第3章 语句与控制结构

CHAPTER 3

C++程序由一系列语句组成，这些语句按照一定的控制结构组织起来，按逻辑顺序运行。控制结构包括不同选择分支的计算与处理，以及对相同语句组的循环执行。C++语言提供了判断、循环和转移等语句来控制程序。

学习目标

- 掌握 if、switch 分支语句的使用；
- 掌握 for、while、do…while 循环语句的使用，并能根据不同要求灵活选择不同的循环语句，熟练使用循环的嵌套；
- 掌握 break、continue 语句的联系和区别，根据不同场合正确地使用；
- 了解 goto 语句的转移用法。

3.1 语句格式

C++的语句类似于自然语言中的句子，它由不同类型的数据组成。数据类型是程序中最基本的元素，而语句则是程序中可以独立执行的最小单元。C++的语句以分号作为结束标记，相当于自然语言中的句号。语句通常由表达式末尾加上分号构成，C++认为不执行任何操作的空语句是合法的。

例如：

```
;                                   //不执行任何操作的空语句
a + b;                              //一般表达式语句
a = a + b;                          //赋值语句
```

空语句一般在循环结构中用来延迟一段时间，空语句的分号是必不可少的。分号的存在是语句成立的必要条件。

在C++中，声明变量的语句称为声明语句，声明语句可以出现在程序中任何可以出现语句的地方，这提高了变量声明的灵活性。但是在C语言中，变量的声明不是语句，只能在模块的首部集中声明。

某些情况下，可以用一对花括号将多条语句括起来组成一个块语句。

例如：

```
{
 int a = 6, b = 3;
```

```
 a = (a + b)/2;
 cout <<"a = "<< a << endl;
}
```

块语句的右括号后没有分号，块语句一般存在于选择和循环结构中。

3.2 控制结构

C++是解决客观世界实际问题的工具，它对每个具体问题的解决是通过不同算法实现的。例如：计算 1＋2＋…＋100 的累加和。

```
int i = 1,sum = 0;
for ( i = 1;i <= 100;i++)
{
  sum = sum + i;
}
```

上例是使用循环结构进行 100 次累加的算法。算法是解决问题的步骤序列，合法的算法具有如下特征：

（1）有穷性：一个算法必须保证执行有限步之后结束；

（2）确定性：算法的每一步必须有确切的定义，不能出现二义性；

（3）输入：一个算法有 0 个或多个输入；

（4）输出：一个算法有 1 个或多个输出，没有输出的算法是毫无意义的；

（5）可行性：算法原则上能够精确地运行。

可以使用多种方法对算法进行描述，如自然语言、流程图、判定表、判定树、伪代码等。其中流程图是使用最广泛的表示方法，它具有简洁、直观、准确的特点，流程图的常用符号如图 3-1 所示。

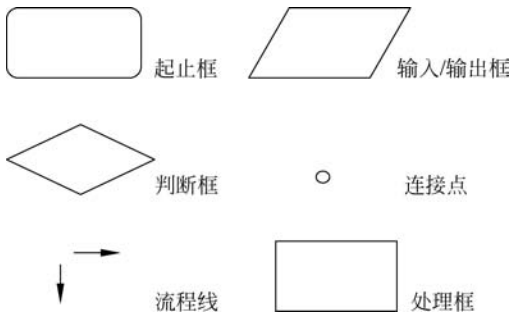

图 3-1　流程图的常用符号

C++的控制结构包括顺序结构、选择结构和循环结构。

（1）顺序结构是最简单的结构，其特点是各个块按照先后次序依次执行。

（2）选择结构是先对条件进行判断，然后选择执行相应语句。C++的选择结构主要是 if 语句和 switch…case 语句。

（3）循环结构也是先对条件进行判断，然后执行语句，并且在执行完语句后继续进行下一次的条件判断，直到条件为假时才不执行结构中的语句。C++的循环结构主要包括 for 语

句、while 语句和 do…while 语句。

3.3　if 语句

if 语句属于判断选择结构的语句。C++的判断选择结构也称为条件分支结构，一般包括 if 语句和 switch 语句。

3.3.1　基本 if 语句

基本 if 语句是最简单的条件语句，其功能是根据指定的条件判断是否执行相应的语句。其语法格式如下：

```
if (条件表达式)
语句;
```

其中条件表达式一般是一个关系或者逻辑表达式，其值或者为 true，或者为 false，并且必须写在一对圆括号中；语句可以是单语句也可以是多条语句组成的语句块(块语句)，多条语句组成的语句块必须包含在一对花括号中，单条语句可以包含在花括号中也可以没有花括号。但是为了程序编写的整洁、规范，建议即使是单语句也将其包含在一对花括号中。

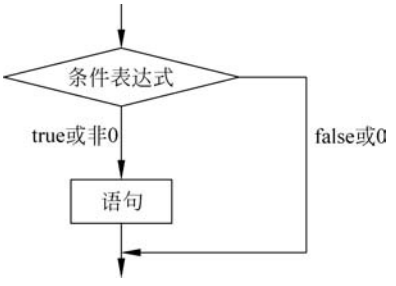

图 3-2　基本 if 语句流程

if 语句执行过程为：首先计算条件表达式的值，如果值为 true，则执行语句；否则跳过语句，执行 if 语句的后继语句，基本 if 语句流程如图 3-2 所示。条件表达式也可以是常量，0 表示 false，非 0 表示 true。

【例 3-1】 从键盘输入学生成绩，如果大于或等于 60 分，输出“通过”。

```
1   /**************************************************
2                       程序文件名:Ex3_1.cpp
3                       基本 if 语句的使用
4    **************************************************/
5   #include <iostream>
6   using namespace std;
7   main()
8   {
9     int n;
10      cout <<"Please enter the score:"<< endl;
11      cin >> n;
12      if (n > = 60)                    //条件表达式为 n >
 = 60
13        {
14          cout <<"通过"<< endl;
15        }
16  }
```

```
Please enter the score:
66
通过
Press any key to continue
```
图 3-3　例 3-1 程序运行结果

程序运行结果如图 3-3 所示。

例 3-1 中的整型变量 n 用来保存输入的成绩，第 12 行语句 if(n>=60)的判断条件是表达式 n>=60，如果成立，则从第 13 行开始执行语句，输出“通过”；否则从第 12 行直接跳转到第 16 行。

3.3.2 if…else 语句

if…else 语句的语法格式如下：

```
if (条件表达式)
 语句 1;
else
 语句 2;
```

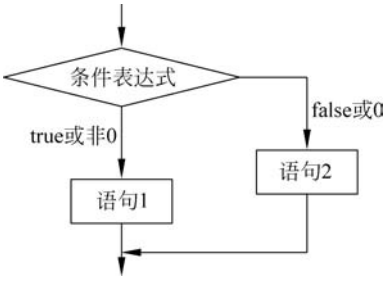

图 3-4 if…else 语句流程

其执行过程为：首先计算条件表达式的值，如果值为 true，则执行语句 1；否则执行语句 2。if…else 语句流程如图 3-4 所示。

else 前面的部分称为 if 分支子句，else 及其后的部分称为 else 分支子句；虽然 if 分支的语句 1 和 else 分支的语句 2 在形式上是独立的两条语句，但是在语法上，从“if”开始到“语句 2；”结束的整体是一条不可分割 if 语句。

【例 3-2】 从键盘输入学生成绩，如果大于或等于 60 分，输出“通过”，否则输出“不通过”。

```
1  /*************************************************
2                    程序文件名:Ex3_2.cpp
3                    if…else 语句的使用
4   *************************************************/
5  #include <iostream>
6  using namespace std;
7  main()
8  {
9    int n;
10   cout<<"Please enter the score:"<<endl;
11   cin>>n;
12   if (n>=60)
13      {                          //第 13～15 行是语句 1
14         cout<<"通过"<<endl;
15      }
16   else
17      {                          //第 17～19 行是语句 2
18         cout<<"不通过"<<endl;
19      }
20 }
```

```
Please enter the score:
56
不通过
Press any key to continue
```
图 3-5 例 3-2 程序运行结果

程序运行结果如图 3-5 所示。

在例 3-2 中，第 13～15 行语句块充当了语句 1；第 17～19 行的语句块充当了语句 2。如果条件表达式 n>=60 成立，则从第 13 行开始执行语句 1，输出“通过”；否则从第 17 行开始执行语句 2，输出“不通过”。if…else 语句中的关键字“else”是必不可少的。

3.3.3　嵌套 if 语句

如果遇到的问题有多重选择，仅使用 if 语句或 if…else 语句无法解决。为了解决多重选择问题，C++提供了嵌套的 if 语句。在 if 语句中，如果分支子句也是 if 语句，就构成了嵌套的 if 语句。

在形式上，嵌套的 if 语句有两种形式：一种是嵌套在 else 分支中，另一种是嵌套在 if 分支中。

1. else 分支的嵌套

else 分支嵌套的 if 语句语法格式为：

```
if(条件表达式 1)
   语句 1;
else if (条件表达式 2)
   语句 2;
  …
    else if (条件表达式 n)
   语句 n;
else
   语句 n + 1;
```

else 分支的嵌套图表示如图 3-6 所示。

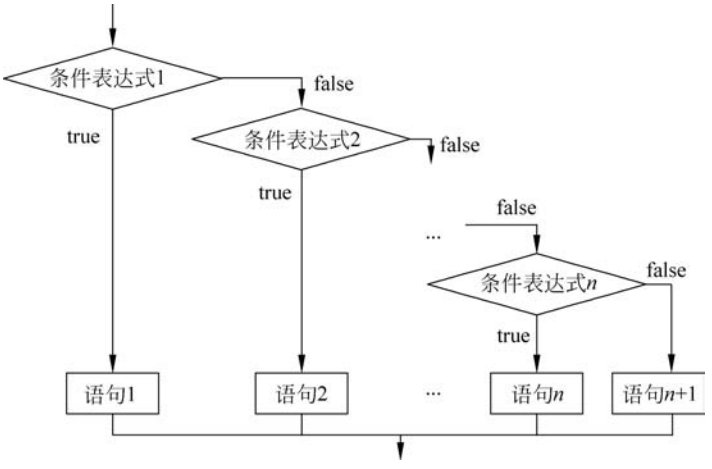

图 3-6　else 分支的嵌套

【例 3-3】 用嵌套 if 语句将百分制成绩按等级输出。

```
1   /**************************************************
2                       程序文件名:Ex3_3.cpp
3                       嵌套 if 语句的使用
4    **************************************************/
5   #include <iostream>
6    using namespace std;
7    main()
8    {
9    int n;                          //变量 n 用来存储从键盘输入的成绩
```

```
10    char grade;
11    cout <<"Please enter the score:"<< endl;
12    cin >> n;
13    if (n < 60)
14        grade = 'E';
15    else if(n < 70)
16        grade = 'D';
17    else if(n < 80)
18        grade = 'C';
19    else if(n < 90)
20        grade = 'B';
21    else if(n < = 100)
22        grade = 'A';
23    else
24        grade = 'F';
25    cout <<"The grade is "<< grade << endl;
26  }
```

在例 3-3 中，首先在第 13 行判断关系表达式 n＜60，如果结果为真，则执行第 14 行，把字符'E'赋给 grade，然后跳过其他语句，直接执行第 25 行的语句。这种嵌套的 if 语句，只允许选择满足条件的一个分支，其他分支不再判断执行。如果第 13 行结果为假，也就是 n≥60，则跳过第 14 行而执行第 15 行的判断。第 24 行表示如果输入的成绩大于 100，grade 的值为'F'。

2. if 分支的嵌套

if 分支嵌套的 if 语句语法格式为：

```
if(条件表达式 1)
   if (条件表达式 2)
      语句 1;
   else
      语句 2;
else
   语句 3;
```

if 分支的嵌套图如图 3-7 所示。

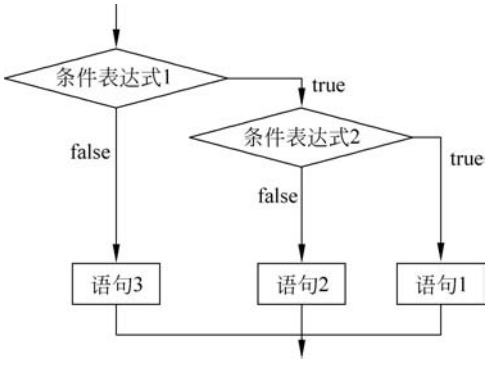

图 3-7　if 分支的嵌套图

使用 if 分支的嵌套形式修改例 3-3 中相应的语句为：

```
if (n>=60)
  if(n>=70)
    if(n>=80)
      if(n>=90)
         grade='A';
      else
        grade='B';
    else
      grade='C';
  else
    grade='D';
else
  grade='E';
```

在使用 if 分支嵌套时，要注意 else 与 if 的匹配原则：else 与上方离它最近的未匹配过的 if 相匹配。

3.3.4　条件运算符

当 if…else 语句是简单语句时，可用条件运算符“?：”来进行简化，其语法格式为：

```
(条件表达式)?语句 1: 语句 2
```

例如，求两个数 a 和 b 中较大的值，使用 if 语句：

```
if(a>b)
   max=a;                          //语句 1
else
   max=b;                          //语句 2
```

将 if 语句替换成等价的条件运算符为：

```
max=(a>b)?a:b;
```

3.4　switch 语句

嵌套的 if 语句能够实现多分支选择，C++还提供了另外一种实现多分支选择的结构——switch 语句。switch 语句也称为开关语句，它根据给定的条件，从多个分支中选择一条语句作为执行的入口。switch 语句虽然与嵌套的 if 语句功能类似，但是 switch 语句每次都计算同一表达式的值，而嵌套的 if 语句要分别计算各个 if 分支语句表达式的值，所以在处理多分支选择问题时，switch 语句更加简便、直观。其语法格式如下：

```
switch(表达式)
{
  case 常量表达式 1:[语句块 1;][break;]
  case 常量表达式 2:[语句块 2;][break;]
  …
  case 常量表达式 n:[语句块 n;][break;]
```

```
  [default:语句块 n+1;]
}
```

（1）switch 后面圆括号中的“表达式”只能是整型、字符型、枚举型或布尔型等离散类型，而不能是实型(float、double 型)等连续类型。

（2）“case”起到标号的作用，其后“常量表达式”的类型必须与“表达式”的类型匹配，并且所有 case 后常量表达式的值不能重复。

（3）当“表达式”的值与某一个 case 后“常量表达式”的值相等时，就执行这个 case 后的语句；如果所有 case 后“常量表达式”的值都不与“表达式”的值匹配时，就执行“default”后的语句。

（4）符号“[ ]”表示其中的内容是可选的；语句块、break 和 default 都是可选的，语句块由一条语句或者一个复合语句组成。

switch 语句的执行过程为：

① 计算 switch 后表达式的值；

② 将表达式的值依次与 case 后常量表达式的值相比较，如果表达式的值与某一常量表达式的值相等，则执行该 case 后的语句块，直到遇到 break 或 switch 语句的右花括号；

③ 如果表达式的值与 case 后所有常量表达式的值都不相等，则执行 default 后的语句；如果 switch 语句不包含 default，则不执行任何操作。

【例 3-4】 用 switch 语句将百分制成绩按等级输出。

```
1   /**************************************************
2                       程序文件名:Ex3_4.cpp
3                       switch语句的使用
4    **************************************************/
5   #include<iostream>
6   using namespace std;
7   main()
8   {
9     int n;
10    cout<<"Please enter the score:"<<endl;
11    cin>>n;
12    switch (n/10)
13    {
14        case 10:
15            cout<<"The grade is A"<<endl;
16            break;
17        case 9:
18            cout<<"The grade is A"<<endl;
19            break;
20        case 8:
21            cout<<"The grade is B"<<endl;
22            break;
23        case 7:
24            cout<<"The grade is C"<<endl;
25            break;
26        case 6:
27            cout<<"The grade is C"<<endl;
```

```
28              break;
29          default:
30              cout <<"The grade is D"<< endl;
31      }
32  }
```

```
Please enter the score:
98
The grade is A
Please enter the score:
88
The grade is B
Please enter the score:
78
The grade is C
Please enter the score:
66
The grade is C
Please enter the score:
56
The grade is D
```

图 3-8　例 3-4 运行结果

程序运行结果如图 3-8 所示。

上例中，当 n/10 的值为 10 或 9 时，case 后的语句相同，输出等级都是"A"；同样，当 n/10 的值为 7 或 6 时，case 后的语句也相同，输出等级都是"C"。C++规定，当多个 case 共用相同语句时，可以对相同语句进行简化处理，如例 3-4 中的第 14～19 行代码可替换为：

```
case 10:
case 9:
     cout <<"The grade is A"<< endl;
```

同理，第 23～27 行代码也可替换为：

```
case 7:
case 6:
     cout <<"The grade is C"<< endl;
```

但是下面写法是错误的：

```
case 10, 9:
     cout <<"The grade is A"<< endl;
```

如果 switch 语句中 case 标号后省略了 break，则程序会从第 1 个匹配的 case 开始不加判断地执行 switch 语句中剩下的所有语句，直到遇到第 1 个 break 为止。例如：

```
1   switch(a)
2   {
3     case'A':
4         cout <<" It is A"<< endl;
5     case 'B':
6         cout <<" It is B"<< endl;
7     case'C':
8         cout <<" It is C"<< endl;
9      break;
10  default:
11     cout <<" It is D"<< endl;
```

当字符变量 a 的值为'B'时，程序的执行结果为：

```
It is B
It is C
```

如果去掉第 9 行的 break；则程序的结果变为：

```
It is B
It is C
It is D
```

3.5 for 循环语句

在C++程序中，常常会出现在满足给定条件下，需要重复执行相同操作的情况。这种重复执行相同的操作就是循环，C++提供了3种实现循环的语句：for语句、while语句和do…while语句。在循环语句中，被重复执行的操作叫作循环体，循环体可以由一条语句组成，也可以由一个语句块或空语句组成。

3.5.1 for 语句

for语句的语法格式如下：

```
for(表达式1;表达式2;表达式3)
   循环体
```

其中，for是关键字；表达式1用来给循环变量赋初值，表达式1在循环变量中仅执行一次；表达式2是循环继续进行的条件；表达式3是对循环变量的值进行修改。

for语句的执行过程为：

① 计算表达式1的值；

② 计算表达式2的值，如果值为false或0，则结束循环，转到⑥；

③ 如果表达式2的值为true或非0，则执行循环体；

④ 计算表达式3的值；

⑤ 转到②；

⑥ 执行for语句的后续语句。

for语句流程如图3-9所示。

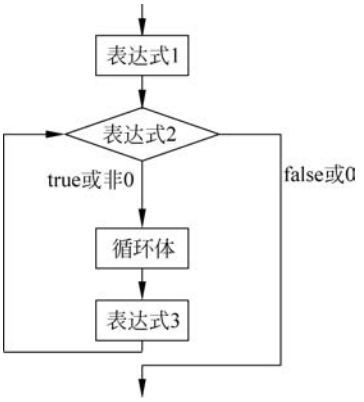

图3-9 for语句流程

for语句不仅可以用于循环次数确定的情况，还可以用于循环次数虽不确定但给出了循环继续条件的情况。

【例3-5】 用for语句计算1＋2＋3＋…＋100。

```
/*****************************************************
                    程序文件名:Ex3_5.cpp
                    for 语句计算 1 + 2 + 3 + … + 100
*****************************************************/
#include <iostream>
using namespace std;
main()
{
   int i,sum = 0;                          // 将累加变量 sum 初始化为 0
     for(i = 1; i <= 100; i++)             // i 为循环变量,循环次数为 100
   {
       sum += i;                           // 实现累加
   }
    cout <<"sum = "<< sum << endl;
```

```
15  }
```

```
sum= 5050
```

图 3-10　例 3-5 运行结果

程序运行结果如图 3-10 所示。

根据程序的不同要求，对 for 语句中的表达式可以进行相应默认。

(1) 表达式 1 可以默认。由于表达式 1 的作用是给循环变量赋初值，所以若默认表达式 1，则应在 for 语句之前有给循环变量赋初值的语句。虽然默认表达式 1，但是其后的分号不能默认。

例如：

```
int a = 0;
for( ; a < 100 ; a++)                    //默认表达式 1,但分号必须保留
   sum += a;
```

(2) 表达式 2 可以默认。表达式 2 的作用是判断循环继续进行的条件，若默认表达式 2，则循环会无条件永远进行下去。虽然默认表达式 2，但是其后的分号同样不能默认。

例如：

```
for(a = 0; ;a++)                         //默认表达式 2,但分号必须保留
   sum += a;
```

等价于：

```
for(a = 0;1 ;a++)                        //默认表达式 2 相当于循环条件永远为真
   sum += a;
```

(3) 表达式 3 可以默认。表达式 3 的作用是对循环变量进行修改，因此若默认表达式 3，则在每次循环结束后，应该有对循环变量进行修改的语句，否则循环一旦开始运行则永远无法正常结束。

例如：

```
for(a = 0;a < 100 ; )                    //默认表达式 3,循环变量的值永远为 0,程序将永远进行下去
   sum += a;
```

若默认表达式 3 并让程序能够正常结束，则需要添加如下修改循环变量的语句：

```
for(a = 0;a < 100 ; )
{
   sum += a;
   a++;                                  //使循环变量 a 自增 1
```

for 语句不但可以默认 3 个表达式中的一个，而且可以同时默认 3 个表达式中的任意两个，C++认为最极端的同时默认 3 个表达式的写法也是合法的。不管表达式默认与否，for 语句圆括号中的两个分号缺一不可。

3.5.2　for 语句的循环嵌套

在一个循环中又包含其他循环的结构称为循环嵌套。for 语句可以实现循环嵌套。

【例 3-6】 用 for 语句实现在屏幕上输出以下图案：

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

```
1  /*************************************************
2                    程序文件名:Ex3_6.cpp
3                    for 语句实现循环嵌套
4   *************************************************/
5  #include <iostream>
6  using namespace std;
7  main()
8  {
9   int i,j,k;
10 for (i=0;i<=3;i++)                    //外重循环,循环 4 次,显示 4 行
11 {
12   for (j=0;j<=2-i;j++)                //内重循环
13   {
14     cout<<" ";
15   }
16   for (k=0;k<=2*i;k++)                //内重循环
17   {
18     cout<<" *";
19   }
20     cout<<"\n";                       //外重循环中的语句
21 }
22 }
```

【程序解释】

(1) 在例 3-6 中,共有两重循环嵌套,第 10 行的 for 语句对应的是外重循环,主要控制对应图案显示的行数。

(2) 第 12 行和第 16 行的 for 语句对应的都是内重循环,分别显示每一行的空格符和“*”。

(3)第 20 行是外重循环的语句,用来实现每行输出结束后换行,其中“\n”的作用和 endl 的作用完全相同。

3.6 while 循环语句

3.6.1 while 语句

while 循环语句也称为当型循环,当满足判断条件,则执行循环；语句的语法格式如下：

```
while(条件表达式)
    循环体
```

其中,while 是关键字；条件表达式是 C++中的一个合法表达式,表示是否执行循环体的判断条件。循环体可以是一条语句,也可以是复合语句。如果循环体是一条语句,则其两边有无花括号皆可；如果循环体是复合语句,则复合语句必须被包含在一组花括号中。

while 语句的执行过程为：

① 计算条件表达式的值，如果值为 false 或 0，则结束循环，转到④；

② 如果条件表达式的值为 true 或非 0，则执行循环体；

③ 转到①；

④ 执行 while 语句的后续语句。

while 语句流程如图 3-11 所示。

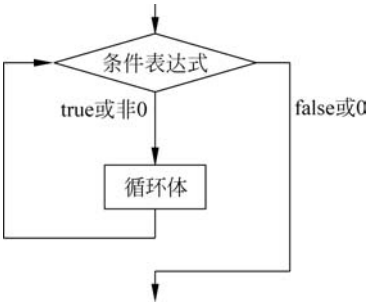

图 3-11　while 语句流程

【例 3-7】 用 while 语句计算 1＋2＋3＋…＋100。

```
1   /**************************************************
2                   程序文件名:Ex3_7.cpp
3                   while 语句计算 1 + 2 + 3 + … + 100
4    **************************************************/
5   #include <iostream>
6   using namespace std;
7   main()
8   {
9    int i = 1,sum = 0;
10   while(i <= 100)                    // 判断 i < 100 是否成立
11   {
12     sum += i;                        // 实现累加
13     i++;
14   }
15   cout <<"sum = "<< sum << endl;
16  }
```

```
sum= 5050
```

图 3-12　例 3-7 运行结果

程序运行结果如图 3-12 所示。

(1) 对比例 3-5 和例 3-7 可以看出，while 前“i＝1”的作用相当于 for 语句的表达式 1，用来给循环变量赋初值。

(2) while 语句中条件表达式“i＜＝100”的作用相当于 for 语句的表达式 2。

(3) while 语句中“i＋＋”的作用相当于 for 语句的表达式 3，用来修改循环变量。

(4) while 语句和 for 语句可以相互替换使用。

3.6.2　do…while 语句

do…while 循环语句也称为直到型循环，语句的语法格式如下：

```
do
  循环体
while(条件表达式);
```

其中，do 和 while 都是关键字；循环体可以是一条语句，也可以是复合语句；条件表达式是 C++中的一个合法表达式，给出是否继续执行循环体的判断条件。

do…while 语句的执行过程为：

① 执行循环体；

② 计算条件表达式的值，如果值为 false 或 0，则结束循环，转到④；

③ 如果条件表达式的值为 true 或非 0，则转到①；

④ 执行 do…while 语句的后续语句。

do…while 语句流程如图 3-13 所示。

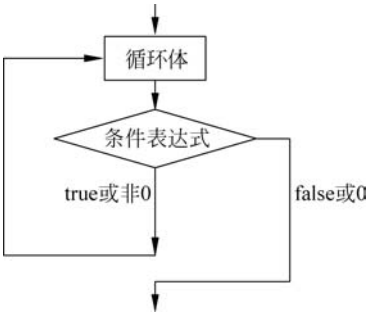

图 3-13　do…while 语句流程

【例 3-8】 用 do…while 语句计算 1＋2＋3＋…＋100。

```
/ ************************************************
                    程序文件名:Ex3_8.cpp
                    do…while 语句计算 1 + 2 + 3 + … + 100
  ************************************************ /
#include < iostream >
using namespace std;
main()
{
 int i = 1, sum = 0;
 do                                    // 不判断条件,先执行一次循环体
 {
   sum += i;                           // 实现累加
   i++;
 } while(i < = 100);                   //判断 i < = 100 是否成立,分号不能漏掉
 cout <<"sum = "<< sum << endl;
}
```

```
sum= 5050
```

图 3-14　例 3-8 运行结果

程序运行结果如图 3-14 所示。

(1) 对比 do…while 语句和 for 语句、while 语句可以看出，不管条件表达式是否成立，do…while 语句中循环体最少执行次数为 1，而 for 语句和 while 语句的最少执行次数为 0。

(2) do…while 语句条件表达式后的分号一定不能漏掉。

3.7　转移语句

C++语言提供了一些用以改变程序中语句执行顺序的转移语句，转移语句能够使程序从某一条语句有目的地转移到另一条语句执行。常用的转移语句有 break 语句、continue 语句和 goto 语句。其中 break 语句和 continue 语句是半结构化的控制语句，goto 语句是非结构化的控制语句。

3.7.1　break 语句

break 语句用在 switch 语句和所有循环语句中。

break 语句在 switch 语句中，用来跳出 switch 语句，继续执行 switch 后的语句；在循环语句中，用来跳出包含它的最内层循环体。例如，下面的代码在执行了 break 语句后，继续执行“sum＝100”处的语句，而不是跳出所有循环：

```
for(i = 0;i < 100;i++)
{
  for(j = 0;j < i;j++)
  {
      if(j > 100)
      {
        break;
      }
      sum += j;
  }
  sum = 100;
}
```

【例 3-9】 输出 10～100 的全部素数(素数 i 是指除 1 和 i 之外，不能被 2～(i－1)的任何整数整除的数)。

```
1   /**************************************************
2                       程序文件名:Ex3_9.cpp
3                       输出 10～100 的全部素数
4    **************************************************/
5   #include <iostream>
6   using namespace std;
7   main()
8   {
9    int i = 11, j, counter = 0;
10   for( ; i <= 100; i += 2)                //外循环:为内循环提供一个整数 i
11   {
12     for(j = 2; j <= i - 1; j++)           //内循环:判断整数 i 是否是素数
13     {
14       if(i % j == 0)                      //i 不是素数:因为能被 2～(i－1)的某个数整除
15           break;                          //强行结束内循环,执行第 17 行语句
16     }
17     if(counter % 10 == 0)                 //输出 10 个数后换行
18       cout <<"\n";                        //换行
19      if(j >= i)                           //整数 i 是素数:输出后计数器加 1
20      {
21       cout <<" "<< i;
22       counter++;
23      }
24   }
25   cout <<"\n";
26  }
```

程序运行结果如图 3-15 所示。

图 3-15　例 3-9 运行结果

【程序解释】

(1) 在例 3-9 中，共有两重循环嵌套，第 10 行的 for 语句对应外重循环，第 12 行的 for 语句对应内重循环。

(2) 当执行到第 15 行的 break 时，跳出内重循环，继续执行第 17 行。

3.7.2　continue 语句

continue 语句仅用在循环语句中。

continue 语句的功能是从包含 continue 的最内层循环体的当前位置，跳过循环体中本次循环尚未执行的语句，接着进行下一次是否执行循环的判定。也可以这样理解：在循环体中如果遇到了 continue 语句，则立即结束循环体的本次循环，接着开始下一次循环的判定。

例如：

```
1    for(int i = 100;i <= 200;i++)
2    {
3      if(i % 3 == 0)                          //判断 i 能否被 3 整除
4        continue;
5      cout << i << endl;
6    }
```

这段代码用来输出 100～200 所有不能被 3 整除的整数。

其执行过程为：

① 判断表达式 2(即 i＜＝200)是否成立；

② 如果值为 false 或 0，则结束循环，转到⑧；

③ 如果值为 true 或非 0，则转到④；

④ 判断“i%3＝＝0”是否成立，如果不成立，转到⑤，否则转到⑥；

⑤ 执行第 5 行，转到⑦；

⑥ 执行第 4 行“continue”，转到⑦；

⑦ 计算表达式 3(即 i＋＋)的值，转到①；

⑧ 执行 for 的后续语句。

continue 语句与 break 语句的区别是：遇到 continue 语句，立即结束本次循环，但不终止包含它的整个循环；遇到 break 语句则立即结束循环，并且终止包含它的整个循环。

3.7.3　goto 语句

goto 语句使程序跳转到语句标号所指的位置开始执行，其格式如下：

goto 语句标号;

语句标号属于标识符,其形式如下:

语句标号: 语句

这里的语句可以是任何语句,如 while 语句、for 语句等。

在 C++中。goto 语句只能用在函数体。在同一函数体中,语句标号应该是唯一的。例如:

```
i = 1,sum = 0;
wen: sum += i;
if(i <= 100)
   goto wen;
cout <<"sum is"<< sum < endl;
```

编译器遇到了 goto 语句会无条件地转向语句标号后的语句。但是在循环语句中,只能利用 goto 语句从循环体内跳出,而不能从循环体外跳到循环体内。

3.8　实例应用与剖析

【例 3-10】 输出 100～200 所有不能被 3 整除也不能被 7 整除的整数。

```
1   /**************************************************
2                       程序文件名:Ex3_10.cpp
3                       for 语句求素数
4    **************************************************/
5   #include <iostream>
6   using namespace std;
7   main()
8   {
9      cout <<"100～200 不能被 3 和 7 整除的数为\n";
10     for(int i = 100;i <= 200;i++)
11     {
12       if(i % 3!= 0&&i % 7!= 0)
13         cout << i <<" ";
14     }
15    cout <<"\n";
16  }
```

程序运行结果如图 3-16 所示。

```
100-200不能被3和7整除的数为
100     101     103     104     106     107     109     110     113     115
116     118     121     122     124     125     127     128     130     131
134     136     137     139     142     143     145     146     148     149
151     152     155     157     158     160     163     164     166     167
169     170     172     173     176     178     179     181     184     185
187     188     190     191     193     194     197     199     200
Press any key to continue
```

图 3-16　例 3-10 运行结果

【程序解释】

（1）第 10 行使用了 for 语句，表达式 1“int i＝100”在定义循环变量 i 的同时给其赋初值 100。

（2）for 语句从第 10 行开始，到第 14 行结束，循环体循环 201 次。


视频

【例 3-11】 中国古代数学史上著名的百钱买百鸡问题：“今有鸡翁一，值钱五；鸡婆一，值钱三，鸡雏三，值钱一。凡百钱买鸡百只，问鸡翁、鸡婆、鸡雏各几何？”

```
1   /*************************************************
2                      程序文件名:Ex3_11.cpp
3                      嵌套 for 语句计算百钱买百鸡问题
4    *************************************************/
5   #include<iostream>
6   using namespace std;
7   main()
8   {
9     int cock,hen,chick;
10    cout<<" 鸡翁 鸡婆 鸡雏"<<endl;
11    for(cock=0;cock<=20;cock++)
12      for(hen=0;hen<=33;hen++)
13      {
14          chick=100-cock-hen;
15          if(cock*5+hen*3+chick/3==100 && chick%3==0)
16            cout<<" "<<cock<<" "<<hen<<" "<<chick<<endl;
17      }
18  }
```

程序运行结果如图 3-17 所示。

```
鸡翁  鸡婆  鸡雏
 0    25    75
 4    18    78
 8    11    81
12     4    84
```

图 3-17　例 3-11 运行结果

【程序解释】

（1）这是双重 for 循环嵌套问题。

（2）由 cock＋hen＋chick＝100 和 5＊cock＋3＊hen＋chick/3＝100 可知，百钱最多能买鸡翁 20 只或鸡婆 33 只，这与第 11 行和第 12 行的两个循环继续条件相对应。

（3）第 15 行的判断条件为鸡翁、鸡婆和鸡雏的总和为 100，并且鸡雏的数目要被 3 整除。

【例 3-12】 编写程序，从键盘输入一个整数，逆向输出其各位数字，同时求出其位数以及各位数字之和。

```
1   /*************************************************
2                      程序文件名:Ex3_12.cpp
3                      while 语句求逆序问题
4    *************************************************/
5   #include<iostream>
6   using namespace std;
7   main()
8   {
9     int num,sum=0,k,i=0;
10    cout<<"请输入一个整数:";
11    cin>>num;
12    cout<<"逆序为:";
```

```
13    while(num > 0)
14    {
15      k = num % 10;
16      cout << k;
17      sum += k;
18      i++;
19      num/ = 10;
20    }
21   cout <<"\n 各个位数数字之和为:"<< sum << endl;
22   cout <<"你输入的是:"<< i <<<" 位数"<< endl;
23  }
```

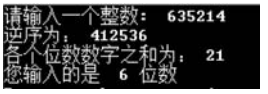

图 3-18　例 3-12 运行结果

程序运行结果如图 3-18 所示。

【程序解释】

(1) 根据提示输入整数,如“635214”,编译器会自动生成逆序“412536”以及各位数字之和“21”。

(2) num 存储被输入的整数；第 15 行“k＝num％10”,表示每次循环时,k 变量的值是 num 的个位数字。

(3) 每次循环结束后,num＝num/10,使编译器能够依次获取每位数字；编译器将每次获取的个位数字输出,其输出的组合便是原数字的逆序。

(4) 变量 sum 的作用是将每次循环计算得到的 k 值相加,所得结果便是各位数字之和,循环的次数便是输入数字的位数。

3.9　建模扩展与优化

【例 3-13】 基于例 3-11 的百钱买百鸡问题进行扩展如下：鸡翁一,值钱五；鸡婆一,值钱三,鸡雏三,值钱一。凡 n 钱买鸡 n 只,问鸡翁、鸡婆、鸡雏各几何?

```
1   /*************************************************/
2                       程序文件名:Ex3_13.cpp
3                       降低时间复杂度计算 n 钱买 n 鸡问题
4   /*************************************************/
5   #include <bits/stdc++.h>
6   using namespace std;
7   main(){
8   int n,cock,hen,chick,s = 0;
9   cout <<"请输入钱数:";
10  cin >> n;
11  for(cock = 0; cock <= n/5;cock++)
12  {
13    hen = (n - 7 * cock)/4;
14    chick = n - cock - hen;
15    if (5 * cock + 3 * hen + chick/3 == n&&chick % 3 == 0)
16    {
17     if (hen >=  0 && cock >= 0)
18     {
19       s++;
```

```
20          cout <<"鸡翁"<< cock <<"只"<<" 鸡母"<< hen <<"只 鸡雏"<< chick <<"只"<< endl;
21       }
22     }
23   }
24   if(s == 0)
25   {
26       cout <<"无解";
27   }
28   else
29   {
30      cout << endl <<"共有"<< s <<"种方案组合"<< endl;
31   }
32 }
```

程序运行结果如图 3-19 所示。

图 3-19　例 3-13 运行结果

【程序解释】

（1）例 3-11 的时间复杂度为 $O(n^2)$，本题将时间复杂度降低为 O(n)。

（2）第 5 行导入万能头文件 stdc++.h。

（3）设鸡翁 cock 只，鸡婆 hen 只，则鸡雏 chick=n−cock−hen 只，可得到方程：5 * cock+3 * hen+(n−cock−hen)/3=n。

（4）第 11～23 行，用一层循环来枚举鸡翁数量 cock。

（5）第 14 行计算鸡母数量为 hen=(n−7 * cock)/4，则第 17 行只需判断鸡雏、鸡婆个数是否为非负数即可。

思考：

在使用多重循环求解问题时，如何减少循环嵌套数量进行枚举，以降低时间复杂度。

【例 3-14】 一诺是小动物保护协会成员，她在家里养了 n 只兔子，除去生病和走失的情况，兔子的数量每个月以 1.72 倍的速度递增。请计算 k 个月后，一诺家共有多少只兔子？（如果计算出的数字是小数，则小数部分按照一只进行计算。）

```
1   /*************************************************/
2                    程序文件名:Ex3_14.cpp
3                    兔子繁殖问题
4    /*************************************************/
5    #include <iostream>
```

```
#include <cmath>
using namespace std;
main( )
{
 int n,k;
 cout <<"请输入兔子初始数量 n:";
 cin >> n;
 cout <<"请输入经历的月数 k:";
 cin >> k;
 double s = 1.0 * n;
 for (int i = 1;i <= k;i++)
 {
   s = s * 1.72;
 }
 cout <<"经过"<< k <<"个月后,兔子的数量变为"<< ceil(s)<<"只"<< endl;
}
```

程序运行结果如图 3-20 所示。

图 3-20　例 3-14 运行结果

【程序解释】

(1) 初始兔子数量 1＜n≤10,月份 0＜k≤12。

(2) 经过 k 月后,经过 1.72 倍递增后的值可能包含小数部分,故第 15 行中数量 s 的数据类型定义为 double。

(3) 第 20 行 ceil()是取整函数,返回大于等于 s 的最小整数。

小结

(1) 在表达式后添加分号构成语句；语句包括单语句和块语句,若干语句通过{}组成块语句。

(2) if 语句是最简单的判断控制语句,包括 if 和 if…else 两种基本形式,简单的 if…else 语句可以与条件运算符相互替换使用。

(3) switch…case 语句常用来解决多重选择分支问题,使程序的结构更加清晰；case 语句可以有不同的精简表示形式,与 break 语句结合使用。

(4) C++的循环语句包括 for 语句、while 语句和 do…while 语句，for 语句通常可与 while 语句互相替换。当循环次数不确定时,常使用 while 语句；当循环次数不确定且循环体至少执行一次时,常使用 do…while 语句；当循环次数已知时,常使用 for 语句。for 语句的 3 个表达式有不同的功能,在某些情况下可以缺省,但是表达式间的分号必须保留。

(5) C++语言提供 3 种转移语句,其中 break 语句在 switch 语句和循环语句中,用以结束 switch 语句或其所在的最内层循环体；continue 语句仅在循环语句中使用,用以其所在循环体的本次循环继续进行下一次循环,建议尽量少使用 goto 语句。

习题 3

1. 选择题

(1) 下列正确的 if 语句是(　　)。

A. if(1<=m<=n)m++;

B. if(m==n) m++;

C. if(m<n) {m++ n++;}

D. if(m!=n) cout<<m;　else cout<<n;

(2) 下列程序的输出结果是(　　)。

```
main()
{
  int a(24),b(35),c(46);
  if(a>b)
    a=b; b=c; c=a;
  cout<<"a="<<a<<",b="<<b<<",c="<<c;
 }
```

A. a=24,b=35,c=46　　B. a=24,b=46,c=24

C. a=35,b=46,c=24　　D. a=35,b=46,c=35

(3) 下列程序段的输出结果是(　　)。

```
int i(2),j(3), k(9), m(8), n(7);
if (i<j)
if(k<m)
n=1;
else if(i<k)
if(j<m)
n=6;
else n=3;
else n=7;
else n=11;
cout<<n;
```

A. 3　　B. 6　　C. 7　　D. 11

(4) 下列程序的输出结果是(　　)。

```
main()
{
 int i(0),j(0),k(1);
 switch(k)
 {
   case 0:j++;
   case 1:i++;
   case 2:i++;j++;
 }
 cout<<"i="<<i<<"j="<<j;
```

```
}
```

A. i=2,j=2　　B. i=1,j=1　　C. i=1,j=0　　D. i=2,j=1

(5) j=(i>0? 1: i<0? -1: 0);的功能等同于下列哪个 if 语句？(　　)

A.
```
j = -1;
if(i)
if(i>0) j=1;
else if(i==0) j=0;
else j=-1;
```

B.
```
if(i)
  if(i>0) j=1;
  else if(i<0) j=-1;
```

C.
```
if(i>0) j=1;
else if(i<0) j=-1;
else j=0;
```

D.
```
j=0;
if(i>=0)
if(i>0) j=1;
else j=-1;
```

(6) 下列程序段的结果是(　　)。

```
int i(1),j(10) ;
do
{
  j-= i;i++;
}while(j-- >=9);
cout<<i<<","<<j;
```

A. i=3,j=11　　B. i=3,j=5　　C. i=1,j=-1　　D. i=4,j=9

(7) 下列的程序段循环了多少次？(　　)

```
int k=10;
 while (k=3)
  k=k-1;
```

A. 0 次　　B. 3 次　　C. 7 次　　D. 死循环

(8) 下列与“for(式 1；式 2；式 3；)循环体；”功能相同的语句是(　　)。

A. 式 1；while(式 2){循环体；式 3；}

B. 式 1；while(式 2){式 3；循环体；}

C. 式 1；do{循环体；式 3；} while(式 2)

D. do{式 1；循环体；式 3；} while(式 2)

(9) 下列循环执行的次数是(　　)。

```
for(int x=0,y=0;(y=1)&&(x<3);x++)
```

A. 1 次　　B. 2 次　　C. 3 次　　D. 4 次

(10) 下列程序段的运行结果是(　　)。

```
int i(2),j(1);
for(;i<=100;i++)
{
  if(j>=10) break;
  if(j%3==1)
```

```
    {
      j += 3;
      continue;
    }
    i = i + 1;
  }
  cout << i;
```

A. 101　　B. 6　　C. 5　　D. 4

2. 编程题

(1) 求 sn＝a＋aa＋aaa＋…＋aa…aa(n 个 a)之值,其中 a 是一个数字(例如：3＋33＋333＋3333,此时 n＝4,n 由键盘输入)。

(2) 从键盘输入一个整数,判断该数是否为回文数(所谓回文数就是从左向右读和从右向左读一样的数,如 7887、23432)。

(3) 编写程序,分别正向、逆向输出 26 个大写英文字母。

3. 读程序写结果。

(1)
```
#include <iostream>
using namespace std;
int sum,k;
main()
{
   for(sum = 0,k = 1;k < 10;k++)
   {
      if(k % 2 == 0)
        continue;
      sum += k;
   }
 cout <<"sum = "<< sum << endl;
}
```

(2)
```
#include <iostream>
using namespace std;
main()
{
   int a(18),b(21),m(0);
   switch(a % 3)
   {
      case 0: m++;break;
      case 1: m++;
              switch(b % 2)
              {
                default: m++;
                case 0: m++;break;
              }
   }
   cout << m;
 }
```