第 5 章 循环结构程序设计

【本章要点】

- 循环结构程序设计的基本思想。
- for 语句、while 语句、do…while 语句的使用方法。
- 循环嵌套的应用方法。
- 循环控制语句 break 和 continue 的用法。

循环结构是程序设计中实现重复性与规律性操作的核心构造，可高效解决代码冗余、重复操作烦琐等编程问题，它与顺序结构、选择结构共同构成了程序设计的三大基本结构，是构建复杂程序逻辑的基础模块；循环结构的特点是在指定的循环条件成立时，重复执行循环体，直至条件不成立为止，C 语言提供了三种循环控制语句来实现这一机制，即 while 语句、do...while 语句和 for 语句。

5.1 while 语句

while 语句适合循环次数不确定，仅通过条件判断控制循环的场景。while 语句的特点是：先计算条件表达式，值为真才执行循环体语句，否则不执行。

1. while 语句的一般格式

```
while(表达式)
{
   循环体;
}
```

其中，“表达式”是循环条件，可以为任意合法的表达式，一般是关系表达式或逻辑表达式；“循环体”语句是重复执行的程序段，可以是单条语句、空语句，也可以是复合语句。

2. while 语句的执行过程

先计算表达式，当表达式的值为“真”或非 0 时，执行循环体，再判断表达式的值，如果表达式的值为真，重复执行循环体，直到表达式的值为“假”或 0 时，结束循环，执行 while 语句后面的语句。while 语句的执行过程如图 5-1 所示。

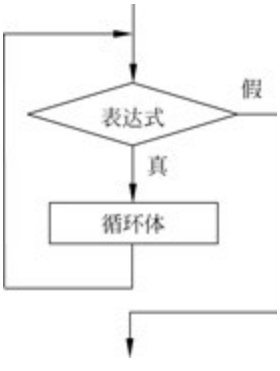

图 5-1 while 语句的执行过程

【示例 5-1】 输出 1～5 的数字。

【分析】 输出数字的语句如下。

```
printf("%d\n",1);
printf("%d\n",2);
printf("%d\n",3);
printf("%d\n",4);
printf("%d\n",5);
```

(1) 定义整型变量 i 表示整数 1～5 的任意一个数，初始值为 1。

(2) 以上 5 条语句即可表示为：当 i≤5 时，重复执行以下

语句。

```
printf("%d\n",i);
i++;
```

该程序的算法流程图如图 5-2 所示。

程序代码如下：

```
#include<stdio.h>
int main(void)
{
   int i=1;
   while(i<=5)
   {
      printf("%d\n",i);
      i++;
   }
   return 0;
}
```

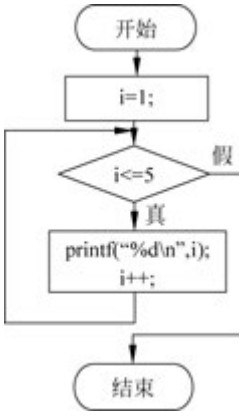

图 5-2　示例 5-1 的算法流程图

程序运行结果如下：

```
1
2
3
4
5
```

【示例 5-2】 编程计算 1＋2＋3＋…＋100 的值。

【分析】 本题的特点是重复执行加法运算，将 1，2，…逐个累加起来直至加到 100 为止。

（1）定义变量。

sum（也称累加器）：整型，存储累加和的值，初始值为 0。i：整型，存储所加的数，初始值为 1，其值的变化为 1，2，3，…，100，同时，该变量也作为循环变量，表示重复执行加法运算的次数。

（2）本题的求和问题可表示为：当 i≤100 时，重复执行以下语句。

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

该程序的算法流程图如图 5-3 所示。

程序代码如下：

```
#include<stdio.h>
int main(void)
{
   int i=1,sum=0;                    /* 定义并初始化变量 i、sum */
   while(i<=100)
   {
      sum=sum+i;                     /* 累加 i 的值 */
      i++;                           /* 取 i 的下一个值 */
   }
   printf("sum=%d\n",sum);
   return 0;
}
```

程序运行结果如下：

```
sum=5050
```

【示例 5-3】 从键盘输入若干字符，直到输入特殊符号“!”为止（如输入“student!”），统计

输入字符的个数。

【分析】 根据题意，本题可使用循环结构完成。

(1) 定义以下变量。

ch：字符型，用来存储键盘输入的一个字符。count：整型，用来统计输入字符的个数，初始值为 0。

(2) 本题可表示为：当 ch!='!'时，重复执行以下操作。

利用 getchar()函数接收一个字符，ch=getchar()。统计录入的字符的个数，count++。

该程序的算法流程图如图 5-4 所示。

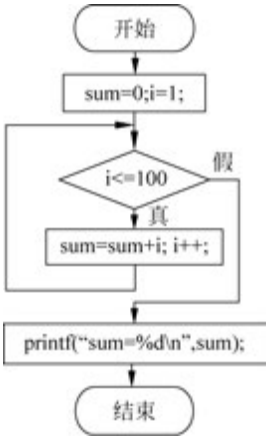

图 5-3 示例 5-2 的算法流程图

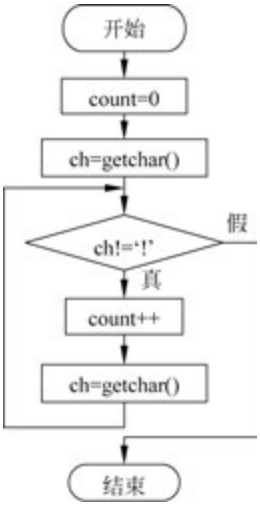

图 5-4 示例 5-3 的算法流程图

程序代码如下：

```
#include <stdio.h>
int main(void)
{
   int  count = 0;                    //计数器初值为 0
   char ch;
   ch = getchar() ;                   //输入第一个字符到变量 ch 中
   while(ch!= '!')                    //如果输入的不是"!",则进入循环体
   {
      count++;                        //统计输入的字符个数
      ch = getchar();                 //输入下一个字符
   }
   printf("输入的字符总数是: %d\n",count);
   return 0;
}
```

程序运行结果如下：

```
student!  ↵
输入的字符总数是: 7
```

5.2 do…while 语句

do…while 语句的特点是先执行一次循环体，再判断条件，因此循环体至少会执行一次，适用于需要"先执行后判断"的场景。

1．do…while 语句的一般形式

```
do
{
   循环体;
} while(表达式);
```

其中，do 必须与 while 联合使用，在 while 的表达式后面必须要有分号，表示该语句的结束；“表达式”是循环条件，可以是任何类型，通常是关系表达式或逻辑表达式；“循环体”语句为重复执行的程序段，可以是单个语句，空语句，也可以是复合语句。

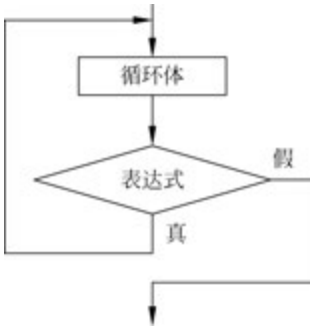

图 5-5　do…while 语句的执行流程

2．do…while 语句的执行过程

do…while 语句的执行过程是：先执行循环体，然后计算表达式，根据表达式的值是否为真决定是否继续执行循环体。

do…while 语句的执行流程如图 5-5 所示。

【示例 5-4】　用 do…while 语句编程，计算 1＋2＋…＋100 的值。

程序代码如下：

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

程序运行结果如下：

```
sum=5050
```

3．while 语句和 do…while 语句的区别

通常，while 语句和 do…while 语句只是格式不同，都可以用来解决同一个问题。只有一种情况下两者处理同一个问题时结果不同，即当最初的循环条件不满足时，对于 while 语句，循环体一遍都没有执行，而对于 do…while 语句，循环体已经执行了一遍，如示例 5-5 所示。

【示例 5-5】　while 和 do…while 语句循环的比较。

程序代码如下：

```
//(1)
#include<stdio.h>
int main(void)
{
int i=10,sum=0;
while(i<10)
{
   sum=sum+i;
   i++;
}
printf("sum= %d,i= %d\n",sum,i);
return 0;
}
```

```
//(2)
#include<stdio.h>
int main(void)
{
int i=10,sum=0;
do{
   sum=sum+i;
   i++;
}
while(i<10);
printf("sum= %d,i= %d\n",sum,i);
return 0;
}
```

程序运行结果如下：

```
sum = 0, i = 10
```

程序运行结果如下：

```
sum = 10, i = 11
```

while 语句和 do…while 语句的具体区别如下。

(1) 执行顺序不同。while 语句先判断条件表达式是否为真，如果为真才执行循环体，否则一次都不执行。而 do…while 语句首先执行一次循环体，再判断条件表达式是否为真，如果为真就继续执行循环体，否则结束循环。

(2) 语法上不同。while 语句中表达式后面没有分号，而 do…while 语句中表达式后有分号。

(3) 最小执行次数不同。当初始条件为假时，while 语句循环体不执行。而 do…while 语句无论条件是否为真都执行一次循环体。

5.3 for 语句

for 语句比 while 语句更为灵活方便，一般用于循环次数已经确定的情况。

1. for 语句的一般形式

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

【说明】

(1) 表达式 1 通常为循环控制变量赋初值。

(2) 表达式 2 通常是循环条件，常用关系表达式或逻辑表达式。

(3) 表达式 3 的作用是用来修改循环变量的值，即循环控制变量在每次循环结束后变化的方式。

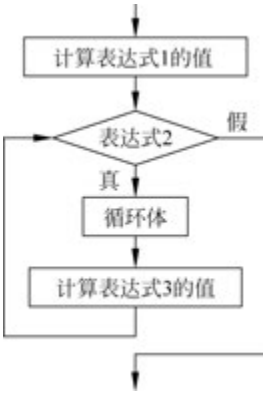

图 5-6 for 语句的执行流程

(4) 3 个表达式用分号隔开，且 3 个表达式都可省略，但分号不能省略。每个表达式可由多个表达式组成，如用逗号表达式来表示每个表达式。

2. for 语句的执行过程

for 语句的执行过程如下。

(1) 执行表达式 1。

(2) 计算表达式 2，若值非 0(循环条件成立)，执行步骤(3)；否则，结束 for 语句，继续执行后续的语句。

(3) 执行循环体语句，并计算表达式 3，然后转向步骤(2)。

for 语句的执行流程如图 5-6 所示。

【示例 5-6】 计算 1＋2＋…＋100 的值。

程序代码如下：

```
#include <stdio.h>
int main(void)
{
   int i,sum = 0;                    /* 定义循环变量 i、初始化 sum */
   for(i = 1;i <= 100;i++)
      sum = sum + i;                 /* 累加求和 */
```

```
    printf("sum = %d\n",sum);
    return 0;
}
```

程序运行结果如下：

```
sum = 5050
```

3. for 语句的变形

for 语句的书写格式非常灵活，其有以下几种变形形式。

(1) for 语句中的表达式 1、表达式 2、表达式 3 均可省略，但其间的分号不能省略。

例如，在示例 5-6 中，省略表达式 1 可以写成以下形式：

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

在示例 5-6 中，省略表达式 3 可以写成以下形式：

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

在示例 5-6 中，省略表达式 1 和表达式 3 可以写成以下形式：

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

这种形式等价于 while 语句。

若在示例 5-6 中省略表达式 2，即循环条件始终为真，则构成无限循环，循环将无终止地进行下去，程序流程如图 5-7 所示。

为使循环能正常结束，此时要在循环体内使用 break 语句终止循环。

程序代码如下：

```
i = 1;
for( ; ; )                /* 相当于 for(;1;) */
{
    sum += i;
    i++;
    if(i > 100)
        break;
}
```

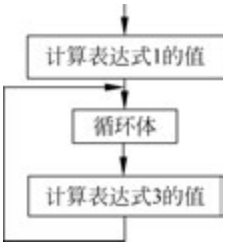

图 5-7　省略循环条件的 for 循环流程

(2) for 语句中的逗号表达式。

for 语句中的表达式 1 和表达式 3 可以是逗号表达式，例如，示例 5-6 中的表达式 1 可以写成如下形式：

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

特别是在有两个循环变量参与循环控制的情况下，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：

```
int i,j;
for(i = 1,j = 10;i < = j;i++,j -- )
    printf("i = %d,j = %d\n",i,j);
```

程序运行结果如下：

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

(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如下：

```
for(表达式 1; 表达式 2; 表达式 3)
;
```

例如，计算 1＋2＋3＋…＋100 的值，可以由如下循环语句完成：

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

在循环体为空的结构中，其核心作用是将循环的所有逻辑(初始化、条件判断、迭代操作)都浓缩在 for 循环的三个表达式中，无须循环体执行额外操作，适用于逻辑简单、无须在循环体内操作的场景。

5.4 循环的嵌套

若在一个循环内又包含另一个完整的循环结构，则称为循环的嵌套。三种基本循环结构(while 循环、do…while 循环和 for 循环)可以互相嵌套，理论上循环嵌套层数是没有限制的。如果是两层循环嵌套，则称为双重循环；如果是三层循环嵌套，则称为三重循环；如果是 N 层循环嵌套，则可以称为 N 重循环。在实际编程中，过多的循环嵌套会影响程序的可读性和执行效率。

【示例 5-7】 从键盘输入一个正整数 n，在屏幕上输出由“*”构成的图案。例如，输入 5，输出如图 5-8 所示的三角形图案。

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

图 5-8 三角形图案

【分析】 在输出的三角形图案中，每行输出均是由“*”及换行符组成的。其规律如下。

第 1 行：1 个 *，1 个换行符。

第 2 行：3 个 *，1 个换行符。

第 3 行：5 个 *，1 个换行符。

第 4 行：7 个 * ,1 个换行符。

第 5 行：9 个 * ,1 个换行符。

由此可推断第 i 行：2×i－1 个 * ,1 个换行符。

(1) 定义变量如下。

- n 代表总行数。
- i 表示行的编号,值为 1～n。
- j 表示每行输出的 * 个数：第 1 行 1 个,第 2 行 3 个,……,第 i 行 2×i－1 个。

(2) 输出图案需要双重循环,外层循环的控制变量 i 表示输出的行数,值为 1～n；内层循环的控制变量 j 表示输出的 * 的个数,值为 1～2×i－1。

程序代码如下：

```
#include<stdio.h>
int main(void)
{
   int i,j,n;
   scanf("%d",&n);
   for(i=1;i<=n;i++)                    /* 按行依次输出 */
   {
      for(j=1;j<=2*i-1;j++)             /* 输出当前行的*  */
         printf("*");
      printf("\n");                     /* 当前行换行 */
   }
   return 0;;
}
```

程序运行结果如下：

```
6↵
*
***
*****
*******
*********
***********
```

【示例 5-8】 在屏幕上输出如图 5-9 所示的九九乘法口诀表。

```
1*1=1
1*2=2  2*2=4
1*3=3  2*3=6  3*3=9
1*4=4  2*4=8  3*4=12  4*4=16
1*5=5  2*5=10  3*5=15  4*5=20  5*5=25
1*6=6  2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7  2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8  2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9  2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

图 5-9　九九乘法口诀表

【分析】 九九乘法口诀表由 9 行 9 列构成,具体结构如下。

第一行：输出 1＊1＝1,换行。

第二行：输出 1＊2＝2,2＊2＝4,换行。

第三行：输出 1＊3＝3,2＊3＝6,3＊3＝9,换行。

……

第 i 行：输出 i 个表达式及换行，i 个表达式依次是：1 * i=i，2 * i=2i，3 * i=3i，…，j * i=ji，…，i * i=i^2，换行。

因此要输出九九乘法口诀表，需要使用双重循环，外层循环的控制变量 i 表示输出的行数，值为 1～9；内层循环的控制变量 j 表示输出的列数，值为 1～i。

程序代码如下：

```
#include <stdio.h>
int main(void)
{
   int i,j;
   for(i=1;i<=9;i++)                                  /* 输出第 1～9 行 */
   {
      for(j=1;j<=i;j++)                               /* 输出当前行的 1～i 列 */
         printf("%d*%d=%-2d",j,i,i*j);                /* 输出每一行 i*j 的结果 */
      printf("\n");                                   /* 输出每行的换行符 */
   }
   return 0;
}
```

5.5 break 语句与 continue 语句

循环程序通常会按照事先设定的循环条件正常地开始和结束，在实际应用中，由于某种情况，有时需要在循环进行的过程中从本循环体中提前退出，或者加快循环速度。也就是说，要改变循环执行的状态。C 语言提供了 break 和 continue 语句，用于控制循环过程。

▶ 5.5.1 break 语句

break 语句可以用在循环语句和 switch 语句中，其作用是跳出控制结构语句。在循环语句中用来终止本层循环，然后执行本层循环结构外的下一条语句。通常，break 语句总是与 if 语句一起使用，即满足条件时便跳出本层循环结构。

break 语句的格式如下：

```
break;
```

【示例 5-9】 从键盘输入两个正整数 x、y(2≤x，y≤1000)，输出 1000 以内最大的既不是 x 的倍数，也不是 y 的倍数的正整数。

【分析】 根据题意，需要使用循环结构，定义变量 i 作为循环变量，从大到小逐一判断，找到第一个既不是 x 的倍数，也不是 y 的倍数的正整数，输出该数后，用 break 语句结束循环，不再进行判断。

程序代码如下：

```
#include <stdio.h>
int main(void)
{
   int x,y,i;
   printf("请输入两个正整数 x 和 y: ");
   scanf("%d%d",&x,&y);
   for(i=1000;i>=2;i--)
   {
```

```
        if(i % x!= 0 && i % y!= 0){
            printf(" % d\n",i);
            break;
        }
    }
    return 0;
}
```

程序运行结果如下：

```
请输入两个正整数 x 和 y: 2 3 ↵
997
```

【示例 5-10】 从键盘输入一个正整数 n，判断 n 是否为素数(质数)。

【分析】 素数是只能被 1 和自身整除的整数。判断整数 n 是否为素数，可以用 n 除以 $2\sim\sqrt{n}$ 的整数，如果 n 能被其中的一个整数整除，则表明 n 不是素数；如果 n 不能被 $2\sim\sqrt{n}$ 中的任何一个整数整除，则表明 n 是素数。

该程序的执行流程如图 5-10 所示。

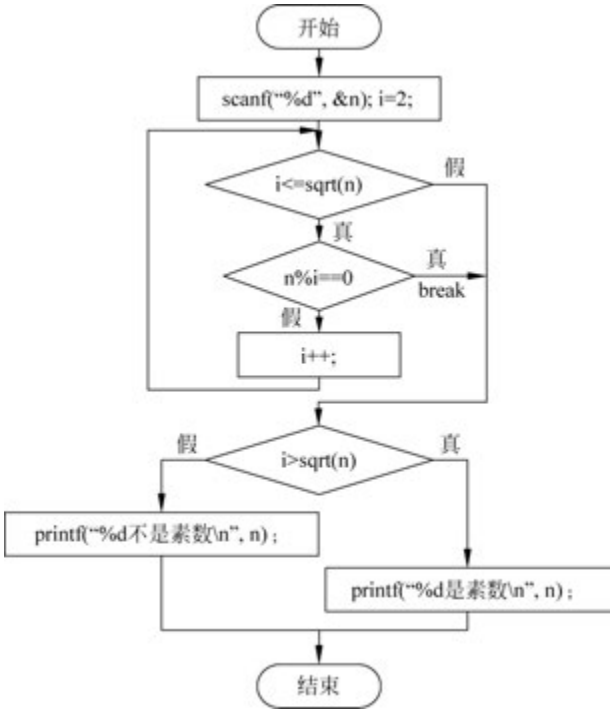

图 5-10　示例 5-10 的程序流程图

程序代码如下：

```
#include <stdio.h>
#include <math.h>
int main(void)
{
    int i,n,k;
    scanf(" % d",&n);
    k = sqrt(n);
    for(i = 2;i <= k;i++)
        if(n % i == 0) break;
    if(i > k)
```

```
        printf("%d是素数\n",n);
    else
        printf("%d不是素数\n",n);
    return 0;
}
```

程序运行结果如下：

```
17↵
17是素数
```

▶ 5.5.2 continue语句

continue语句不会终止循环，而是在本次循环体语句没有执行完的情况下，结束当前循环，提前进入下一次循环。

continue语句的一般格式如下：

```
continue;
```

【示例 5-11】 输出50～100不能被13整除的数，并且每行输出10个数。

【分析】 设整型变量i表示50～100的任意整数，整型变量count用来计数。

根据题意，使用循环结构实现。

重复执行以下操作：

判断整数i，若不能被13整数，则输出i，count加1，若count的值能被10整除，表明当前行已输出10个数，则输出换行符；若i的值能被13整除，则取i的下一个值继续判断。

程序的流程图如图5-11所示。

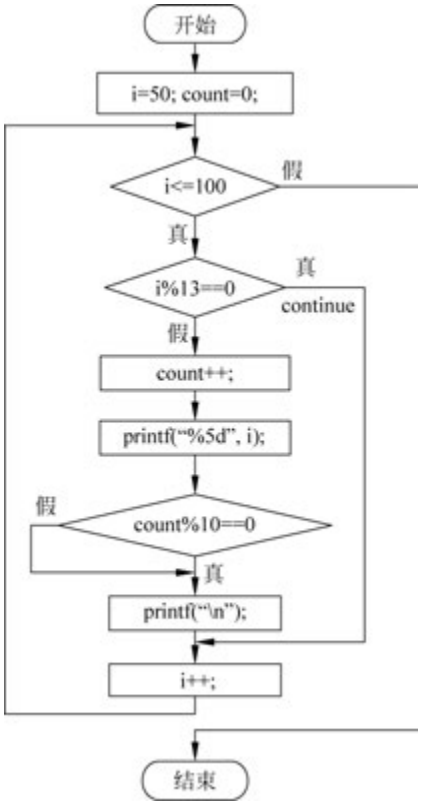

图 5-11 示例 5-11 的程序流程图

程序代码如下：

```
#include<stdio.h>
int main(void)
{
   int i,count=0;                    /*count用来计数*/
   for(i=50;i<=100;i++)
   {
      if(i%13==0)
         continue;                   /*能被3整除的数跳过*/
      count++;
      printf("%5d",i);
      if(count%10==0)                /*若count能被10整除,则输出换行符*/
         printf("\n");
   }
   return 0;
}
```

程序运行结果如下：

```
   50   51   53   54   55   56   57   58   59   60
   61   62   63   64   66   67   68   69   70   71
   72   73   74   75   76   77   79   80   81   82
   83   84   85   86   87   88   89   90   92   93
   94   95   96   97   98   99  100
```

程序中的循环体语句也可以使用以下语句：

```
if(i%13!=0)
{
   count++;
   printf("%5d",i);
   if(count%10==0)  printf("\n");
}
```

▶ 5.5.3　break 语句与 continue 语句的比较

break 语句和 continue 语句常常和 if 语句配合使用，达到控制循环的目的。

(1) break 语句可强行结束循环，转去执行循环语句的下一条语句。该语句能应用于循环语句和 switch 语句中。

(2) continue 语句可结束本次循环，继续进行下一次循环。对于 for 循环，可跳过循环体其余语句，转去执行表达式 3；对于 while 和 do…while 循环，可跳过循环体其余语句，转向循环条件的判定。该语句只能用于循环语句中。

(3) 循环嵌套时，break 语句和 continue 语句只影响包含它们的本层循环，与其他循环无关。

5.6　几种循环的比较

下面是 while、do…while 和 for 循环语句的比较。

(1) 通常情况下，三种循环语句都能够处理同一个问题，但当循环次数为已知时，使用 for 循环比较方便。

(2) while 循环和 for 循环都是先判断条件再执行循环体语句，因此，有可能一次也不执行

循环体语句；而 do…while 循环无论条件是否成立都会先执行一次循环体语句。

使用循环结构时要注意以下几方面的问题。

(1) 循环体语句为复合语句时，要使用花括号。

(2) 避免使程序发生无限循环。

(3) 循环嵌套时，内外层循环要使用不同的循环变量。

5.7 典型例题解析

【示例 5-12】 用公式$\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots$，计算 π 的近似值，直到某一项的绝对值小于 10^{-6} 为止。

【分析】 设变量 sum 表示 $1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots$的累加和，其初值为 0，则 π＝sum×4。本题属于累加问题，需要使用循环实现。

(1) 表达式中加数的特点。

第一项为正，第二项为负，……，即奇数项为正，偶数项为负；分母为连续的奇数：1，3，5，7，…。

(2) 定义变量。

item 表示所加的项，其初值为 1。i 表示分母，其初值为 1，其值的变化为 1，3，5，…。s 表示所加项的正负号，其初值为 1，即第一项的符号，其值的变化为 1，－1，1，－1，…。变量 pi 表示 π。

(3) 循环体语句为：

```
sum = sum + item;
i = i + 2;
s = - s;
item = s/i;
```

(4) 循环条件为某加数项的绝对值大于或等于 10^{-6}，即 fabs(item)>=1e－6。

程序代码如下：

```
#include <stdio.h>
#include <math.h>
int main(void)
{
   float sum,item,i,s,pi;
   item = 1; sum = 0; i = 1; s = 1;        /* 变量初始化 */
   while(fabs(item)> 1e - 6)
   {
      sum = sum + item;
      i = i + 2;                           /* 计算分母 */
      s = - s;                             /* 计算所加项的正负号 */
      item = s/i;                          /* 计算所加的项 */
   }
   pi = sum * 4;
   printf("pi = %f\n",pi);
   return 0;
}
```

程序运行结果如下：

```
pi = 3.141594
```

【示例 5-13】 从键盘输入一个 3×3 矩阵的数据，每个矩阵元素均为整数，求出对角线（从左下角到右上角）元素之和。

例如：输入

```
1 2 3
3 4 5
6 7 8
```

则输出为 13。

【分析】

（1）定义 3 个变量：i、j、sum，分别表示矩阵元素的行编号和列编号，以及和的值。

（2）对于 3×3 的矩阵，假设行编号和列编号都从 1 开始，则对角线元素的特点是行编号＋列编号的值等于 4，即求和元素满足的条件是 i＋j 的值等于 4。

使用循环嵌套实现求和。

程序代码如下：

```
#include <stdio.h>
int main(void)
{
   int i,j,x,sum = 0;
   for(i = 1;i <= 3;i++)
     for(j = 1;j <= 3;j++)
       {
          scanf("%d",&x);
          if(i + j == 4)
             sum = sum + x;
       }
   printf("%d\n",sum);
   return 0;
}
```

程序运行结果：

```
1 2 3↵
3 2 1↵
5 8 9↵
10
```

【示例 5-14】 小朋友们玩一种数字游戏，编程预测结果。游戏开始时，每个小朋友都会得到一个数字 n，且所有小朋友的初始分数均为 0。对当前持有的数字 n 执行以下判断与操作，每完成一次判断与数字更新，小朋友的分数就加 1。如果 n 是奇数，那么小朋友就会把它乘以 3 后再加 1；如果 n 是偶数，那么这个数就会被除以 2。当 n 的值等于 1 时，游戏结束，此时的分数就是这个小朋友在这局游戏中的最终得分。从键盘输入一个正整数 n，计算并输出小朋友的得分。

例如：

当 n＝5 时，游戏中数字变化的完整过程是：5→16→8→4→2→1，因此得分是 5。

【分析】

（1）设置两个整型变量 n、cnt，分别表示数字 n 和得分值。

（2）根据得分规则，当 n 的值不等于 1 时，重复执行以下语句。

```
if(n%2!=0)
   n=n*3+1;
else
   n=n/2;
cnt++;
```

程序代码如下：

```
#include<stdio.h>
int main(void)
{
   int n,cnt=0;
   printf("请输入数字 n:");
   scanf("%d",&n);
   while(n!=1)
   {
      if(n%2!=0)
         n=n*3+1;
      else
         n=n/2;
      cnt++;
   }
   printf("得分是：%d\n",cnt);
return 0;
}
```

程序运行结果如下：

```
请输入数字 n:112↵
得分是：20
```

【示例 5-15】 编程计算：a＋aa＋aaa＋…＋aa…a(n 个 a)的值，n 和 a 的值由键盘输入。

【分析】

(1) 表达式中所加项的特点为：从第二项开始，以后的每项等于前一项×10＋a。

(2) 可设置变量 sum 表示总和，item 表示当前所加项，则此后的每项 item＝item×10＋a，item 的初值为 a，n 为所加项的位数，n、a 由键盘输入。

程序代码如下：

```
#include<stdio.h>
int main(void)
{
   int n,i,a,sum=0,item;
   printf("input n:");
   scanf("%d",&n);
   printf("input a:");
   scanf("%d",&a);
   i=1;item=a;
   while(i<=n)
   {
      sum=sum+item;
      item=item*10+a;
      i++;
   }
   printf("%d\n",sum);
   return 0;
}
```

程序运行结果如下：

```
input n:3 ↵
input a:2 ↵
246
```

【示例 5-16】 从键盘输入一个正整数 x(100≤x≤500),输出小于 x 的三个最大的素数。

【分析】

(1) 定义整型变量 n 用来表示从 x－1 开始,依次递减的,需要判断是否为素数的整数。整型变量 count 表示素数的个数。

(2) 由于要求小于 x 的三个最大的素数,因此从 n＝x－1 开始,且 count 小于 3 的条件下重复以下操作。

判断 n 是否为素数,若是素数,则 count 加 1。

程序代码如下:

```
#include<stdio.h>
#include<math.h>
int main(void)
{
   int n,i,k,x,count=0;
   printf("请输入 x: ");
   scanf("%d",&x);
   printf("小于%d的三个最大的素数是: \n",x);
   for(n=x-1;n>=2;n--)
   {
      k=sqrt(n);
      for(i=2;i<=k;i++)
         if(n%i==0) break;
      if(i>k)
         {
            printf("%d\n",n);
            count++;
            if(count==3) break;
         }
   }
return 0;
}
```

程序运行结果如下:

```
请输入 x: 350 ↵
小于 350 的三个最大的素数是:
349
347
337
```

【示例 5-17】 输入一个正整数 n,再输入 n 个学生的成绩值,计算成绩的最大值、最小值和平均值。

【分析】

(1) 设置整型变量 score、max、min、sum,分别表示输入的成绩值、最大值、最小值、总和,浮点型变量 avg 表示成绩的平均值。

(2) 本题需要用循环结构实现。

重复执行以下操作 n 次:

输入一个成绩,累加成绩,判断最大值及最小值。

程序代码如下:

```
#include<stdio.h>
int main(void)
{
    int n,i;
    int score,max,min,sum=0;
    float avg;
    max=0;min=100;
    printf("请输入 n: ");
    scanf("%d",&n);
    printf("请输入 %d 个成绩值: \n",n);
    for(i=1;i<=n;i++)
    {
        scanf("%d",&score);
        sum=sum+score;
        if(max<score) max=score;
        if(min>score) min=score;
    }
    avg=1.0*sum/n;
    printf("max=%d,min=%d,avg=%f\n",max,min,avg);
return 0;
}
```

程序运行结果如下：

```
请输入 n: 7 ↵
请输入 7 个成绩值:
85 75 88 90 92 93 95 ↵
max=95,min=75,avg=88.285713
```

本章小结

本章主要介绍了 C 语言的循环结构语句 for、while、do…while，以及辅助控制语句 break 和 continue。

for 语句和 while 语句的执行特点是先进行循环条件判断，如果循环条件值第一次判断就为假，则循环体一次也不执行；而 do…while 语句是先执行循环体，然后判断循环条件，因此 do…while 语句至少执行一次循环体。

通常，这三种循环语句可以相互替代，除非循环条件第一次判断就为假。for 语句更适合处理循环次数确定的问题。这三种循环语句可以自由组合构成循环嵌套，在使用循环嵌套时，注意内外层循环要使用不同的循环变量，以免构成死循环。

辅助控制语句 break 用于无条件退出当前层循环结构，而 continue 语句用于结束本次循环，继续执行下一次循环。

习题 5

一、选择题

1. 在 while 循环中，while 后圆括号中表达式的值决定了循环体是否进行，因此，进入循环后，一定有能使此表达式的值变为(　　)的操作，否则，循环将会无限制地进行下去。

A. 0　　B. 1　　C. 成立　　D. 2

2. 在 do…while 循环中，while 表达式后的(　　)不能丢，它表示该循环的结束。

A. 0　　B. 1　　C. ；　　D. ，

3. 与语句 while(a)等价的语句是(　　)。

A. while(a＝＝0)　　B. while(a!＝0)

C. while(a!＝1)　　D. while(a＝1)

4. for 语句中的 3 个表达式可以部分或全部省略，但两个(　　)不可省略。

A. 0　　B. 1　　C. ；　　D. ，

5. 下面程序段的运行结果是(　　)。

```
int x = 23;
do{
   printf("%d",x--);
}while(!x);
```

A. 321　　B. 23　　C. 2322　　D. 不打印

6. 对于关系表达式 60≤x＜80，代码描述正确的是(　　)。

A. a <＝x < b　　B. x >＝60&&x < 80

C. x >＝60&&x < 80　　D. 60 <＝x||< 80

7. 执行语句“for(i＝1;i＋＋< 4;);”后，变量 i 的值是(　　)。

A. 3　　B. 4　　C. 5　　D. 不定

8. 以下描述正确的是(　　)。

A. continue 语句的作用是结束整个循环的执行

B. 只能在循环体内和 switch 语句内使用 break 语句

C. 在循环体内使用 break 语句和 continue 语句的作用相同

D. 从多层循环嵌套中退出时，只能使用 goto 语句

9. 有以下程序，程序的运行结果是(　　)。

```
#include < stdio.h >
void main()
{
   int x = 3;
   do
   {
        printf("%3d",x-= 2);
   }while(--x);
return 0;
}
```

A. 1　　B. 31　　C. 1 －2　　D. 死循环

10. 当输入“12345?”时，下面程序的运行结果是(　　)。

```
#include < stdio.h >
int main(void)
{
   char c;
   c = getchar();
   while((c = getchar())!= '?')
   putchar(++c);
return 0;
}
```

A. 12345　　B. 23456　　C. 3456　　D. 2345

二、填空题

小言年薪 N 万，准备在某市附近买一套房子，现在的价格是 200 万。假设房子价格以每年百分之 K 增长，但未来小言年薪不变，每年所得年薪全部积攒起来。从键盘输入 N 和 K 的值，如果在第 20 年或者之前能买下这套房子，则输出最早能买下房子的年数，否则输出 Impossible。

```
#include<stdio.h>
int main(void)
{
   float s,n,k,f=200;
   int i;
   scanf("%f%f",&n,&k);
   s=    ①    ;
   if(s>=200){
      printf("1年就可以买下房子\n");
      return 0;
   }
   for(i=2;i<=    ②    ;i++){
      f=    ③    ;
      s=s+n;
      if(s>=f){
          printf("%d年就可以买下房子\n",i);
          return 0;
      }
   }
   printf("Impossible!\n");
return 0;
}
```

三、改错题

求两个整数的最大公约数。

算法：用一个整数和另一个整数求余，然后取被除数除以余数，直到余数为 0 为止，此时的被除数即为最大公约数。

在每一行/********** ERROR **********/的下方有一处错误，请更正，不得改动程序的其他部分。

```
#include<stdio.h>
int main(void )
{
   int m,n;
   int r;
   printf("请输入两个整数 m, n: ");
/********** ERROR **********/
   scanf("%d %d",m,n);
/********** ERROR **********/
   r=m/n;
   while(r!=0)
   {
      m=n;
/********** ERROR **********/
      n=m;
      r=m%n;
   }
   printf("最大公约数是: %d",n);
return 0;
}
```

四、编程题

1. 输出所有的卡常数。卡常数是一个四位数，将它平分成两段，十位和个位组成一个新的数 a，千位和百位组成一个新的数 b，如果满足$(a+b)^2$ 等于这个数本身，这个数就是一个卡常数。

例如，3025 被拆成 30 和 25 两个数，$(30+25)^2=3025$，所以 3025 是卡常数。

2. 计算下列表达式的值并输出。

$$s=\left(1+\frac{1}{2}\right)+\left(\frac{1}{3}+\frac{1}{4}\right)+\cdots+\left(\frac{1}{2*n-1}+\frac{1}{2*n}\right)$$

若输入 n=10(n 的值为 1～100)，则输出结果为 s=3.597739。

3. 从键盘输入一个正整数 n，输出由 * 构成的等腰三角形图案。

例如，输入 5，输出如图 5-12 所示的图案。

图 5-12　第 3 题要输出的图案

4. 从键盘输入一个正整数 n(n≥100)，输出不大于 n 的最大完全数。一个数如果等于它不包括自身的因子之和，这个数就是完全数。例如，6 的因子为 1、2、3，且 6=1+2+3，因此，6 就是完全数。

5. 求斐波那契数列的前 40 项。斐波那契数列是：1，1，2，3，5，8，13，…。该数列有如下特点：前两项分别为 1、1，从第 3 项开始，依次等于其前面两项之和。