for(…)
{
…
}
}while(…);

⑥ do
{ …
do
{
…
}while(…);
…
}while(…);

【例 5.10】　输出由数字组成的如下所示的金字塔图案。

```
1
222
33333
4444444
555555555
66666666666
7777777777777
888888888888888
99999999999999999
```

分析：输出图案一般可由多重循环实现，外循环来控制输出的行数，内循环控制每行的空格数和字符个数。程序如下：

```
#include<stdio.h>
void main( )
{
    int i, k, j;
    for (i = 1; i<= 9; i++)                    //外循环控制输出行数
    {
        for (k = 1; k<= 10 - i; k++)           //每行起始输出位置
        {
            printf(" ");                       //输出空格符
        }
        for (j = 1;j<= 2 * i - 1; j++)         //内循环控制输出字符个数
        {
            printf("%c",'0' + i);              //输出内容
        }
        printf("\n");                          //换行
     }
}
```

循环可以嵌套使用，即循环体内还可以包含另一个完整的循环。循环的嵌套可以是二重的，也可以是多重的。

循环的嵌套形式是多种多样的，前面介绍的几种循环语句，都可以互相嵌套。例如在 while 的循环体中包含一个 for 循环，或是 for 循环中包含一个 do-while 循环等。

【例 5.11】　求 1!+2!+…+10!。

```
#include<stdio.h>
void main( )
{ int i,j;
  long mul,sum = 0;
  for(i = 1;i<= 10;i++)
   { mul = 1;
   for(j = 1;j<= i;j++)
     mul = mul * j;
```

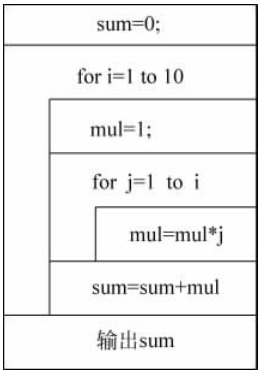

图 5-10　求 1!+2!+…+10!

```
    sum = sum + mul;
  }
   printf("sum = %ld",sum);
}
```

使用循环嵌套时，要注意几个问题：

（1）外层循环必须完全嵌套内层循环，严禁交叉嵌套；

（2）内、外层循环变量尽量不要同名，否则结果不可预料。

例如：

```
for(i = 1;i <= 10;i++)
for(i = 1;i <= 10;i++)
     printf("*");
```

一共打印了多少个*？请大家思考。

5.7 3种循环语句比较

一般情况下，3种循环语句可以相互代替，表5-1列出了各种循环语句的区别。

表5-1 几种循环语句的区别

格　　式	for(表达式1；表达式2；表达式3){循环体；}	while(表达式){循环体；}	do{循环体；}while(表达式)；
循环类别	当型循环	当型循环	直到型循环
循环变量初值	一般在表达式1中	在while之前	在do之前
循环控制条件	表达式2的值	表达式的值	表达式的值
提前结束循环	break	break	break
改变循环条件	一般在表达式3	循环体中用专门语句	循环体中用专门语句

说明：

（1）3种循环中for语句功能最强大，使用最多，任何情况的循环都可使用for语句实现。

（2）当循环体至少执行一次时，使用do-while语句与while语句等价。如果循环体可能一次也不执行，则只能使用while语句或for语句。

5.8 程序举例

许多程序都要用到循环结构，关于循环的算法很多，这里只列举一些常用的算法。

【例5.12】 输入两个正整数m和n，求它们的最大公约数。

程序分析：

求最大公约数可以用“辗转相除法”：将大数m作为被除数，小数n作为除数，二者余数为r。如果r≠0，则将n→m，r→n，重复上述除法，直到r＝0为止。此时最大公约数就是n，流程图如图5-11所示。

```
#include <stdio.h>
void main( )
{   int m,n,r,t;
    printf("input m and n:\n");
    scanf("%d%d",&m,&n);
    if(m<n)
    { t=m;m=n;n=t;}
    r=m%n;
    while (r!=0)
    { m=n;
      n=r;
      r=m%n;
    }
  printf("%d",n);
}
```

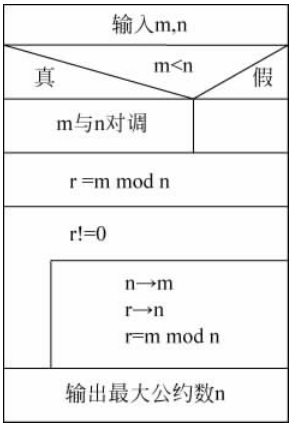

图 5-11　求最大公约数

【例 5.13】 打印 Fibonacci 数列的前 20 项，每行打印 5 个数。该数列前两个数是“1、1”，以后的每个数都是其前两个数之和。

程序分析：

这要用到递推法。所谓递推，是指根据前面的一个或多个结果推导出下一个结果。这里设 3 个变量 f1、f2、f3，其中 f3＝f1＋f2。流程图如图 5-12 所示。

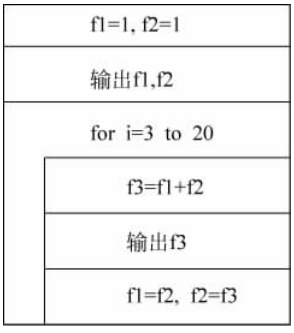

图 5-12　输出 20 项 Fibonacci 数列

```
#include <stdio.h>
void main( )
{   int f1,f2,f3,i;
    f1=1;f2=1;
    printf("%10d%10d",f1,f2);
    for(i=3;i<=20;i++)  /* 求后面18个数 */
    {  f3=f1+f2;
       printf("%10d",f3);
       if(i%5==0) printf("\n");
    f1=f2; f2=f3;
    }
}
```

【例 5.14】 打印出所有的“水仙花”数。所谓“水仙花”数，就是一个 3 位数，其各位数字的立方和等于该数本身。例如 407 就是一个“水仙花”数，因为 $407=4^3+0^3+7^3$。

程序分析：

本题可以采用穷举法。穷举法就是把所有的可能组合一一考虑到，对每种组合都判断是否符合要求，符合则输出。

```
#include <stdio.h>
void main( )
{  int i,j,k,m,n;
   for(i=1;i<=9;i++)
      for(j=0;j<=9;j++)
         for(k=0;k<=9;k++)
             {  m=i*100+j*10+k; n=i*i*i+j*j*j+k*k*k;
                if(m==n) printf("%10d",m);
```

```
            }
}
```

【例 5.15】 从键盘输入一行字符，要求将所有大写字母转换成小写，小写字母转换成大写，然后输出该字符串。

```
#include <stdio.h>
void main( )
{ char ch;
   while((ch=getchar())!='\n')
   {  if(ch>='a'&&ch<='z') ch=ch-32;          /* 小写变大写 */
      else if(ch>='A'&&ch<='Z') ch=ch+32;     /* 大写变小写 */
      putchar(ch);
   }
}
```

如果上例去掉 else，即改成

```
if(ch>='a'&&ch<='z') ch=ch-32;
    if(ch>='A'&&ch<='Z') ch=ch+32;
```

后，会得到什么结果？

【例 5.16】 输入一正整数 n，在屏幕中央打印 n 行三角形。如 n=4，则打印：

```
   *
  ***
 *****
*******
```

程序分析：

要想将图形打印在屏幕中央，应在每行输出第一个 * 之前打印若干个空格作为占位符。这里假设图形输出在 10 列处。

```
#include <stdio.h>
void main( )
{  int i,j,n;
   printf("input n:\n");
   scanf("%d",&n);
   for(i=1;i<=n;i++)
    {  for(j=1;j<=10;j++)
          printf(" ");                          /* 打印 10 个空格,占位 */
       for(j=1;j<=2*i-1;j++)
           printf("*");                         /* 打印 *号 */
           printf("\n");                        /* 打印完一行,换行 */
    }
}
```

【例 5.17】 编一个程序验证哥德巴赫猜想：一个大于等于 6 的偶数可表示为两个素数的和。例如：6=3+3，8=3+5，10=3+7。

程序分析：

设 n 为大于等于 6 的任一偶数，将其分解为 n1 和 n2 两个数，使得 n1＋n2＝n，分别判断 n1 和 n2 是否为素数，若都是素数，则为一组解。若 n1 不是素数，就不必再检查 n2 是否为素数。从 n1＝3 开始判断，直到 n1＝n/2 为止。事实上，在判断一个数 m 是否是素数时，可以减少循环次数，因为 m＝sqrt(m) * sqrt(m)，所以，当 m 能被大于 sqrt(m)的整数整除时，在 2～ sqrt(m)之间，至少存在一个能整除 m 的数，因此只要判断 m 能否被 2，3，…，sqrt(m)整除即可。

程序代码如下：

```
#include<stdio.h>
#include<math.h>
void main( )
{
    int n, n1, n2, j, k;
    printf("Enter a number n = ?\n");
    scanf("%d", &n);
    for (n1 = 3; n1 <= n / 2; n1++)
    {
        k = sqrt(n1);                                    //求 n1 的平方根
        for (j = 2; j <= k; j++)
        {
            if (n1 % j == 0)
            {
                break;
            }
        }
        if (j < k)
        {
            continue;
        }
        n2 = n - n1;
        k = sqrt(n2);
        for (j = 2; j <= k; j++)
        {
            if (n2 % j == 0)
            {
                break;
            }
        }
        if (j > k)
        {
            printf("%d = %d + %d\n",n,n1,n2);
            }
        }
}
```

输入及程序运行过程：

```
Enter a number n = ?
64 ↵
64 = 3 + 61
64 = 5 + 59
```

```
64 = 11 + 53
64 = 17 + 47
64 = 23 + 41
```

习题 5

1. 输入一行字符，分别统计出其中字母、数字和其他字符的个数。

2. 编程求 $1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\cdots+\frac{1}{99}-\frac{1}{100}$。

3. 编写程序，对数据进行加密。从键盘输入一个数，对每一位数字均加 2，若加 2 后大于 9，则取其除 10 的余数。如，2863 加密后得到 4085。

4. 输出 3～100 之间的所有素数。

5. 验证 2000 以内的哥德巴赫猜想：对于任何一个大于 4 的偶数，均可以分解为两个素数之和。

6. 求 $\frac{1}{1\times2}+\frac{1}{2\times3}+\cdots+\frac{1}{n\times(n+1)}$，直到某一项小于 0.001 时为止。

7. 100 匹马驮 100 担货，大马一匹驮 3 担，中马一匹驮 2 担，小马两匹驮 1 担，求大马、中马、小马的数目，要求列出所有的可能。

8. 假设我国国民经济总值按每年 8%的比率增长，问几年后翻番。

9. 从键盘上输入 10 个整数，求其中的最大值和次大值。

10. 从键盘输入 n，输出 n 行 * 组成的倒等腰三角形，如 n=4，则输出：

```
*******
 *****
  ***
   *
```

11. 用迭代法求 $X=\sqrt{a}$。迭代公式为：$X_{n+1}=\frac{1}{2}\left(X_n+\frac{a}{X_n}\right)$，要求迭代精度满足 $|X_{n+1}-X_n|<0.00001$。

12. 求解爱因斯坦数学题。有一条长阶梯，若每步跨 2 阶，则最后剩 1 阶；若每步跨 3 阶，则最后剩 2 阶；若每步跨 5 阶，则最后剩 4 阶；若每步跨 6 阶，则最后剩 5 阶；若每步跨 7 阶，最后一阶都不剩，问总共有多少级阶梯？

13. 打印如下的九九乘法表：

```
1  2  3  4  5  6  7  8  9
-------------------------------------------------
1
2  4
3  6  9
4  8  12 16
5  10 15 20 25
.................................................
9  18 27 36 45 54 63 72 81
```