第5章 循环结构程序设计

循环结构是程序中一种很重要的结构。其特点是，在给定条件成立时，反复执行某程序段，直到条件不成立为止。给定的条件称为循环条件，反复执行的程序段称为循环体。C 语言提供了多种循环语句，可以组成各种不同形式的循环结构。

(1) 用 goto 语句和 if 语句构成循环(此法已不提倡使用，本书不做介绍)。

(2) 用 while 语句。

(3) 用 do…while 语句。

(4) 用 for 语句。

5.1 while 语句以及用 while 语句构成的循环结构

5.1.1 while 循环的一般形式

由 while 语句构成的循环也称“当”循环，while 循环的一般形式如下：

```
while(表达式) 语句
```

其中表达式是循环条件，语句为循环体。

while 语句的语义是：计算表达式的值，当值为真(非 0)时，执行循环体语句，否则退出循环。

说明：

(1) while 是 C 语言的关键字，要小写。

(2) while 后一对小括号中的表达式可以是 C 语言中任意合法的表达式，但不能为空，由它来控制循环体是否执行。

(3) 在语法上，循环体只能是一条可执行语句，若循环体内有多条语句，则必须用一对大括号{}括起来，构成一条复合语句。

(4) 对于任何循环，必须掌握两点内容：一是循环条件是什么？二是循环体是谁？

(5) 如何结束循环，一般是两种方式：一是正常结束(即不满足循环条件了)；二是中途结束(用 break 语句，具体使用方法的介绍见 5.4 节)。

5.1.2 while 循环的执行过程

while 循环的执行过程如下：

(1) 计算 while 后小括号中表达式的值。当值为非 0 时，执行步骤(2)；当值为 0 时，

执行步骤(4)。

(2) 执行循环体一次。

(3) 转去执行步骤(1)。

(4) 退出 while 循环。

其执行过程如图 5.1 所示。

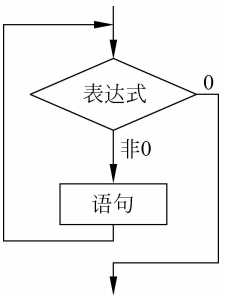


图 5.1 while 循环语句的执行流程

请初学者注意：

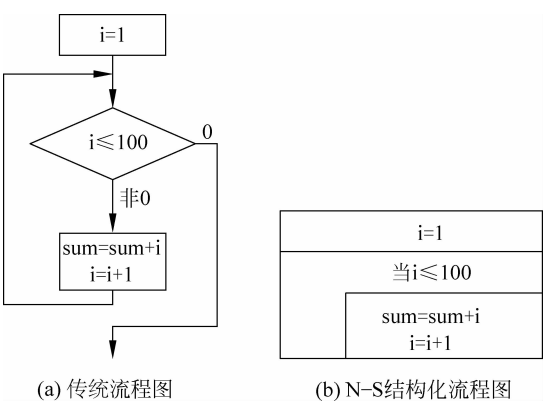
(1) while 语句的循环体可能一次都不执行，因为 while 后小括号中的条件表达式可能一开始就为 0。

(2) 不要把由 if 语句构成的分支结构与由 while 语句构成的循环结构混同起来。若 if 后条件表达式的值为非 0，其后的 if 子句只可能执行一次；而 while 后条件表达式的值为非 0 时，其后的循环体语句可能重复执行。在设计循环时，通常应在循环体内改变条件表达式中有关变量的值，使条件表达式的值最终变为 0，以便能结束循环。

(3) 当循环体需要无条件循环，条件表达式可以设为 1(恒真)，但在循环体内要有带条件的非正常出口(break 等)。

【例 5.1】 用 while 语句求 s＝1＋2＋3＋…＋100。

用传统流程图和 N-S 结构化流程图表示算法，如图 5.2(a)和图 5.2(b)所示。



(a) 传统流程图　　(b) N-S结构化流程图

图 5.2 例 5.1 算法流程图

根据流程图写出程序：

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

程序执行后输出以下结果：

```
sum = 5050
```

本例利用 while 循环语句实现了数据累加运算和循环控制变量 i 自增运算的重复执行。

【例 5.2】 用 while 语句求 s=1+1/(2 * 2)+1/(3 * 3)+1/(4 * 4)+…+1/(100 * 100)

分析：本题的算法也是求累加和，只是求和的每一项与例 5.1 不同。本题是求分式的累加和，因此需要注意：求和变量 sum 应该是 double 或 float 类型。由于分式中每一项的分子小于分母，因此为使分式的值不为零，累加的每一项应该写成 1.0/(i * i)。编写程序如下：

```
#include<stdio.h>
void main( )
  {  double sum = 0;
     int i;
     i = 1;
     while(i<=100)
        {
          sum = sum + 1.0/(i * i);
          i++;
        }
     printf("sum = %f\n",sum);
  }
```

程序执行后输出以下结果：

```
sum = 1.634984
```

【例 5.3】 用$\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}$…公式求 π 的近似值，直到最后一项的绝对值小于 10^{-6}为止。

分析：本题的基本算法也是求累加和，但比例 5.2 稍复杂。

(1) 用分母的值来控制循环的次数。若用 n 存放分母的值，则每累加一次 n 应当增 2。每次累加的数不是整数，而是一个实数，因此 n 应当定义成 float 类型。

(2) 可以看成隔一项的加数是负数。若用 t 来表示相加的每一项，则每加一项之后，t 的符号 s 应当改变，这可用交替乘 1 和−1 来实现。

(3) 从以上求 π 的公式来看，不能决定 n 的最终值应该是多少，但可以用最后一项的绝对值小于 10^{-6}来作为循环的结束条件。

程序如下：

```
#include<stdio.h>
#include<math.h>              /* 调用 fabs 函数时要求包含 math.h 文件 */
void main( )
  {
    float n,t,pi;
    int s;
    s = 1;                    /* 用 s 存放符号位,其值在 1～-1 之间变化 */
    n = 1.0;                  /* 用 n 存放每项的分母 */
```

```
    t = 1.0;                    /* 用 t 存放每项的值,初值为 1 */
    pi = 0;                     /* 用 pi 存放所求 π 的值,初值为 0 */
    while(fabs(t)>= 1e-6)       /* fabs(t)就是 t 的绝对值 */
        { pi += t;
          n += 2;
          s =- s;               /* 改变符号 */
          t = s/n;
        }
    pi = pi * 4;
    printf("pi = %f\n",pi);
}
```

程序执行后输出以下结果：

```
pi = 3.141594
```

while 语句一般用于事先并不知道循环次数的循环，例如通过控制精度等进行的计算可用 while 循环来实现。

5.2 do…while 语句以及用 do…while 语句构成的循环结构

5.2.1 do…while 语句构成的循环结构

do…while 循环结构的形式如下：

```
do
  循环体
while(表达式);
```

do…while 语句的语义是：先执行循环体中的语句，然后再判断表达式是否为真，如果为真则继续循环；如果为假，则终止循环。因此，do…while 循环至少要执行一次循环语句。

说明：

(1) do 是 C 语言的关键字，必须和 while 联合使用。

(2) do…while 循环由 do 开始，至 while 结束。必须注意的是：在 while(表达式)后的“;”不可丢，它表示 do…while 语句的结束。

(3) while 后一对小括号中的表达式，可以是 C 语言中任意合法的表达式，由它控制循环是否执行。

(4) 按语法，在 do 和 while 之间的循环体只能是一条可执行语句。若循环体内需要多个语句，应该使用复合语句。

5.2.2 do…while 循环的执行过程

do…while 循环的执行过程如下：

(1) 执行 do 后面循环体中的语句。

(2) 计算 while 后一对小括号中表达式的值。当值为非 0 时,转去执行步骤(1);当值为 0 时,执行步骤(3)。

(3) 退出 do…while 循环。

其执行过程可如图 5.3 所示。

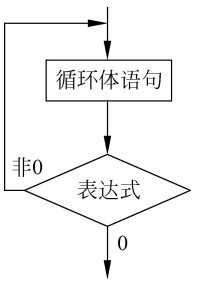


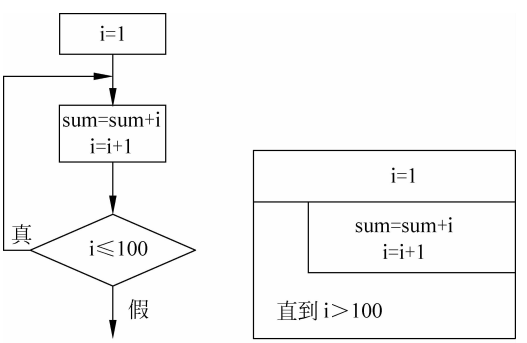
图 5.3 do…while 语句的执行流程

由 do…while 构成的循环与 while 循环十分相似,它们之间的重要区别是:while 循环的控制出现在循环体之前,只有当 while 后面条件表达式的值为非 0 时,才可能执行循环体,因此循环体可能一次都不执行;在 do…while 构成的循环中,总是先执行一次循环体,然后再求条件表达式的值,因此,无论条件表达式的值是 0 还是非 0,循环体至少要被执行一次。

和 while 循环一样,在 do…while 循环体中,一定要有能使 while 后表达式的值变为 0 的操作,否则,循环将会无限制地进行下去,除非循环体中有带条件的非正常出口(break 等)。

【例 5.4】 用 do…while 语句求 $s=1+2+3+\cdots+100$。

用传统流程图和 N-S 结构化流程图表示算法,如图 5.4(a)和图 5.4(b)所示。



(a) 传统流程图　　(b) N-S结构化流程图

图 5.4 例 5.3 算法流程图

根据流程图写出程序:

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

程序执行后输出以下结果：

```
5050
```

【例 5.5】 用迭代法求方程 $\cos(y)=y$ 的根，要求误差小于 10^{-6}。

分析：迭代法的求解思想是从一个初始值开始，将初始值代入迭代公式，得到一个迭代输出值。再次迭代时，将上一次的迭代输出当作本次的迭代输入，不断重复以上过程，直到满足要求为止。根据迭代法的基本思想，求解步骤如下：

(1) 取 $y1$ 的初值为 0.0。

(2) $y0=y1$，把 $y1$ 的初值赋给 $y0$。

(3) $y1=\cos(y0)$，求出一个新的 $y1$。

(4) 判断 $|y0-y1|<10^{-6}$ 是否成立，若 $y0-y1$ 的绝对值小于 10^{-6}，则执行步骤(5)，否则执行步骤(2)。

(5) 所求 $y1$ 就是方程 $\cos y=y$ 的根。计算结束，输出结果。

程序如下：

```
#include<stdio.h>
#include<math.h>
void main( )
  {   double y0,y1 = 0.0;
      do{
          y0 = y1;
          y1 = cos(y0);
        }
      while(fabs(y0 - y1)>= 1e - 6);
      printf("y1 = %f\n",y1);
  }
```

程序执行后输出以下结果：

```
y1 = 0.739086
```

【例 5.6】 计算 Fibonacci 数列，直到某项大于 1000 为止，并输出该项的值。

分析：Fibonacci 数列，$f_0=0$，$f_1=1$，$f_2=1$，$f_3=2$，$f_4=3$，…，$f_n=f_{n-2}+f_{n-1}$。程序中可定义 3 个变量 f_1、f_2 和 f，给 f_1 赋初值 0，f_2 赋初值 1，然后进行以下步骤：

(1) $f=f_1+f_2$；$f_1=f_2$；$f_2=f$；

(2) 判断 f_2 是否大于 1000，若不大于，重复步骤(1)继续循环；否则执行步骤(3)。

(3) 循环结束，输出 f_2 的值。

程序如下：

```
#include<stdio.h>
void main( )
  {  int f1,f2,f ;
     f1 = 0; f2 = 1;
     do
         { f = f1 + f2;
           f1 = f2;
```

```
            f2 = f;
        } while(f2 <= 1000);
    printf("F = %d\n",f2);
}
```


程序执行后输出以下结果：

```
F = 1597
```

5.3 for 语句以及用 for 语句构成的循环结构

5.3.1 for 语句构成的循环结构

for 语句构成的循环结构通常称为 for 循环。for 循环的一般形式如下：

```
for(表达式 1; 表达式 2; 表达式 3) 循环体
```

for 是 C 语言的关键字，其后的一对小括号中通常含有 3 个表达式，各表达式之间用“;”隔开。这 3 个表达式可以是任意形式的表达式，通常主要用于 for 循环的控制。紧跟在 for(...)之后的循环体语句在语法上要求是一条语句，若在循环体内需要多条语句，应该使用复合语句。

for 语句最简单的应用形式也是最容易理解的形式如下：

```
for(循环变量赋初值; 循环条件; 循环变量增量) 循环体
```

例如：

```
for(i = 1; i <= 100; i++) sum = sum + i;
```

以上 for 循环的作用是求 s=1+2+3+ … +100，它相当于以下语句：

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

显然，用 for 语句简单、方便。对于以上 for 语句的一般形式也可以改写成 while 循环的形式：

```
表达式 1;
while(表达式 2)
{
  循环体
  表达式 3;
}
```

在 C 语言中，for 语句使用最为灵活，不仅可以用于循环次数已经确定的情况，而且可以用于循环次数不确定而只给出循环结束条件的情况，它完全可以代替 while 语句。

5.3.2 for 循环的执行过程

for 循环的执行过程如下：

(1) 求解表达式 1。

(2) 求解表达式 2，若其值为非 0，转去执行步骤(3)；若其值为 0，转步骤(5)。

(3) 执行一次 for 循环体。

(4) 求解表达式 3，转向步骤(2)。

(5) 循环结束。

其执行过程如图 5.5 所示。

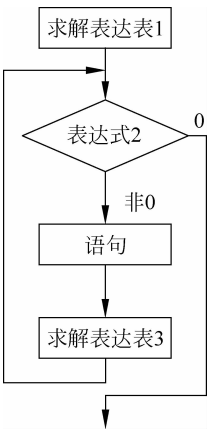


图 5.5　for 语句的执行流程

由 for 循环的执行过程，可以看出“表达式 1”只能被执行一次，它可以是设置循环控制变量初始值的赋值表达式，也可以是与循环控制变量无关的其他表达式。“表达式 2”的值决定了是否继续执行循环。“表达式 3”的作用通常是不断改变循环控制变量的值，最终使“表达式 2”的值为 0。由于第一次计算“表达式 2”时，其值可能等于 0，因此 for 循环语句的循环体可能一次也不执行。

5.3.3 有关 for 语句的说明

1. 表达式的“省略”

如果在 for 语句之前给循环变量赋了初值，则表达式 1 可以省略，但其后的分号不可省略。

对于从 1～100 求和的 for 循环语句：

```
for(i = 1; i <= 100; i++) sum = sum + i;
```

如果省略表达式 1，可以写成如下形式：

```
i = 1;                              /* 在 for 语句之前给循环变量赋初值 */
for( ; i <= 100; i++) sum = sum + i;
```

如果省略表达式 3，则应在 for 语句的循环体内修改循环控制变量，例如：

```
for(i = 1; i <= 100;)
      {   sum = sum + i;
          i++;                      /* 修改循环控制变量 */
      }
```

如果表达式 1 和表达式 3 都省略，则 for 语句就相当于 while 语句，例如：

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

如果 3 个表达式都省略，则 for 是无循环终止条件的循环，可以利用 break 语句终止循

环，例如：

```
i = 1;
for( ; ;)
    {   sum = sum + i;
        i++;
        if(i > 100)break;       /* 如果 i 大于 100,则退出循环 */
    }
```

总之，for 语句中的 3 个表达式可以都省略，也可以部分省略，但是起分隔作用的两个分号是绝不能省略的。

2. for 语句中的逗号表达式

逗号运算符的主要应用就是在 for 语句中。for 语句中的表达式 1 和表达式 3 可以是逗号表达式，特别是在有两个循环变量参与对循环控制的情况下。若表达式 1 和表达式 3 为逗号表达式，将使程序显得非常清晰，例如：

```
#include <stdio.h>
void main( )
  {   int i,j;
      for(i = 1,j = 10;i < j;i++,j--)
      printf("i = %d,j = %d\n",i,j);
  }
```

运行结果如下：

```
i = 1,j = 10
i = 2,j = 9
i = 3,j = 8
i = 4,j = 7
i = 5,j = 6
```

以上程序中 for 语句的表达式 1 是逗号表达式，它为两个循环变量赋初值；i=1,j=10。表达式 3 也是逗号表达式，它们的作用是修正两个循环控制变量的值：i++,j--。

3. 循环体为空语句

对于 for 语句，循环体为空语句的一般形式为：

```
for(表达式 1;表达式 2;表达式 3);
```

例如求 s=1+2+3+ … +100 可以用如下循环语句完成。

```
for(sum = 0,i = 1;i <= 100;sum += i,i++);
```

上述 for 语句的循环体为空语句，不做任何操作。实际上已把求累加和的运算放入表达式 3 中了。

C 语言中的 for 语句书写灵活，功能较强。在 for 后的一对小括号中，允许出现各种形式的与循环控制无关的表达式，虽然这在语法上是合法的，但这样会降低程序的可读性。建议初学者在编写程序时，在 for 后面的一对小括号内，仅含有能对循环进行控制的表达式，其他操作尽量放在循环体内去完成。

【例 5.7】 根据公式 $S=1+1/(1+2)+1/(1+2+3)+\cdots+1/(1+2+3+\cdots+n)$，

求前 20 项之和。

分析：本题主要是利用 for 循环来实现累加及递推运算，可由题中所给计算公式获得递推关系，程序中需要有累加和变量 sum，初值为 0，循环变量 i 从 1～20，以及一个累计量 t，$t=1+2+3+\cdots+i$。编写程序如下：

```
#include<stdio.h>
void main( )
  {   float sum=0.0;
      int t=0,i;
      for(i=1;i<=20;i++)
          {
            t+=i;
            sum+=1.0/t;
          }
      printf("the result is :sum= %f\n",sum);
  }
```

程序执行后输出以下结果：

```
the result is :sum=1.904762
```

【例 5.8】 从键盘输入一个正整数，判断它是否为素数(质数)。

分析：按照素数的定义，如果一个数只能被 1 和它本身整除，则这个数是素数。反过来说，如果一个数 x 能被 2～$x-1$ 之间的某个数整除，则这个数 i 就不是素数；而一个非素数至少有两对及以上因子，除一对是本身和 1 之外，另一对或以上因子中至少有一个要小于或等于原数的一半和原数开平方后的取整；由此推理可得判断一个正整数 x 是否为素数的方法有 3 个：

(1) x 被 2 ～$x-1$ 来除，若都不能被整除，则 x 就是素数。

(2) x 被 2 ～$x/2$ 来除，若都不能被整除，则 x 就是素数。

(3) x 被 2 ～$\sqrt{x}$来除，若都不能被整除，则 x 就是素数。

解法 1：利用标志变量法，就是在程序中设置一个标志，假定所有数都是素数，通过程序来动态改变标志，一旦标志变化则说明该数就不是素数，否则该数就是素数；编写程序如下：

```
#include<stdio.h>
void main( )
  {   int x,i,f=1;              /* f 是标志变量 */
      scanf("%d",&x);
      for(i=2;i<x;i++)
      if(x%i==0) {f=0;break;}
      if(f==1) printf("是素数");
      else printf("不是素数");
  }
```

解法 2：利用判断循环是如何退出的方法，编写程序如下：

```
#include<stdio.h>
void main( )
```

```
{   int x,i;
    scanf("%d",&x);
    for(i=2;i<x;i++)
    if(x%i==0) break;
    if(i==x) printf("是素数");  /* 如果 i 等于 x 的话,说明循环是正常退出的 */
    else printf("不是素数");
}
```

上述两种解法具有异曲同工之妙,请读者认真体会!

5.4 break 语句和 continue 语句在循环结构中的应用

5.4.1 break 语句

用 break 语句可以使流程跳出 switch 语句体,也可用 break 语句在循环结构中终止本层循环体,从而提前结束本层循环。

【例 5.9】 计算 $s=1+2+3+\cdots+i$,直到累加到 s 大于 5000 为止,并输出 s 和 i 的值。

```
#include<stdio.h>
void main( )
  {   int i,s;
      s=0;
      for(i=1; ;i++)
          { s=s+i;
            if(s>5000)break;
          }
      printf("s=%d,i=%d\n",s,i);
}
```

程序的输出结果如下:

```
s=5050,i=100
```

这是在循环体中使用 break 的示例。在例 5.9 中,如果没有 break 语句,程序将无限循环下去,成为死循环。但当 i=100 时,s 的值为 100*101/2=5050,if 语句中的条件表达式:s>5000 为“真”(值为 1),于是执行 break 语句,跳出 for 循环,从而终止循环。

break 语句的使用说明:

(1) 只能在循环体内和 switch 语句体内使用 break 语句。

(2) 当 break 出现在循环体中的 switch 语句体内时,其作用只是跳出该 switch 语句体,并不能终止循环体的执行。若想强行终止循环体的执行,可以在循环体中,但并不在 switch 语句中设置 break 语句,当满足限定条件时则跳出本层循环体。

5.4.2 continue 语句

continue 语句的作用是跳过本次循环体中余下尚未执行的语句,立刻进行下一次的循环条件判定,可以理解为仅结束本次循环。注意:执行 continue 语句并没有使整个循环终止。

在 while 和 do…while 循环中，continue 语句使得流程直接跳到循环控制条件的测试部分，然后决定循环是否继续进行。在 for 循环中，遇到 continue 后，跳过循环体中余下的语句，而去对 for 语句中的“表达式 3”求值，然后进行“表达式 2”的条件测试，最后根据“表达式 2”的值来决定 for 循环是否执行。在循环体内，不论 continue 是作为何种语句中的语句成分，都将按上述功能执行，这点与 break 有所不同。

【例 5.10】 把 100～200 之间不能被 3 整除的数输出。

```
#include<stdio.h>
void main( )
  {   int n;
      for(n=100;n<=200;n++)
          { if(n%3==0)
            continue;
            printf("%d",n);
          }
      printf("\n");
}
```

当 n 能被 3 整除时，执行 continue 语句，结束本次循环（即跳过 printf 函数语句），只有 n 不能被 3 整除时才执行 printf 函数。

当然，例 5.10 中的循环体也可以改用一个 if 语句处理：

```
if(n%3!=0) printf("%d",n);
```

在程序中使用 continue 语句无非为了说明 continue 语句的作用。

5.5 循环的嵌套

在一个循环体内又包含了另一个完整的循环结构，称为循环的嵌套。前面介绍的 3 种类型的循环都可以互相嵌套，循环的嵌套可以多层，但每一层循环在逻辑上必须是完整的。

在编写程序时，为了增强程序的可读性，循环嵌套的书写要采用缩进形式，像以下例题程序中所示，内循环中的语句应该比外循环中的语句有规律地向右缩进 2～4 列，这样的程序层次分明，易于阅读。

【例 5.11】 输出 $n\times n$ 个字符“*”。

分析：

（1）n 行“*”的输出，可用下列循环控制。

```
for(i=1;i<=n;i++)
```

（2）每行 n 个“*”的输出，可用下列循环语句实现：

```
for(j=1;j<=n;j++)
  putchar('*');
putchar('\n');
```

所以输出 $n\times n$ 行“*”可用双重循环语句实现，编写程序如下：

```
#include<stdio.h>
void main( )
{   int i,j;
    for(i=1;i<=n;i++)
          { for(j=1;j<=n;j++)          /* 输出一行"*"*/
                putchar('*');
            putchar('\n');             /* 换行 */
          }
}
```

这是循环控制变量之间没有依赖关系的多重循环。在许多情况下,内循环的循环控制变量的初值或终值依赖于外循环控制变量。

【例 5.12】 编写程序输出如下图形:

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

分析:

(1) 用循环控制变量 i(1≤i≤5)控制输出行。

```
for(i=1;i<=5;i++)
```

(2) 每行上的“*”个数是随着行控制变量 i 的值变化而变化的。

i=1 时,执行 1 次 putchar('*');

i=2 时,执行 2 次 putchar('*');

…

i=5 时,执行 5 次 putchar('*');

输出第 i 行时执行 i 次“putchar('*');”,所以内循环体语句应如下:

```
for(j=1;j<=i;j++)
    putchar('*');                  /* 输出一行'*'*/
```

因此,可用双重循环语句编写程序如下:

```
#include<stdio.h>
void main( )
{   int i,j;
    for(i=1;i<=5;i++)
          {   for(j=1;j<=i;j++)        /* 输出第 i 行'*'*/
                putchar('*');
              putchar('\n');
          }
}
```

【例 5.13】 编写程序,找出 2～100 以内的所有素数(质数)。

分析:由例 5.8 可知,如何判定给定的一个正整数是否为素数。因此,要找出一个范围内的所有素数,可采用双重循环的方法来实现。设循环变量 i 从 2～100 做外循环,判断素

数的过程作内循环，利用标识变量法编写程序如下：

```
#include<stdio.h>
void main( )
  {   int x,i,j,f;                          /* f是标志变量 */
      for(i=2;i<=100;i++)
            { f=1;
              for(j=2;j<i;j++)
                   if(i%j==0) {f=0;break;}
              if(f==1) printf("%d,"i);
            }
  }
```

【例 5.14】 编写程序，找出大于正整数 m 且靠近 m 的 k 个素数，m 和 k 均从键盘输入。

分析：本题中要求找出大于 m 且靠近 m 的 k 个素数，因此需要从 $m+1$ 开始循环查找素数，如果是素数则将其输出，同时计数器 n 加 1，直到 $n=k$ 为止。而验证一个数 i 是不是素数，则需要利用循环从 $2\sim i-1$ 逐个去取余，都除不尽则是素数。编写程序如下：

```
#include<stdio.h>
void main( )
{   int i,j,m,k,n=0;
    printf("please enter m and k:");
    scanf("%d%d",&m,&k);
    for(i=m+1;n<k;i++)
          { for(j=2;j<i;j++)
                if(i%j==0) break;
            if(j>=i) {printf("%d",i); n++;}
          }
}
```

程序执行过程中输入：

```
20 5↙(回车)
```

则输出结果如下：

```
23 29 31 37 41
```

以上 4 个例子都是两重 for 循环嵌套。另外，3 种循环(while 循环、do…while 循环和 for 循环)也可以互相嵌套。例如，下面几种都是合法的形式：

```
(1) while( )
      { …
        while( )
          {…}
      }

(2) do
      { …
        do
          {…}
```

```
            while();
          }

    (3) for(;;)
          {
            for(;;)
              {…}
          }

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

    (5) for(;;)
          {…
            while()
              {…}
              …
          }

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

注意：循环嵌套的程序中，要求内循环必须被包含在外循环的循环体中，不允许出现内外层循环体交叉的情况。

5.6 3种循环的比较

3种循环都可以用来处理同一个问题，一般可以互相代替。例如，用3种循环方法都实现了求s=1+2+3+ … +100(即从1～100的和)。

for循环语句和while循环语句都是先检查循环条件是否成立，后执行循环体，因此循环体可能一次也不执行；而do…while循环语句是先执行循环体，后检查循环条件是否成立，因此循环体至少被执行一次。

对于while和do…while循环语句，对“表达式”中循环控制变量赋初始值是在执行这两个循环语句之前完成的；而对于for循环语句，对“表达式2”中的循环控制变量赋初始值既可在“表达式1”中完成，又可在执行for循环语句之前完成。

为了防止出现“死循环”，while和do…while循环语句的循环体中一般应包括改变“表达式”中循环控制变量值的语句，以便使循环操作趋于结束；而for循环语句通常是在“表达式3”中包含改变循环控制变量的值，进而使循环趋于结束。

5.7 程 序 举 例

【例 5.15】 计算并输出 100 以内(包括 100),能被 3 或 7 整除的所有自然数的倒数之和。

分析:本例中要找到小于等于 100 的每一个整数,即用变量 i 从 1～100 循环,判断条件能否被 3 或者被 7 整除即 i%3==0 || i%7==0,最后求得符合要求的倒数的累加和,编写程序如下:

```
#include<stdio.h>
void main( )
  { int i;
    double sum=0.0;
    for(i=1;i<=100;i++)
        { if(i%3==0 || i%7==0)
          sum+=1.0/i;
        }
    printf("sum= %f",sum);
}
```

运行结果如下:

```
sum=1.728235
```

【例 5.16】 求分数序列 1/2,2/3,3/5,5/8,… 前 10 项之和。

分析:本例每一个分数是 Fibonacci 数列 1,1,2,3,5,8,21,…从第三项起前后两项的商。该分数序列从第二项起,每一项的分子是前一项的分母,每一项的分母是前一项的分子与分母之和。用变量 s 求和,s 初值为 0,用变量 m 来求各项的分子,变量 n 求各项的分母,k 为中间变量,初值 $m=1$,$n=2$。用变量 i 从 1～10 循环,$s=s+m/n$; $k=m+n$; $m=n$; $n=k$;即可求该序列的和,编写程序如下:

```
#include<stdio.h>
void main( )
  { int i;
    float k,s=0,m=1,n=2;
    for(i=1;i<=10;i++)
          { s=s+m/n;
            k=m+n;
            m=n;
            n=k;
          }
    printf("s= %.f",s);
}
```

运行结果如下:

```
s=6.097960
```

【例 5.17】 找出 100～999 之间的所有“水仙花”数,所谓“水仙花”数是指一个 3 位数,

其各位数字的立方和等于该数本身，例如 $153=1^3+3^3+5^3$，所以 153 是“水仙花”数。

分析：设 $100\leqslant n\leqslant 999$，$i$、$j$、$k$ 分别代表数 n 百位、十位、个位上的数字，则：

```
i = n/100
j = n/10 % 10
k = n % 10
```

如果 $i^3+j^3+k^3==n$，则 n 即是所求。编写程序如下：

```
#include <stdio.h>
void main( )
  { int i,j,k,n;
    for(n = 100;n <= 999;n++)
          { i = n/100;
            j = n/10 % 10;
            k = n % 10;
            if(n == (i * i * i + j * j * j + k * k * k))
                  printf(" % d\n",n);
          }
}
```

运行结果如下：

```
153
370
371
407
```

【例 5.18】 从键盘输入两个正整数 m 和 n，求这两个数的最大公约数和最小公倍数。

分析：求解最大公约数，最小公倍数的步骤如下：

(1) 输入两数 m 和 n，将较大者保存在变量 a 中，较小者保存在变量 b 中，采用 while( ) 循环的方法求解最大公约数，结束条件是余数为 0。

(2) 求解最大公约数的思想俗称辗转相除法，即将 a 对 b 求余，如果余数为 0，则 b 即为两数的最大公约数；如果余数不为 0，则将 b 赋给 a，余数赋给 b，继续执行 a 对 b 求余运算，如此反复，直到余数为 0。

(3) 最小公倍数等于两数的乘积除以最大公约数。

编写程序如下：

```
#include <stdio.h>
void main( )
  {   int a,b,c,t,m,n,max,min;          /* 用 max 来放最大公约数,用 min 放最小公倍数 */
      scanf(" % d % d",&m,&n);
      t = m * n;                        /* 把 m 和 n 的乘积放在变量 t 中 */
      if(m < n) { a = n;b = m;}
      else { a = m;b = n;}              /* 让 a 放 m 和 n 中大的数 */
      c = a % b;
      while(c! = 0)
          {
            a = b;
            b = c;
```

```
            c = a % b;
          }
      max = b;                          /* b 的值就是最大公约数 */
      min = t/max;                      /* 最小公倍数等于两数的乘积除以最大公约数 */
      printf("max = %d,min = %d\n",max,min);
}
```

程序执行过程中输入：

```
12 21 ↙(回车)
```

则输出结果如下：

```
max = 3,min = 84
```

【例 5.19】 从键盘输入正整数 n，程序的功能是将 n 中各位上为偶数的数取出，并按原来从高位到低位相反的顺序组成一个新的数，将其输出。

分析：本例中要将数 n 中的个位取出，应将 n%10，得到的是 n 的低位，存放到变量 t 中，由于题中要求按原来从高位到低位的相反的顺序组成一个新数 x，因此每次 x 乘 10 再加 t，即 x＝x＊10＋t，这样低位就到高位上了，用 n 做循环变量，每次取 n 的个位，取完后应将 n 除以 10，让十位变成个位，为下次循环做好准备。编写程序如下：

```
#include<stdio.h>
void main( )
  {   long n,x = 0;
      int t;
      scanf("%ld",&n);
      while(n)
          { t = n % 10;
            if(t % 2 == 0)
            x = x * 10 + t;
            n = n/10;
          }
      printf("x = %ld\n",x);
}
```

程序执行过程中输入：

```
27638496 ↙(回车)
```

则输出结果如下：

```
x = 64862
```

【例 5.20】 电文加密问题。已知电文加密的规律为：将字母变成其后面的第 3 个字母，其他字符保持不变。例如，a→e，A→E，W→A。编写一个程序，输入一行字符，要求转换成加密电文输出。

分析：输入字符 ch，如果 ch 是字母，则进行加密处理 ch＋＝3；判断机密后 ch 是否超出字母的范围，如果超过，则 ch－＝26；循环控制条件用 ch!＝'\n'。编写程序如下：

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

```
void main( )
  { char ch;
    while((ch = getchar( ))! = '\n')
        {if((ch >= 'a'&&ch <= 'z') || (ch >= 'A'&&ch <= 'Z'))
                  { ch += 3;
                    if((ch>'Z'&&ch<'a') || (ch>'z'))     /* ch 在加密后越界 */
                          ch -= 26;
                  }
         printf("%c",ch);              /* 输出电文 */
       }
}
```

程序运行时,输入:

```
student123 ↙
```

输出结果为:

```
vwxghqw123
```

习 题 5

一、选择题

5.1 有以下程序段:

```
int k,j,s;
for(k = 2;k < 6;k++,k++)
{   s = 1;
    for(j = k;j < 6;j++) s += j;
}
```

程序段的输出结果是(　　)。

A) 11　　B) 9　　C) 1　　D) 10

5.2 有以下程序段:

```
int n = 10;
while(n > 7)
{ n--;
  printf("%d\n",n);
}
```

程序段的输出结果是(　　)。

A) 10　　B) 9　　C) 10　　D) 9

9　　8　　9　　8

8　　7　　8　　7

7　　6

5.3 有以下程序段:

```
int x = 3;
do
```

```
{ printf(" %3d",x-=2);
}
while(!(--x));
```

程序段的输出结果是(　　)。

A) 3 0　　B) 1　　C) 1 -2　　D) 死循环

5.4　有以下程序段：

```
int n,t=1,s=0;
scanf(" %d",&n);
do{ s=s+t; t=t-2;} while(t!=n);
```

为使此程序段不陷入死循环,从键盘输入的数据应该是(　　)。

A) 任意正奇数　　B) 任意负偶数　　C) 任意正偶数　　D) 任意负奇数

5.5　若变量已正确定义,有下列程序段：

```
i=0;
do printf(" %d,",i); while(i++);
printf(" %d\n",i);
```

程序段的输出结果是(　　)。

A) 0,0　　B) 0,1　　C) 1,1　　D) 程序进入死循环

5.6　下面的 for 语句的循环次数为(　　)。

```
for(x=1,y=0;(y!=19)&&(x<6);x++);
```

A) 无限循环　　B) 循环次数不定

C) 最多执行 6 次　　D) 最多执行 5 次

5.7　有以下程序：

```
#include <stdio.h>
void main( )
{ int i,sum;
  for(i=1;i<6;i++)
  sum+=sum;
  printf(" %d\n",sum);
}
```

程序的输出结果是(　　)。

A) 14　　B) 0　　C) 不确定　　D) 15

5.8　有以下程序：

```
#include <stdio.h>
void main( )
{ int y=10;
  for(;y>0;y--)
  if(y%3==0)
    {printf(" %d",--y); continue;
    }
}
```

程序的输出结果是(　　)。

A) 963　　B) 852　　C) 741　　D) 875421

5.9　有以下程序：

```
#include"stdio.h"
void main( )
{int i;
for(i=1;i<=5;i++)
  {if(i%2) printf("*");
           else continue;
    printf("#");
    }
    printf("$\n");
}
```

程序的输出结果是(　　)。

A) *#*#*#$　　B) #*#*#*$

C) *#*#$　　D) #*#*$

5.10　有以下程序：

```
void main( )
{ int i,j,x=0;
  for(i=0;i<2;i++)
    { x++;
      for(j=0;j<=3;j++)
        { if(j%2) continue;
          x++;
        }
      x++;
    }
    printf("x=%d\n",x);
}
```

程序执行后的输出结果是(　　)。

A) x=4　　B) x=8　　C) x=6　　D) x=12

5.11　有以下程序：

```
#include"stdio.h"
void main( )
{ int x,i;
   for(i=1;i<=100;i++)
    { x=i;
      if(++x%2==0)
        if(++x%3==0)
          if(++x%7==0)
          printf("%d",x);
      }
    printf("\n");
}
```

程序的输出结果是(　　)。

A) 28 70　　B) 42 84　　C) 26 68　　D) 39 81

二、填空题

5.12　下列程序的输出结果是_____。

```
#include"stdio.h"
void main( )
{ int n=12345,d;
  while(n!=0) {d=n%10; printf("%d",d);n/=10;}
}
```

5.13　以下程序的输出结果是_____。

```
#include"stdio.h"
void main( )
{ int x=27;
  while(x>20&&x<30)
    { x++;
      if(x/3) {x++;break;}
      else continue;
    }
    printf("%d\n",x);
}
```

5.14　以下程序段的输出结果是_____。

```
int k,n,m;
n=10;m=1;k=1;
while(k++<=n) m*=2;
printf("%d\n",m);
```

5.15　以下程序的输出结果是_____。

```
#include"stdio.h"
void main( )
{ int x=2;
  while(x--);
  printf("%d\n",x);
}
```

5.16　下列程序的输出结果是_____。

```
void main( )
{ int i=0,sum=0;
  do
    { if(i==(i/2)*2)
      continue;
      sum+=i;
      } while(++i<100);
  printf("%d\n",sum);
}
```

5.17　以下程序段的输出结果是_____。

```
int i = 0,sum = 1;
do{sum += i++;} while(i<5);
printf(" %d\n",sum);
```

5.18　下列程序运行后的输出结果是_____。

```
void main( )
{ char c1,c2;
  for(c1 = '0',c2 = '9';c1 < c2;c1++,c2 -- )
    printf(" %c %c",c1,c2);
  printf("\n");
}
```

5.19　有以下程序段：

```
s = 1.0;
for(k = 1;k <= n;k++)
    s = s + 1.0/(k * (k + 1));
printf(" %f\n",s);
```

请填空，使下面的程序段的功能完全与之等同。

```
s = 0.0;
_____;
k = 0;
do
{ s = s + d;
  _____;
  d = 1.0/(k * (k + 1));
}while(_____);
printf(" %f\n",s);
```

5.20　下列程序的功能是计算：s＝1＋12＋123＋1234＋12345。请填空。

```
void main( )
{ int t = 0,s = 0,i;
  for(i = 1;i <= 5;i++)
    { t = i + _____; s = s + t;}
    printf("s = %d\n",s);
}
```

5.21　以下程序的功能是：从键盘输入若干学生的成绩，统计并输出最高成绩和最低成绩，当输入负数时结束输入。请填空。

```
#include"stdio.h"
void main( )
{ float x,amax,amin;
  scanf(" %f",&x);
  amax = x;amin = x;
  while(_____)
    { if(x > amax) amax = x;
```

```
        if(_____) amin = x;
        scanf("%f",&x);
      }
printf("\namax = %f\namin = %f\n",amax,amin);
}
```

5.22　下面程序的输出结果是_____。

```
void main( )
{ int i = 0,a = 0;
  while(i < 20)
    { for(; ;)
      { if((i % 10) == 0) break;
        else i -- ;
      }
      i += 11; a += i;
    }
  printf("%d\n",a);
}
```

三、编程题

5.23　编写程序，求 1－3＋5－7＋ … －99＋101 的值。

5.24　从键盘输入一个正整数 n，计算该数的各位数字之和并输出。例如，输入 576，则计算 5＋7＋6＝18 并输出。

5.25　一个整数等于该数所有因子之和，则称该数是一个完数。例如，6 和 28 都是完数。因为 6＝1＋2＋3，28＝1＋2＋4＋7＋14，输出 3 位数中所有的完数。

5.26　编写程序，求 e 的值，$e \approx 1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\cdots$。

(1) 用 for 循环计算前 50 项。

(2) 用 while 循环，要求直到最后一项的值小于 10^{-6}。

5.27　编写程序，按公式 $S=1+(1+2^{0.5})+(1+2^{0.5}+3^{0.5})+\cdots+(1+2^{0.5}+3^{0.5}+\cdots+N^{0.5})$ 计算 N＝20 时，该序列的和。