第3章 控制语句

使用高级语言编写的程序是由控制语句组成的。普通的语句都是顺序执行的，即按照先后顺序一条一条地执行。除此之外，C++还提供了两种结构的控制语句——选择结构和循环结构。

3.1 选择结构

选择结构的语句是用于根据条件控制程序的执行流向，是程序中最常用的语句。C++的选择结构包含以下几种语句：if-else 选择语句、嵌套的 if-else 语句、if-else if 语句和 switch 语句。

3.1.1 if-else 选择语句

if-else 选择语句的语法结构为：

```
if (表达式)语句 1;
else 语句 2;
```

执行顺序为：首先求出表达式的值，如果表达式的值为 true 或为非零值，则执行语句 1；若表达式的值为 false 或 0，则执行语句 2。图 3.1 为 if-else 语句的流程图。

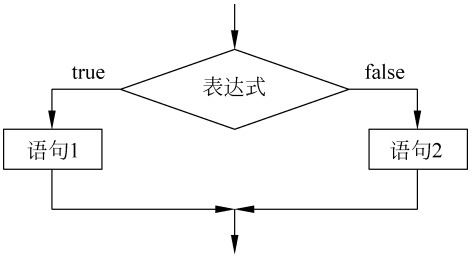

图 3.1 if-else 语句流程图

if-else 语句中的语句 1 和语句 2 可以是一条语句，也可以是由花括号括住的多条语句。语句 2 可以为空，当语句 2 为空时，else 可以省略。

例 3.1 输入两个数，判断是否相等，并输出判断结果。

```
#include<iostream>
```

```
using namespace std;
void main()
{   int i,j;
    cin >> i >> j;
    if(i == j)
        cout <<"i == j"<< endl;
    else
        cout <<"i!= j"<< endl;
}
```

图 3.2 为例 3.1 程序的运行结果。

```
100
50
i!=j
```

图 3.2 例 3.1 程序的运行结果

3.1.2 嵌套的 if-else 语句

如果在 if 或 else 后面的语句中又出现了 if-else 语句,这种结构就是嵌套的 if-else 语句。其语法形式为:

```
if(表达式 1)
   if(表达式 2)  语句 1
   else     语句 2
else
   if(表达式 3)  语句 3
   else     语句 4
```

其中的语句 1、2、3、4 可以是由花括号括住的多条语句。

嵌套的 if-else 语句构成了一个梯级选择结构,可以完成复杂的逻辑判断,使用非常频繁。使用时需注意 else 和 if 的匹配问题,即确定 else 和哪个 if 属于同一个逻辑层次。匹配的原则是:else 和它前面的最近一个没有被花括号括住的 if 相匹配。

例 3.2 输入两个整数,比较它们的大小并输出比较结果。

```
#include < iostream >
using namespace std;
void main()
{   int i,j;
    cin >> i >> j;
    if(i!= j)
        if(i > j)
            cout <<"i > j"<< endl;
        else
            cout <<"i < j"<< endl;
    else
        cout <<"i = j"<< endl;
}
```

例 3.2 程序的运行结果如图 3.3 所示。

```
10
100
i<j
```

图 3.3　例 3.2 程序的运行结果

3.1.3　if-else if 语句

if-else if 语句语法形式如下：

```
if (表达式 1)   语句 1;
else if (表达式 2)   语句 2;
else if (表达式 3)   语句 3;
      ⋮
else   语句 n
```

这是一种常用的选择结构，它和如下梯级结构的嵌套 if-else 语句是等价的，只是为了容易理解和书写格式清晰而使用的另一种写法。

```
if (表达式 1)   语句 1;
else
    if (表达式 2)   语句 2;
    else
        if (表达式 3)   语句 3;
        ⋮
        else   语句 n
```

例 3.3　用 if-else if 语句编程实现例 3.2 的问题。

```
#include <iostream>
using namespace std;
void main()
{   int i,j;
    cin >> i >> j;
    if(i == j)      cout <<"i = j"<< endl;
    else if(i < j)  cout <<"i < j"<< endl;
    else            cout <<"i > j"<< endl;
}
```

3.1.4　switch 语句

对于深层嵌套的选择结构，如果所有的选择都依赖于同一个整型或字符型的变量或表达式的取值，则可以用 switch 语句来代替 if-else 或者 else-if 的梯级结构。switch 语句的语法形式如下：

```
switch(变量名或表达式)
{
    case   常量表达式 1: 语句 1; break;
    case   常量表达式 2: 语句 2; break;
                ⋮
```

```
    case   常量表达式 n: 语句 n; break;
    default:   语句 n + 1;
}
```

switch 的执行顺序是：首先求出变量或表达式的值，然后用该值分别和各 case 后的常量表达式的值相比较，若找到一个和该值相等的常量表达式，则执行其后的语句，语句执行完毕后，再由 break 语句直接退出由花括号括住的 switch 语句体；如果没有找到相等的常量表达式，则执行 default 后的语句，语句执行完毕后退出 switch 语句体。图 3.4 为 switch 语句的流程图。

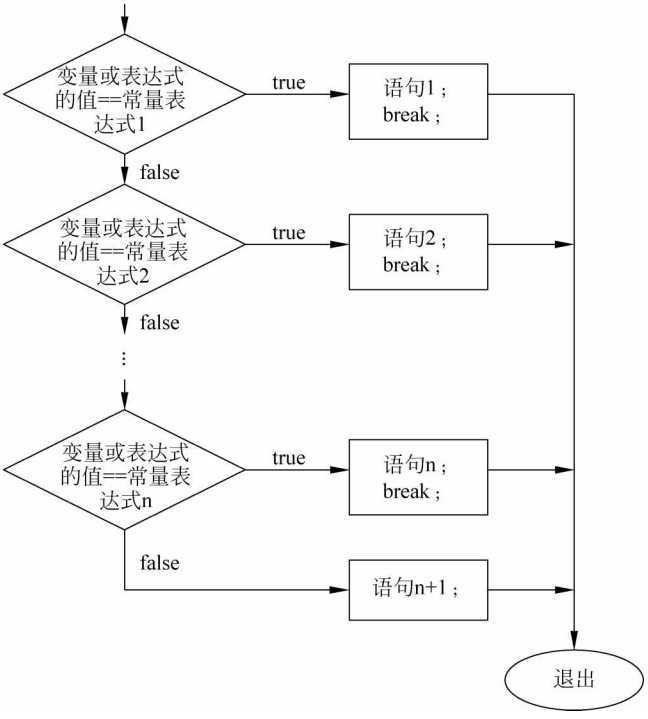

图 3.4 switch 语句流程图

使用 switch 语句时，需注意以下几点。

(1) switch 后的变量或表达式的类型必须是整型或字符型。

(2) 各 case 后的语句 1、语句 2、…、语句 n 和 default 后的语句 n+1 可以是一条语句也可以是多条语句；如果是多条语句，也不使用大括号{}括住。

(3) 各个 case 后的常量表达式的值都不能相同。

(4) 每个 case 后面的 break 语句都可以没有。如果条件变量或表达式的值和某个 case 后的常量表达式的值相等，而且该 case 后面没有 break 语句，则以该 case 为入口点开始执行后面的语句，直到遇到第一个 break 语句跳出 switch 语句体，若其后的所有 case 都没有 break 语句，则一直执行到 switch 的结束点。

(5) 当多个 case 分支后的语句完全相同时，则可以把它们组合成一个 case 分支。

例 3.4 输入一个'A'～'D'的大写字符表示考试成绩的等级，并按照该等级输出对应的百分制的分数段。

```
#include<iostream>
using namespace std;
void main()
{   char grade;
    cout<<"请输入一个 A 到 D 的大写字母\n";
    cin>>grade;
    switch(grade)
    {
      case 'A':  cout<<"85～100\n"; break;
      case 'B':  cout<<"70～84\n";  break;
      case 'C':  cout<<"60～69\n";  break;
      case 'D':  cout<<"60 以下\n"; break;
      default :   cout<<"输入的字母错误\n";
    }
}
```

例 3.4 程序的运行结果如图 3.5 所示。

```
请输入一个A到D的大写字母
C
60~69
```

图 3.5 例 3.4 程序的运行结果

3.2 循环结构

循环结构的语句根据条件重复地执行程序中的某一条或某一段语句，也是程序中最常用的语句。C++包含以下几种循环结构的控制语句：while 循环语句、do-while 循环语句和 for 循环语句。

3.2.1 while 循环语句

while 是最常用的循环语句，其语法形式为：

```
while(表达式)
  循环体语句;
```

其中，循环体语句部分可以是一条语句，也可以是由花括号括住的多条语句。

执行顺序如下。

(1) 先计算并判断表达式的值，若为 0 或 false，则跳转到第(4)步。

(2) 若表达式的值非零或为 true，则执行循环体中的语句。

(3) 返回第(1) 步。

(4) 退出循环体，接着执行循环体后的语句。

图 3.6 为 while 语句的流程图。

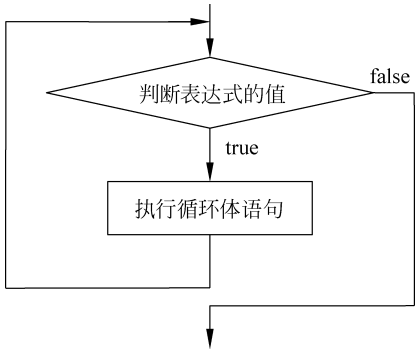

图 3.6 while 语句流程图

例 3.5 输入一个正整数 n,求其阶乘 n!(n!=1×2×3×…×(n−1)×n)。

```
#include<iostream>
using namespace std;
void main()
{   unsigned short n;
    int i=1;
    unsigned long result=1;
    cout<<"请输入一个不大于 30 的正整数\n";
    cin>>n;
    while(i<=n)
    {
       result*=i;
       i++;
    }
    cout<<n<<"!="<<result<<endl;
}
```

程序中先提示用户输入一个不大于 30 的正整数,这是因为阶乘值的增大速度非常快,虽然使用了无符号长整型(unsigned long)作为阶乘结果的数据类型,但是当 n 的值大于 33 后,n 的阶乘值就将溢出。

变量 result 用来存放计算结果,其初始值为 1。

循环的条件为 i<=n,变量 i 的初值为 1,每次循环中首先用 i 乘以 result,并把乘积再赋值给 result,然后执行 i++。当最后退出循环时,result 中的值就是 n 的阶乘。图 3.7 为例 3.5 程序的运行结果。

```
请输入一个不大于30的正整数
6
6!=720
```

图 3.7 例 3.5 程序的运行结果

注意:使用 while 语句实现循环时,在循环体中应该包含改变循环条件表达式值的语句。否则,会造成无限循环。

3.2.2 do-while 循环语句

do-while 语句的语法形式如下:

```
do
    循环体语句;
while(表达式);
```

其中,循环体语句可以是一条语句,也可以是由花括号括住的多条语句。while 语句的后面一定要有分号。

语句的执行顺序如下。

(1) 先执行循环体中的语句。

(2) 计算并判断表达式的值。

(3) 若表达式的值非零或为 true,则跳转到第(1) 步。否则退出循环,接着执行后面的

语句。

图 3.8 是 do-while 语句的流程图。

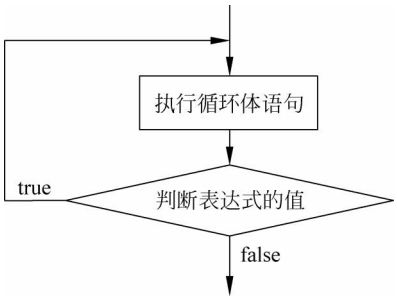

图 3.8 do-while 语句流程图

例 3.6 使用 do-while 语句实现例 3.5 中求正整数阶乘的程序。

```
#include <iostream>
using namespace std;
void main()
{   unsigned short n;
    int i = 1;
    unsigned long result = 1;
    cout <<"请输入一个不大于 30 的正整数\n";
    cin >> n;
    do
    {
       result * = i;
       i++;
    }
    while(i <= n);
    cout << n <<"!= "<< result << endl;
}
```

while 循环语句和 do-while 循环语句的功能在多数情况下是等价的，然而它们也存在区别。while 语句和 do-while 语句的区别是：while 语句是先判断循环条件，后执行循环体语句；而 do-while 语句是先执行循环体语句，后判断循环条件。如果循环条件表达式的初始值就为零或 false，则 while 循环的循环体语句一次也不执行，而 do-while 循环的循环体语句执行了一次。

3.2.3 for 循环语句

for 循环语句的语法形式为：

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

其中，表达式 1 通常用来初始化循环控制变量的值，所以又叫循环控制变量初始化表达式；表达式 2 用来判断是否满足循环条件，又叫循环条件表达式；表达式 3 通常用来修改循环控制变量的值。循环体语句可以是一条语句也可以是由花括号括住的多条语句。

for 语句的执行顺序如下。

(1) 计算表达式 1 的值。

(2) 计算并判断表达式 2 的值，若表达式 2 的值非零或为 true，则接着执行第(3)步；否则跳转到第(4)步。

(3) 执行循环体语句，计算表达式 3 的值。返回第(2)步。

(4) 退出循环，接着执行后面的语句。

图 3.9 是 for 语句的流程图。

例 3.7 用 for 循环语句实现例 3.5 中求正整数阶乘的程序。

```
#include <iostream>
using namespace std;
void main()
{   unsigned short n;
    unsigned long result = 1;
    cout <<"请输入一个不大于 30 的正整数\n";
    cin >> n;
    for(int i = 1;i <= n;i++)
       result * = i;
    cout << n <<"!= "<< result << endl;       }
```

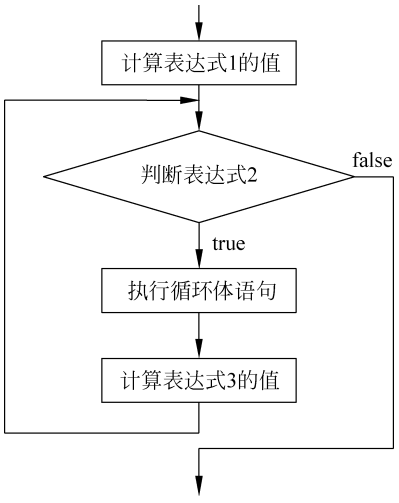

图 3.9 for 语句流程图

for 语句通常用于已知循环次数的情况。

for 语句的用法非常灵活,for 后面的三个表达式哪个都可以被省略,也可以同时被省略。虽然表达式可以省略,但是不能省略用于分隔它们的分号。最简单的 for 语句如下所示:

```
for( ; ; )
{…}
```

这样的 for 循环语句和下面的 while 循环语句是等价的,它们都是无限循环(死循环)。

```
while(true)
{…}
```

如果省略表达式 1,则在程序中 for 循环语句的前面,应该有对循环控制变量或循环条件的初始化语句;如果省略表达式 3,则应该在循环体中包含修改循环控制变量和循环条件的语句,以防止出现死循环。

例 3.8 省略表达式 1 和 3 的 for 循环语句。

```
#include <iostream>
using namespace std;
void main()
{
    int i = 0;
    for(;i <= 2;)
    {
        cout <<"Welcome to C++\n";
        i++;
    }
}
```

程序的功能是使用 for 循环语句输出三行字符串“Welcome to C++”。for 语句中省略了表达式 1 和表达式 3,在 for 语句的前面初始化循环控制变量 i 的值为 1;而循环体内部的语句 i++在每次循环执行时,修改循环控制变量 i 的值。这样的 for 循环等价于一个类似的 while 循环。图 3.10 为例 3.8 程序的运行结果。

```
Welcome to C++
Welcome to C++
Welcome to C++
```

图 3.10 例 3.8 程序的运行结果

C++ 11 新引入了一种基于范围的 for 循环语句，用这种 for 循环语句访问数组和集合中的元素时，比使用传统的 for 循环语句书写更加方便，程序更容易理解。本书将在第 4 章中结合数组介绍这种基于范围的 for 循环语句。

3.2.4 嵌套的循环语句

如果在循环体语句中又出现了循环结构语句，就形成了嵌套的循环结构。通常把嵌套的循环语句称为多重循环语句。在嵌套的循环结构中，外部的循环叫外层循环；而被包含在其他循环内部的循环叫内层循环。这里外层和内层都是相对而言的。

例 3.9 输出一个由星号 * 组成的三角形图案。

```
#include<iostream>
using namespace std;
void main()
{
    for(int i=1;i<=5;i++)
    {
        for(int j=0;j<=10-i;j++)
            cout<<' ';
        for(int k=1;k<=(2*i-1);k++)
            cout<<'*';
        cout<<endl;
    }
}
```

图 3.11 为例 3.9 程序的运行结果。

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

图 3.11 例 3.9 程序的运行结果

程序中包含嵌套的循环结构。外层的 for 循环语句负责输出星号的行数。内层的第一个 for 循环语句负责输出每行中星号前面的空格字符；内层的第二个 for 循环语句负责输出每行中的星号。

3.1.2 节中介绍过选择结构 if 语句的嵌套，本节又介绍了循环语句的嵌套结构。选择结构和循环结构也可以相互嵌套。

C++ 11 中新增了一种基于范围的 for 循环语句，这种循环语句常用于遍历数组或容器中的元素，在第 4 章中将详细介绍这种 for 循环语句。

3.3 其他流控制语句

以上几节介绍了 C++中实现选择和循环结构的语句。本节先介绍两个选择和循环结构中常用的有条件转移语句——break 语句和 continue 语句；然后介绍无条件转移语句——goto 语句。

3.3.1 break 语句和 continue 语句

break 语句在介绍 switch 语句时,已经见到过了。它用在 switch 语句和循环语句中,功能是:立即从包含它的 switch 语句体或包含它的最内层的循环体中退出,开始顺序执行后面的语句。

continue 语句用在循环语句中,功能是:立即结束本次的循环执行,转到判断循环条件的语句判断是否进行下一次循环。

例 3.10 和例 3.11 揭示了 break 语句和 continue 语句用在循环语句中的区别。

例 3.10 在循环中使用 break 语句。

```
#include<iostream>
using namespace std;
void main()
{
    int i=0;
    while(i<10)
    {
        if(i++==5) break;
        cout<<"欢迎学习 C++\n";
    }
}
```

例 3.11 在循环中使用 continue 语句。

```
#include<iostream>
using namespace std;
void main()
{
    int i=0;
    while(i<10)
    {
        if(i++==5) continue;
        cout<<"欢迎学习 C++\n";
    }
}
```

以上两例的程序几乎完全相同,唯一的区别是:例 3.10 程序的循环中使用了 break 语句,而例 3.10 程序的循环中使用了 continue 语句。

例 3.10 中,当 i 的值为 5 时,执行 break 语句,马上退出循环体,程序运行结束。结果是只输出了 5 行字符串。

例 3.11 中,当 i 的值为 5 时,执行 continue 语句,马上退出本次循环,循环体中后面的语句被跳过。但是,循环并没有终止,从 i=6 开始继续执行下一次循环。结果是输出了 9 行字符串。

break 语句和 continue 语句只能使程序控制转到一个确定的地方,所以可以把它们叫作有条件的转移语句。

3.3.2 goto 语句

C++还有一个功能更加强大的转移语句——goto 语句。它和语句标号相配合，可以使控制转移到程序中的任何地方，所以叫无条件转移语句。goto 语句的使用会使程序的结构变得杂乱无章，违反了结构化程序设计的原理，所以应尽可能少用或不用 goto 语句。

小结

C++语言的控制语句形式简洁、功能强大，主要包括选择结构和循环结构两种类型。两种结构的控制语句可以相互嵌套，完成复杂的控制逻辑。

选择结构的语句用于根据条件控制程序的执行流向。C++的选择结构包含以下几种语句：if-else 选择语句、嵌套的 if-else 语句、if-else if 语句和 switch 语句。

循环结构的语句根据条件重复地执行程序中的某一条或某一段语句。C++包含以下几种循环结构的控制语句：while 循环语句、do-while 循环语句和 for 循环语句。

除了选择结构和循环结构的语句之外，C++还包括几个特殊的流控制语句，它们是 break 语句、continue 语句和 goto 语句。

习题

3.1 编写一段程序，提示用户输入一个英文字母，使用 if 语句判断用户输入的字母是大写字母还是小写字母，然后输出相关信息。

3.2 以下程序段输出若干行字符串“How are you!”，行数由用户输入的整数决定，如果用户输入 0，则输出一行字符串“Hello!”，请找出下面程序中存在的语法错误。

```
int n;
cout <<"请输入一个整数：";
cin >> n;
if(n = 0)
    cout <<"Hello!\n";
else
    for(int i = 0;i < n;i++);
        cout <<"How are you!\n";
```

3.3 编写一段程序，连续输入若干个学生的考试成绩，根据成绩判断其所在的等级，并输出相关信息。判断成绩等级的规则如下：

$$\text{等级}=\begin{cases}\text{A 级} & 90\leqslant\text{分数}\leqslant100\\ \text{B 级} & 80\leqslant\text{分数}<90\\ \text{C 级} & 70\leqslant\text{分数}<80\\ \text{D 级} & 60\leqslant\text{分数}<70\\ \text{E 级} & \text{分数}<60\end{cases}$$

例如，如果第 3 个学生的成绩为 82 分，则应输出一行字符串“学生 3 的成绩为 B 等”。

要求使用 while 循环和 switch 选择语句。

3.4 编写程序,输出右边的由星号组成的倒三角形。

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

3.5 请举例说明 break 语句和 continue 语句各自的用法和区别。