5.2　循环结构的概述

循环结构（*iteration*）是高级程序设计语言三大控制结构之一，其特点是能够重复执行相同的代码。图 5-1 是循环结构的流程图。从图中可以清晰地看到一个回路的存在。

循环结构用程序设计语言中的循环语句来实现。一般情况下，循环语句都包含一个测试条件和一个执行部分（称为“**循环体**”）。当测试条件为真时，循环体将会被反复执行，直到条件不成立为止。如果条件一直为真，那么这个循环将会无休止地进行下去，可能会将计算机资源（特别是 CPU 资源）耗尽，从而使别的程序不能流畅运行。这种无限的循环称为“**死循环**”。因此，在循环语句的执行部分中，一定会有一些修改测试条件的语句，使循环能在有限次的运行后结束。这样的循环是“**有出口的**”，是一种良好的设计。

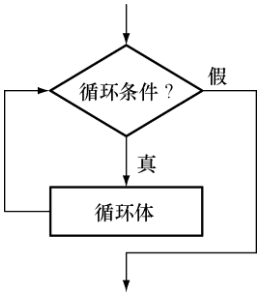

图 5-1　循环结构的流程图

要使循环正确运行的另一个条件是：循环必须从一个确定的起点开始。这称为循环的“**初始化**”。没有起点的循环可能会从一个随机的起点开始，从而导致不正确的结果。

一般地，一种程序设计语言中的循环语句有多种，但都分为两大类：当型和直到型。当型循环的结构是当测试条件成立时循环；直到型循环的结构是执行循环，直到条件不成立。C 语言的 3 种循环语句都属于当型循环。

5.3　while 语句

while 语句是 C 语言中较常使用的循环形式之一，它特别适合已知循环条件、却不明确循环的具体次数的情况。

5.3.1　while 语句的语法

当程序员明确知道循环初始条件，以及循环次数可以为 0 次的情况下，可以使用 while 循环，其语法如下：

```
while(表达式)
{
    循环体
 }
```

while 语句的功能是：当表达式的值为真（非 0 值），执行一遍循环体语句，然后再计算表达式的值，当值为真，则继续执行循环体语句，并以此类推；如果表达式的值为假（0 值），则退出循环，执行紧跟在 while 后面的语句。特别地，当第一次表达式的值为假时，则直接退出循环，而不会做循环体语句。图 5-2 是 while 语句的流程图。

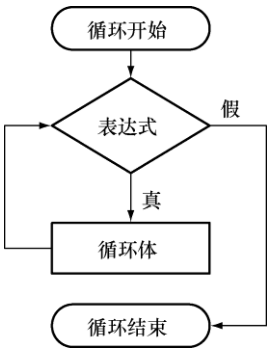

图 5-2　while 语句的流程图

while 语句的条件，表达式中往往包含一个变量，用于控制循环是否继续。这个变量称为“**循环控制变量**”，一般用于计数。当计数达到指定数量时，循环结束。

【例 5-1】求自然数 1 到 10000 的和。

【解题思路】题目是一种典型的级数求和问题。现实中求和常使用的方法是累加。所谓累加，

就是设定一个中间变量，然后依次将级数的每一项都加到这个中间变量上。当级数的最后一项加完后，中间变量的值就是累加结果。累加的具体操作描述如下。

首先令中间变量 sum = 0，然后依次做如下累加操作：

sum ← sum + 1；(符号“←”的含义是将右面的结果存回到左面的变量中)

sum ← sum + 2；

sum ← sum + 3；

……

sum ← sum + 10000；

最后的 sum 即是所求的累积和。

从上面的操作中可以清楚地看到每一步完成的加法操作是完全相同的，只是参与操作的数不同。因此，上述的累加操作可以用循环来解决。可以制定这样的算法：

(1) 令 sum = 0，i = 1;

(2) 如果 i> 10000，转到第 6 步；

(3) 否则，sum += i；

(4) ++i；

(5) 转到第 2 步；

(6) 结束。

上述算法的流程图如图 5-3 所示。

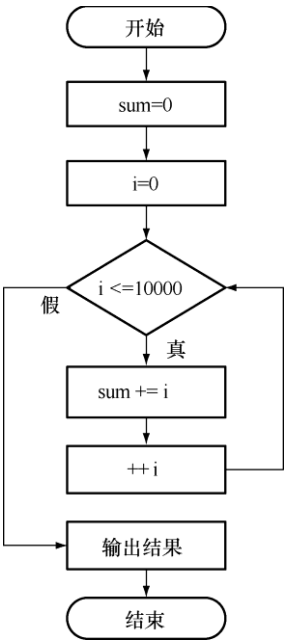

图 5-3　例 5-1 的流程图

根据上述算法写成的程序如下所示。

```
//5-1.c
#include<stdio.h>

int main()
{
     int i = 1, sum = 0;

     while (i<=10000)
     {
          sum += i;
          ++i;
     }

     printf("1 到 10000 的和=%d\n",sum);

     return 0;
}
```

程序的运行结果如下：

```
1 到 10000 的和=50005000
```

5.3.2　死循环

当 while 的条件为永真时就构成了一个死循环，例如，

```
while (1) { … }
```

甚至简单到

```
while (1);
```

死循环常常为某些特定的应用而设，这里提醒读者尽量避免死循环的出现。

> 初学者使用 while 语句时常犯一些错误。
>
> 1. 存在多余的分号。

```
i = 0;
while (i< 100); //请注意这个分号的存在
{…}
```

while 后面的分号将其后的复合语句排除在循环体之外，而真正的循环体只是一条空语句。这样一来，由于循环控制变量 i 在循环中不会改变，因此不可避免地造成死循环。

2．在循环体中没有改变循环控制变量的值，从而造成死循环。例如：

```
i = 0; sum = 0;
while (i< 100)//i 的值永远是 0，因此循环不会终止
{sum += i;}
```