第5章

循环结构程序设计

通常，许多问题的求解都包含一些重复执行的操作。例如，输入 N 名学生的成绩、数值计算中的方程迭代求根、集合中的数据遍历访问等。在程序设计中对于那些需要重复执行的操作可以利用循环结构来进行处理。

循环结构是结构化程序设计的 3 种基本结构之一，其特点是在给定条件成立时，反复执行某程序段，直到条件不成立时为止。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，常用的有 while 语句、do…while 语句和 for 语句。灵活、巧妙地掌握和使用这些语句可以实现各种不同的复杂程序功能。

本章将介绍 3 种循环语句的用法以及循环结构程序设计的基本方法，具体内容如下。

- for 循环。
- while 循环。
- do…while 循环。
- break 和 continue 语句。

5.1 for 循 环

for 循环是最常用的循环结构，通常用于循环次数确定的情况，也可用于循环次数不确定但给出循环结束条件的情况。

学一学

for 循环的语法格式为

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

for 循环的执行过程如下(流程图如图 5-1 所示)。

① 计算表达式 1。

② 计算表达式 2 并判断其值；若其值为真，则转步骤③执行循环体；若其值为假，则转步骤⑤结束循环。

③ 执行循环体语句。

④ 计算表达式 3，转步骤②继续执行。

⑤ 结束循环，执行 for 循环后的下一个语句。

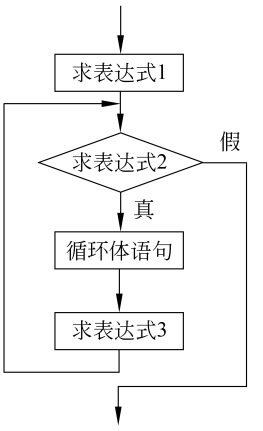

图 5-1　for 循环的流程图

说明

① for 语句的第一行被称为循环控制行，由 3 个表达式组成。循环控制行最常见的形式为

```
for(初始表达式; 循环条件表达式; 循环变量表达式)
```

例如：

```
for(i=0; i<n; i++)
```

初始表达式一般用来对循环变量赋初值，如上例 i＝0，变量 i 称为循环变量，通常用来记录循环执行的次数。循环条件表达式一般判断循环变量有没有达到需要重复执行的次数，如上例 i＜n，满足此条件时执行循环体，否则退出整个循环；循环变量表达式一般用来控制每次循环结束后循环变量的变化，如上例 i＋＋，使得表达式 2 在一定条件下不满足，结束循环。

② 若循环体语句只有一条，大括号可以省略，否则必须加上。

③ 循环条件表达式一般为关系表达式或逻辑表达式。

试一试

例 5-1　编写满足下列要求的 for 语句的循环控制行。

① 循环变量从 1 计数到 100。

② 循环变量按 2、4、6、8…计数到 100。

③ 循环变量从 100 开始反向计数，即 99、98、97、…、0。

④ 循环变量从'a'变到'z'。

解题思路

分析：根据 3 个表达式的作用(循环变量赋初值、循环条件控制、循环变量变化)，不难写出对应的控制行代码。

具体如下。

① for(i＝1；i＜＝100；i＋＋)

② for(i＝2；i＜＝100；i＝i＋2)

③ for(i＝100；i＞＝0；i－－)

④ for(c＝'a'；c＜＝'z'；c＋＋)

例 5-2　编写程序计算 1＋2＋3＋…＋10，然后输出结果。

解题思路

分析：本题是典型的累加求和问题，循环次数已知(10 次)，可以使用 for 循环进行实

现。具体步骤如下。

① 变量定义：定义循环变量 i 和累加和变量 s 并初始化 s=0。

② 计算处理：利用 for 循环累加求和，先编写循环控制行 for(i=1; i<=10; i++)，再编写循环体 s=s+i;。循环体只有一行，可以不用加大括号{}。

③ 输出结果：输出 s 的值。

程序代码

```
#include "stdio.h"
void main()
{
    int i, s=0;
    for(i=1; i<=10; i++)
        s=s+i;
    printf("1+2+…+10=%d\n", s);
}
```

程序运行结果如下。

```
1+2+…+10=55
```

程序注解

① 若循环变量 i 在定义时已经初始化，则表达式 1 可省略，但分号不能省略，例如：

```
int i=1, s=0;
for(; i<=10; i++)
```

表达式 3 也可以省略，可以将表达式 3(步骤④) 提前合并到循环体(步骤③) 中，例如：

```
for(i=1; i<=10;)
    s=s+i++;
```

若表达式 1 和表达式 3 同时省略，则等同于后面介绍的 while 循环。若三个表达式同时省略，循环体将无限制地执行，形成无限循环(死循环)。

② 表达式 1 可以是逗号表达式，可以将多个变量的初始化放在表达式 1 中，例如：

```
int i, s;
for(i=1, s=0; i<=10; i++)
```

表达式 3 也可以是逗号表达式，可以将简单循环体语句(步骤③) 合并到表达式 3(步骤④) 中，例如：

```
int i, s;
for(i=1, s=0; i<=10; s=s+i, i++)
```

③ 要理解循环变量与数据项之间的关系，本例中累加的数据项等于循环变量。若计算 2+4+…+20，循环变量从 2 变化到 20，每次增加 2，其他情况下不进行任何操作，对

应程序可以编写如下。

```
for(i=2; i<=20; i=i+2)
    s=s+i;
```

也可这样思考：10 项数据累加，循环变量从 1 到 10，每次循环累加的数据项为当前循环变量的 2 倍，对应程序可以编写如下。

```
for(i=1; i<=10; i++)
    s=s+i*2;
```

练一练

① 编程实现输出 5!。

② 编程实现输出 100 以内所有是 3 的倍数或者含有 3 的正整数。

5.2 while 循 环

在 for 循环中，表达式 1 和表达式 3 均可省略，此时可以将表达式 1 放在 for 循环前的变量初始化语句中，表达式 3 可以放在循环体语句中，例如：

```
int i=1, s=0;
for(; i<=10;)
    s=s+i++;
```

为了简化，将 for 循环的循环控制行修改为 while(i＜＝10)，这就是循环的另外一种结构：while 循环。

学一学

while 循环是“当型”循环结构，其语法格式为

```
while(条件表达式)
{
    循环体语句;
}
```

while 循环的执行过程如下(流程图如图 5-2 所示)。

① 计算条件表达式的值，判断其值是否为真，若为真，转步骤②执行，否则转步骤③执行。

② 执行循环体语句。

③ 结束循环，执行 while 循环后的下一个语句。

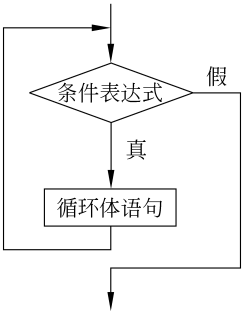

图 5-2 while 循环的流程图

说明

① 若循环体语句只有一条，大括号可以省略，否则必须加上。

② 条件表达式一般为关系表达式或逻辑表达式。

③ 避免条件表达式始终成立，使程序进入“死循环”状态。

试一试

例 5-3 计算 $s=1+2+3+\cdots$ 直到 s 大于 1000 为止。

解题思路

分析：本题也是典型的累加求和问题，不同于例 5-2 已知循环次数，题目给出了循环终止的条件 $s>1000$，相对应的循环条件表达式为 s<=1000。具体解题步骤如下。

① 变量定义：定义循环变量 i 和累加和变量 s 并初始化 s=0。

② 计算处理：利用 while 循环累加求和。

③ 输出结果：输出 s 的值。

程序代码

```
#include "stdio.h"
void main()
{
    int i=0, s=0;

    while(s<=1000)
    {
        s=s+i;
        i++;
    }
    printf("s=%d\n", s);
}
```

程序运行结果如下。

```
s=1035
```

程序注解

不同于 for 循环有初始化表达式进行循环变量的初始化，如果使用 while 循环，注意在定义循环变量时要进行循环变量初始化(本例中的 i=0)，在循环体中不要忘记编写循环变量变化表达式(本例中的 i++)，否则容易出错。

例 5-4 计算 $1-1/2+1/3-1/4+1/5-\cdots$ 直到某项的绝对值小于等于 10^{-6}(该项不累加)。

解题思路

分析：本题也是累加求和问题，不同于例 5-3，其累加项为实型数据且正负交替，在编写程序时需要注意变量类型的定义、flag 标记的设置以实现正负交替。具体步骤如下。

① 变量定义：定义循环变量 n 和累加和变量 s 并初始化 s=0，同时定义通项变量 item 以及标记变量 flag。

② 计算处理：利用 while 循环累加求和，当 item 的绝对值>1e-6 时进行累加，然后改

变 flag 标记，计算下一项的值并保存在 item 中，重复执行直至 item 的绝对值<=1e-6。

③ 输出结果：输出 s 的值。

程序代码

```
#include "stdio.h"
#include "math.h"
void main()
{
    int flag=1;
    double s=0, item=1, n=1;
    while(fabs(item)>1e-6)
    {
        s=s+item;
        flag=-flag;
        n=n+1;
        item=flag / n;
    }
    printf("s=%.6f\n", s);
}
```

程序运行结果如下。

```
s=0.693148
```

程序注解

① 本例中使用了 C 库函数中的求绝对值函数 fabs，需要在本文件的开头加上文件包含指令：#include "math.h"。

② 当循环变量 n 定义为整型时，计算 item 应避免使用两个整型变量直接相除，导致结果为 0，可使用 item=flag * 1.0/n;将参与计算数据转变为 double 类型进行计算。

练一练

用公式$\frac{\pi}{4}\approx 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots$求 π 的近似值，直到发现某一项的绝对值小于 10^{-6} 为止(该项不累加)。

5.3 do…while 循环

学一学

除了 for 循环和 while 循环外，C 语言还提供了 do…while 语句来实现循环结构。其语法格式为

```
do
{
    循环体语句;
} while(条件表达式);
```

do…while 循环的执行过程如下(流程图如图 5-3 所示)。

① 执行循环体语句。

② 计算条件表达式的值,判断其值是否为真,若为真,转步骤①执行,否则转步骤③执行。

③ 结束循环,执行 while 循环后的下一个语句。

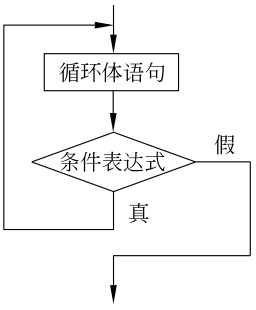

图 5-3　do…while 循环的流程图

说明

① do…while 循环的特点是先无条件执行循环体,然后判断循环条件表达式是否成立,因此循环体至少会执行一次。

② 若循环体语句只有一条,大括号可以省略,否则必须加上。

③ 条件表达式一般为关系表达式或逻辑表达式。

试一试

例 5-5　使用 do…while 循环计算 5!。

解题思路

分析:本题是典型的连乘求积问题,注意在定义连乘积变量时初始化其值为 1。具体步骤如下。

① 变量定义:定义循环变量 i 和连乘积变量 s 并初始化 s=1。

② 计算处理:利用 do…while 循环计算连乘积,先执行 s=s * i,然后更改循环变量,判断循环条件 i<=5 是否成立,条件成立时继续执行循环体,直至 i>5 结束循环。

③ 输出结果:输出 s 的值。

程序代码

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

程序运行结果如下。

```
s=120
```

程序注解

当 while 后面的表达式一开始就为假时，while 和 do…while 两种循环的结果是不同的。

练一练

求出 100～999 的所有水仙花数。水仙花数是指一个三位数其各位数字的立方和等于该数本身(例如 $1^3+5^3+3^3=153$)。

5.4 循环的嵌套

学一学

一个循环体内又包括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。前面介绍的 3 种循环之间可以互相嵌套。

嵌套循环有很多应用场景，图形输出是嵌套循环的典型应用之一，下面通过例题讲解如何利用两层循环进行图形的输出。

试一试

例 5-6 编写程序，输出图 5-4 中的九九乘法表。

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

图 5-4 九九乘法表

解题思路

分析：图形问题一般有行有列，本题中的九九乘法表共有 9 行，设计一个循环(一般称为外循环)，控制循环打印 9 行。针对第 i 行共有 i 列，再设计一个循环(一般称为内循环)，控制循环打印每行的 i 列数据。每行结束后需要换行，列与列之间可以填充一个空格。具体步骤如下。

① 变量定义：定义两个循环变量 i 和 j，分别用来控制打印行和打印列。

② 外循环：先设计一个外循环，i 从 1 变化到 9，循环体负责打印第 i 行数据。

③ 内循环：在外循环的循环体内再设计一个内循环打印第 i 行数据，j 从 1 变化到 i，在内循环的循环体内打印第 i 行第 j 列数据 i * j。

程序代码

```
#include "stdio.h"
```

```
void main()
{
    int i=1, j=1;
    for(i=1; i<=9; i++)
    {
        for(j=1; j<=i; j++)
        {
            printf("%d*%d=%-2d ", i, j, i*j);
        }
        printf("\n");
    }
}
```

程序注解

① 外循环的循环体包括一个内循环和一个换行语句，因此外循环的循环体需要加上大括号，将两者作为一个整体。

② %－2d 用于限制打印的数据占两个字符宽度并左对齐。

练一练

编写程序，打印输出 1～5 的数字金字塔，如图 5-5 所示。

```
    1
   121
  12321
 1234321
123454321
```

图 5-5　数字金字塔

“百钱买百鸡”类问题是嵌套循环的另一典型应用，下面通过例题讲解如何利用两层循环解决此类问题。

试一试

例 5-7　已知公鸡每只 5 元，母鸡每只 3 元，小鸡 1 元 3 只。要求用 100 元钱正好买 100 只鸡，问公鸡、母鸡、小鸡各多少只？

解题思路

分析：公鸡可能取值 0～20，母鸡可能取值 0～33，对所有可能的公鸡取值、母鸡取值进行穷举。先设计一个循环对公鸡可能取值 x 进行穷举。当公鸡取值 x 固定时，再设计一个循环对母鸡可能取值 y 进行穷举。当 $5*x+3*y+(100-x-y)/3==100$ 时，满足题目要求，输出对应结果，然后继续对下一组可能取值进行判断。具体步骤如下。

① 变量定义：定义两个循环变量 x 和 y，分别用来控制公鸡可能取值和母鸡可能取值。

② 外循环：先设计一个外循环对公鸡可能取值进行循环，x 从 0 变化到 20。

③ 内循环：在外循环的循环体内再设计一个内循环对母鸡可能取值进行循环，y 从 0 变化到 33，在内循环的循环体内负责对 x、y、100－x－y 这样一组数据进行判断，满足要

求则输出，否则更改循环变量，继续进行下一组数据的判断。

程序代码

```
#include "stdio.h"
void main()
{
    int x=0, y=0, c=1;
    for(x=0; x<=20; x++)
    {
        for(y=0; y<=33; y++)
        {
            if(5*x+3*y+(100-x-y)/3==100 && (100-x-y)%3==0)
            {
                printf("%d: 公鸡%d只,母鸡%d只,小鸡%d只\n",
                    c++, x, y, 100-x-y);
            }
        }
    }
}
```

程序运行结果如下。

```
1: 公鸡 0 只,母鸡 25 只,小鸡 75 只
2: 公鸡 4 只,母鸡 18 只,小鸡 78 只
3: 公鸡 8 只,母鸡 11 只,小鸡 81 只
4: 公鸡 12 只,母鸡 4 只,小鸡 84 只
```

程序注解

小鸡取值需要被 3 整除，因此在对每组数据判断时增加了(100－x－y)％3＝＝0。

练一练

某工地需要搬运砖块，已知男人一人搬 3 块，女人一人搬 2 块，小孩两人搬 1 块。问 45 人正好搬 45 块砖有多少种搬法？

5.5 break 语句和 continue 语句

以上介绍的循环都是根据事先指定的循环条件正常终止的，但有时在某种情况下需要提前终止正在执行的循环，这时可以用 break 语句和 continue 语句实现。

5.5.1 用 break 语句提前终止循环

学一学

break 语句只能用在 switch 语句和循环语句(for、while、do…while 循环)中。当