第3章 控制结构

✧ 引言

一个程序往往由一系列语句组成，程序的执行原则是由上而下、一行一行顺序执行。但程序中往往遇到这种情况：根据不同的选择或计算结果程序要进行不同的处理；有的语句要反复执行多次，这就需要程序要有控制转向能力与反复执行语句的能力。C++提供了判断、转移、循环控制语句来实现对程序的转移与循环控制。

✧ 学习目标

1. 理解并掌握分支语句 if、switch 的使用；

2. 理解并掌握 for、while、do…while 这 3 种循环的语法和使用场合，控制执行的顺序，能根据要求选择合适的循环语句；

3. 理解并掌握跳转语句 break、continue 语句的语法和使用场合，能根据要求选择合适的跳转语句。

3.1 C++语言的语句

语句是程序中可以独立执行的最小单元，类似于自然语言中的句子。与句子由句号结束一样，语句一般由分号结束。语句通常是由表达式构成的，表达式尾部加上分号构成**表达式语句**。但表达式不是语句，所以表达式不能在程序中独立存在。例如：

```
a = b + c;          //赋值语句
i + j;
;                   //空语句
```

均是 C++语言的合法语句。

在上述语句中，第 1 条语句是一条由赋值表达式构成的语句，通常称其为赋值语句；第 2 条语句是一条由算术运算表达式构成的语句。应当说明的是，该语句尽管是合法的，但没有什么实际意义，一般在程序中不使用这样的语句；第 3 条语句是由一个空的表达式构成的语句，这样的语句叫做**空语句**。空语句不仅合法，而且在程序中有其实用价值，它常被用于在程序中某处根据语法要求应该有一条语句，而实际上又没有什么操作可执行的场合。

在 C++语言中，变量的说明必须以分号结束，所以变量的说明也是语句，叫做**说明语句**。正是由于在 C++语言中存在说明语句这一概念，才使变量说明可以出现在程序中任何一个语句可以出现的地方，提高了变量说明的灵活性。对比其他语言，比如 C 语言，由于说明不

是语句,所以变量只能集中地在一个模块的首部进行说明。

由一对花括号{}括起来的多条语句叫做一个块语句。例如:

```
{
    int i = 5;
    i = (i + 5)/2;
    cout << i << endl;
}
```

块语句也叫复合语句,它在语法上等价于一条语句。块语句主要用于这样的场合:在程序的某处需要执行一项必须由多条语句才能完成的操作,而从语法上讲,程序在此处只允许存在一条语句。花括号是C++语言中的一个标点符号,左括号标明了块语句的起始位置,右括号标明了块语句结束,它的作用就像单条语句中的分号一样。因此,右花括号后边不再需要分号。

块语句与本章将介绍的判断、循环语句均属于结构语句。结构语句决定一个程序的结构与流程。

3.2 判断与循环

学习了数据类型、表达式、赋值语句和数据的输入输出后,就可以编写程序完成一些简单的功能了,但是学习者现在能写的程序还只是一些顺序执行的程序序列。实际人们所面对的客观世界远不是这么简单,除了最简单的程序外,顺序的程序执行过程对于人们必须要解决的问题来说是不够的。

例如,有一数学函数如下,要求输入一个x值,求出y值。

$$y=\begin{cases} x & (x\geqslant 0) \\ -x & (x<0) \end{cases}$$

这个问题人工计算起来并不复杂,但如何将这一计算方法用计算机语言描述清楚,使计算机能够计算呢?很显然用顺序执行的语句序列是无法描述的,这里必须进行判断,需要用到判断选择结构。

再比如要统计某门课程的平均成绩。这个问题人工计算的方法很简单,但是当数据量很大时,人们就不得不借助于计算机了。计算机计算速度很快,但计算方法必须由人们准确地描述清楚。这个算法的核心就是进行累加,显然不可能用顺序执行的语句来进行大量数据的罗列,这种情况需要用循环控制结构。

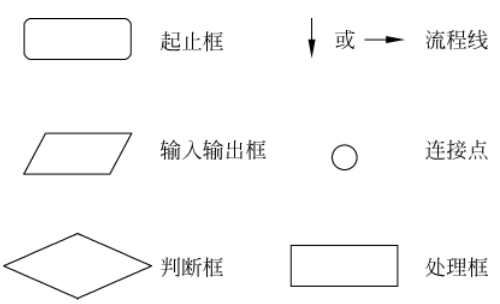

图 3-1 流程图的标准符号

为了描述程序的流程,人们使用一种流程图的工具。**流程图**是用来描述算法(程序)的工具,它具有简洁、直观、准确的优点,一些常用的流程图符号如图3-1所示。在本章中将使用它来描述语句的功能。

3.3 if…else…if 判断式

判断选择结构又称条件分支结构，是一种基本的程序结构类型。在程序设计中，当需要进行选择、判断和处理的时候，就要用到条件分支结构。条件分支结构的语句一般包括 if 语句、if…else 语句、switch…case 语句。

3.3.1 基本的 if 语句

if 语句是专门用来实现判断选择型结构的语句。基本的 if 语句具有如下的一般形式：

```
if (表达式)  语句;
```

其中，表达式通常是一个关系表达式或逻辑表式，语句可以是一个单条语句，或是一个块语句，甚至是一个空语句。它的执行过程是：首先计算表达式的值，如果表达式的值为真，则执行语句段；否则就跳过语句，直接执行 if 语句的后继语句，如图 3-2 所示。在 C++中，if 后面实际可以跟任意一个可计算出结果的表达式，甚至可以直接用常量 0 代表逻辑假，用非 0 代表逻辑真。

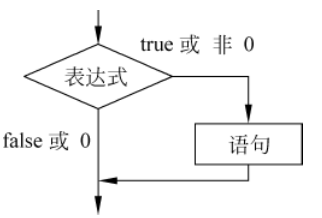

图 3-2 if 语句流程图

☆注意：

➢ *表达式两边的括号必不可少。*

例如：

```
if (i>10)     i=i-5;
cout<<i<<endl;
```

执行过程如下：先对 i 的值进行判断；如果 i 的值大于 10，则将 i 的值减 5，然后输出；否则直接输出 i 的值。

3.3.2 完整的 if 语句

完整的 if 语句的一般形式是

```
if (表达式) 语句 1;
else  语句 2;
```

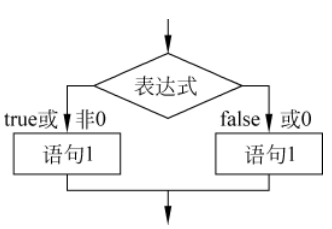

图 3-3 if…else 语句流程图

它的执行过程是：首先计算表达式的值，如果表达式的值为真，则执行语句 1；否则就执行语句 2，如图 3-3 所示。通常，将前者叫做 if 分支，将后者叫做 else 分支。应当说明的是，尽管完整的 if 语句中存在两个语句段，且有两个表示语句结束的分号，但整个语句在语法上只是一条语句。尤其要注意 if 分支后边的分号是不可缺少的(除非这里是一条复

合语句)。

例如:

```
if (x> y)   cout << x << endl;
else cout << y << endl;
```

实现了从 x 和 y 中选择较大的一个输出。

3.3.3 if 语句的嵌套

由于 if 语句的语句段要求是一条 C++语句,而 if 语句本身就是一条合法 C++语句,所以可以将 if 语句用作 if 语句的语句段。这就是所谓的 if 语句的嵌套。if 语句的嵌套常用于多次判断选择。

【例 3-1】 将百分制的成绩按等级分输出。

分析:等级分为四等:A、B、C、D,分别对应的分数段为 90～100、80～89、60～79、0～59,转换时需要进行多次判断,要用多重选择结构,这里将选用嵌套的 if 语句。

编写多重分支的程序,为了防止错误,应该首先画出流程图,然后按照流程图写语句。流程如图 3-4 所示。

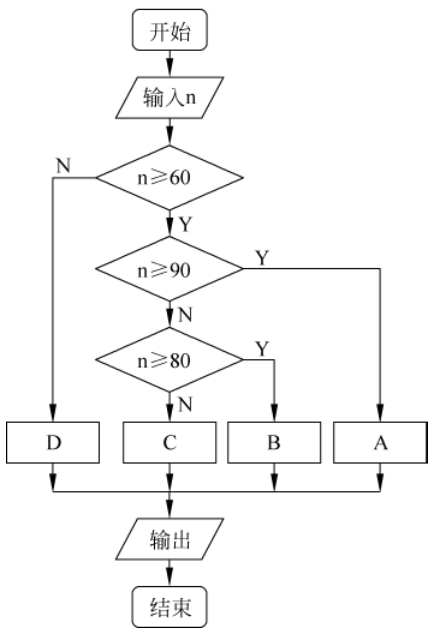

图 3-4 流程图

```
1  /*****************************************
2   *  程序名: p3_1.cpp                     *
3   *  功  能: 将百分制的成绩按等级分输出   *
4   *****************************************/
5  #include<iostream>
```

```
6   using namespace std;
7   int main()
8   {
9      int n;
10     cout<<"Enter the score:";
11     cin>>n;
12     if (n>=60)
13        if (n>=90)
14           cout<<"The degree is A"<<endl;
15        else if (n>=80)
16           cout<<"The degree is B"<<endl;
17        else
18           cout<<"The degree is C"<<endl;
19     else
20        cout<<"The degree is D"<<endl;
21     return 0;
22  }
```

运行结果：

```
Enter the score:86↙
The degree is B
Enter the score:74↙
The degree is C
Enter the score:48↙
The degree is D
```

程序解释：

由于 if 语句存在两种形式，当发生嵌套时就面临理解的问题，在程序 p3-1.cpp 中就出现下列嵌套形式：

```
if(表达式 1)
          if(表达式 2)
             语句 1;
          else
             语句 2;
```

其中，else 解释为属于第 1 个 if 与属于第 2 个 if 会有完全不同的结果。else 究竟属于哪个 if？对于上述歧义，C++规定 else 与前面最近的没有 else 的 if 语句配对。因此上述嵌套的 if 语句解释如下：

```
if (表达式 1)
          if (表达式 2)  语句 1;  else  语句 2;
```

如果需要将 else 与第 1 个 if 语句配对，程序要进行如下的修改：

```
if (表达式 1) {
          if (表达式 2)
             语句 1;
         }
```

```
else
        语句 2;
```


修改后相应的解释为：

```
if (表达式 1)
   { if (表达式 2)  语句 1;}
else
   语句 2;
```

3.4 switch…case 判断式

在有的问题中，虽然需要进行多次判断选择，但是每一次都是判断同一表达式的值，这样就没有必要在每一个嵌套的 if 语句中都计算一遍表达式的值，为此 C++中有 switch 语句专门用来解决这类问题。switch 语句的语法形式如下：

```
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]
}
```

其中：

- 表达式可以是任意一个合法的 C++表达式，但其值只能是字符型或者整型。
- 常量表达式是由常量组成的表达式，其值也只能是字符型常量或者整型常量，各常量表达式的值不可以重复(相等)。
- 符号[ ]表示其中的内容可选，语句块是可选的，它可以由一条语句或一个复合语句组成。
- break 语句、default 语句也是可选的。

switch 语句的执行过程是：

① 先求出表达式的值。

② 将表达式的值依次与 case 后面的常量表达式值相比较，若与某一常量表达式的值相等，则转去执行该 case 语句后边的语句序列，直到遇到 break 语句或 switch 语句的右花括号为止。

③ 若表达式的值与 case 语句后的任一常量表达式的值都不相等，如果有 default 语句，则执行其后边的语句序列。如果没有 default 语句，则什么也不执行。

由 switch 语句的执行过程可以看出，若某一分支要求执行的是不止一条语句，则所要求执行的多条语句不必写成块语句的形式。default 语句可以放在 switch 语句中的任意位置，但一般放在最后作为 switch 语句的最后一个分支。

☆**注意：**

➢ 在使用 switch…case 语句时经常容易丢失必要的 break 语句，这样程序会产生结果的错误，此类错误往往不易发觉。

switch 语句是一个很好的多路分支选择语句，常用它来实现较为复杂的多路分支程序。程序 p3_2.cpp 是程序 p3_1.cpp 的 switch 语句的实现。

【例 3-2】 将百分制的成绩按等级分输出的 switch…case 语句的实现。

```
/*************************************************************
 *   程序名：p3_2.cpp                                        *
 *   功   能：将百分制的成绩按等级分输出 switch-case 实现    *
 *************************************************************/

#include<iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Enter a score:";
    cin>>n;
    switch(n/10)
    {
        case 9: case 10:
            cout<<"The degree is A"<<endl;
            break;
        case 8:
            cout<<"The degree is B"<<endl;
            break;
        case 7: case 6:
            cout<<"The degree is C"<<endl;
            break;
        default:
            cout<<"The degree is D"<<endl;
    }
   return 0;
}
```

if 语句和 switch 语句的比较：

if 语句和 switch 语句都可以用来处理程序中的分支问题，在许多场合可以互相替代。但是它们之间还是有一些差别，其主要表现为：

① if 语句常用于分支较少的场合，而 switch 语句常用于分支较多的场合。

② if 语句可以用来判断一个值是否落在一个范围内，而 switch 语句则要求其相应分支的常量必须与某一值严格相等。例如，设 i 为一整型变量，若 i 的值为 1、2、3 或 4 时，就执行某一操作语句，则相应的 if 语句和 switch 语句分别为

```
if(i>=1&&i<=4)
        执行语句;
        //…
```

```
switch(i) {
        case 1:
        case 2:
        case 3:
        case 4:
        }
```

若值的范围较大时,显然 if 语句要优于 switch 语句。特别是当表达式的值是一个实数时,通常只能使用 if 语句。

3.5 for 循 环

3.5.1 for 语句

在一个程序中,常常需要在给定条件成立的情况下,重复地执行某些操作。C++语言为实现这一目的提供了 3 种循环语句:for 语句、while 语句和 do…while 语句。在循环语句中,重复执行的操作叫做循环体。循环体可以是单条语句、块语句甚至是空语句。

for 循环的使用非常灵活,既可以用于循环次数确定的情况,也可以用于循环次数未知的情况。for 语句的语法形式如下。

```
for(表达式 1;表达式 2;表达式 3) 语句
```

- 上述格式可理解为:for(循环变量赋初值;循环条件;循环变量增值)循环体。
- for 是关键字。
- 表达式 1、表达式 2 和表达式 3 是任意表达式。
- 语句为循环体,它可以是一条语句,也可以是复合语句,还可以是空语句。
- for 循环语句的执行过程如下:
  ① 计算表达式 1 的值。
  ② 计算表达式 2 的值,并进行判断,如果表达式 2 的值为 0(false)则退出该循环,执行该循环体后面的语句;如果表达式 2 的值为非 0(true)转③。
  ③ 执行循环体的语句。
  ④ 计算表达式 3 的值。
  ⑤ 转 ②。

for 循环语句执行过程如图 3-5 所示。

在程序中,for 循环的基本用法为:先说明一个整型或字符型变量作为循环变量;然后在初始化部分为循环变量置一初值,在循环条件部分用一关系表达式给出当循环变量的值在何范围时继续循环,在增量部分用一赋值表达式给出循环变量的变化量;最后,退出循环体。

【例 3-3】 求 1+3+5+7+…+99。

分析:求 1~100 之间的奇数和就是一个累加的算法,累加过程是一个循环过程,可以用 for 语句实现。

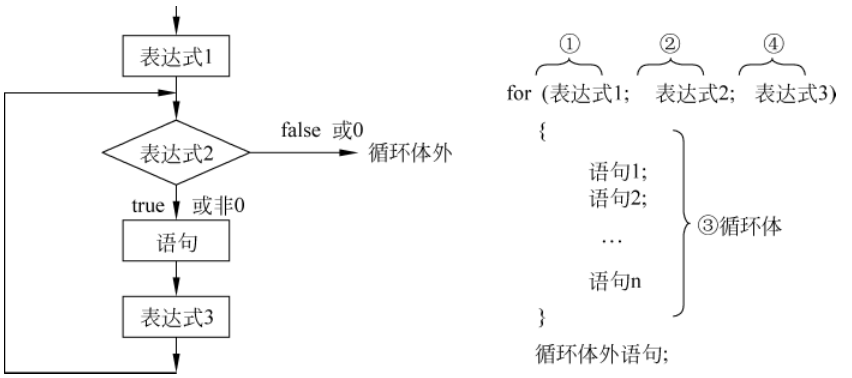

图 3-5　for 循环语句执行过程

```
1   /*********************************************
2    *   程序名: p3_3.cpp                        *
3    *   功   能: 用 for 语句计算 1 + 3 + … + 99   *
4    *********************************************/
5   #include<iostream>
6   using namespace std;
7   int main()
8   {
9       int i,sum = 0;
10      for(i = 1; i<100; ++i, ++i)
11        sum = sum + i;
12      cout<<"sum = "<<sum<<endl;
13      return 0;
14  }
```

运行结果：

```
sum = 2500
```

程序解释：

- for 循环还有各种不同的格式，使用中可以灵活选择。
- for 语句一般形式中的表达式 1 可以省略，此时表达式 1 应出现在循环语句之前。
- for 循环语句一般形式中的表达式 3 可以省略，即将循环变量增量的操作放在循环体内进行。
- 表达式 1 不仅可以设置循环变量的初值，同时还可以通过逗号表达式设置一些其他变量的值。类似地，表达式 2 和表达式 3 都可以这样，正如程序第 10 行 for 语句中，使用了++i,++i 表达式。
- for 语句一般形式中的表达式 2 如果省略，即不判断其循环条件，则循环将无终止地进行下去。要结束循环，需要在循环体内设置 break 语句，退出循环。

3.5.2　用 for 语句实现嵌套循环

for 循环可以嵌套，所谓循环结构嵌套是指一个循环体内可以包含另一个完整的循环结

构,构成多重循环结构。下面举例说明。

【例 3-4】 用嵌套 for 语句显示乘法九九表。


```
/*************************************************
 *   程序名: p3_4.cpp                             *
 *   功   能: 用嵌套 for 语句显示乘法九九表       *
 *************************************************/
#include<iostream>
using namespace std;
int main()
{
   const int line=9;
   int i,j;
   for(i=0; i<line; i++)              //显示 9 行
   {
      for(j=0; j<=i; j++)             //每一行显示的内容
        cout<<j+1<<"×"<<i+1<<"="<<(j+1)*(i+1)<<"\t";
      cout<<endl;
   }
  return 0;
}
```

运行结果:

```
1×1=1
1×2=2  2×2=4
1×3=3  2×3=6   3×3=9
1×4=4  2×4=8   3×4=12  4×4=16
1×5=5  2×5=10  3×5=15  4×5=20  5×5=25
1×6=6  2×6=12  3×6=18  4×6=24  5×6=30  6×6=36
1×7=7  2×7=14  3×7=21  4×7=28  5×7=35  6×7=42  7×7=49
1×8=8  2×8=16  3×8=24  4×8=32  5×8=40  6×8=48  7×8=56  8×8=64
1×9=9  2×9=18  3×9=27  4×9=36  5×9=45  6×9=54  7×9=63  8×9=72  9×9=81
```

【例 3-5】 百钱百鸡问题:鸡翁一、值钱五;鸡婆一、值钱三;鸡雏三,值钱一;百钱买百鸡。问鸡翁、鸡婆、鸡雏各几?

分析: 鸡翁最多有 20 个,鸡婆最多有 33 个,鸡雏最多有 100 个。采用穷举的方式,考查每一种可能,是否满足百钱买百鸡。

```
/***********************************
 *   程序名: p3_5.cpp                *
 *   功   能: 求解百钱百鸡问题      *
 ***********************************/
#include<iostream>
using namespace std;
int main()
{
   const int cock=20, hen=33, chick=100;   //分别表示鸡翁、鸡婆、鸡雏的最大数
```

```
10      int i,j,k;
11
12
13      for(i = 0; i <= cock; i++)
14          for(j = 0; j <= hen; j++)
15            for(k = 0; k <= chick; k++)
16              //鸡的个数与钱数必须为整数
17              if ((i + j + k) == 100&&(5 * i + 3 * j + k/3) == 100&&k % 3 == 0)
18                cout <<"鸡翁、鸡婆、鸡雏各有: \t"<< i <<"\t"<< j <<"\t"<< k << endl;
19      return 0;
20  }
```

运行结果:

```
鸡翁、鸡婆、鸡雏各有: 0   25    75
鸡翁、鸡婆、鸡雏各有: 4   18    78
鸡翁、鸡婆、鸡雏各有: 8   11    81
鸡翁、鸡婆、鸡雏各有: 12   4    84
```

程序解释:

➢ 由于计算的数据是整数,为了保证 k/3 是整数,还需加上 k%3==0 的判断。

for 语句是 C++语言中最为灵活的循环语句,可以毫不夸张地讲,C++语言中的 for 语句可以解决编程中的所有循环问题。

对用多重循环实现的程序,有时可以进行循环优化,循环优化是指通过减少循环的层次以及每层循环体执行的次数,以节省系统资源(时间),提高程序运行效率。百钱百鸡问题可以进行多种循环优化,留给读者作为练习。

3.6 while 循 环

在 C++中 while 循环有两种循环控制语句实现,while 语句和 do…while 语句。

3.6.1 while 语句

while 语句的语法形式如下:

```
while(条件表达式) 语句
```

其中:

➢ while 是关键字。

➢ 条件表达式给出是否执行循环体的判断条件,常用关系表达式或逻辑表达式作为条件表达式,也可以用其他表达式或常量。

➢ 语句是 while 循环的循环体,它可以是一条语句,也可以是复合语句。如果在 while 循环头下面有花括号,则循环体将是由花括号括起的复合语句。如果在 while 循环