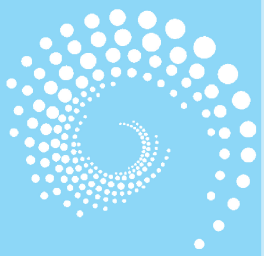
第 5 章

循环结构程序设计

CHAPTER 5



经过对前几章的学习，读者已经掌握了结构化程序设计的顺序结构和选择结构。在本章将继续讲解循环结构，它可以解决许多循环控制问题。通过对本章的学习，读者将能够综合运用结构化编程思想解决一些问题。

5.1 循环结构程序的概念

【引例 5-1】 编程计算一个学生三门课程的平均成绩。

【思路】 把一个学生三门课程的成绩相加，再除以 3，得到平均成绩。使用顺序结构，就可以实现上述功能。

编写程序段如下：

```
float score1, score2, score3, aver;
scanf("%f,%f,%f",&score1,&score2,&score3);
aver=(score1+score2+score3)/3;
printf("aver=%f\n", aver);
```

【引例 5-2】 一个班 30 名学生，求每个学生三门课程的平均成绩。

【思路】 在引例 5-1 中，程序段实现的功能是求一个学生三门课程的平均成绩。也就是说，可以利用上述同一个程序段来求每个学生的平均成绩。如果有 30 名学生，则要将上述程序段重复书写 30 遍，来实现求 30 名学生每人三门课程的平均成绩的功能。

【程序代码】

```
float score1, score2, score3, aver;
//求第 1 名学生三门课程的平均成绩
scanf("%f,%f,%f",&score1,&score2,&score3);
aver=(score1+score2+score3)/3;
printf("aver=%f\n",aver);
      …
//求第 30 名学生三门课程的平均成绩
```

```
scanf("%f,%f,%f",&score1,&score2,&score3);
aver=(score1+score2+score3)/3;
printf("aver=%f\n",aver);
```

【存在问题】　将同一个程序段重复书写 30 遍，存在着工作量大、程序易出错、可阅读性差、可维护性差等问题。引例 5-2 涉及的学生数是 30 名，通过复制重复程序段还是可以完成的。但是，试想如果涉及的学生数是几千、几万、几十万，则采用上述方法来完成相应的功能是无法想象的。

【总结】　引例 5-2 的主要程序采用顺序结构，重复地执行同一个程序段，总共重复了 30 次。上述这类问题，可以采用本章将要讲解的循环结构来解决。

循环结构程序的概念：循环结构程序就是重复执行一个程序段的程序。在 C 语言中，循环结构的实现语句有三种，分别为 while 语句、do-while 语句和 for 语句。下面，将对由上述三种语句实现的 while 循环、do-while 循环和 for 循环分别给予详细阐述。

5.2　while 循环

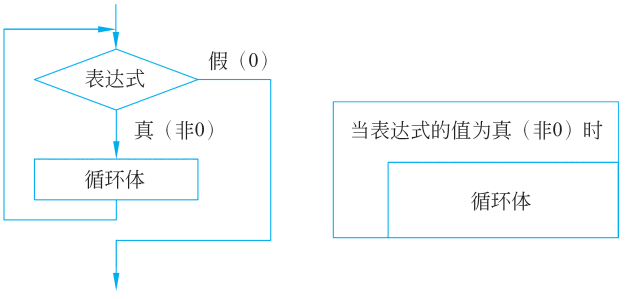
while 语句的一般形式如下：

```
while(表达式)　循环体
```

其中，表达式是循环条件，由它来控制循环体是否执行。循环体可能是一条语句，也可能是多条语句组成的复合语句。

while 循环流程图如图 5.1 所示。如果表达式的值为真(非 0)，则执行循环体，如此反复，直到表达式的值为假(0)时，跳出 while 循环，while 循环结束。从图中可以看出，while 语句实现的是“当型”循环结构。

while 循环的特点是先判断表达式，后执行循环体。



(a) 传统流程图　　(b) N-S图

图 5.1　while 循环流程图

现在，尝试着用 while 循环来完成引例 5-2 所要实现的功能，其流程图如图 5.2 所示。这里，从第 1 个学生开始，一直到第 30 个学生结束，用 n 来记录学生的次序。所以，n 初始值为 1，表达式为 n≤30，每次循环最后，n 都加 1；循环结束的条件是 n>30，即当 n=31 时，跳出 while 循环。这里，将 n 称为循环变量。

【程序代码】

```
1   #include <stdio.h>
```

```
void main()
{
    float score1,score2,score3,aver;
    int n=1;                          //循环变量初始化
    while(n<=30)                      //循环条件
    {   //循环体
        scanf("%f,%f,%f",&score1,&score2,&score3);
        aver=(score1+score2+score3)/3;
        printf("aver=%f\n",aver);
        n++;                          //循环变量加 1
    }
}
```

【例 5-1】 求 s=1+2+3+…+100 的值。

【问题解析】

首先,用流程图来描述这个问题,具体如图 5.3 所示。

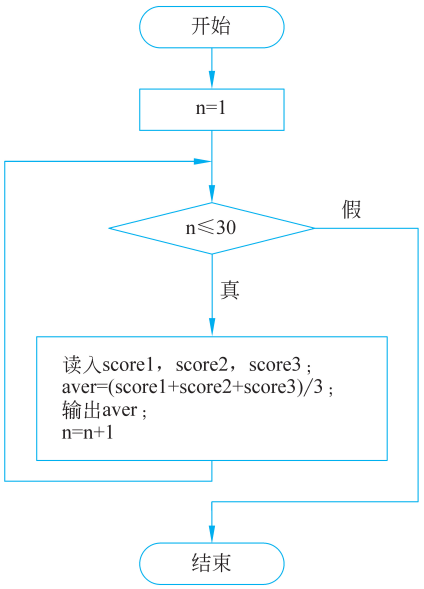


图 5.2 用 while 循环实现引例 5-2 的程序流程图

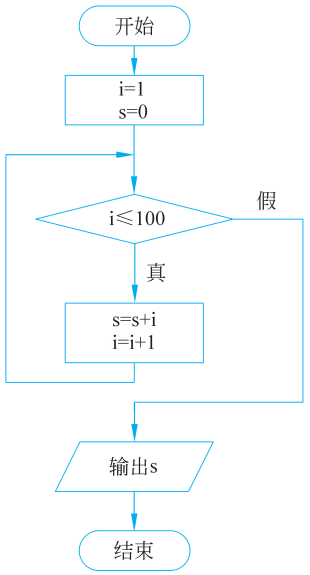


图 5.3 例 5-1 程序流程图

【程序代码】

```
#include <stdio.h>
void main()
{   int i=1, s=0;
    while (i<=100)
    {   s=s+i;
        i++;
    }
    printf("s=%d\n", s);
}
```

【运行结果】

```
s=5050
```

【说明】

(1) 如果循环体的语句超过一条,必须使用复合语句,即利用{ }把多条语句括起来,形

成复合语句；对于循环体只有一条语句的情况，可以不加{}。例如：例 5-1 的循环体包括两条语句“s＝s＋i; i＋＋;”，应该用{ }括起来，如果不加{ }，则循环体只包括一条语句“s＝s＋i;”。另外，要注意 while 语句的表达式之后没有分号，如果加上分号，则表示循环体是一条空语句。

（2）循环体中应有使循环趋向结束的语句。例如：例 5-1 的表达式是“i＜＝100”，循环结束的条件是“i＞100”，所以循环体中的语句“i＋＋;”是使循环趋向结束的语句。如果没有这条语句，则循环变量 i 的值一直保持不变，永远都是初始值 1，则 while 循环永远都不会结束，从而形成死循环。

思考：若求 s＝1＋2＋3＋…＋n 的值，应该如何修改程序呢？其中 n 是由键盘输入的随机正整数。

注意：

（1）对于循环来说，循环体可能一次也不执行，如例 5-1 中，若 i 的初值为 101，进入循环的条件为假，则循环一次也不执行。

（2）在循环题中必须有使循环趋向结束的操作，否则循环将无限进行（死循环）。如例 5-1 中的“i＋＋;”语句。

（3）在循环体中，语句的先后位置必须符合逻辑，否则会影响运算结果。如例 5-1 中循环体的“s＝s＋i; i＋＋;”两句的顺序就不能颠倒，否则计算的结果就完全不同。

【例 5-2】 说出下列程序的功能。

```
#include <stdio.h>
void main()
{   int i=1,n,a=0,b=0,c=0;
    whlie (i<=10)
    {   scanf("%d",&n);
          if (n>0) a++;
          else if (n==0) b++;
          else c++;
       i=i+1;
    }
    printf("a=%d,b=%d,c=%d\n",a,b,c);
}
```

【问题解析】

这个程序的功能是统计从键盘上输入的 10 个整数中，大于 0、小于 0、等于 0 的整数个数。其中，a、b、c 的值分别代表大于 0、等于 0 和小于 0 的整数个数。

【运行结果】

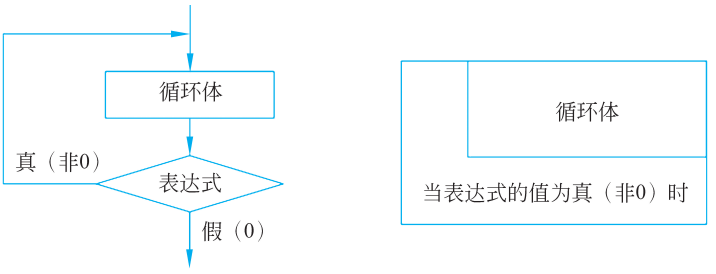
```
10 -13 35 47 0 0 -59 98 0 88
a=5, b=3, c=2
```

5.3　do-while 循环

do-while 语句的一般形式如下：

```
do
    循环体
while (表达式);
```

do-while 循环流程图如图 5.4 所示。先执行一次循环体,然后判断表达式,如果表达式的值为真(非 0),则重新执行循环体,如此反复,直到表达式的值为假(0)时,跳出 do-while 循环,do-while 循环结束。



(a) 传统流程图　　(b) N-S图

图 5.4　do-while 循环流程图

do-while 循环的特点是先执行循环体,后判断表达式。

现在,尝试用 do-while 循环来完成引例 5-2 所要实现的功能,其流程图如图 5.5 所示。

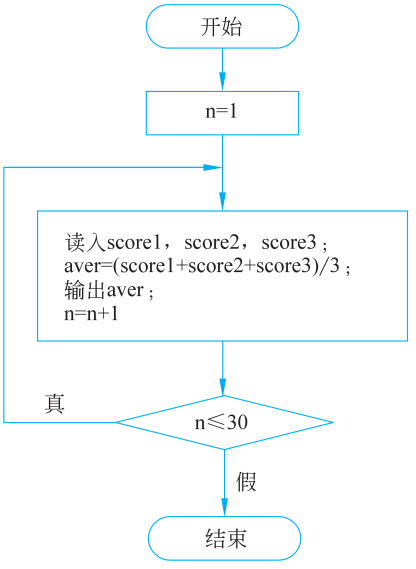


图 5.5　用 do-while 循环实现引例 5-2 的流程图

【程序代码】

```
#include <stdio.h>
void main()
{   float score1, score2, score3, aver;
      int n=1;              //循环变量初始化
      do
      {   //循环体
         scanf("%f,%f,%f",&score1,&score2,&score3);
         aver=(score1+score2+score3)/3;
         printf("aver=%f\n",aver);
          n++;              //循环变量加 1
       }
```

```
12          while (n<=30);              //循环条件
13  }
```

【例 5-3】　要求用 do-while 语句实现例 5-1。

【问题解析】

首先，用流程图来描述这个问题，具体如图 5.6 所示。

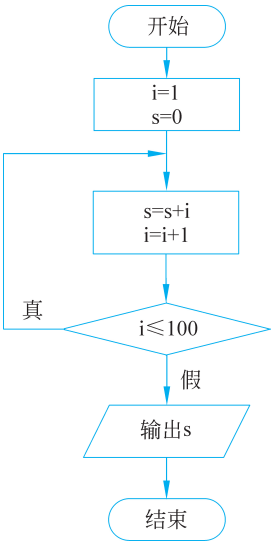


图 5.6　例 5-3 程序流程图

【程序代码】

```
1   #include <stdio.h>
2   void main()
3   {   int i=1, s=0;
4       do
5       {   s=s+i;
6           i++;
7       }
8       while(i<=100);
9       printf("s=%d\n", s);
10  }
```

【运行结果】

```
s=5050
```

【例 5-4】　已知 s=1+3+5+…+99，求 s 以及奇数个数 j。

【问题解析】

设循环变量 i=1,3,5,…,99，初始值为 i=1，s=0，j=0，循环条件为 i<=99，循环变量增值为 i=i+2，在循环体中除最后一条语句“i=i+2;”外的其他语句为“s=s+i; j=j+1; ”。

【程序代码】

```
1   #include <stdio.h>
2   void main()
3   {   int i=1,s=0,j=0;
4       do
5       {   s=s+i;
6           j=j+1;
7           i=i+2;
8       }
9       while(i<=99);
10      printf("s=%d\n",s);
11      printf("j=%d\n",j);
12  }
```

【运行结果】

```
s=2500
j=50
```

由前面的学习可知，同样的问题可以用 while 循环和 do-while 循环来分别进行实现，但是在实现的过程中会发现 while 和 do-while 既有相同点，又有不同点。下面，就来比较一下 while 和 do-while，它们的异同点具体总结如下。

相同点如下。

（1）while 和 do-while 的循环体基本相同。

（2）while 和 do-while 的循环变量初始化都在循环体外进行。

(3) 当 while 和 do-while 具有相同的循环体时,如果 while 后的表达式值第一次为“真”,则 while 和 do-while 的执行结果相同。

不同点如下。

(1) do-while 先执行循环体,后判断表达式;while 先判断表达式,后执行循环体。

(2) 当 while 和 do-while 具有相同的循环体时,如果 while 后的表达式值第一次就为“假”,则 while 和 do-while 的执行结果不同。此时,while 循环体一次都不能执行,但是 do-while 循环体能执行一次。

【例 5-5】 求任意整数 n 的阶乘 j,即 j=n!,要求分别用 while 和 do-while 实现。

【问题解析】

j=1 * 2 * 3 * … * n,设循环变量 i=1,2,3,…,n,初始值为 i=1,j=1,循环条件为 i<=n,循环变量增值为 i=i+1,在循环体中除最后一条语句“i=i+1;”外的其他语句为“j=j * i;”。由于 j 的值可能较大,所以要将 j 定义为 long 类型。

(1) 用 while 实现,编写程序如下:

```
#include <stdio.h>
void main()
{   int n, i;
    long j;
    printf("请输入 n:");
    scanf("%d",&n);
    i=1; j=1;
    while (i<=n)
    {   j=j * i;
        i++;
    }
    printf("%d!=%ld\n",n,j);
}
```

【运行结果】

```
请输入n:6
6!=720
```

(2) 用 do-while 实现,编写程序如下:

```
#include <stdio.h>
void main()
{   int n, i;
    long j;
    printf("请输入 n:");
    scanf("%d",&n);
    i=1; j=1;
    do
    {   j=j * i;
        i++;
    }
    while (i<=n);
    printf("%d!=%ld\n",n,j);
}
```

【运行结果】

```
请输入n:6
6!=720
```

5.4　逗号表达式

逗号表达式的定义：逗号表达式是用一个逗号运算符“,”将两个表达式连接起来的式子，例如：1＊2，5＋7。

逗号表达式的一般形式如下：

```
表达式 1,表达式 2
```

逗号表达式的求解过程为：先求解表达式 1，再求解表达式 2。表达式 2 的值作为整个逗号表达式的值。注意，逗号运算符的优先级在所有运算符中是最低的。

例如：1＊2，5＋7 表达式的值是 12。

问：a＝4＊2，a＊4 表达式的值是多少？

【问题解析】

由于逗号运算符“,”的优先级比赋值运算符“＝”的优先级低，所以应该先求解赋值表达式 a＝4＊2，相当于(a＝4＊2),a＊4。注意第一个表达式求解后 a＝8，第一个表达式最终的值是 8，求解第二个表达式 a＊4 时 a 的值为 8，第二个表达式最终的值是 32，整个逗号表达式的值是 32。注意：在对每个表达式进行求解的过程中，要实时跟踪各个变量值的更新情况，以便在随后的表达式求解过程中使用这些变量的最新值。

逗号表达式的扩展形式如下：

```
表达式 1，表达式 2，表达式 3，…，表达式 n
```

其求解过程为：先求解表达式 1，再求解表达式 2，直到求解表达式 n。表达式 n 的值作为整个逗号表达式的值。

问：a＝2，b＝3＋a，c＝3＊a＋b，求解后 a、b、c 以及整个表达式的值分别是多少？

【问题解析】

先求解第一个表达式 a＝2，此时 a 的值为 2，并且第一个表达式的值为 2；再求解第二个表达式 b＝3＋a，此时 b 的值为 3＋2，等于 5，并且第二个表达式的值为 5；最后求解第三个表达式 c＝3＊a＋b，此时 c 的值为 3＊2＋5，等于 11；所以 a 最终的值为 2，b 最终的值为 5，c 最终的值为 11，整个表达式的值为第三个表达式的值 11。

逗号表达式将几个表达式连接在一起，其主要目的是求解各个表达式的值，而不是求解整个表达式的值。逗号表达式常被用于 for 循环中，详见 5.5 节。

例如 for (i＝1，s＝0；i＜＝100；s＝s＋i，i＋＋)；，这里，i＝1，s＝0 属于逗号表达式，s＝s＋i，i＋＋也属于逗号表达式，一般只关心各个表达式的执行结果，并不关注整个逗号表达式的值。

最后，需要特别说明的是：在后面要学习的函数中，存在函数调用，例如 max(x，y，z)，此时的 x，y，z 是函数 max 的三个参数，x，y，z 不是逗号表达式。

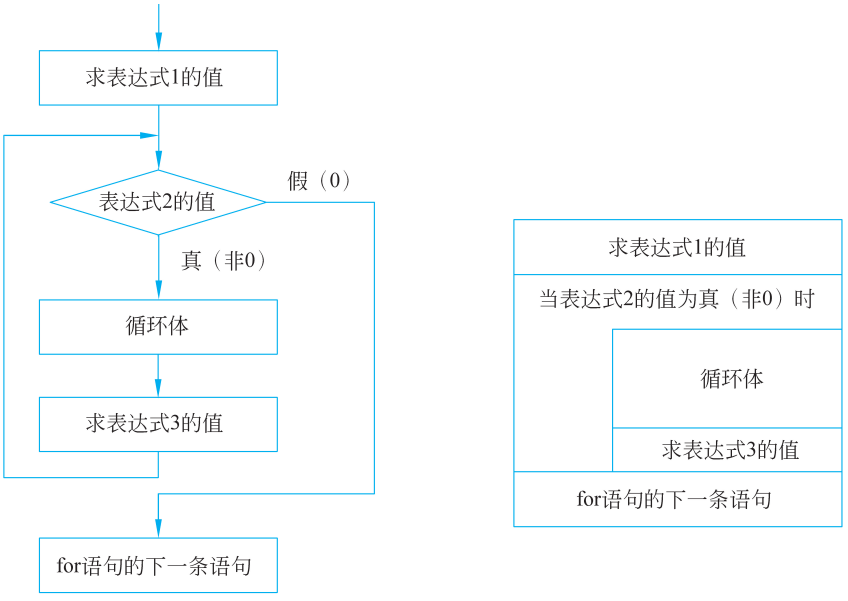
5.5　for 循环

for 语句的一般形式如下：

```
for (表达式 1; 表达式 2; 表达式 3) 循环体
```

这里,for 是关键字,圆括号中是三个合法的表达式,循环体可以是一条语句,也可以是复合语句。

for 循环流程图如图 5.7 所示,它的求解过程具体描述如下。



(a) 传统流程图　　(b) N-S图

图 5.7　for 循环流程图

(1) 求表达式 1 的值,它只执行一次。

(2) 求表达式 2 的值。若值为真(非 0),则执行循环体,然后执行第(3)步;若值为假(0),则跳出 for 循环,for 循环结束,继续执行 for 语句的下一条语句。

(3) 求表达式 3 的值,并转至第(2)步。

for 语句常用的简单形式如下:

```
for(循环变量赋初值; 循环条件; 循环变量增值) 循环体
```

需要说明的是:

(1) 循环变量赋初值对应表达式 1,循环条件对应表达式 2,循环变量增值对应表达式 3。例如:

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

(2) 表达式 1 也可以不是循环变量赋初值的表达式。例如:

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

(3) 表达式 3 也可以不是循环变量增值的表达式。例如:

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

(4) 表达式 1 和表达式 3 可以是一个简单表达式,也可以是一个逗号表达式。例如:

```
s=0;
for(m=0, n=0; m<10 && n<20; m++, n=n+2) s=s+m+n ;
```

(5) 表达式 2 可以是关系表达式、逻辑表达式、数值和字符表达式，只要其值为真(非 0)，就执行循环体，否则退出循环。例如：

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

由于循环条件为数值 10，是一个非 0 的数值，所以循环条件一直满足，可以永远循环下去。

现在，用 for 循环来完成引例 5-2 所要实现的功能，其流程图如图 5.8 所示。

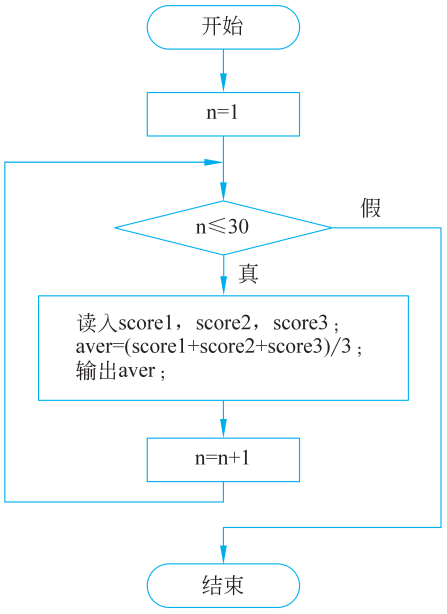


图 5.8　用 for 循环实现引例 5-2 的程序流程图

【程序代码】

```
1   #include <stdio.h>
2   void main()
3   {   float score1, score2, score3, aver;
4       int n;
5       for(n=1; n<=30; n++)
6       {   scanf("%f,%f,%f",&score1,&score2,&score3);
7           aver=(score1+score2+score3)/3;
8           printf("aver=%f\n",aver);
9       }
10  }
```

下面是两个简单且常用的程序，用 for 语句来实现。通过举例，让大家学习 for 语句的用法。

例如：求 s=1+2+3+4+…+100，用 for 语句实现。

【问题解析】

设循环变量 i=1,2,3,4,…,100，累加和变量为 s；初始值为 i=1，s=0，循环条件为 i<=100，循环变量增值为 i=i+1，循环体为 s=s+i。

(1) 用 for 语句实现的编码为

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