第3章 语句和函数

语句是程序的最小单元，C++ 的程序就是由一条一条的语句组成的。程序是由程序员写给计算机，并让其执行的语句序列。函数是一个可以独立完成某个功能的语句块，它可以被反复使用，也可以作为一条语句放在程序的任何地方被使用。

3.1 赋值语句

赋值语句是由赋值表达式组成的语句，作用是把一个数据赋给一个变量或数组元素，是最基本的计算机语句，赋值语句的语法结构为：

<变量标识符><赋值运算符><表达式>；

符号“＝”是最简单的赋值运算符，其语义是将表达式的值赋给变量标识符所代表的变量。变量标识符在赋值运算符的左边，所以也称为左值表达式，简称左值，而赋值运算符“＝”符号右边的表达式称为右值。例如：

```
int x;
x=5+3;
```

就是将 5＋3 表达式的值 8 赋给变量 x。

【例 3.1】 赋值语句。

```
int x,y;
float a,b,c;
x=36;                          //赋值语句对 x 赋值 36
y=x*2+2*3-12;                  //赋值语句对 y 赋 x*2+2*3-12 表达式的值
a=36.5;                        //赋值语句对 a 赋值 36.5
b=2.5*a+1.2;                   //赋值语句对 b 赋 2.5*a+ 1.2 表达式的值
c=2.5*3.14-2.7/1.35+2.2;       //赋值语句对 c 赋 2.5*3.14-2.7/1.35+2.2 表达式的值
```

上面实现了 5 个赋值语句。

a＝b 是一个赋值表达式，是将变量 b 的值赋给 a，这是赋值的最简单的情况。

另外还有四个复合赋值运算符，如表 3 .1 所示。

表 3.1　赋值运算符

赋值运算符	示　例	语　义
=	x=a;	a 的值赋给 x
+=	x+=a;	相当于 x=x+a
-=	x-=a;	相当于 x=x-a
*=	x*=a;	相当于 x=x*a
/=	x/=a;	相当于 x=x/a

例如：

```
int a=8;
int b=3;
```

语句 a+=3 等价于 a=a+3，其作用是把 a 的值 8 与 3 相加，再赋值给变量 a，于是变量 a 的值变成 11。

语句 a/=b+1 等价于 a=a/(b+1)，其作用是先把变量 b 的值 3 与 1 相加，得到 4，再用变量 a 的值 8 除以 4，得到 2，最后将 2 赋值给变量 a，于是变量 a 的值变成 2。

从这两个例子，可以看出复合赋值运算符就是将基本的四种算术运算符+(加)、-(减)、*(乘)、/(除)与=相结合得到的。这四种复合赋值运算符都是二元运算符，其优先级和=是一致的，结合性都是从右到左。

【例 3.2】 分析下面的赋值语句。

```
int a=3;
a+=a*=2;
cout<<a;
```

得到输出：

```
12
```

这里出现了两个复合赋值运算符，首先要知道，赋值表达式的值和赋值运算后的左值是一样的，从而它可以当做操作数继续进行别的运算。由于赋值号的结合性是自右向左，所以先进行 a*=2 的赋值运算，变量 a 的值变为 6，接着 6 作为左边的复合赋值运算符+=的右值进行赋值运算，于是左值 a 的值变为 12。

如果赋值号两边的数据类型不一致，编译器会在赋值前将表达式值的类型转换成与左值相同的类型。赋值运算符的两边如果类型不一致，处理原则是：

(1) 如果是将字符型的数据赋给整型变量，虽然字符型的数据也可以看做是特殊的整型数据，但两者所占的字节数是不同的，字符数据占用一个字节，而整型数据占用的字节数要多于字符数据。系统处理的方法是将字符型数据放在整型变量的低 8 位，变量剩余高位的处理方法是，如果系统将字符看做是无符号的，则将剩余的高位全部补 0，但如果系统将字符看做是有符号的，那么为了保持数值不变，则要进行符号扩充，符号扩充的意思，就是要看字符数据的最高一位是 0 还是 1，是 0 则整型变量除了低八位外以外，所有位补 0，是 1 则剩余高位全部补 1。

(2) 如果是将 int 型的数据赋给 long int 型的变量，这种情况和上面的情况相类似，也是两种数据所占的字节数不同，转换就将 int 型的数据放在 long int 型变量的低 16 位，long int 型变量的高 16 位全部补 0。如果是将 unsigned 型的数据赋给 long int 型的变量，则与上述所说的，系统将字符看做是无符号的情况一样，在高位全部补 0。

(3) 将非 unsigned 型的数据赋给长度相同的 unsigned 型的变量，数据不变。

(4) 如果是将实型的数据赋给一个整型的变量，则舍弃实数的小数部分，而不是四舍五入。例如，执行下面的三条语句：

```
int a;
a=3.9;
cout<<" a="<<a;
```

得到结果：a=3。

(5) 如果是将整型的数据赋给实型变量，则数值的大小不变，只是整型数据值根据是赋给单精度变量还是双精度变量，变为与其一致的形式，主要就是补足小数点的位数。

3.2 选 择 语 句

3.2.1 条件语句

条件语句就是一种选择结构，几乎所有高级程序设计语言都提供条件语句，条件语句也称为 if 语句。它用来判定所给出的条件是否满足，并根据判定的结果(真或假)来决定要执行的操作。

条件语句的一般形式为：

```
if(<表达式>)<语句 1>else<语句 2>
```

if 语句的表达式也称为条件表达式，其求值结果必须是取其逻辑类型的值。表达式必须用小括号括起来，含义是当表达式求值结果为真时执行语句 1，否则执行语句 2。

在C++ 中花括号括起来的语句序列看做是一条语句，单独一对花括号是一个空语句，单独写一个分号“;”也是一个空语句，空语句什么都不执行，但它同样是语句。如果 else 后面为空语句，则 else 是可以省略的，这时 if 语句变为如下的形式：

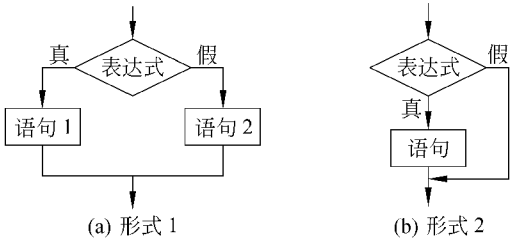
```
if(<表达式>)<语句>
```

两种形式的 if 语句的执行顺序如图 3.1 所示。

【例 3.3】 判断一个数是否大于 60。

程序如下：

```
#include<iostream.h>
void main()
{    int a;
```



(a) 形式 1　　(b) 形式 2

图 3.1　两种形式的 if 语句的示意图

```
    cin>>a;
    cout<<"大于 60 吗?"<<endl;
    if(a>60)cout<<"是";
    else cout<<"否";
}
```

这个程序的功能是读入一个数,判断它与 60 比较的大小,如大于 60 则输出“是”,否则输出“否”,从中可以体会 if 语句的用法。iostream.h 是输入输出头文件,本书为了简洁有时会省略,但实际编写程序时是不能省略的。

【例 3.4】 判断一个整数是正数、负数还是 0。

程序如下:

```
#include<iostream.h>
void  main()
{    int a;
     cin>>a;
     if(a>=0){if(a==0)cout<<"零";
              else cout<<"正数";}
     else cout<<"负数";
}
```

if 语句允许嵌套:

```
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  if(<表达式 n>)<语句 n>
else<语句 n+1>
```

即所有 if 语句的嵌套都发生在若干个 else 分支语句中。执行顺序如图 3.2 所示。

3.2.2　开关语句

开关语句的一般的形式为:

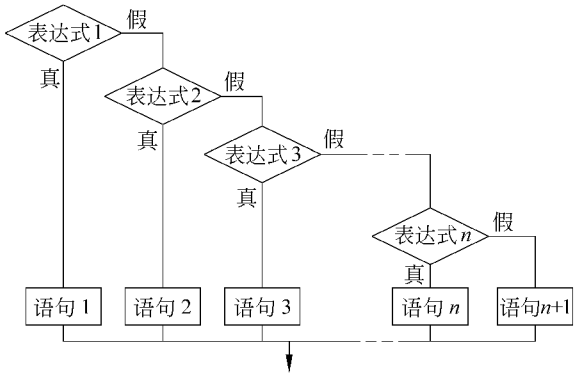


图 3.2　嵌套的 if 语句

```
switch(<开关表达式>){
     case<常量表达式 1>:<语句序列 1>;break;
   case<常量表达式 2>:<语句序列 2>;break;
       ⋮
   case<常量表达式 n>:<语句序列 n>;break;
   default            :<语句序列 n+1>;
}
```

开关语句也称 switch 语句，当程序的流程到达 switch 语句时，首先对开关表达式进行计算，用其值从上到下在所有的 case 常量表达式中寻找相等者，从找到的第一个匹配的 case 后的第一条语句开始执行，当所有的 case 常量表达式都不匹配时，从 default 后面的语句开始执行，当没有 default 并且所有的 case 常量都不匹配时，流程不进入该 switch 语句。

【例 3.5】 一个产品分为 m、p、q 三个等级，根据等级可打印相应的产品的价格。

程序如下：

```
switch(grade){
            case 'm': cout<<"1500";break;
            case'p':cout<<"1000";break;
            case'q':cout<<"500";break;
            defaut :cout<<"没有这个等级";
}
```

最后一个 default 语句，是当常量表达式全部不匹配时执行，意思是所输入的不是合法的产品等级。

switch 语句应该注意以下几点：

(1) 虽然括号内的表达式可以是任意的类型，但最好使用整型、字符型和枚举型的表达式。case 后面的常量表达式的类型必须与开关表达式的类型相匹配。

(2) 各常量表达式的值必须不能相同，否则程序不知该从哪里执行，会出现编译错误。各常量值出现的次序并不影响执行的最后结果。

(3) 每个 case 的后面可跟若干条语句，且不必用{}括起来，因为每个 case 只是一个入口的标志，一旦进入某个入口，便会一直被执行下去，并非遇到下一个 case 就终止。这样，若干个 case 可以共享语句。为了让每个 case 后面的语句执行完后，终止 switch 语句的执行，则必须在每个 case 后的若干条语句的最后加上 break，break 语句的作用就是使程序流程跳出 switch 语句。

【例 3.6】 将月份的阿拉伯数字转换成对应月份的英文单词。

程序如下：

```
#include<iostream.h>
void main()
{    int month;
     cin>>month;
     switch(month){
     case 1:cout<<"January"<<endl;break;
     case 2:cout<<"February"<<endl;break;
     case 3:cout<<"March"<<endl;break;
     case 4:cout<<"April"<<endl;break;
     case 5:cout<<"May"<<endl;break;
     case 6:cout<<"June"<<endl;break;
     case 7:cout<<"July"<<endl;break;
     case 8:cout<<"August"<<endl;break;
     case 9:cout<<"September"<<endl;break;
     case 10:cout<<"October"<<endl;break;
     case 11:cout<<"November"<<endl;break;
     case 12:cout<<"December"<<endl;break;
     default:cout<<"没有这个月份.";
     }
}
```

3.3 循环语句

3.3.1 while 循环语句

while 语句有两种形式。

第一种形式：

```
while(<条件表达式>)
<语句>                                    //循环体
```

其控制机理为：当程序的流程到达 while 语句时，首先对条件表达式进行判断，若其值为真(非 0)，便执行语句，即循环体，否则跳过 while 语句。每执行完一次循环体，都要再计算一次条件表达式，以便判断下一步操作是执行 while 语句，还是跳过 while 语句。

第二种形式：

```
do<语句>                          //循环体
while(< 条件表达式> )
```

其控制机理为：先执行一次循环体，再根据条件表示式判断是否要再执行循环体。

注意：两种形式的区别是，第一种形式是“先判定，后执行”，可能不执行循环体，第二种形式是“先执行，后判定”，至少会执行一次循环体。

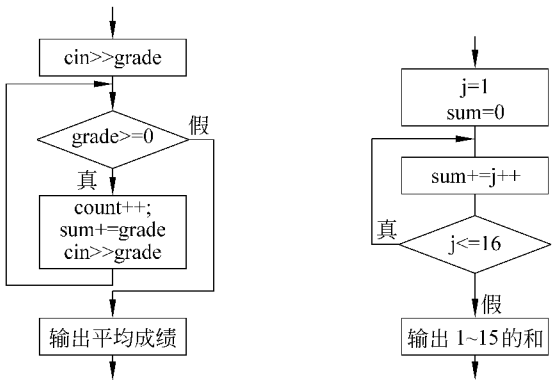
结合例子，来看一下 while 循环语句的执行顺序。

【例 3.7】 读入若干个学生的成绩，计算他们的平均成绩。

程序如下：

```
#include<iostream.h>
void main()
{    int sum=0,count=0,grade;
     cin>>grade;
     while(grade>=0)                          //负数表示输入结束
     {    count++;
          sum+=grade;
          cin>>grade;
     }
     if(count>0)  cout<<"平均成绩是："<<sum/count<<endl;
}
```

这是根据 while 循环语句的第一种形式编写的，其执行示意图如图 3.3(a)所示。



(a) while语句　　(b) do-while语句

图 3.3　while 和 do-while 语句的示意图

再看一个关于 do-while 的例子。

【例 3.8】 求 1～15 的和。

程序如下：

```
#include<iostream.h>
void  main()
```

```
{    int j=1,sum=0;
     do  sum+=j++;
     while(j<16);
     cout<<"1~15 的和是: "<<sum<<endl;
}
```

图 3.3(b)是程序执行的流程示意图。

3.3.2 for 循环语句

C++ 中的 for 语句相对 while 和 do-while 语句较为灵活。它不仅可以用于循环次数已经确定的情况,而且还可以用于循环次数不确定,只给出循环结束条件的情况。for 语句的一般形式为:

for(<表达式 1;表达式 2;表达式 3>)<语句>

其中,表达式 1 是初始化表达式,用来为循环变量赋初值;表达式 2 是条件表达式,用来决定什么时候退出循环;表达式 3 是增量表达式,用来决定循环变量的变化方式。

for 语句的执行顺序如图 3.4 所示。

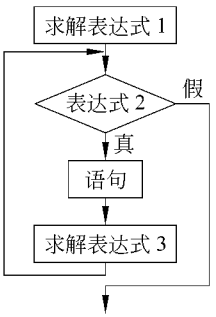


图 3.4 for 语句执行顺序

【例 3.9】 用 for 语句求 1～15 的和。

程序如下:

```
int sum=0;
for(int j=1;j<=15;j++)sum+=j;
```

第一个表达式 int j=1;的作用是定义一个整型的循环变量 j,并将整数 1 赋予此变量,这是在刚进入到 for 语句时执行的。第二个表达式 j<=15;是决定是否进入循环,其值为布尔类型,根据该值决定是否进入循环体,如果为 true(非 0)则进入循环体,如果为 false(0)则跳出 for 语句。第三个表达式 j++的作用是用来改变循环变量 j 的值。

for 语句括号内的三个表达式都可以省略,分号不可以省略,即可以为 for(;;),表示不设初值,不判断条件(认为表达式 2 为真),循环变量无增值,无终止执行循环体,此时,需要在循环体中有跳出循环的控制语句。如果是省略表达式 1,则应该在 for 语句之前,给循环变量赋初值。如果是省略表达式 2,即不判断结束条件,循环无终止进行下去,相当于 while(true)。如果是省略表达式 3,则在循环体内应该有改变循环变量值的语句,以保证循环能够正常结束。

3.3.3 break 和 continue 语句

前面已经提到,使用 break 语句从 switch 语句中跳出来,在循环语句中也可以使用 break 语句,用来从最近的封闭循环体中跳出,除此之外,不能使用 break 语句。

【例 3.10】 读入 10 个学生的成绩，计算其平均成绩，如果遇到负数则报告出错。

程序如下：

```
#include<iostream.h>
void main()
{    int record,sum=0;
     bool flag=true;
     for(int j=1;j<=10;j++)
     {    cin>>record;
          if(record<0){flag=false;break;}
          sum+=record;
     }
     if(flag)cout<<"平均成绩是："<<sum/10.0<<endl;
     else cout<<"出错"<<endl;
}
```

有时可能会遇到另外一种情况，即要跳过循环体里的一些还未执行的语句，进行下一次的循环操作，这时就要用到 continue 语句。continue 语句用于循环语句，作为结束当前循环（跳过循环体中还未执行的语句），接着进行下一次是否执行循环的判定。

【例 3.11】 统计一句话（以“.”结束）里字符 a 的数目。

程序如下：

```
#include<iostream.h>
void main()
{    int count=0;
     char c;
     while(c!='.')
     {    cin>>c;
          if(c!='a')continue;                    //如果不是字符'a',则读下一个字符
          count++;                               //否则计数器 count 加 1
     }
     cout<<count;
}
```

这里 continue 跳过后面的 count＋＋语句，直接回到 while 语句的开头。

3.3.4 多重循环

有时候仅仅使用简单的循环结构，不能够达到编程目的，例如，要给一个二维数组的各个元素赋初值，就要用到二重循环。二重循环其实就是循环语句嵌套循环语句。

【例 3.12】 给一个二维数组的各个元素赋初值。

程序如下：

```
#include<iostream.h>
```

```
void main()
{    int a[3][4];
     cout<<"输入整数: ";
     for(int j=0;j<3;j++)
         for(int k=0;k<4;k++)
         {    cout<<"a["<<j<<"]["<<k<<"]=: ";
              cin>>a[j][k];
         }
}
```

可以看到第二个 for 语句处于第一个 for 语句的循环体当中，对于第一个 for 语句的循环变量 j 的每个值，第二个 for 语句都执行一遍。while 和 for 语句同样也是可以相互嵌套。

3.4 函　　数

编写一个较大程序时，通常将其分成若干个程序模块，每一个模块用来实现一个特定的功能，函数就是一个可以独立完成某个功能的程序模块。在面向对象语言里，函数有着非常重要的作用。一个C++ 程序的入口定义为主函数，也称为主程序，就是前面已经出现过的 main()函数。在一个C++ 程序里只可以定义一个主函数，但对于其余函数的数量是没有限制的，可以根据需要来编写。各个函数包括主函数之间都是平等的，不能在一个函数的内部定义另一个函数，但是各个函数可以互相调用，通过调用各个函数联系在一起，共同构成一个完整的程序。但是，主函数是不可以被别的函数调用的。

函数既可以是用户根据具体的需要定义的，也可以是一些系统定义好可供用户使用的标准函数，又称为库函数。在使用C++ 系统之前应该仔细了解这个系统所提供的库函数。

3.4.1 函数的定义

函数定义的一般形式是：

<类型标识符><函数名>(<形式参数表>)
{<函数体>}

【例 3.13】 编写一个函数，功能是返回一个整数的绝对值。

程序如下：

```
int absolute(int s)
{    int z;                                  //函数体里的变量声明
     if(s>=0)z=s;
     else z=-s;
```