第5章　循环结构

教学目标：

(1) 掌握 for 循环结构。

(2) 掌握 while 和 do…while 循环结构。

(3) 掌握 continue 语句和 break 语句。

(4) 掌握循环的嵌套。

5.1　循环概述

在实际问题中，有许多具有规律性的重复操作。对应于程序，就需要重复执行某些语句，这种重复现象称循环。循环是程序设计语言中反复执行某些代码的一种计算机处理过程，常见的有按照次数循环和按照条件循环。循环结构一般由四部分组成：

(1) 循环条件(进入或退出循环的条件)。一般由控制循环的变量(称循环变量)的值来决定，循环变量就是在循环体内，使循环次数同步变化的变量。

(2) 循环初始值。一般包括两个部分：循环变量的初值和循环体中所需变量的初值。

(3) 改变循环控制变量值的语句。决定循环控制变量的增加或减少，一般是赋值语句。

(4) 循环体。需要完成的功能(一组被重复执行的语句)。

循环结构也称重复结构，实现程序循环结构的语句称为循环语句。循环结构的示意图如图 5-1 所示。

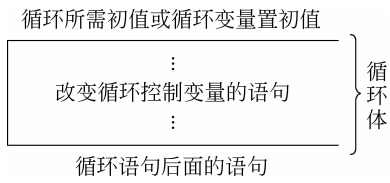


图 5-1　循环结构

C++ 中提供了三种循环语句：for 语句、while 语句、do…while 语句。

5.2　while 语句

while 语句是一种形式较为简单的循环语句，语法图如图 5-2 所示。

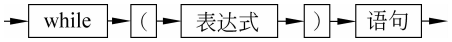


图 5-2　while 语句

语法格式：

```
while(表达式)
    语句
```

说明：

(1) while 为关键字，称 while 循环。

(2) 表达式是 while 循环的条件，它用于控制循环是否继续进行。

(3) 语句称 while 循环的循环体，它是被重复执行的代码行。while 循环的循环体可以是单条语句(不能省略语句后面的分号“;”)，也可以是由花括号括起来的复合语句。

while 语句的执行过程：

(1) 计算表达式的值，若此值不等于 0(即循环条件为“真”)，则转步骤(2)；若此值等于 0(即循环条件为“假”)，则转步骤(4)；

(2) 执行一遍循环体语句；

(3) 转步骤(1)；

(4) 结束 while 循环。

while 语句执行的 UML 活动图如图 5-3 所示。

while 语句将在表达式成立的情况下重复执行语句(循环体)；若在第一次进入 while 循环时表达式就不成立，则语句(循环体)一次也不会执行，即 while 的循环体执行 0 次以上。

【例 5-1】 用 while 循环计算 $\sum_{i=1}^{n} i = 1+2+3+\cdots+n$ 的累加和，其中 n 由键盘输入。

设计思路：需要先后将 n 个数相加。要重复 n−1 次加法运算，可用循环实现。由于后一个数是前一个数加 1 而得，加完上一个数 i 后，使 i 加 1 可得到下一个数，其 UML 活动图如图 5-4 所示。

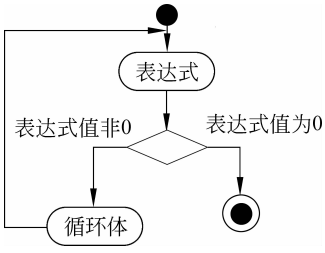


图 5-3　while 语句

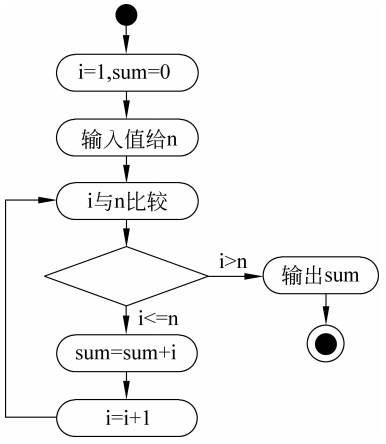


图 5-4　累加和

程序代码：

```
1. #include<iostream>
2. using namespace std;
3. int main( )
4. {//循环初始值
5.   int i=1,sum=0;
6.   int n;
7.   cin>>n;
8.  //循环条件
9.   while (i<=n)
10.    {//累加和,循环体
11.      sum=sum+i;
12.      i++;//控制变量增值
13.    }
14.  cout<<"sum="
15.     <<sum<<endl;
16. }
```

第5行，定义循环控制变量i并赋初值，sum循环体中累加和需要的初始值。
第9行，循环条件，i<=n成立时，执行循环体。
第10～13行，循环体，由两条语句构成：累加和改变控制变量值。

输入：100
输出结果：

```
sum=5050
```

【例5-2】 求最小公倍数和最大公约数。

最小公倍数：两个整数公有的倍数称为它们的公倍数，其中一个最小的公倍数是它们的最小公倍数，同样地，若干个整数公有的倍数中最小的正整数称为它们的最小公倍数。求最小公倍数算法：最小公倍数＝两整数的乘积÷最大公约数。

设计思路：设有两整数M和N，求最大公约数算法，其UML活动图由读者完成。

方法1　辗转相除法

(1) M % N得余数R；
(2) 若R＝0，则N即为两数的最大公约数，退出；
(3) 若R≠0，则M＝N，N＝R，再回去执行(1)。

如，求27和15的最大公约数过程为：
27÷15余12，15÷12余3，12÷3余0，故，3即为最大公约数。
程序代码：

```
1. #include<iostream>
2. using namespace std;
3. void main()   /*  求最大公约数 */
4. {
5.    int m, n, num1, num2, remainder;
6.    cout<<"输入两个整数:\n";
7.    cin>>num1>>num2;
8.    m=num1;
9.    n=num2;
```

```
10.    /*辗转相除法,余数不为0,继续相除,直到余数为0 */
11.    while(num2 !=0)
12.    {
13.       remainder=num1%num2;
14.       num1=num2;
15.       num2=remainder;
16.    }
17.    cout<<"最大公约数:"<<num1<<endl;
18.    cout<<"最小公倍数:"<<m * n / num1<<endl;
19. }
```

第11～16行,辗转相除法。循环变量初始值由键盘输入,第11行为循环条件,第12～16行由复合语句构成的循环体。
第15行,改变循环控制变量值。

运行结果:

```
输入两个整数:
27 15
最大公约数:3
最小公倍数:135
```

注意:

(1) 因while语句中缺少对循环控制变量进行初始化的结构,故在使用while循环之前对循环控制变量进行初始化。

如例5-1:

```
int i=1,sum=0;
```

(2) 在while循环体中不要忘记对循环控制变量的值进行修改,以使循环趋向结束。

如例5-1:

```
i++;
```

方法2　相减法

(1) 若M>N,则M=M−N;

(2) 若M<N,则N=N−M;

(3) 若M=N,则M(或N)即为两数的最大公约数,退出;

(4) 若M≠N,则再回去执行(1)。

如求27和15的最大公约数过程为:

27-15=12(15>12),15-12=3(12>3),12-3=9(9>3),9-3=6(6>3),6-3=3(3==3),故,3即为最大公约数。

方法3　穷举法

(1) MIN=M与N的最小的;

(2) 若M、N能同时被MIN整除,则MIN是最大公约数,结束;

(3) MIN--;

(4) 若MIN>0,则再回去执行(2)。

方法2、方法3的活动图和程序代码,请读者自行完成。

5.3 do…while 语句

while 循环遵循“先判断条件再执行循环体”。C++ 也提供了“先执行循环体再判断条件”的循环结构 do…while。它的功能类似于 while 语句，只是将循环的判定条件移到了循环体之后，语法图如图 5-5 所示。

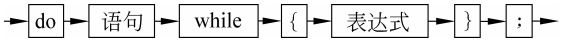


图 5-5 do…while 语句

语法格式：

```
do
    语句
while(表达式);
```

说明：

(1) 本循环中，do 和 while 为关键字，称 do…while 循环或 do 循环。

(2) 语句称 do…while 循环的循环体，它是要被重复执行的代码行，do…while 循环的循环体可以是单条语句(不能省略语句后面的分号“;”)，也可以是由花括号包围起来的复合语句。

(3) 表达式是 do…while 循环的条件，它用于控制循环是否继续进行。

do…while 语句的执行过程：

(1) 执行一遍循环体语句；

(2) 计算表达式的值，若此值不等于 0(即循环条件为“真”)，则转向步骤(1)；若此值等于 0(即循环条件为“假”)，则转向步骤(3)；

(3) 结束 do…while 循环。

do…while 语句执行的 UML 活动图如图 5-6 所示。

【例 5-3】 用 do…while 循环计算阶乘 $\prod_{i=1}^{n} i = n! = 1\times2\times3\times\cdots\times n$，其中 $n < 20$ 由键盘输入。

设计思路：需要先后将 n 个数相乘。要重复 n−1 次乘法运算，可用循环实现。由于后一个数是前一个数加 1 而得，乘完上一个数 i 后，使 i 加 1 可得到下一个数，其 UML 活动图如图 5-7 所示。

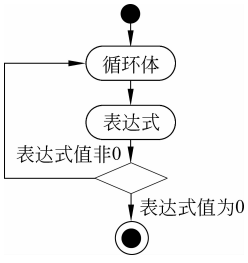


图 5-6 do…while 语句

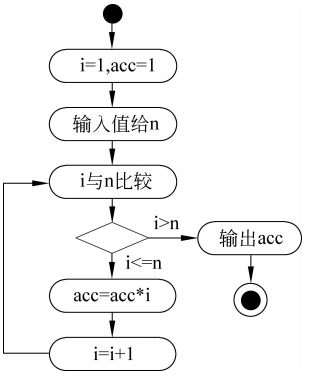


图 5-7 n! 活动图

程序代码：

```
1. #include<iostream>
2. using namespace std;
3. int main( )
4. {  //循环初始值
5.   int i=1, acc=1;
6.   int n;
7.   cin>>n;
8.
9.   do
10.    {  //循环体
11.      acc=acc * i;
12.      i++;     //控制变量增值
13.     //循环条件
14.  } while (i<=n);
15.   cout<<" acc="
16.       <<acc<<endl;
17. }
```

第 5 行，定义循环控制变量 i 并赋初值，sum 循环体中累乘积需要的初始值。
第 9 行，循环条件，i<=n 成立时，执行循环体。
第 10～13 行，循环体，由两条语句构成：累乘和改变控制变量值。

输入：5
输出结果：

```
acc=120
```

注意：

(1) 在使用 do…while 循环之前需要对循环控制变量进行初始化；在 do…while 循环中不要忘记对循环控制变量进行修改，以使循环结束。

(2) 在 do…while 语句中最后的分号不能丢掉，它用来表示 do…while 语句的结束。

5.4　for 语句

for 语句是 C++ 中最常用且功能最强的循环语句，语法图如图 5-8 所示。

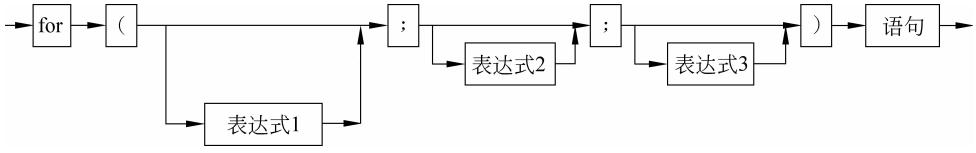


图 5-8　for 语句

语法格式：

```
for([表达式 1];[表达式 2];[表达式 3])
    语句
```

说明：

(1) for 为关键字，称 for 循环。

(2) 表达式 1 是 for 循环的初始化部分，一般为赋值表达式，可以是逗号表达式，给控制变量赋初值，还可以是变量声明。表达式 1 可省略。这时，应在 for 语句之前给循环控制变量赋初始值。

注意： 省略表达式 1 时，其后的分号不能省略。

(3) 表达式 2 是 for 循环的条件部分，可以是任何表达式，可以是逗号表达式，但一般是关系表达式或逻辑表达式，是循环的控制条件，条件成立(条件结果是 true 或表达式 2 的值非 0)。表达式 2 可省略，但其后面的分号不能省略。这时，for 语句将不再判断循环条件，循环会无限次地执行下去，即“死循环”。

如：

```
for(int i=0;; i++)
```

此时，等价于 while(true)，或 for(;|;)。

(4) 表达式 3 是 for 循环的循环控制变量的更改部分。一般为赋值表达式，可以是逗号表达式，更改循环控制变量的值。表达式 3 可省略，此时，应在循环中更改循环控制变量的值，以确保循环能够正常结束。

(5) 语句称为 for 循环的循环体，它是要被重复执行的代码行，for 循环体可以是单条语句(不能省略语句后面的分号“;”)，也可以是由花括号包围起来的复合语句。

(6) 三个表达式可同时省略，但两个分号“;”不能省略。这时，for 语句显然是“死循环”。如：for(;;)等价于 while(true)。

(7) 表达式 1、表达式 2 和表达式 3 都可以是任何类型的 C++ 表达式。

for 语句的执行过程：

(1) 计算表达式 1 的值。

(2) 计算表达式 2 的值，若此值不等于 0(循环条件为“真”，即成立)，则转向步骤(3)；若此值等于 0(循环条件为“假”，即不成立)，则转向步骤(5)。

(3) 执行一遍循环体语句。

(4) 计算表达式 3 的值，然后转向步骤(2)。

(5) 结束 for 循环。

for 语句的 UML 活动图如图 5-9 所示。

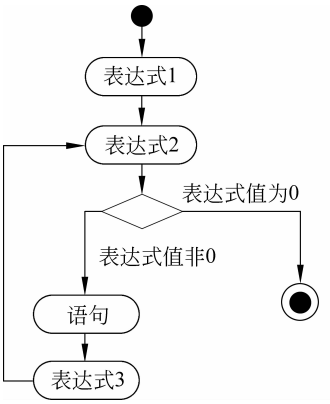


图 5-9 for 语句

【例 5-4】 若一对兔子每月生一对兔子；一对新生兔，从第二个月起就开始生兔子；假定每对兔子都是一雌一雄，试问一对兔子，一年能繁殖成多少对兔子？

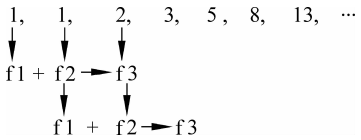
分析：先看前几个月的情况，第一个月有一对刚出生的兔子，即 F(1)=1；第二个月，这对兔子长成成年兔，即 F(2)=1；第三个月，这对成年兔生出一对小兔，共有两对兔子，即 F(3)=2；第四个月，成年兔又生出一对小兔，原出生的兔子长成成年兔，共有三对兔子，即 F(4)=3；第五

个月，原成年兔又生出一对小兔，新成年兔也生出一对小兔，共有五对兔子，即 F(5)＝5；……以此类推，可得每个月的兔子对数，组成数列：1，1，2，3，5，8，13，21，34，55，89，144，…，即斐波那契数列，其中的任一个数，都叫斐波那契数。于是有关系式：

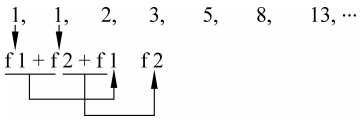
$$\begin{cases} F(1) = F(2) = 1 & (n = 1, n = 2) \\ F(n) = F(n-1) + F(n-2) & (n \geqslant 3) \end{cases}$$

设计思路：

方法 1：一次求一个值 f1＋f2→f3，f2 赋值给 f1，f3 赋值给 f2，示意图如下：



方法 2：一次求两个值 f1＋f2，得到新的 f1，新的 f1＋f2，得到新的 f2，示意图如下：



程序代码：

```
1. #include<iostream>
2. #include<iomanip>
3. using namespace std;
4.  int main()
5. {
6.     long f1, f2;
7.     int i;
8.    //一次输出两个值
9.     f1=f2=1;
10.    for(i=1; i<6; i++)
11.    {
12.       cout<<setw(12)
13.           <<f1<<setw(12)
14.           <<f2;
15.       //一行输出四个值
16.       if(i%2==0)
17.         cout<<endl;
18.
19.        f1=f1+f2;
20.        f2=f1+f2;
21.  }
22. return 0;
23. }
```

第 6 行，为表示范围更大，定义为 long 类型。

第 7～9 行是循环初值，第 7 行是循环控制变量初值。
第 11～21 行由复合语句构成的循环体。

第 12～14 行，为输出，setw(12)设定 f1、f2 输出都占 12 列。

第 16～17 行，确定每行输出 4 项。

注意：f2＝f1＋f2 中 f1 是 f1＝f1＋f2 中“＝”左边的 f1，f2＝f1＋f2 中“＝”右边的 f2 是 f1＝f1＋f2 中“＝”右边的 f2。

运行结果：

```
 1   1   2    3
 5   8  13   21
34  55  89  144
```

本题，重要思想是迭代，每一次迭代得到的结果会作为下一次迭代的初始值。

5.5 三种循环的比较与循环嵌套

1. 三种循环的比较

C++ 中的三种循环的比较，如表 5-1 所示。

表 5-1 三种循环的比较

	类型	循环体需要的初始值与循环控制变量初值	循环体被执行的次数	循环控制变量的更改位置	一般应用环境
while	当型	循环语句前	0 次以上	循环体中	不确定次数
do…while	直到		1 次以上		
for	当型	在表达式 1 中或循环前	0 次以上	表达式 3 或循环体中	确定次数

(1) 三种循环都可以用来处理同一问题，一般情况下它们可以互相代替。

(2) while 和 do…while 循环，是在 while 后面指定循环条件的，在循环体中应包含使循环趋于结束的语句(如 i++，或 i=i+1 等)。

do…while 循环的循环体在前，循环条件在后，故 do…while 循环体在任何条件下(即使不满足循环条件)都至少被执行一次。而 while 循环条件在前，循环体在后，当条件不满足时，循环体有可能一次也不会执行。这正是在构造循环结构时决定使用 while 语句还是 do…while 语句的重要依据。

for 循环可以在表达式 3 中包含使循环趋于结束的操作，甚至可以将循环体中的操作全部放到表达式 3 中。故 for 语句的功能更强，凡用 while 循环能完成的，用 for 循环都能实现。

(3) 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。

(4)三种循环都可以执行死循环，比如 do{}while(1) ;for(;1;);while(1)。

注意：

(1) for、do、while 等语句各自占一行，其他语句不得紧跟其后；不论该语句块中有多少行语句都要用“{}”括起来，这样可以防止书写失误。当准备开始书写语句块时，首先写下一对“{}”，然后再在里面书写语句，这样可以避免不易察觉的逻辑错误。

(2) 空循环体应使用{}或 continue，而不是一个简单的分号。

2. 循环嵌套

在一个循环结构中又完整地包含着另一个循环结构称循环的嵌套。C++ 中三种类型的循环语句都可以相互嵌套，并且嵌套的层数没有限制。编程中有许多问题需要使用

循环结构的嵌套来解决。

若循环语句的循环体中又出现循环语句，就构成多重循环结构，也称循环嵌套。一般常用的有二重循环和三重循环。二重循环结构如图 5-10 所示，其中的“框”表示循环，被嵌套的循环称为内循环，嵌套循环的循环称为外层循环。图中，外层循环里面包含两个并列的内层循环。内、外层循环的概念是相对的，如图 5-11 所示的三重循环，其中所谓的“中层循环”，对内层循环而言是外层循环，对外层循环而言是内层循环。

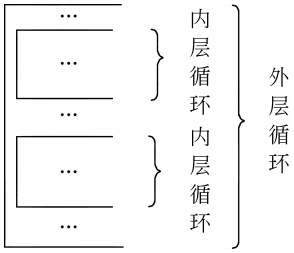


图 5-10　二重循环

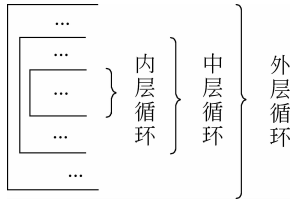


图 5-11　三重循环

循环层数越多，运行时间越长，程序越复杂。

三种循环（while 循环、do…while 循环和 for 循环）可以互相嵌套，如表 5-2 所示。

表 5-2　循环嵌套

		内层循环		
		while	do…while	for
外层循环	while	√	√	√
	do…while	√	√	√
	for	√	√	√

这里的“√”表示允许。即外层循环可以是 while 循环 do…while 循环和 for 循环三种循环中的一种，内层循环也可以是 while 循环 do…while 循环和 for 循环三种循环中的一种，这样就构成表 5-2 中的九种情况。

【例 5-5】 输出如下图形。

```
    *
   ***
  *****
 *******
*********
```

设计思路：观察上图，实际该图由空格（在计算机中，空格是一个重要的常用的可输出的字符，但不像其他可输出字符观察明显）和星号组成。为使上述图形设计清楚，这里用字符“-”（代替空格）和星号（*）组成：

```
----*        i=1时，由4个空格和1个*组成
---***       i=2时，由3个空格和3个*组成
--*****
-*******
*********    i=n=5时，由0个空格和2i-1个*组成
```