第 6 章　循环结构程序设计

在实际应用中，经常会遇到需要重复处理相同或相似操作的情况，例如，连续生成 50 个随机整数、求 40 个整数之和等。为了有效地描述这种相同或相似操作的重复执行，C 语言提供了循环语句。

本章首先介绍了 while 循环语句、do-while 循环语句和 for 循环语句三种循环结构，并且对这三种循环结构进行了比较；然后介绍了循环嵌套和转移语句；最后通过实例，让读者进一步掌握循环结构程序设计的基本思想。

学习目标

- 掌握 while 循环语句的使用方式
- 掌握 do-while 循环语句的使用方式
- 掌握 for 循环语句的使用方式
- 掌握三种循环语句的区别以及嵌套使用方式
- 掌握 break 和 continue 语句的应用，了解 goto 语句的使用方法
- 掌握循环结构程序的编写和应用

6.1　while 循环语句

试讲章节：6.1、6.2、6.3

在 C 语言所有循环语句中，while 语句是最简单也是最基本的。while 语句的语法格式为：

```
while(表达式)      语句          /*循环体*/
```

圆括号内的表达式是控制表达式，圆括号右边的语句是循环体，循环体可以是一条简单的语句，也可以是多条语句组成的复合语句（用花括号括起来）。

while 语句的执行流程如图 6-1 所示。执行该语句时，先计算表达式的值，如果它为真，则执行循环体；接着再次判定表达式的值，如果它仍为真，继续执行循环体，否则循环结束，执行 while 语句后的下一条语句。

提　示

while 语句是“先判断，后执行”。如果刚进入循环时条件就不满足，则循环体一次也不执行。还需要注意的是，一定要有语句修改表达式的值，使其有结果为 0 的时候，否则将出现“死循环”。

在 while 语句中没有包含设置初始状态的功能，因此这一工作需要在 while 语句之前完成。对于循环相关状态的修改是在循环体中完成，因此除了少数特殊情况下，while 语句的循环体一般都是复合语句。

【例 6-1】　输入整数 n 的值，求 S=1+2+3+…+n 的值。

【问题分析】　这是一个累加的问题，需要先后将 n 个数相加。要重复进行 n 次加法运算，因此，可以用循环结构来实现。重复执行循环体 n 次，每次加一个数。继续分析发现每次累加的数是有规律的，后一个数是前一个数加 1。因此，只须在加完上一个数 i 后，使 i 加 1 就可得到下一个数。具体流程如图 6-2 所示。

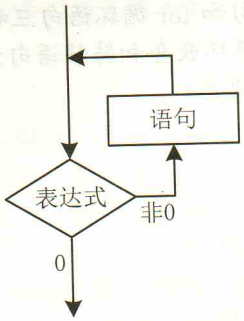

图 6-1　while 语句流程图

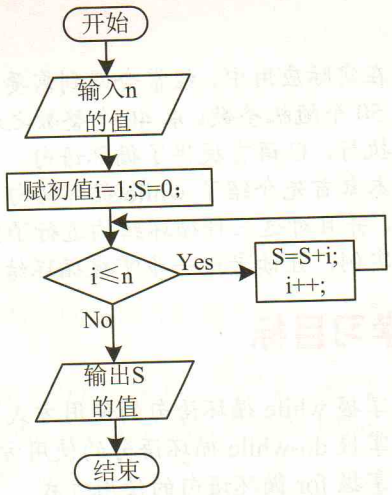

图 6-2　例 6-1 流程图

【参考代码】

```
#include <stdio.h>
int main()
{
    int i,n,S;                  /*定义变量*/
    printf("请输入 n 的值:");    /*输出提示语*/
    scanf("%d",&n);             /*输入 n 的值*/
    i=1;                        /*给 i 赋初值*/
    S=0;                        /*给 S 赋初值*/
    while(i<=n)                 /*循环，当 i>n 时结束*/
    {
        S+=i;                   /*求和，将结果放入 S 中*/
        i++;                    /*循环控制变量 i 加 1*/
    }
    printf("S=%d\n",S);         /*输出 S 的值*/
    return 0;
}
```

【运行结果】　程序运行结果如图 6-3 所示。

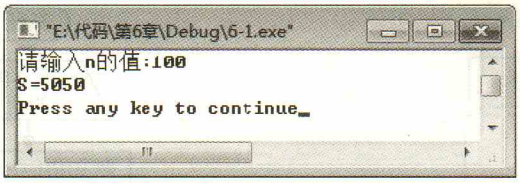

图 6-3 例 6-1 程序运行结果

【程序说明】 在这段代码中，循环开始时初始状态的设置是由变量 i 和 S 的初始化操作来完成的。循环的执行条件是 i<=n，在满足这一条件的情况下，i 的值被累加到变量 S 中，然后由语句 i++修改循环控制变量 i 的值。当 while 语句执行完毕后，变量 S 中就保存了从 1 到 n 的 n 个自然数的累加结果。

在使用 while 语句时，需要注意两点：

（1）变量初始化描述要完整、准确。例如，在上面的例子中，在 while 语句前要对变量 i 和 S 进行初始化。

（2）在循环体中应有使循环趋向于结束的语句。例如，在例 6-1 中循环结束的条件是“i>n”，因此，在循环体中应该有使 i 增值并最终大于 n 的语句，这里用“i++;”语句来达到此目的，如果没有这条语句，则 i 的值始终不变，就形成了死循环。

6.2 do-while 循环语句

除了 while 语句以外，C 语言还提供了 do-while 语句来实现循环结构。有些情况下，无论条件是否满足，都至少执行一次循环体，这时可以使用 do-while 语句，其语法格式为：

```
do
    语句                          /*循环体*/
while(表达式);
```

do-while 语句的执行流程如图 6-4 所示。首先执行循环体中的语句一次，然后计算表达式的值，若为真（非 0）则继续执行循环体，并再计算表达式的值，当表达式的值为假（0）时，终止循环，执行 do-while 语句后的下一条语句。

提 示

do-while 语句中，条件放在 while 后面的圆括号中，并且最后必须加上一个分号，这是很多初学者容易遗漏的。

【例 6-2】 用 do-while 语句求 S=1+2+3+…+n 的值。

【问题分析】 解题思路与例 6-1 相似，只是用 do-while 语句，就是用直到型循环结构来实现，流程图如图 6-5 所示。

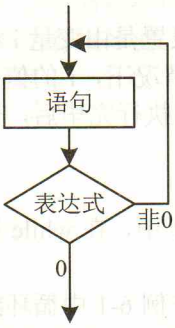

图 6-4　do-while 语句流程图

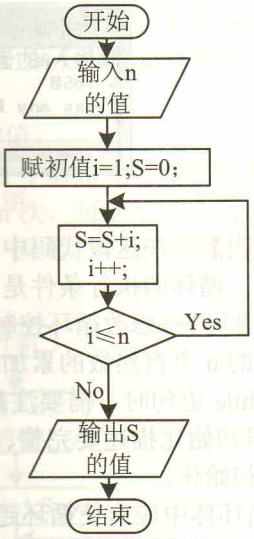

图 6-5　例 6-2 流程图

【参考代码】

```
#include <stdio.h>
int main()
{
    int i,n,S;                  /*定义变量*/
    printf("请输入 n 的值:");    /*输出提示语*/
    scanf("%d",&n);             /*输入 n 的值*/
    i=1;                        /*给 i 赋初值*/
    S=0;                        /*给 S 赋初值*/
    do                          /*开始循环*/
    {
        S+=i;                   /*求和*/
        i++;                    /*循环控制变量 i 加 1*/
    }while(i<=n);               /*当 i>n 时结束*/
    printf("S=%d\n",S);         /*输出 S 的值*/
    return 0;
}
```

【运行结果】　程序运行结果如图 6-6 所示。

从例 6-1 和例 6-2 中可以看到，对同一个问题既可以用 while 语句实现也可以用 do-while 语句实现。do-while 语句结构可以转换成 while 结构，如图 6-7 所示，图 6-4 与图 6-7 完全是等价的，而图 6-7 中虚线框部分就是一个 while 结构。可见，do-while 结构是由一个“语句”加一个 while 结构构成的。

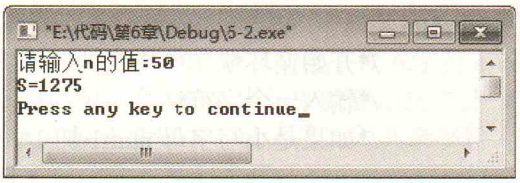

图 6-6 例 6-2 程序运行结果

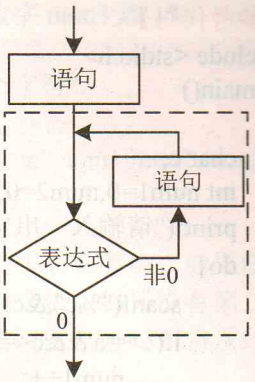

图 6-7 do-while 转换成 while 流程图

【例 6-3】 输入一串字符，以“#”结束，输出其中小写字母个数、大写字母个数以及数字个数。

【问题分析】 输入字符包括字母（A～Z，a～z），数字（0～9）和其他符号（如+、-、%等）。我们只统计其中的小写字母个数、大写字母个数和数字个数。

用变量 c 表示输入字符，并说明为字符类型。用 num1、num2 和 num3 分别表示小写字母个数、大写字母格式和数字个数，并说明为整型。

首先读入一个字符，判断它是否为小写字母，是则将小写字母个数加 1；否则，判断它是否是大写字母，是则将大写字母个数加 1；否则，判断它是否为数字，是则将数字个数加 1。当输入字符不是“#”时，应重复执行循环。直到输入“#”时结束循环，输出统计结果。执行流程如图 6-8 所示。

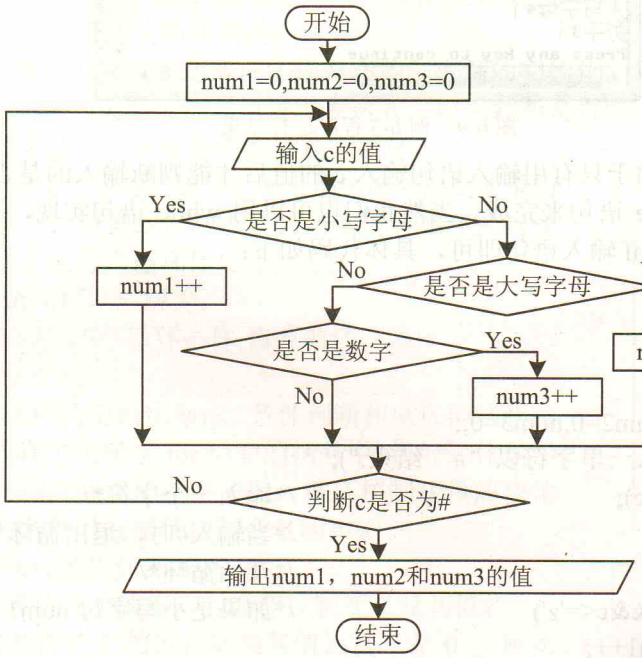

图 6-8 例 6-3 流程图

【参考代码】

```
#include <stdio.h>
int main()
{
    char c;
    int num1=0,num2=0,num3=0;;
    printf("请输入一串字符以“#”结束:");
    do{                                          /*开始循环*/
        scanf("%c",&c);                          /*输入一个字符*/
        if(c>='a'&&c<='z')                       /*如果是小写字母 num1 加 1*/
            num1++;
        else if(c>='A'&&c<='Z')                  /*如果是大写字母 num2 加 1*/
            num2++;
        else if(c>='0'&&c<='9')                  /*如果是数字 num3 加 1*/
            num3++;
    }while(c!='#');                              /*当输入#时，退出循环*/
    printf("小写字母:%d 个\n 大写字母%d 个\n 数字%d 个\n",num1,num2,num3);
    return 0;
}
```

【运行结果】 程序运行结果如图 6-9 所示。

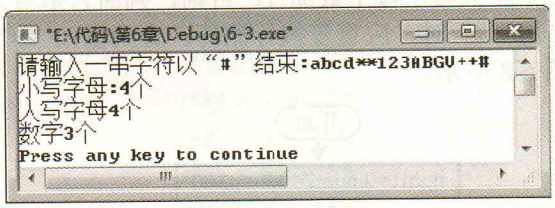

图 6-9 例 6-3 程序运行结果

【程序说明】 由于只有用输入语句输入 c 的值后才能判断输入的是否为“#”，所以本题更适合用 do-while 语句来完成。当然我们也可以用 while 语句实现，只需要在 while 循环语句前加一句 scanf 输入语句即可，具体代码如下：

```
#include <stdio.h>
int main()
{
    char c;
    int num1=0,num2=0,num3=0;;
    printf("请输入一串字符以“#”结束:");
    scanf("%c",&c);                              /*输入一个字符*/
    while(c!='#')                                /*当输入#时，退出循环*/
    {                                            /*开始循环*/
        if(c>='a'&&c<='z')                       /*如果是小写字母 num1 加 1*/
            num1++;
        else if(c>='A'&&c<='Z')                  /*如果是大写字母 num2 加 1*/
            num2++;
```

```
        else if(c>='0'&&c<='9')                    /*如果是数字 num3 加 1*/
            num3++;
        scanf("%c",&c);                            /*输入一个字符*/
    }
    printf("小写字母:%d 个\n 大写字母%d 个\n 数字%d 个\n",num1,num2,num3);
    return 0;
}
```

在程序的两个不同的地方安排了读入字符 c 的语句，这是很有必要的。如果没有第一个读入 c 的语句，在 while 语句中的表达式“c!='#'”就无法确定值。如果没有第二个读入 c 的语句（它放在循环中）就无法读入其余字符，循环也无法结束，因为 c 将永远是第一次读入的字符，它不等于“#”。

6.3 for 循环语句

在 C 语言中，除了可以用 while 和 do-while 语句实现循环外，还可以用 for 语句实现循环。

6.3.1 for 语句的一般形式

对循环状态的初始化和对循环控制变量的修改是循环语句中必不可少的两个组成部分。for 语句将这两部分作为表达式写入到圆括号中，更便于描述、阅读和检查程序，其语法格式为：

```
for(表达式 1;表达式 2;表达式 3)
    语句                                    /*循环体*/
```

圆括号中的三个表达式的作用是：

表达式 1：通常为赋值表达式，实现循环控制变量的初始化，只执行一次。

表达式 2：通常为关系表达式或逻辑表达式，用来判断是否继续循环。在每次执行循环体前先执行此表达式，决定是否继续执行循环。

表达式 3：通常为表达式语句，用来描述循环控制变量的变化，多数情况下为自增或自减表达式，实现对循环控制变量的修改。它是在执行完循环体后才执行的。

因此，for 语句可以理解为：

```
for(循环变量赋初值;循环条件;修改循环变量)
    语句
```

for 语句把循环的初始化操作、条件判断和循环控制状态的修改都一并放在了关键字 for 后面的括号中，很好地体现了正确表达循环结构应注意的三个问题：循环控制变量的初始化、循环控制的条件以及循环控制变量的更新。

for 语句的执行流程如图 6-10 所示。

（1）计算表达式 1 的值，为循环控制变量赋初值。

（2）计算表达式 2 的值，如果其值为真（非 0），则执行循环体语句，然后执行第（3）步。如果为假（0），则退出

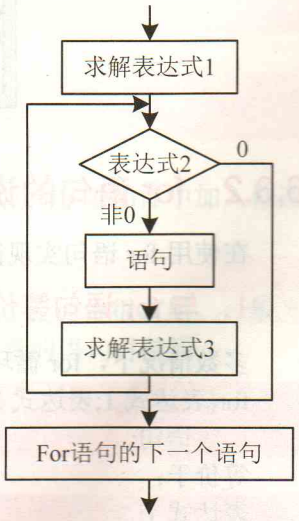

图 6-10 for 语句执行流程图

循环，执行 for 循环后的语句。

（3）计算表达式 3 的值，调整循环控制变量的值。

（4）返回执行第（2）步，重新计算表达式 2 的值，依此重复过程，直到表达式 2 的值为假（0）时，退出循环。

例如：

```
for(i=1;i<=10;i++)
    语句;
```

先给 i 赋初值 1，判断 i 是否小于等于 10，若是则执行语句，之后 i 的值增加 1。再重新判断 i 是否小于等于 10，直到条件为假，即 i>10 时，结束循环。

【例 6-4】 用 for 语句求 S=1+2+3+…+n 的值。

【参考代码】

```
#include <stdio.h>
int main()
{
    int i,n,S;                      /*定义变量*/
    printf("请输入 n 的值:");        /*输出提示语*/
    scanf("%d",&n);                 /*输入 n 的值*/
    S=0;                            /*给 S 赋初值*/
    for(i=1;i<=n;i++)               /*循环，当 i>n 时结束*/
        S+=i;                       /*求和，将结果放入 S 中*/
    printf("S=%d\n",S);             /*输出 S 的值*/
    return 0;
}
```

【运行结果】 程序运行结果如图 6-11 所示。

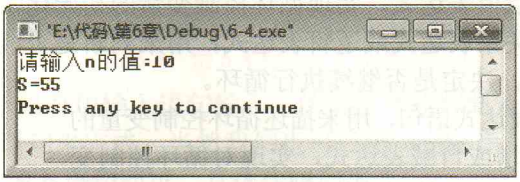

图 6-11 例 6-4 程序运行结果

6.3.2 for 语句的说明

在使用 for 语句实现循环时需要说明以下几点。

1. 与 for 语句等价的 while 语句形式

多数情况下，for 循环结构可以用等价的 while 循环表示。for 语句的一般形式为：

```
for(表达式 1;表达式 2;表达式 3)
    语句;
```

等价于：

```
表达式 1;
while(表达式 2)
```

```
{
    语句;
    表达式 3;
}
```

2. 在 for 语句中省略表达式

for 循环中的“表达式 1”“表达式 2”和“表达式 3”都是可选项，即可以缺省，但表达式之间的分号“;”绝对不能缺省。

（1）省略“表达式 1”，即不设置初值，语句格式为：

```
for(;表达式 2;表达式 3)
```

应该注意的是，由于省略了“表达式 1”，没有对循环变量赋初值，因此，为了能正常执行循环，应在 for 语句之前给循环变量赋以初值。即“表达式 1”可以写在 for 语句结构的外面。

例如：

```
n=1;
for(;n<100;n++)
    语句;
```

它等价于

```
for(n=1;n<100;n++)
    语句;
```

一般使用这种格式的原因是：循环控制变量的初值不是已知常量，而是需要通过前面语句的执行计算得到。

（2）省略“表达式 2”，表示不用判断循环条件是否成立，循环条件总是满足的，则如果不做其他处理，便成了死循环。语句格式为：

```
for(表达式 1;;表达式 3)
```

等价于：

```
while(1)
```

例如：

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

循环无终止地进行，i 的值不断加大，s 的值也不断累加。

（3）省略“表达式 3”，则不对循环控制变量进行操作，这时可在语句体中加入修改循环控制变量的语句。语句格式为：

```
for(表达式 1;表达式 2;)
```

C 语言允许在循环体内改变循环控制变量的值，这在某些程序设计中很有用。一般当循环控制变量呈非规则变化，并且在循环体中有更新循环控制变量的语句时使用。

例如：

```
for(n=1;n<=100;)
{
    …
    n=4*n-1;
    …
```

```
}
```

循环控制变量的变化为：1，3，11，43，…

（4）省略 3 个表达式，语句格式为：

```
for(;;)
```

这是一个无限循环语句，与 while(1)的功能相同，一般处理方法是：在循环体内的适当位置，利用条件表达式与 break 语句的配合中断循环，即当满足条件时，用 break 语句跳出 for 循环。

例如：

```
for(;;)
{
    …
    if(x==0) break;
    …
}
```

表示当 x 等于 0 时，使用 break 语句退出循环。

3．在 for 语句中省略语句

for 语句的循环体可以是空语句，表示当循环条件满足时进行空操作。一般用于延时处理。语句格式为：

```
for(表达式 1;表达式 2;表达式 3);
```

例如：

```
for(i=1;i<=20000;i++);
```

表示循环变量空循环了 20000 次，占用了一定的时间，起到了延长时间的效果。

4．在 for 语句中逗号表达式的应用

在 for 语句中，表达式 1 和表达式 3 可以是一个简单的表达式，也可以是逗号表达式，即包含一个以上的简单表达式，中间用逗号间隔，语句格式为：

```
for(逗号表达式 1;表达式 2;逗号表达式 3)
```

例如：

```
for(n=1,m=100;n<m;n++,m--)
    s=n+m;
```

其中：表达式 1 同时为 n 和 m 赋初值，表达式 3 同时改变 n 和 m 的值。表示循环可以有多个控制变量，但是，逗号表达式可以与循环变量有关，也可以与循环变量无关。例如：

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

s 的赋初值可以写在 for 语句之前，也可以写在表达式 1 中。

for 结构不是狭义上的计数式循环，是广义上的循环结构，它不仅能进行已知循环次数的循环，也能够处理循环次数未知的情况。

【例 6-5】　编程实现，输出 1～1000 之间能同时被 3、5、7 整除的数。

【问题分析】　这是一题已知循环次数的情况，可以设循环变量 i 从 1 到 1000 进行循环，然后判断每个自然数 i 是否能够同时被 3、5、7 整除，将满足条件“i%3==0&&i%5==0&&i%7==0”的数进行输出，执行流程如图 6-12 所示。

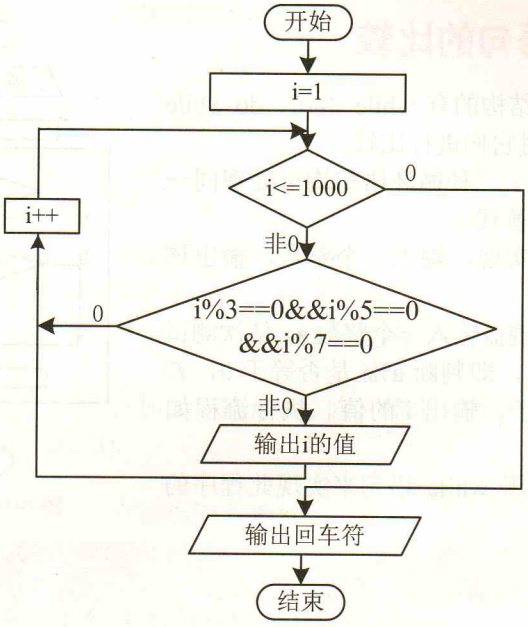

图 6-12 例 6-5 执行流程图

【参考代码】

```
#include<stdio.h>
int main()
{
    int i=1;                                    /*定义变量 i*/
    for(i=1;i<=1000;i++)                        /*i 从 1 到 1000 循环 1000 次*/
    {
        if(i%3==0&&i%5==0&&i%7==0)              /*如果满足条件则输出 i 的值*/
            printf("%4d",i);
    }
    printf("\n");                               /*输出回车符*/
    return 0;
}
```

【运行结果】 程序运行结果如图 6-13 所示。

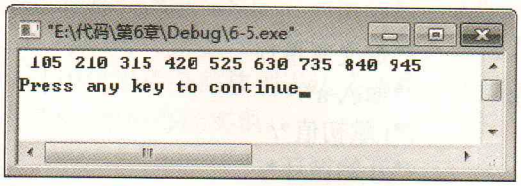

图 6-13 例 6-5 程序运行结果