第5章

循环结构程序设计

从1＋2＋3＋…＋100求和谈起——

高斯从小家境贫寒，他的父亲是一家杂货铺里的算账先生。高斯小时候，父亲就经常给他讲自己在工作中积累的一些简便算法。高斯聪明而且用心，对这些算法，不仅仅快速理解，而且还能举一反三，灵活运用，从小培养了对数学的浓厚兴趣。可是，在高斯上小学的时候，他的数学老师白尔脱先生总觉得农村小孩都不会学习也不爱学习，自然也不认真讲课，还经常无缘无故地训斥学生。

有一天，白尔脱心情不太好，就黑着脸，在黑板上列出题目：1＋2＋3＋…＋100＝？说："现在你们自己算这道题，谁算完就回家吃饭，算不完不许回家。"吓得同学们赶紧拿出练习本，开始计算。白尔脱人呢？却坐到一旁，悠闲自在地看起小说。可他两页还没有看完，小高斯就高高地举起手，说："老师，我算完了。"

谁知白尔脱不耐烦地挥挥手，"算完了？这么快能算对吗？再算算看吧！"

"老师，不会错的，我都检查过了，还验算了一遍。"高斯站起来，理直气壮地对老师说。

白尔脱走到高斯座位前，看到他的结果竟然是正确的，不禁大吃一惊，答案确实是"5050"。高斯讲述了一下他的计算过程：1＋100＝101，2＋99＝101，……，49＋52＝101，50＋51＝101，一共有50对101的数目，所以答案正好是50×101＝5050。

从这件事情以后，白尔脱一下子改变了对农村学生的偏见，慢慢发现高斯的数学天赋，还经常对高斯进行个别辅导。高斯也不负期望，对数学的兴趣越来越浓，造诣越来越深，17岁时就发现了数论中的二次互反律。

高斯采用了巧妙方法进行了计算。如果用计算机编程实现，就可以使用循环结构。实际生活中的许多问题属于循环结构，例如，假期—学习—假期—学习的循环，上课—下课—上课—下课的循环，人日出而作，日落而息，周而复始；求有规律的数列之和，求阶乘，求全班某门课的平均成绩，求素数，等等，而计算机最擅长的也正是重复操作，表现到程序上就是循环的结构，大多数应用程序中的重复操作都有一定的规律，因此都会包含循环结构。计算机比起人类的优势，就是不知疲倦，可以反复循环。也正因如此，机器把人类从重复烦琐的劳动中解放出来。那么如果编程实现这道题，会是怎么样呢？

本章介绍循环结构的有关问题，主要包括while语句、do…while语句、for语句、continue语句及其具体应用。

这样，程序设计的三种基本结构——顺序结构、选择结构和循环结构就都介绍了。这三种结构是结构化程序设计的基本结构，是复杂程序的基本构成单元。大家需要通过示例，全

面理解三种基本结构，弄清楚它们的应用场合、结构特点、区别联系、编程技巧等，以求熟练掌握有关内容，达到本课程的学习目标。

5.1 while 语句与用 while 语句构成的循环结构

5.1.1 while 语句

while 语句的一般语法格式如下：

```
while(表达式)循环体
```

说明：

(1) while 语句用来实现“当型”循环结构，while 语句构成的循环也称“当型”循环。while 是 C 语言的关键字。“当型”的意思是“当”条件满足时。

(2) while 后面括号中的表达式可以是 C 语言中任意合法的表达式，用以控制循环体是否执行。

(3) 若循环体由多个语句组成，应用花括号括起来，组成复合语句。

5.1.2 while 构成的循环结构

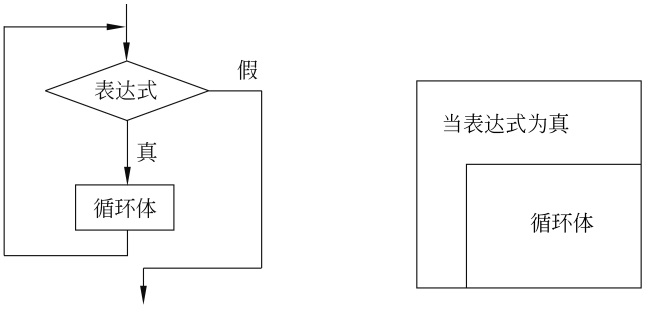
while 循环的执行过程如下。

(1) 计算 while 后表达式的值，当值为非零(条件满足，“真”)时，执行步骤(2)；当值为零(条件不满足，“假”)时，退出循环。

(2) 执行循环体。

(3) 执行步骤(1)。

流程图和 N-S 图如图 5.1 所示。



(a) 流程图　　(b) N–S图

图 5.1　while 语句与用 while 语句构成的循环结构的流程图和 N-S 图

使用 while 语句时，需注意如下问题。

(1) while 语句是先判断表达式的值，当值为非零(条件满足，“真”)时，执行循环体中的语句，如果表达式的值一开始就为零，则循环体一次也不执行。

(2) while 后圆括号中表达式的值决定了循环体是否执行，因此，循环体中应该有使表

达式的值变为零的语句，否则，循环将无限制地执行下去，即死循环。一般在嵌入式系统中经常要用到无限循环。因为设备一上电就要一直工作。


视频

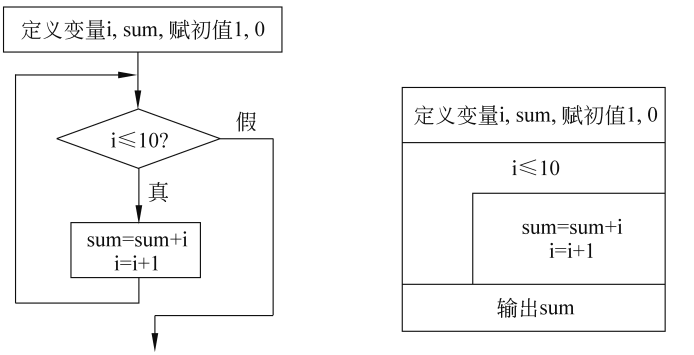
【例 5.1】 试求 1＋2＋3＋…＋10 的值。

分析：

(1) 本题是一个累加的问题，需要重复进行 10 次加法运算，显然可以用循环结构。而所累加的数从 1 到 10 是有规律的：后一个数比前一个数增 1，可在循环中定义一个整型变量 i 作为计数器，初始值为 1，每循环一次 i 加 1，当 i 的值超过 10 时结束循环。

(2) 对累加和，可定义一个变量 sum，初始值为 0，每循环一次，把 i 的值加到 sum 中，当循环结束时，sum 中的值即为所求的结果。

流程图和 N-S 图如图 5.2 所示。



(a) 流程图　　(b) N-S图

图 5.2　例 5.1 的流程图和 N-S 图

程序如下：

```
main( )
{
   int i=1,sum=0;
   while (i<=10)
   {
      sum=sum+i;
      i++;
   }
   printf("sum=%d\n",sum);
}
```

运行结果为：

```
sum=55
```

说明：

(1) 本程序循环体中包括两个语句，因此，用“{}”括起来。

(2) sum 为累加和，应在循环前赋初值，其初值应为零。

(3) 循环体中的 i＋＋使得 i 的值不断增加，以便最终导致 i＞10 而退出循环。

(4) 循环变量 i 的初值与循环体中改变其值语句的位置应相适应。请思考：若以下面的程序段代替例 5.1 中的循环体，会出现什么结果？

```
while (i<=10)
{
```

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

思考：你能否用 while 循环编程实现小高斯遇到的问题 1＋2＋3＋…＋100＝？

5.2 do…while 语句与用 do…while 语句构成的循环结构

5.2.1 do…while 语句

do…while 语句的一般格式如下：

```
do
循环体
while(表达式);
```

说明：

(1) do…while 语句用来实现直到型循环结构，do…while 语句构成的循环也称为直到型循环。直到型的意思是直到条件不满足为止。

(2) while 后面括号中的表达式，可以是 C 语言中任意合法的表达式，用以控制循环体是否执行。

(3) do 和 while 是 C 语言的关键字，do…while 是一个整体，必须联合使用。

(4) do…while 语句以 do 开始，以 while 结束，while 后的“;”不可省，以此表明 do…while 是一个语句。

(5) 循环体由多个语句组成时，应用“{}”把循环体括起来。

5.2.2 do…while 构成的循环结构

do…while 循环的执行过程如下。

(1) 执行 do 后面的循环体。

(2) 计算 while 后括号中表达式的值，当值为非零(条件满足，“真”)时，执行(1)；当值为零(条件不满足，“假”)时，退出 do…while 循环。

流程图和 N-S 图如图 5.3 所示。

使用 do…while 语句时，需注意如下问题。

(1) do…while 语句是先执行循环体，再判断表达式的值。因此，无论开始表达式的值是非零还是零，循环体都要首先被执行一次，这是同 while 语句的最大区别。

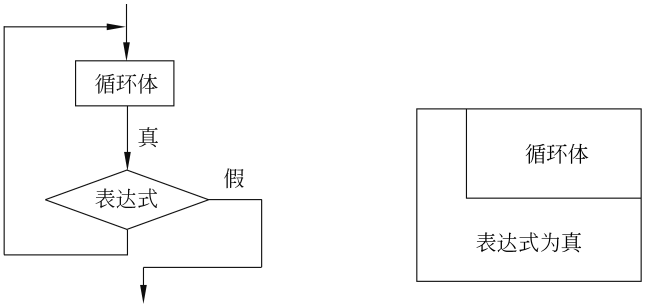
(2) 在循环体中同样应该有使表达式的值逐步变为零的语句，否则将成为死循环。

(3) C 语言中 do…while 语句是在表达式为非零(为“真”)时执行循环体。

【例 5.2】 试用 do…while 语句计算 1＋1/2＋1/4＋…＋1/50 的值。

分析：

(1) 该题也是一个累加和问题，需要重复进行多次加法运算；而各累加项分母的变化是



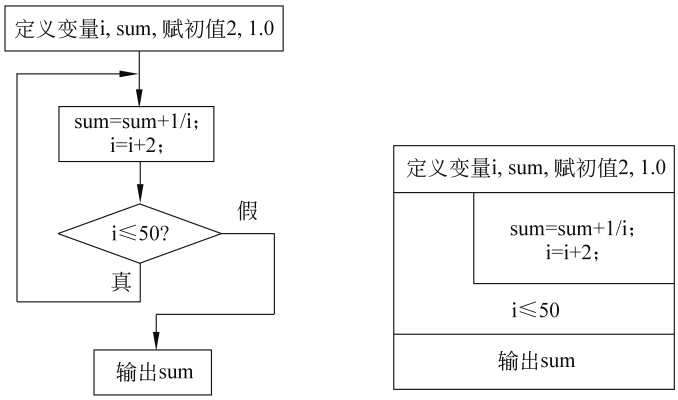
(a) 流程图　　(b) N–S图

图 5.3　用 do…while 语句构成的循环结构的流程图和 N-S 图

有规律的，其递推关系是后一项的分母比前一项增 2，因此，可用循环结构实现。

(2) 可设置一实型变量 sum，初始值为 0，累加和放在变量 sum 中。

流程图和 N-S 图如图 5.4 所示。



(a) 流程图　　(b) N–S图

图 5.4　例 5.2 的流程图和 N-S 图

程序如下：

```
main( )
{
   int i;
   float sum;
   sum=1.0;
   i=2;
   do
   {  sum+=1.0/i;
      i+=2;
   }while (i<=50);
   printf("sum=%f\n",sum);
}
```

运行结果为：

```
sum=2.907979
```

请思考：

(1) sum 的初值应与 i 的初值相对应，思考 sum 的初值是否可为零。

(2) sum＋＝1.0/i;语句中为什么要使用 1.0 而不是 1。

思考：你能否用 do…while 循环编程实现小高斯遇到问题 1＋2＋3＋…＋100＝？

5.3 for 语句与用 for 语句构成的循环结构

5.3.1 for 语句

for 语句的一般格式如下：

```
for(表达式 1;表达式 2;表达式 3)循环体
```

说明：

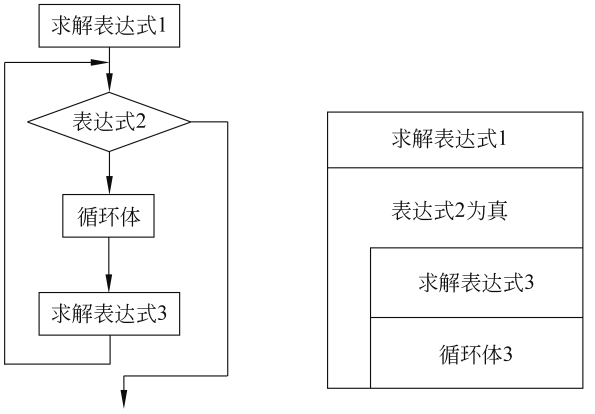
(1) for 是 C 语言的关键字。

(2) for 后括号中的三个表达式用“;”分隔，它们可以是 C 语言中任意合法的表达式，主要用于循环控制。

(3) 循环体若由多条语句构成，应用“{}”把它们括起来，组成复合语句。

5.3.2 for 语句构成的循环结构

for 循环的流程图和 N-S 图如图 5.5 所示。



(a) 流程图　　(b) N–S图

图 5.5　for 语句与用 for 语句构成的循环结构的流程图和 N-S 图

for 循环的执行过程如下。

(1) 计算表达式 1。

(2) 计算表达式 2，若其值为非零(条件满足，“真”)时，执行步骤(3)；若其值为零(条件不满足，“假”)时，执行步骤(5)。

(3) 执行一次循环体。

(4) 计算表达式 3，转向步骤(2)。

(5) 结束循环。

使用 for 语句时，需注意如下问题。

(1) for 语句中的表达式 1 可省略，可在循环之前为循环变量赋初值，但其后的“;”不

能省。

(2) for 语句中的表达式 2 也可省略,但其后的“;”不能省,此时应另设法使循环可以正常结束,否则将是死循环。例如:

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

相当于:

```
i=1;
while(1)
{
   sum=sum+1;
   i++;
}
```

(3) for 语句中的表达式 3 同样可省略,但应另设法使循环可以正常结束。例如:

```
for(i=1; i<=10;)
{
   sum=sum+1;
   i++;
}
```

(4) 表达式 1 和表达式 3 可同时省略,只有表达式 2。例如:

```
for(; i<=10;)
{
   sum=sum+1;
   i++;
}
```

相当于:

```
while (i<=10)
{
   sum=sum+1;
   i++;
}
```

(5) 三个表达式都可省略。例如:

```
for (; ;) 循环体
```

相当于:

```
while (1) 循环体
```

(6) 表达式 1 可以是设置循环变量初值的赋值语句,也可以是与设置循环变量无关的其他语句。例如:

```
for(sum=0;j<=10;j++)sum=sum+j;                    /*j为循环控制变量*/
```

(7) 表达式 1 和表达式 3 可以是简单的表达式,也可以是逗号表达式。例如:

```
for (sum=0,j=1;j<=10;j++) sum=sum+j;
```

(8) 在逗号表达式内按自左至右的顺序求值,整个逗号表达式的值等于最右边表达式的值。例如:

```
for (j=1;j<=100;j++,j++)sum=sum+j;
```

相当于:

```
for(j=1;j<=100;j=j+2)sum=sum+j;
```

【例 5.3】 计算 10!＝1×2×3×…×10 的值。

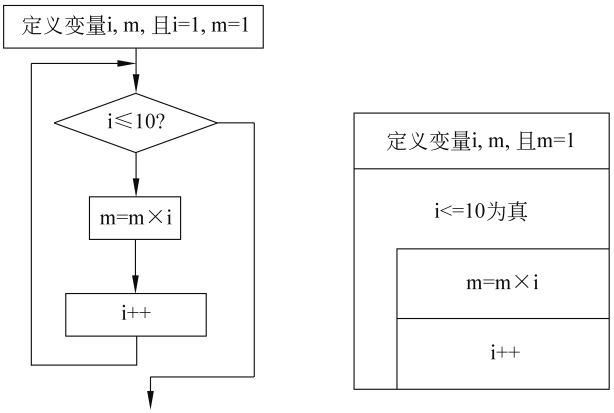
分析：

(1) 本题中包含了多个乘法运算，因此需要用循环实现；各乘数的规律是后一个数比前一个数增 1，并且初值为 1，终值为 10，循环次数容易确定，因此用 for 语句较适合。

(2) 最后的乘积放入变量 m 中，并且 m 的初值应为 1。

(3) 变量 m 应定义为 long 型，而不是 int 型。int 型最大值为 32 767，1×2×3×…×10 的值早已超过它。

流程图和 N-S 图如图 5.6 所示。



(a) 流程图　　(b) N–S图

图 5.6　例 5.3 的流程图和 N-S 图

程序如下：

```
main ( )
{
   int i; long m;
   m=1;
   for(i=1;i<=10;i++)
     m=m*i;
   printf("m=%ld\n",m);
}
```

运行结果为：

```
m=3628800
```

请思考：

(1) 如果计算 1×2×3×…×100 的值，修改这个程序能达到目的吗？

(2) 例 5.1 和例 5.2 的问题也可以用 for 语句完成，请分析并实现。

思考：你能否用 for 循环编程实现小高斯遇到的问题 1＋2＋3＋…＋100＝？

5.4 三种循环的比较和嵌套

5.4.1 三种循环的比较

(1) 一种循环可以解决的问题,使用另外两种循环同样可以解决,只是方便程度不同而已。三种循环的关系如下:while 循环修改循环控制条件后相当于 do…while 循环,for 循环省略表达式 1 和表达式 3 后相当于 while 循环。

(2) for 循环和 while 循环是先判断条件是否为真,再执行循环体,因此,可出现循环一次也不执行的情况;do…while 语句是先执行循环体,再判断条件是否为真,因此,循环体至少执行一次。

(3) while 循环和 do…while 循环的表达式只有一个,只起控制循环的作用;for 循环有三个条件表达式,除控制循环外,还可以赋初值和使循环变量的值改变。

(4) 用 while 和 do…while 循环时,循环变量初始化的操作应在 while 和 do…while 语句之前完成,而 for 语句可以在表达式 1 中实现循环变量的初始化。

(5) while 循环和 do…while 循环一般用于循环次数不定的情况,for 循环一般用于循环次数事先已知的情况。do…while 循环一般用于至少需要执行一次循环的情况。

5.4.2 三种循环的嵌套

如果在一个循环体中,又包含了另外一个完整的循环,称为循环的嵌套。内循环中还可以再嵌套循环,这是多重循环。

三种循环语句可以自身嵌套,也可以相互嵌套。应当注意,嵌套不能出现交叉,必须保证每一个循环结构的完整性。

实际应用中,复杂的循环程序往往是多重循环,应仔细分析题目,先确定整体框架即外循环,再确定内循环,必要时可先适当减少循环次数,手工写出具体每次的操作结果,容易发现其循环规律。这也符合结构化程序设计自顶向下,逐步细化的特点。

下面几种都是合法的嵌套形式。

(1) 外循环为 while 结构,内循环也是 while 结构。

```
while( )
{
   while( )
   {…}
}
```

(2) 外循环为 do…while 结构,内循环也是 do…while 结构。

```
do
{
   …
   do
   {
```

```
    …
    }while( );
}while( );
```

(3) 外循环为 for 结构,内循环也是 for 结构。

```
for( ;  ; )
{
   for( ;  ; )
   { … }
}
```

(4) 外循环为 while 结构,内循环是 do…while 结构。

```
while( )
{
   …
   do
   {
   …
   }while( );
}
```

(5) 外循环为 for 结构,内循环是 while 结构。

```
for(; ;)
{
   …
   while( )
   {…}
   …
}
```

(6) 外循环为 do…while 结构,内循环是 for 结构。

```
do
{
   …
   for(; ;)
   { … }

} while( );
```

【例 5.4】 for 结构内嵌套 for 结构。编程打印以下图形。


视频

```
123456789
23456789
3456789
456789
56789
6789
789
89
9
```

分析:

(1) 本题目用一重循环无法实现,需要使用二重循环的嵌套结构。外循环控制打印哪一行(第 1 行、第 2 行、…、第 9 行),内循环控制打印某 1 行数字,从数字 i 打印到数字 9。

(2) 本题目循环次数事先已知,用 for 循环比较合适。