第 3 章　程序控制结构

引言

语句是构成程序的基本单位，程序的运行过程就是执行语句的过程。程序中语句的执行顺序称为流程控制。C++ 支持结构化程序设计，也称面向过程的程序设计，基于具体问题的求解，主要包括顺序结构、选择结构和循环结构这 3 种基本控制结构。顺序结构是指按照程序语句的顺序依次执行相应的语句，选择结构是指依据条件选择执行相应的语句，循环结构是指根据条件重复执行某一程序块。大家要掌握顺序结构、条件结构和循环结构的语法、执行过程和编程；循环嵌套的执行过程和应用；转移语句执行过程和使用，这是结构化程序设计最重要的内容。

学习目标

- 理解：算法流程图表示。
- 熟悉：顺序结构、选择结构和循环结构语法和执行过程。
- 掌握：顺序结构、选择结构、循环结构和转移语句的编程。

课程思政

爱国教育：通过程序流程控制引入求真务实、严谨细致，培养学生严谨的逻辑思维和精益求精的大国工匠精神。

3.1　顺序结构

顺序结构是程序设计中最简单、最常用的结构。在顺序结构中，程序中所有语句按照书写顺序从前往后依次执行，中间没有跳转语句，不漏掉任何一行代码。在程序设计中，顺序结构是最基本的控制结构，也常用作选择结构和循环结构的子结构。

顺序结构程序基本组成包括变量的定义、输入、赋值、数据处理和输出。顺序结构按照语句的先后顺序依次执行，不能改变语句的执行顺序。

【例 3.1】　求 3 个整数的最大值。

```
#include<iostream>
using namespace std;
int main()
{
    long a,b,c,max;                          //a、b、c 为输入的数，max 为最大值
    cout<<"请输入 3 个整数:";
    cin>>a>>b>>c;
    max=a>b? a:b;                            //a 和 b 的大者赋给 max
```

```
    max=max>c?max:c;                                    //max 和 c 的大者赋给 max
    cout<<"max="<<max<<endl;
    return 1;
}
```

程序运行时,输入内容和运行结果如下:

```
请输入 3 个整数:123 -2324  243
max=243
```

在本例中,先求 a 和 b 的大值,然后再把 a 和 b 的大值和 c 进行比较,求出最大值,程序按照语句顺序依次执行。

【例 3.2】 分糖果。有 3 个小朋友,甲有 x 粒糖果,乙有 y 粒糖果,丙有 z 粒糖果。现在他们玩一个游戏。甲将它的糖分成三等份,多余的自己吃掉,然后自己留一份,其余两份分别给乙和丙;乙和丙也这样做。最后甲、乙、丙各有多少粒糖果?输入甲、乙、丙最初的糖果数,输出甲、乙、丙最后的糖果数。


分糖果

```
#include<iostream>
using namespace std;
int main()
{
    int x, y, z, t;
    cout << "请输入甲最初的糖果数:";
    cin >> x;
    cout << "请输入乙最初的糖果数:";
    cin >> y;
    cout << "请输入丙最初的糖果数:";
    cin >> z;
    //甲分
    t = x / 3;
    x = t;
    y = y + t;
    z = z + t;
    //乙分
    t = y / 3;
    y = t;
    x = x + t;
    z = z + t;
    //丙分
    t = z / 3;
    z = t;
    x = x + t;
    y = y + t;
    cout << "甲最后的糖果数:" << x << endl;
    cout << "乙最后的糖果数:" << y << endl;
    cout << "丙最后的糖果数:" << z << endl;
    return 1;
}
```

程序运行时,输入和运行结果如下:

```
请输入甲最初的糖果数:16
请输入乙最初的糖果数:23
请输入丙最初的糖果数:32
甲最后的糖果数:29
乙最后的糖果数:24
丙最后的糖果数:15
```

在本例中，按题目要求定义变量，根据提示输入甲、乙、丙最初的糖果数。甲先分糖果，然后是乙和丙分糖果，最后输出甲、乙、丙最后的糖果数。

3.2 选择结构之一——if 语句

在学习和工作中，人们常常会用百度查询资料，查询到可能满足条件的信息时，系统是如何判定的呢？这里面就涉及条件判定问题，可以用 if 语句实现。if 语句分为单分支 if 语句、双分支 if 语句(if-else 语句)和嵌套的 if 语句。


if 语句

3.2.1 单分支 if 语句

单分支 if 语句的基本格式如下：

```
if (表达式)
    语句
```

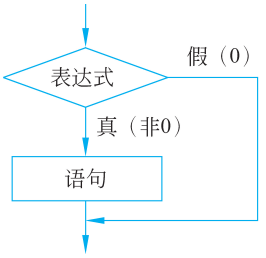

图 3.1 单分支 if 语句的执行过程

if 是关键字，后面紧跟一对圆括号，括号里面是条件表达式。单分支 if 语句的执行过程如图 3.1 所示，如果表达式成立，就执行语句；否则结束 if 语句，执行 if 语句后面的语句。

注意

(1) if 语句中的表达式可以是任意表达式，表达式的值非 0 为真，0 为假。例如：

```
int x=3;
if(x)
    cout<<x;
```

(2) if 语句只会控制后面一条语句。如果要控制多行，需要用花括号括起来，构成复合语句。

【例 3.3】 输入两个整数，按从大到小的顺序输出这两个整数。

```
#include <iostream>
using namespace std;
int main()
{
    int a, b, t;
    cout << "请输入两个整数:";
    cin >> a >> b;;
    if (a < b)
    {
        t = a;   a = b; b = t;
    }
    cout <<"从大到小输出:"<< a << "    " << b << endl;
```

```
    return 1;
}
```

程序运行时，输入和运行结果如下：

```
请输入两个整数:12  34
从大到小输出:34  12
```

在本例中，如果 a 小于 b，则交换 a 和 b 的值，要借助中间变量 t 暂时存放 a 的值。交换 a 和 b 的值有 3 条语句，因此要使用花括号括起来，构成一条复合语句。

【例 3.4】 设分段函数如下：

$$f(x)=\begin{cases} x^{0.5}, & x>0 \\ (x+1)^2+2x+1/x, & x\leqslant 0 \end{cases}$$

输入 x 的值，输出函数 $f(x)$的值。

```
#include <iostream>
#include<cmath>
using namespace std;
int main()
{
    float x, y;
    cin >> x;
    if (x >0)
        y = sqrt(x);
    if (x <= 0)
        y = pow(x + 1, 2) + 2 * x + 1 / x;
    cout << "x=" << x << ",y=" << y;
    return 0;
}
```

程序运行时，输入 5 和运行结果如下：

```
5
x=5,y=2.23607
```

在本例中，函数 sqrt 的原型为 double sqrt(double x)，功能是返回 x 的算术平方根；函数 pow 的原型为 double pow(double x, double y)，功能为返回 x 的 y 次幂。调用 sqrt 和 pow 函数，需要包含头文件 cmath。

注意 数学中的表达式在程序中的书写格式要符合 C++ 的规定。

3.2.2 双分支 if 语句

双分支 if 语句即 if-else 语句，其基本格式如下：

```
if (表达式)
  语句 1
else
  语句 2
```

if-else 语句的执行过程如图 3.2 所示。如果表达式成立，执行语句 1；否则，执行语句 2。语句 1 和语句 2 两条分支只能选择一条执行。如果语句 1、语句 2 是多条语句，需要使用花

括号括起来，构成一条复合语句。

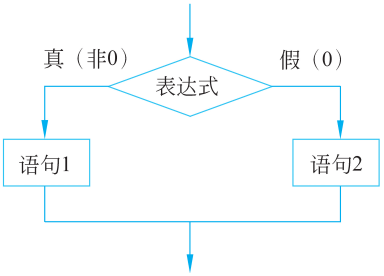

图 3.2　if-else 语句的执行过程

【例 3.5】　输入一个年份，判断是否是闰年。如果是，输出该年是闰年的信息；否则输出该年不是闰年的信息。

```
#include <iostream>
using namespace std;
int main()
{
    int year;
    cout << "请输入一个年份:";
    cin >> year;
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
        cout << year << "年是闰年" << endl;
    else
        cout << year << "年不是闰年" << endl;
    return 1;
}
```

程序运行时，输入和运行结果如下：

```
请输入一个年份:2024
2024 年是闰年
```

在本例中，要注意闰年判断的条件表示。判断条件为：闰年是能被 4 整除但不能被 100 整除的年份或能被 400 整除的年份。

3.2.3　嵌套的 if 语句

嵌套的 if 语句是指 if 语句里面再嵌套 if 语句，其基本格式为

```
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
```

嵌套的 if 语句要用缩进和对齐体现结构性。内嵌的 if-else 语句可以不加大括号，但为了增强程序的结构性和可读性，也可以加上大括号。同理，语句 1、语句 2、语句 3 和语句 4 如果是多条语句，则要加大括号。

在嵌套的 if 语句中，if 与 else 的配对非常重要。当 if 与 else 出现多次时，在省略大括号的情况下，else 与前面最近的未配对的 if 配对。例如：

```
if(a==b)
    if(b==c)
        cout<<"a==b==c";
    else
        cout<<"a!=b";
```

上面的代码的功能是判断 3 个变量 a、b、c 是否相等。在省略大括号情况下，else 与前面最近的未配对的 if，即第 2 个 if 配对。但是，这样配对的执行过程与条件表达不一致。上面的代码应修改为

```
if(a==b)
{
    if(b==c)
        cout<<"a==b==c";
}
else
    cout<<"a!=b";
```

在程序中，经常会用大括号把一条或者多条语句括起来，构成一个整体，以增强程序的逻辑性和可读性。

判定一个整数的符号用 if-else 的嵌套结构实现的代码如下：

```
#include<iostream>
using namespace std;
int main( )
{
    int x, y;
    cin>>x;
    if(x < 0)
        y = -1;
    else
        if(x == 0 )
            y = 0;
        else
            y = 1;
    cout<<"x="<<x<<",y="<<y;
    return 0;
}
```

程序基本结构为两层 if-else 语句的嵌套结构，它在 else 后面内嵌了一个 if-else 语句，实际上是多分支结构。

【例 3.6】 编程求解一元二次方程 $ax^2+bx+c=0$ 的根。

```
#include<iostream>
```

```
#include<cmath>
using namespace std;
int main()
{
    float a, b, c, d, x1, x2;                    //a、b、c 为方程系数
    cin >> a >> b >> c;
    if (a != 0)                                  //一元二次方程判定
    {
        d = b * b - 4 * a * c;
        if (d >= 0)                              //求实根
        {
            x1 = (-b + sqrt(d)) / (2 * a);
            x2 = (-b - sqrt(d)) / (2 * a);
            cout << "方程实根解为 :"<< endl;
            cout<<"x1 = " << x1 << endl<<"x2 = " << x2 << endl;
        }
        else                                     //求虚根
        {
            x1 = b / (2 * a);
            x2 = sqrt(-d) / (2 * a);
            cout << "方程虚根解为:" << endl;
            cout << "x1 = " << x1 << " + " << x2 << "i" << endl;
            cout << "x2 = " << x1 << " - " << x2 << "i" << endl;
        }
    }
    else                                         //非一元二次方程
        cout << "输入数据有误\n";
    return 1;
}
```

程序运行时，输入和运行结果如下：

```
5  1  2
方程虚根解为:
x1 = 0.1 + 0.6245i
x2 = 0.1 - 0.6245i
```

在本例中，程序整体结构就是 if-else 双分支结构，a!＝0 条件为真时执行的语句块中包括两条语句，分别是 d＝b＊b－4＊a＊c；以及内嵌的 if-else 语句。在内层 if-else 语句中，d＞＝0 条件为真求实根，为假求虚根。要注意大括号的使用。

3.3 选择结构之二——switch 语句

观察下面这些问题有什么共同点，学生成绩分类处理问题，把百分制成绩转换成对应的等级；人口统计分类问题，根据年龄分为老年、中年、青年、少年和儿童；菜单命令操作问题，根据系统菜单命令选择执行不同的操作；个人所得税问题，根据收入分成多个档次。这些问题的共同点就是要把数据分成几类分别进行处理，属于多分支结构。

多分支结构可以用 if 嵌套语句处理，也可以用 switch 语句处理。switch 语句有两种形

式，带 break 的 switch 语句和不带 break 的 switch 语句。

3.3.1 带 break 的 switch 语句


switch 语句

带 break 的 switch 语句基本格式如下：

```
switch(表达式 P)
{
    case e1:
        语句 1;break;
    case e2:
        语句 2;break;
    ⋮
    case en:
        语句 n;break;
    default:
        语句 n+1;break;
}
```

switch 是系统提供的关键字，switch 后面的圆括号中的表达式是算术表达式，其值为整型或者字符型。大括号里面有多个 case，case 后面的 e_1，e_2，…，e_n 是常量，再后面的冒号是标号符。

带 break 的 switch 语句的执行过程如图 3.3 所示。首先计算 switch 后面的圆括号中的表达式 P，如果其值等于某个 case 后面的常量表达式 e_1～e_n，就执行对应的语句，当遇到 break 时，则立即结束 switch 语句的执行，执行大括号后面的语句。如果 P 不等于 e_1～e_n 中的任何一个常量，当有 default 时，就执行 default 后面的语句组；否则，就结束 switch 语句，执行大括号后面的语句。

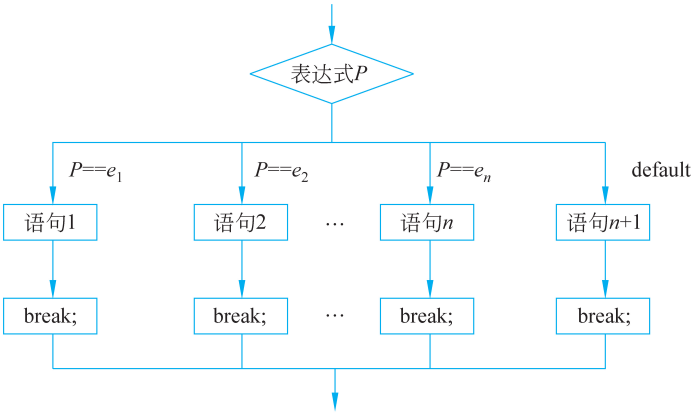

图 3.3 带 break 的 switch 语句执行过程

注意

(1) case 后可包含多条语句，且不必加大括号。

(2) default 是可选的。如果没有一个 case 分支可以执行，同时也没有 default，则不会执行 switch 语句中的任何语句。

(3) e_1，e_2，…，e_n 必须是整型或字符型常量表达式，且值必须不相同。

（4）语句组后面的 break 用来结束 switch 语句，执行大括号后面的语句。

（5）多个 case 可共用一组语句，只保留最后一个 case 后面的语句，前面的可以省略。例如：

```
case  'A':
case  'B':
case  'C':
       cout<<" score>60\n";
       break;
```

（6）最后一条语句可以不用 break，因为遇到右大括号结束。

【例 3.7】 输入成绩等级，等级为 1～5 的整数，5 为 Very good，4 为 Good，3 为 Pass，2 和 1 为 Fail，输出相应的成绩等级。

```
#include <iostream>
using namespace std;
int main()
{
    int score;
    cin >> score;
    switch (score)
    {
        case 5:   cout << "Very good!";      break;
        case 4:   cout << "Good!";           break;
        case 3:   cout << " Pass!";          break;
        case 2:
        case 1:   cout << "Fail!";           break;
        default: cout << " Data error!";     break;
    }
    system("pause");
    return 1;
}
```

程序运行时，输入和运行结果如下：

```
5
Very good!请按任意键继续. . .
```

注意 system 用于调用 Windows 下的 DOS 命令函数，函数原型为

```
int system(char * command);
```

pause 命令的功能是暂停屏幕，以方便用户观察程序运行结果。

3.3.2 不带 break 的 switch 语句

不带 break 的 switch 语句的基本结构如下：

```
switch(表达式 P)
{
    case e1:
        语句 1;
    case e2:
```

```
        语句 2;
    ⋮
    case e_n:
        语句 n;
    default:
        语句 n+1;
}
```

不带 break 的 switch 语句的执行过程如图 3.4 所示。如果表达式 P 的值与常量表达式 e_1,e_2,…,e_n 中的某个匹配,就执行相应的语句,然后往下继续执行其余的语句。实际上不带 break 的 switch 语句不是真正意义上的多分支结构。

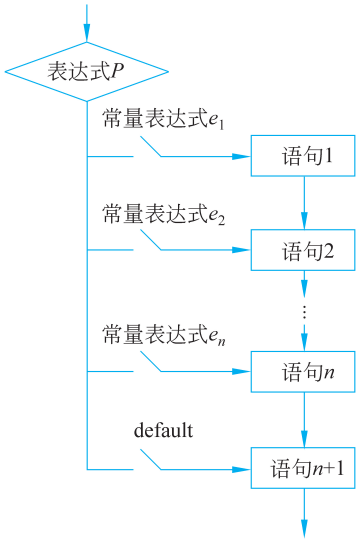

图 3.4 不带 break 的 switch 语句执行过程

将例 3.7 的代码改为不带 break 的 switch 语句,代码如下:

```
switch(score)
{   case 5:   cout<<"Very good!";
    case 4:   cout<<"Good!";
    case 3:   cout<<"Pass!";
    case 2:   cout<<"Fail!";
    default: cout<<"data error!";
}
```

程序运行时,输入和运行结果如下:

```
5
Very good! Good! Pass! Fail! data error!
```

【例 3.8】 输入一个字母,输出相应的问候语。

```
#include<iostream>
using namespace std;
int main()
{
    char c;
    cout <<"Enter m or n or h or other:";
    c = getchar();                                  //输入一个字母赋值给 c
    switch (c)
    {
        case 'm':  cout << "Good morning!\n";  break;
        case 'n':  cout << "Good night!\n";    break;
        case 'h':  cout << "Hello!\n";         break;
        default:   cout << "How are you.\n";   break;
    }
    system("pause");                                //系统函数,暂停屏幕显示
    return 0;
}
```

程序运行时,输入和运行结果如下:

```
Enter m or n or h or other:m
Good morning!
```

在本例中，c=getchar()表示从键盘输入一个字母并赋给 c，getchar 为系统的无参函数，可以接收从键盘输入的任意字符。switch 语句的条件表达式就是变量 c 的取值。如果为 m，则输出“Good morning!”；如果为 n，则输出“Good night!”；如果为 h，则输出“Hello!”；如果不是这 3 个字母，则输出“How are you.”。

【例 3.9】 输入百分制成绩，输出相应的成绩等级。

```
#include<iostream>
using namespace std;
int main()
{
    float score;
    cin >> score;
    switch((int)score / 10)
    {
        case 10:
        case 9:  cout << "优秀!";   break;
        case 8:  cout << "良好!";   break;
        case 7:  cout << "中等!";   break;
        case 6:  cout << "及格!";   break;
        default: cout << "不及格!"; break;
    }
    system("pause");
    return 1;
}
```

程序运行时，输入和运行结果如下：

```
88.5
良好!请按任意键继续. . .
```

在本例中，成绩 score 为浮点型，switch 后面圆括号中的表达式是把 score 强制转换为整型，再除以 10，即去掉个位。然后根据 score 的十位数字判断等级。如果为 10 或者 9，则输出“优秀!”；如果为 8，则输出“良好!”……如果十位数字小于 6，则成绩低于 60 分，输出“不及格!”。

case 后面只能是常量表达式，不能写成条件表达式，不能含有变量。例如，在本例中，case 后面写成 score>=90 && score<=100 是错误的。

【例 3.10】 菜单程序。

```
#include<iostream>
using namespace std;
int main()
{
    void action1(int,int),action2(int,int);  //函数原型声明
    char ch; int a=15,b=23;
    ch=getchar();
    switch(ch)
```

```
    {
        case 'a':
        case 'A': action1(a,b);break;
        case 'b':
        case 'B': action2(a,b);break;
        default: putchar('\a');
    }
    return 0;
}
void action1(int x,int y)                    //定义函数
{
    cout<<"x+y="<<x+y>>endl;
}
void action2(int x,int y)                    //定义函数
{
    cout<<"x*y="<<x*y>>endl;
}
```

在本例中，switch 语句的条件表达式是变量 ch 的值。如果为字母 a，无论大小写，均调用 action1 函数，输出 a、b 的和；如果为字母 b，无论大小写，均调用 action2，输出 a、b 的乘积。如果既不是 a 也不是 b，则输出响铃警报。转义字符'\a'表示响铃警报。

3.4 循环结构之一——while 和 do-while 语句

大家回顾一下如何求 1＋2＋…＋100。数学上的计算过程是从左往右依次进行累加计算。可以设置一个累加器 sum，其初始值为 0，然后利用 sum ＝ sum ＋n 进行累加计算，n 依次取 1，2，…，100。累加需要解决 3 个问题：①将 n 的初始值置为 1；②每次执行 sum ＝ sum＋ n 后，n 增 1；③当 n 增到 101 时停止计算。其中的累加求和 sum＝sum＋n 和 n＝n＋1 需要重复进行，编程求解需要用到循环结构。

循环结构是在给定条件成立的情况下重复执行某些操作。C++ 的循环结构主要有 while 循环、do-while 循环和 for 循环 3 种。本节主要学习 while 语句和 do-while 语句。

3.4.1 while 语句


while 和 do-while

while 语句的一般格式如下：

```
while(表达式)
{
    循环体
}
```

while 是系统提供的关键字，表达式为条件表达式。循环体中的语句如果是两条或两条以上，要用大括号括起来，构成一条复合语句。while 语句的执行过程如图 3.5 所示，当表达式为真(非 0)时，执行循环体语句，然后继续下一次条件判断，如果为真执行循环体；如果表达式为假(0)时就结束。

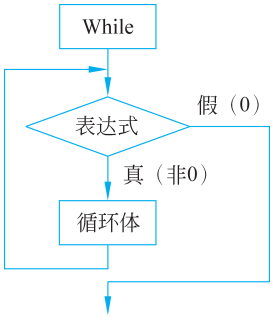

图 3.5 while 语句的执行过程

while 语句特点是先判断后执行，是当循环，当条件成立就执行循环体。

【例 3.11】 利用 while 循环编程输出 1～10 的平方。

```
#include<iostream>
using namespace std;
int main()
{
    int i=1;
    while(i<=10)
    {
        cout<<i*i<<"  ";
        i++;
    }
    return 1;
}
```

程序运行结果如下：

```
1   4   9   16   25   36   49   64   81   100
```

在本例中，循环变量 i 的初始值为 1，循环条件是 i＜＝10，循环体是输出 i＊i 并将循环变量 i 加 1。

注意 循环结构有 3 个要素：

(1) 循环变量赋初始值。

(2) 循环条件。

(3) 循环变量的改变。循环变量一定要向着最终使循环条件为假的方向改变，否则就是一个死循环。

【例 3.12】 用 while 循环编程求 1＋2＋…＋100 的和。

```
#include<iostream>
using namespace std;
void main()
{
    int i,sum=0;
    i=1;
    while(i<=100)
    {
        sum=sum+i;
        i++;
    }
    cout<<sum;
}
```

程序运行结果如下：

```
5050
```

在本例中，定义了循环变量 i 和累加求和变量 sum，sum 的初始值为 0。循环变量的初始值为 1，i＜＝100 为循环条件，100 为循环终值。在循环体中进行累加求和并将循环变量加 1。

【例 3.13】 输入两个整数，求它们的最大公约数和最小公倍数。

```
#include <iostream>
using namespace std;
int main()
{
    int a, b,ma,mb,t,gcd,lcm;
    cout << "请输入两个整数:";
    cin >> a >> b;
    ma = a;  mb = b;
    while(a%b!= 0)
    {
        t = a % b;
        a = b;
        b = t;
    }
    gcd = b;
    lcm = ma * mb / gcd;
    cout << "最大公约数为:" << gcd << endl;
    cout << "最小公倍数为:" << lcm << endl;
    return 1;
}
```

程序运行时，输入和运行结果如下：

```
请输入两个整数:15  25
最大公约数为:5
最小公倍数为:75
```

在本例中，求最大公约数采用的是欧几里得算法，即辗转相除法。其具体思想是：如果 a 除以 b 的余数等于 0，则最大公约数为 b；否则把 b 赋值给 a，把余数赋值给 b，再进行下一次循环判定。在辗转相除的过程中，a 和 b 的值发生了改变，需要定义 ma 和 mb 两个变量分别保存 a 和 b 的值，就可以使用 ma 和 mb 求最小公倍数了。

3.4.2 do-while 语句

do-while 循环语句的一般格式如下：

```
do
{
    循环体
} while(表达式);
```

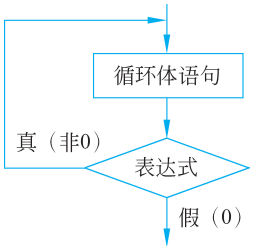

图 3.6 do-while 语句的执行过程

do-while 语句先执行循环体，再进行条件判断，直到条件为假时结束。do-while 语句的执行过程如图 3.6 所示。

【例 3.14】 用 do-while 语句求 1＋2＋…＋100 的和。

```
#include<iostream>
using namespace std;
int main()
{
```

```
    int i,sum=0;
    i=1;
    do
    {
        sum=sum+i;
        i++;
    }while(i<=100);
    cout<<sum;
    return 1;
}
```

在本例中，i 为循环变量，初值为 1；sum 为累加求和变量。先执行 sum＝sum＋i，然后 i 增加 1，然后判断 i＜＝100 是否成立。若成立，则继续执行；否则结束循环，输出 sum 的值。

3.4.3 while 语句和 do-while 语句的区别

while 语句先判断后执行，是当型循环，当条件成立时就执行，循环体有可能一次也不执行。while 表达式后面没有分号。

do-while 语句先执行后判断，是直到型循环，直到条件不成立时结束，循环体至少执行一次。while 表达式后面有分号。

【例 3.15】 分析下列两个程序的运行结果。

（1）do-while 循环：

```
#include<iostream>
using namespace std;
int main()
{
    int i=11,sum=0;
    do
    {
        sum+=1;
        i++;
    }while(i<=10);
    cout<<sum;
}
```

（2）while 循环：

```
#include<iostream>
using namespace std;
int main()
{
    int i=11,sum=0;
    while(i<=10)
    {
        sum+=1;
        i++;
    }
    cout<<sum;
}
```

在 do-while 循环中，循环变量 i=11，先执行累加(sum+=1)，i 自身加 1，循环条件 i<=10 不成立，循环结束，输出结果为 11。在 while 循环中，循环变量 i=11，循环条件 i<=10 不成立，循环体不执行，输出结果为 0。

【例 3.16】 输入一个整数，输出其各位数字之和。

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    int a, x, n, sum = 0;
    cin >> x;
    n = abs(x);
    while (n != 0)
    {
        a = n % 10;
        sum = sum + a;
        n = n / 10;
    }
    cout << x << "各位数之和为:" << sum;
    return 1;
}
```

程序运行时，输入和运行结果如下：

```
-123
-123 各位数之和为:6
```

在本例中，while 语句的循环变量为 n，初值为 x 的绝对值，n!=0 为循环条件。如果满足条件，则执行循环体的 3 条语句，首先将 n 的个位数字赋给 a，然后累加 a，最后循环变量 n 除以 10 取整再赋值给 n，即 n 去掉个位数字。下一轮循环继续判断循环条件，当 n=0 时结束循环，就实现了求整数各位数字之和的目的。本例的算法思想是：求个位数字，累加个位数字，去掉个位数字后开始下一轮循环，直到 n 为 0 时结束。

注意 abs 为系统求绝对值函数，包含在头文件 cmath 中。

【例 3.17】 按以下规则将明文变成密文：将字母 A 变成字母 E，即变成其后的第 4 个字母，W 变成 A，X 变成 B，Y 变成 C，Z 变成 D；小写字母采用相同规则。

```
#include<iostream>
using namespace std;
int main()
{
    char c;
    while ((c = getchar()) != '\n')
    {
        if (c >= 'a' && c <= 'z')
        {
            c = c + 4;
            if (c > 'z')   c = c - 26;
        }
```

```
            if ((c >= 'A' && c <= 'Z'))
            {
                c = c + 4;
                if (c > 'Z') c = c - 26;
            }
            cout << c;
        }
        return 1;
    }
```

程序运行时，输入和运行结果如下：

```
ZDSAGhsa
DHWEKlwe
```

在本例中，while 语句先执行 getchar 函数读入一个字符赋值给 c，再判定 c 是否是换行符。循环体用 if 语句判定是否是大写字母或者小写字母，如果是，则 c＝c＋4，即变成其后的第 4 个字母。然后继续判定 C＋4 后是否已超出字母范围，如果 c＞'Z'，或者 c＞'z'成立，执行 c＝c－26，即 c 往回移 26 个字符位置，以解决字母循环移位的问题。

3.5 循环结构之二——for 语句

分析以下问题的共同点：①$1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\cdots+\frac{1}{99}-\frac{1}{100}$，分母从 1 到 100 递增，每一项符号交替变化；②1！＋2！＋3！＋…＋10！，依次求阶乘，然后累加求和；③输出所有的水仙花数，所谓“水仙花数”是指一个三位数，其各位数字的 3 次方之和等于该数本身（例如 $153=1^3+5^3+3^3$），需要对 100～999 依次进行判定。它们的共同点是：有明确的开始值和结束值以及按一定规律变化的步长，需要逐项进行处理。这类问题适合用 for 循环实现。


for 循环

3.5.1 for 语句的基本形式

for 语句的基本形式如下：

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

for 后面的圆括号中，表达式 1 和表达式 2 后面有分号，表达式 3 后面没有。表达式 1 是循环变量初始值，表达式 2 是循环条件，表达式 3 是循环变量的变化规律。循环体如果有多条语句，要用大括号构成一条复合语句。

for 语句的执行过程如图 3.7 所示。先执行表达式 1，给循环变量赋初始值；再执行表达式 2，判断循环条件是否成立，如果成立，则执行循环体；最后执行表达式 3，使循环变量改变。随后进入下一轮循环，直到循环条件为假时结束循环。

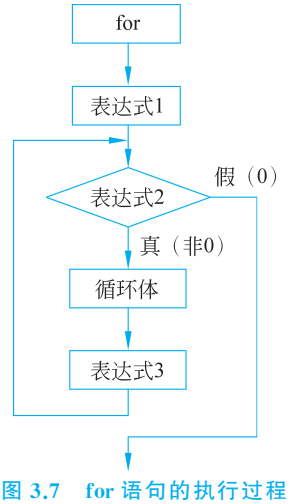

图 3.7 for 语句的执行过程

【例 3.18】 输入一个整数，求它的阶乘。

```
#include<iostream>
#include<iostream>
using namespace std;
int main()
{
    long n, i, s = 1;
    cout << "请输入一个整数:";
    cin >> n;
    for (i = 1; i <= n; i++)
        s *= i;
    cout << n << "!=" << s;
    return 1;
}
```

程序运行时，输入和运行结果如下：

```
请输入一个整数:5
5!=120
```

在本例中，阶乘的值很大，注意变量值的溢出，最好把存放阶乘值的变量 s 定义为取值范围较大的数据类型，例如长整型或者双精度浮点型。累乘变量要赋初始值(为 1)。本例求 n 的阶乘使用的公式为 $n!=1\times2\times\cdots\times n$。

【例 3.19】 输入一个整数 n，求 $1!+2!+\cdots+n!$。

```
#include<iostream>
using namespace std;
int main()
{
    double n, i, s = 0,f=1;
    cout << "请输入一个整数:";
    cin >> n;
    for (i = 1; i <= n; i++)
    {
        f = f * i;
        s = s + f;
    }
    cout << "阶乘和:" << s << endl;
    return 1;
}
```

程序运行时，输入和运行结果如下：

```
请输入一个整数:5
阶乘和:153
```

本例是一个累加求和问题，存放累加结果的变量 s 的初始值为 0。累加项为 i 的阶乘。f 用于存放阶乘的值，其初始值为 1。如果本例求阶乘用例 3.18 的公式，则要用循环的嵌套结构。

3.5.2 for 语句形式的变化

for 语句中的表达式 1、表达式 2 和表达式 3 可以是任意类型的表达式，都可以省略，但分号不可以省略。例如，求 1＋2＋…＋100 有以下 4 种情况。

（1）3 个表达式都不省略，这是基本用法。

```
#include<iostream>
using namespace std;
int main()
{
    int i,sum;
    for(i=1,sum=0;i<=100;i++)
        sum=sum+1;
    cout<<sum;
    return 1;
}
```

（2）省略表达式 1，将其放到 for 循环之前。

```
#include<iostream>
using namespace std;
int main()
{
    int i,sum;
    i=1,sum=0;
    for(;i<=100;i++)
        sum=sum+1;
    cout<<sum;
    return 1;
}
```

（3）在(2)的基础上省略表达式 3，将其放在循环体的最后。

```
#include<iostream>
using namespace std;
int main()
{
    int i,sum;
    i=1,sum=0;
    for(;i<=100;)
    {
        sum=sum+1;
        i++;
    }
    cout<<sum;
    return 1;
}
```

（4）在(3)的基础上省略表达式 2，循环条件默认为真。需要在循环体中用 if 语句对循环条件进行判定，当循环条件为假时要用 break 语句退出循环。

```
#include<iostream>
using namespace std;
int main()
{
    int i,sum;
    i=1,sum=0;
    for(;;)
    if(i<=100)
    {
        sum=sum+1;
        i++;
    }
    else
        break
    cout<<sum;
    return 1;
}
```

【例 3.20】 输出所有的水仙花数。水仙花数是指一个 3 位数，其各位数字立方和等于该数本身，例如，$153=1^3+5^3+3^3$。

```
#include<iostream>
using namespace std;
int main()
{
    int a, b, c, x;
    for (x = 100; x <= 999; x++)
    {
        a = x / 100;
        b = x % 100 / 10;
        c = x % 10;
        if (a*a*a + b * b*b + c * c*c == x)
            cout << x << "   ";
    }
    return 1;
}
```

程序运行结果如下：

```
153   370   371   407
```

在本例中，水仙花数是一个三位数，需要对 100～999 依次进行判定，因此可以用 for 循环实现。定义 x 为循环变量，赋初值为 100，循环条件为 x<＝999，循环体实现求百位(a＝x/100)，求十位(b＝x%100/10)，求个位(c＝x%10)，再用 if 语句对该数是否为水仙花数进行判定，如果是就输出该数。求立方和可以使用 pow( )函数，其原型为 double pow(double x, double y)，求幂运算，表示 x 的 y 次方，如 pow(b,3)为 b 的 3 次方。

【例 3.21】 评委评分程序。在一次运动会方队表演中，学校安排了 10 名老师进行打分。对于每个参赛班级的 10 个打分(百分制整数)，去掉一个最高分和一个最低分，再算出平均分，作为该班级的最后得分，保留 3 位小数输出。