第 5 章

循环结构程序设计

5.1 程序中需要用循环结构

用顺序结构和选择结构可以解决简单的、不出现重复的问题。但是在现实生活中许多问题是需要进行重复处理的。例如，计算一个学生 5 门课的平均成绩很简单，只需要把 5 门课的成绩相加，然后除以 5 即可。如果需要得到一个班 50 个学生每人的平均成绩，就要做 50 次“把 5 门课的成绩相加，然后除以 5”的工作，如果在程序中重复写 50 次相同的程序段显然是不胜其烦的。类似的问题是很多的，如工厂各车间的生产日报表，全国各省、市、自治区的人口统计分析，各大学招生情况统计，学校教职工工资报表等。

事实上，绝大多数的应用程序都包含重复处理。**循环结构就是用来处理需要重复处理的问题的，所以，循环结构又称为重复结构**。

有两种循环：一种是无休止的循环，如地球围绕太阳旋转，永不终止；每天 24 小时，周而复始。另一种是有终止的循环，达到一定条件循环就结束了，如统计完第 50 名学生成绩后就不再继续了。计算机程序只处理有条件的循环，算法的特性是有效性、确定性和有穷性，如果程序永远不结束，是不正常的。

要构成一个有效的循环，应当指定两个条件：①需要重复执行的操作，这称为**循环体**；②**循环结束的条件**，即在什么情况下停止重复的操作。

循环结构是结构化程序设计的基本结构之一，它和顺序结构、选择结构共同作为各种复杂程序的基本构造单元。因此熟练掌握选择结构和循环结构的概念及使用是程序设计的最基本的要求。

C 语言提供了几种能直接实现循环结构的语句，主要有 while 语句、do…while 语句和 for 语句，用起来很方便。下面分别介绍。

5.2 用 while 语句和 do…while 语句实现循环

5.2.1 用 while 语句实现循环

先看一下利用循环的例子。

例 5.1　求 1＋2＋3＋…＋100，即 $\sum_{n=1}^{100} n$。

解题思路：

对此问题可以有不同的求解方法，有的人用心算，把它转换成 50 组头尾两数之和：(1＋100)＋(2＋99)＋(3＋98)＋…＋(49＋52)＋(50＋51)，每个括号内的值都是 101，一共有 50 对括号，所以总和是 50×101，很容易得出 5050。这是适宜于心算的算法。

用计算机算题，计算机是不会按上面的方法自动分组的，而必须事先由人们设计计算的方法。对于这样简单的问题去设计巧妙的算法是没有必要的。计算机的最大特点是快，所以适宜用最“笨”的办法去处理一些简单的问题，就是采取一个一个数累加的方法，从 1 加到 100。对于人来说，这是“笨”办法，对于计算机来说却是“好”办法。

用传统流程图和 N-S 结构流程图表示从 1 加到 100 的算法，见图 5.1(a)和图 5.1(b)。其思路是：变量 sum 是用来存放累加值的，sum 的初值设为 0，i 是准备加到 sum 的数值，让 i 从 1 变到 100，先后累加到 sum 中。具体步骤如下：

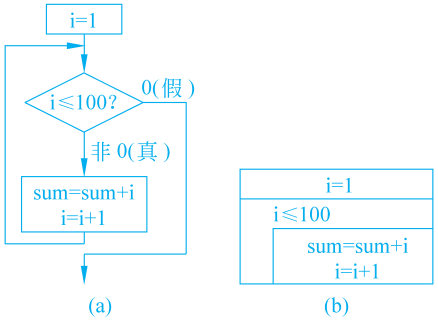

图　5.1

(1) 开始时使 sum 的值为 0，被加数 i 第一次取值为 1。开始进入循环结构。

(2) 判别“i≤100”条件是否满足，由于 i 小于 100，因此“i≤100”的值为真。所以应当执行其下面矩形框中的操作。

(3) 执行 sum＝sum＋i，此时 sum 的值变为 1 了，然后使 i 的值加 1，i 的值变为 2 了，这是为下一次加 2 做准备。流程返回菱形框。

(4) 再次检查“i≤100”条件是否满足，由于 i 的值为 2，小于 100，因此“i≤100”的值仍为真，所以应执行其下面矩形框中的操作。

(5) 执行 sum＝sum＋i，由于 sum 的值已变为 1，i 的值已变为 2，因此执行 sum＝sum＋i 后 sum 的值变为 3。再使 i 的值加 1，i 的值变为 3。流程再返回菱形框。

(6) 再次检查“i≤100”条件是否满足……如此反复执行矩形框中的操作，直到 i 的值变成了 100，把 i 加到 sum 中，然后 i 又加 1 变成 101 了。当再次返回菱形框检查“i≤100”条件时，由于 i 已是 101，大于 100，“i≤100” 的值为假，不再执行矩形框中的操作，循环结构结束。

编写程序：

```
#include <stdio.h>
int main()
  {int i,sum=0;                    //sum 是用来存放累加和的变量,初值为 0
   i=1;
   while (i<=100)                  //当 i 小于或等于 100 时,执行下面花括号中的复合语句
  { sum=sum+i;                     //将 i 的当前值累加到变量 sum 中
    i++;                           //使 i 的值加 1
```

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

运行结果：

```
5050
```

从上面的程序可以看到怎样用 while 语句去实现循环。while 语句的一般形式如下：

```
while (表达式) 语句
```

当表达式为非 0 值(代表逻辑值“真”)时，执行 while 语句中的内嵌语句(如程序中花括号中的复合语句)，其流程图见图 5.2。

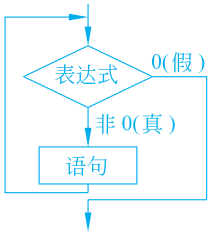

图 5.2

while 循环的特点是：**先判断表达式，后执行循环体(即内嵌语句)**。

注意：

(1) 循环体如果包含一个以上的语句，应该用花括号括起来，以复合语句形式出现。如果不加花括号，则 while 语句的范围只到 while 后面第一个分号处。例如，本例中 while 语句中如无花括号，则 while 语句范围只到“sum=sum+i;”。

(2) 在循环体中应有使循环趋向于结束的语句。例如，在本例中循环结束的条件是“i>100”，因此在循环体中应该有使 i 增值以最终导致 i>100 的语句，现用“i++;”语句来达到此目的。如果无此语句，则 i 的值始终不改变，循环永不结束。

请读者考虑：如果 while 语句中的条件改为“i<100”，情况会怎样？输出结果是什么？

5.2.2 用 do…while 语句实现循环

do…while 语句的特点是**先执行循环体，然后判断循环条件是否成立**。其一般形式为：

```
do
    循环体语句
while (表达式);
```

它是这样执行的：先执行一次循环体语句，然后判别“表达式”，当表达式的值为非 0(真)时，返回重新执行循环体语句，如此反复，直到表达式的值等于 0(假)为止，此时循环结束。可以用图 5.3 表示其流程。请注意 do…while 循环用 N-S 流程图的表示形式(见图 5.3(b))。

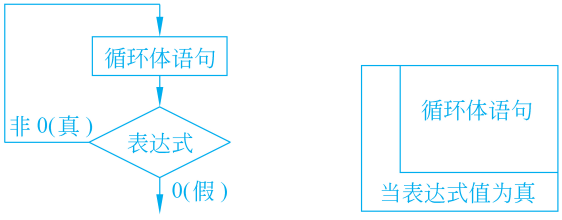

图 5.3

同一个问题既可以用 while 循环处理，也可以用 do…while 循环来处理。二者是可以互相转换的。

例 5.2 用 do…while 循环求 1＋2＋3＋…＋100，即 $\sum_{n=1}^{100} n$。

解题思路：

画出流程图，见图 5.4。

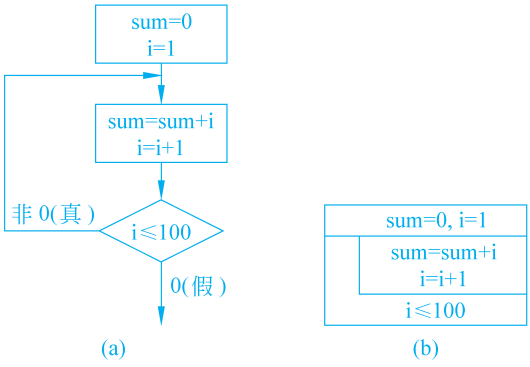

图 5.4

编写程序：

```
#include <stdio.h>
int main()
  { int i,sum=0;
    i=1;
    do                                    //在循环开始时不检查条件,先执行一次循环体
      {sum=sum+i;
```

```
        i++;
      }while(i<=100);
   printf("%d\n",sum);
   return 0;
  }
```

运行结果：

```
5050
```

可以看到，结果和例 5.1 完全相同。

例 5.3 若要募集慈善基金 10000 元，有若干人捐款，每输入一个人的捐款数后，计算机就输出当时的捐款总和。当某一次输入捐款数后，总和达到或超过 10000 元时，即宣告结束，输出最后的累加值。

解题思路：

解此题的思路是设计一个循环结构，在其中输入捐款数，求出累加值，然后检查此时的累加值是否达到或超过预定值，如果达到了，就结束循环操作。

编写程序：

```
#include <stdio.h>
int main()
  {float amount,sum=0;                    //变量 sum 用来存放累加值
    do
       {scanf("%f",&amount);              //输入一个捐款金额
        sum=sum+amount;                   //求出当前的累加值
       }while(sum<10000);                 //如未达 10000 元继续循环
    printf("sum=%9.2f\n",sum);
    return 0;
}
```

运行结果：

```
1000↙                                     (输入捐款额)
1850↙
1500↙
2600↙
2500↙
1200↙
sum=10650.00
```

程序分析：

此题与前面的不同，事先不知道要执行多少次循环，只给出循环的条件(sum＜10000)，每次循环结束时检查此条件是否满足，当某一次 sum 已超过 10000 元时，不再继续执行循环体。

提示：设计循环结构，要考虑两个问题：一是循环体，二是循环结束条件。注意 while 循环中判断的条件是循环继续的条件，而不是结束条件。在上例中循环继续的条件是

sum＜10000，也就是结束循环的条件是 sum＞＝10000。千万不要错写成 while(sum＞10000)。

5.3 用 for 语句实现循环

用 while 语句可以实现循环结构，但是它必须明确地给出继续执行循环的条件(如 sum＜10000)，而在许多情况下，人们给出的往往是执行循环的次数，如统计 100 人的平均工资，求一个学生 5 门课的总成绩等。用 C 语言中的 for 语句更为灵活方便，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。

5.3.1 for 语句的一般形式和执行过程

for 语句的一般形式为

```
for(表达式 1;表达式 2;表达式 3) 语句
```

它的执行过程如下：

(1) 求解表达式 1。

(2) 求解表达式 2，若其值为真(值为非 0)，则执行 for 语句中指定的内嵌语句，然后执行下面第(3)步。若为假(值为 0)，则结束循环，转到第(5)步。

(3) 求解表达式 3。

(4) 转回第(2)步继续执行。

(5) 循环结束，执行 for 语句下面的一个语句。

可以用图 5.5 来表示 for 语句的执行过程。

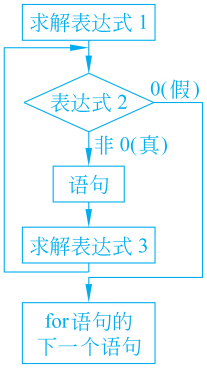

图 5.5

for 语句最简单的应用形式也就是最易理解的如下形式：

```
for(循环变量赋初值;循环条件;循环变量增值) 语句
```

例如：

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

的执行过程与图 5.1 完全一样。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。

for 循环语句功能丰富,使用灵活,方法多变,使用上有许多技巧,可以参阅本章的提高部分。

5.3.2 for 循环程序举例

学习了循环以后,可以实现一些有趣的算法。

例 5.4 国王的小麦。相传古代印度国王舍罕要褒赏聪明能干的宰相达依尔(国际象棋的发明者),国王问他要什么?达依尔回答说:“国王只要在国际象棋的棋盘第 1 个格子中放 1 粒麦子,第 2 个格子中放 2 粒麦子,第 3 个格子中放 4 粒麦子,以后按此比例每一格加一倍,一直放到第 64 格(国际象棋的棋盘是 8×8=64 格),我感恩不尽,其他什么都不要了。”国王想:这有多少!还不容易!让人扛来一袋小麦,但不到一会儿全用没了,再来一袋很快又用完了。结果全印度的粮食全部用完还不够。国王纳闷,怎样也算不清这笔账。现在用计算机来计算一下。

解题思路:

每个格子中的麦子粒数见图 5.6。

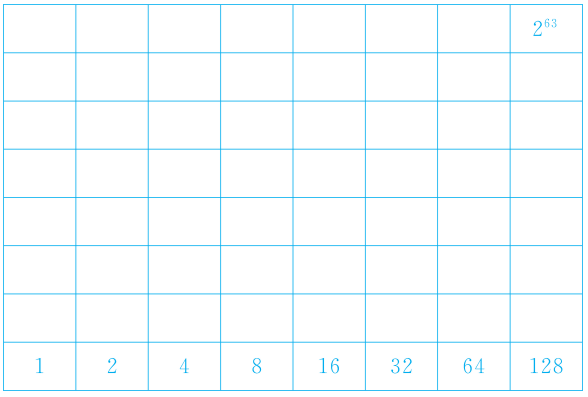

图 5.6

麦子的总粒数是:

$$1+2+2^2+2^3+\cdots+2^{63}$$

分别计算出每一格的麦子粒数,把它们加起来,就得到总粒数。据估算,1 立方米的小麦约有 1.42×10^8 粒,由此可以大致计算出小麦的体积。

可以用 for 语句实现循环。画出流程图(见图 5.7),其中图 5.7(a)是 N-S 流程图,图 5.7(b)是传统流程图。

编写程序:

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

```
{double p=1, t=1, v;
 int i;
 for(i=1; i<64; i++)                    //执行 63 次循环
   { p=p * 2;                           //p 是当前一个格子中的麦子粒数
     t=t+p;                             //t 是当前麦子总粒数
   }
 v=t/1.42e8;                            //v 是总体积,单位为立方米
 printf("total=%e\n",t);                //用指数形式输出麦子总粒数
 printf("volume=%e\n",v);               //用指数形式输出麦子总体积
 return 0;
}
```

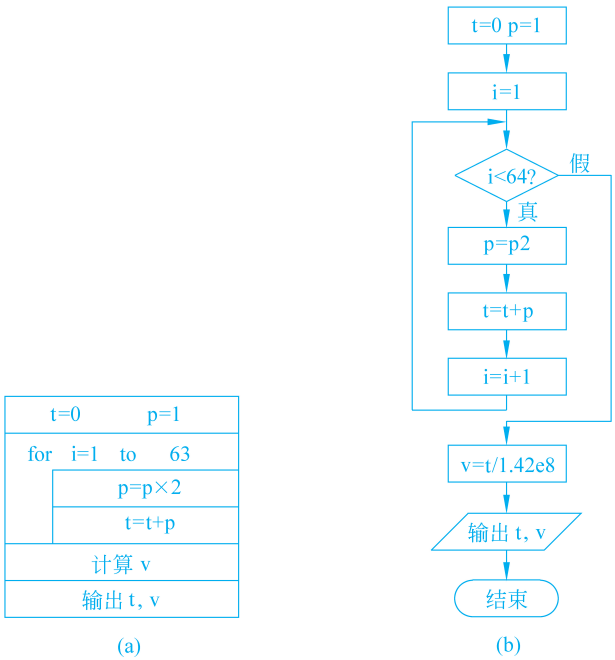

图 5.7

运行结果：

```
total=1.844674e+019
volume=1.299066e+011
```

计算结果为：共有小麦约 1.844674×10^{19} 粒，体积约 $1.3\times10^{11}\,m^3$。相当于全中国 960 万平方千米的土地上，全铺满 1.3cm 厚的小麦，这相当于我国几百年的总产量。

程序分析：

变量 p 用来存放一个格子中的麦子粒数，变量 t 用来存放某一时刻的麦子总粒数，变量 v 用来存放麦子的体积。变量 i 用来控制循环的次数，开始时 i=1，开始第 1 次循环，得到的 p 值是第 2 格的麦子粒数(请思考为什么)，t 是前 2 格的麦子总粒数。在完成第 1 次循环

后,i 的值加 1 变为 2,由于 2<64,所以执行第 2 次循环,此时得到的 p 值是第 3 格的麦子粒数,t 是前 3 格的麦子总粒数。以此类推,当 i 变到 63 时,执行最后一次循环,此时得到的 p 值是第 64 格的麦子粒数,t 是 64 格的麦子总粒数。i 再变为 64,由于 i 不再小于 64 了,不再执行循环。接着计算体积,输出结果。

请读者分析:

(1) 程序执行了 63 次循环,那么怎样实现累加了 64 个格子的小麦呢?

(2) 如果把第 5 行改为:for(i=1;i<=64;i++),结果会怎样?

(3) 如果把第 5 行改为:for(i=0;i<64;i++),结果会怎样?

不妨上机试验一下。

例 5.5 人口增长预测。根据 2020 年末全国人口普查,我国人口为 141178 万人。如果人口的年增长率为 1%,请计算到哪一年中国总人口达到或超过 20 亿人。

解题思路:

计算人口增长和计算存款利息的公式是相同的。假设原来人口为 p_0,一年后的人口为 p:

$$p = p_0 \times (1 + r)$$

其中 r 是年增长率。用此公式依次计算出每年的人口,每算出一年的人口后就检查一下是否达到或超过 15 亿人。如果未达到或超过 15 亿人,就再计算下一年的人口,直到某一年的人口达到或超过 15 亿人为止。

编写程序:

```
#include <stdio.h>
int main()
 {double p=141178e4,r=0.01;
  int year;
  for(year=2020; p<2e9; year++)
    {
      p=p * (1+r);
    }
  printf("year=%d,p=%e\n",year-1,p);
  return 0;
 }
```

运行结果:

```
year=2056,p=2.019931e+009
```

即到 2056 年,中国人口超过 20 亿人。如果 r 变量(表示增长率)改为 0.005,则结果为:

```
year=2090 ,p=2.00165e+009
```

即到 2090 年,中国人口超过 20 亿人。

程序分析:

程序中没有用两个变量 p0 和 p 来代表原来人口和一年后的人口,而用一个变量 p 代表不同年份的人口。开始时 p 的值是原有人口数,把 p 代入 p * (1+r)公式,求出一年后的人

口，然后把它赋给变量 p，此时 p 的值已不是原有人口了，而是一年后的人口了。在第二次循环中，再以这个 p 的新值为基础，计算出下一年的人口，如此不断由 p 的上一个值推算出 p 的新值，直到 p≥20 亿为止。

提示：注意区分变量 p 在不同阶段中的不同含义。可以看到，一个变量开始时有一初值，通过一定的运算，可以推算出一个新的值，再从这个新值又推出下一个新值，即不断用计算出的新值去取代原有的值，这种方法称为迭代(iterate)。上面的计算公式 p*(1+r)称为迭代公式。迭代算法一般是用循环来实现的。迭代是一种常用的算法，用人工实现很麻烦，而用计算机实现却十分方便。

year 代表年份。循环体中只有一个语句，用来计算从 2021 年开始的各年的人口数。在 for 语句中设定的循环条件是 p<20 亿，当某一年的 p 达到或超过 20 亿，就停止循环，输出年份和当年的人口数。

思考：

(1) 在 printf 函数中，输出项为什么是“year-1”，而不是 year?

请仔细分析上面程序中循环体的执行过程。在执行第一次循环时，变量 year 的值是 2020，由于此时 p<2e9，就执行循环体语句，计算出来的 p 的新值是下一年(即 2021 年)的人口。同理，在 year 的值是 2055 时，计算出来的 p 的新值是下一年(即 2056 年)的人口。在执行完循环体后。year 加 1 变为 2056，此时 p 的新值已超 20 亿，循环结束。在 for 循环外输出的 year 值为 2056，p 为 2056 年的人口。

(2) 如果要求输出每一年的人口数，应怎样修改程序?

(3) 如果人口年增长率 $r=1\%$，要求计算 1000 年后的人口，程序怎样修改?

经过计算，结果如下：

```
year=  3020  ,p=2.958972e+013
```

即 1000 年后的 3020 年，我国将有 2958972 亿人口(约 29.589 万亿人口)。全国面积为 960 万平方千米，每平方千米平均有 3082262 人，平均每平方米要居住 3.08 人。如果年增长率为 1.5%(有的不发达国家还高于此值)，则我国在 1000 年后有 41.286 万亿人口，包括大山、河流，沙漠在内，平均每平方米将要居住 430 人。要盖多少的摩天大楼，才能在一平方米的地面上住上 400 多人啊。

5.4 循环的嵌套

一个循环体内又包含另一个完整的循环结构，称为**循环的嵌套**。内嵌的循环中还可以嵌套循环，这就是**多层循环**。

三种循环(while 循环、do…while 循环和 for 循环)可以互相嵌套。例如，下面几种都是合法的形式：