第5章 累加——循环结构

能力目标

- 能运用 for、while 和 do-while 循环语句，理解递归调用方法；
- 能使用加赋值、乘赋值等复合赋值运算符；
- 能运用循环结构编写计算累加、输出金字塔图案等程序。

ch5.1～5.2

5.1 任务预览

本章实训编写的计算累加、输出金字塔图案程序，运行结果如图 5-1 所示。

(a) 累加　　(b) 金字塔

图 5-1　实训程序运行界面

5.2 while 语句

程序有顺序、分支和循环 3 种控制结构，循环语句用于实现循环结构。

Java 语言循环语句有 3 个：while、for 和 do-while。

首先介绍简单易懂的 while 语句，其语法格式如下：

```
while (条件表达式 )
    循环体
```

该语句由关键字 while 开头，表示当圆括号内的条件表达式为 true，则执行后面的循环体，否则不执行。循环体一般是用大括号括起来的若干条语句。执行完一次循环体，再回过

头判断条件表达式，为 true，继续执行循环体。如此循环往复，直到条件表达式为 false 才结束整个语句。

因此，while 语句的条件表达式和循环体都有可能执行多次，条件表达式必须执行一次以上，但循环体也可能一次都不执行，因为首次判断条件时若为 false，就会退出语句。

while 语句的流程图如图 4-5(a)所示。

【例 5-1】 编程，计算 1～10 的累加，即 1＋2＋3＋…＋10。

分析：设初值为 0，则有 10 次加法运算，即 0＋1＋2＋3＋…＋10。把重复的加法运算作为循环体，循环执行 10 次，每次运算结果是上次结果加上递增的次数，次数从 1 开始。

为清晰起见，下面使用算法描述该问题的解决步骤：

(1) sum ← 0(符号←表示赋值，按箭头方向把初值 0 赋给累加变量 sum)。

(2) i ← 1(把初值 1 赋给次数变量 i)。

(3) 当 i≤10 时，执行下一步，否则跳转到步骤(8)。

(4) sum ← sum＋i(sum 累加 i 值，运算结果是上次结果加上递增次数)。

(5) 输出中间结果 sum，即 i 的累加(本步骤可省略)。

(6) i ← i＋1(次数递增)。

(7) 转回步骤(3)。

(8) 输出结果 sum。

其中第(3)～(7)步就是执行多次的循环结构。

用 Java 语言编程实现上面算法，代码如下：

```
public class Ex1 {
    public static void main(String[] args) {
        int sum = 0;                                          //算法步骤(1)
        int i = 1;                                            //算法步骤(2)
        while(i <= 10) {                                      //算法步骤(3)
            sum = sum + i;                                    //算法步骤(4)
            System.out.printf("%d 的累加：%d\n", i, sum);     //算法步骤(5)
            i++;                                              //算法步骤(6)
        }                                                     //算法步骤(7)
        System.out.printf("结果：%d", sum);                   //算法步骤(8)
    }
}
```

进行累加操作之前，先定义存放累加结果的变量 sum 并赋初值 0。使用循环语句，涉及循环次数控制问题，一般使用变量来控制，如 i，称为“循环控制变量”。循环控制变量也要赋初值，如 i＝1。每循环一次，值递增 1，可用自增运算实现，如 i＋＋。

程序运行结果如图 5-2 所示。若删掉 while 循环体内输出语句则只输出“结果：55”。

图 5-2 1～10 累加

注意：循环体有两条以上语句必须用大括号括起来。为规范和便于维护，建议不管循环体有多少条语句，均用大括号括起来。

5.3 复合赋值运算符

ch5.3

在例 5-1 中使用了赋值语句：

```
sum = sum + i;
```

语句中的表达式使用了两个运算符：＋和＝，先把 sum 值加 i 值，再赋给 sum。

可把加法和赋值这两种运算合成一体，用一个复合运算符＋＝表示，如：

```
sum += i;                                    //用一个加赋值运算符 += 执行两种运算
```

复合赋值运算符除了加赋值(累加)，还有减赋值、乘赋值、除赋值和求余赋值等，依次为－＝、＊＝、/＝和％＝。除了算术运算，还有逻辑运算的复合赋值，如逻辑与、逻辑或和异或赋值，运算符为 &＝、|＝和^＝。左、右移位也有复合赋值运算符：<<＝和>>＝。

复合赋值运算由二元运算和赋值运算合成，其表达式语法格式如下：

```
变量 @= 表达式
```

其中@代表二元运算符。执行复合运算时，先求右边表达式的值，再与变量进行@运算，最后执行赋值运算，把结果赋给变量。上式相当于如下表达式：

```
变量 = 变量 @ (表达式)
```

注意：运算符@＝虽表达两种运算功能，但本身是一个运算符，中间不能加空格。

复合赋值运算实质上是两种运算的简化描述。如 sum＋＝i 比 sum＝sum＋i 更简明。

第 2 章介绍过自增自减运算，如 i＋＋和 i－－，有了加赋值和减赋值，还可表示为 i＋＝1 和 i－＝1，不过，对于加 1 和减 1 运算，用自增自减运算符表示更为简练。

复合赋值运算符优先级与普通赋值运算符＝相同，结合性也是右结合的。

运算符＋可用于字符串连接，复合赋值运算符＋＝也可用于追加字符串。如：

```
String str = "We";
str += " are";
str += " students.";
```

执行上述语句后，str 值为"We are students."。

5.4 for 语 句

在循环语句中，for 语句最简洁，使用率最高，当然对初学者也相对难些。

for 语句的语法格式如下：

```
for ( 变量初始化; 条件表达式; 循环变量更新 )
  循环体
```

圆括号内用两个英文分号分隔为 3 部分，其执行顺序可用 while 语句描述如下：

```
变量初始化;
while ( 条件表达式 )
{
  循环体
  循环变量更新;
}
```

在 for 语句中，变量初始化部分只在开始时执行一次，然后判断条件表达式，若为 true，则执行循环体，然后执行循环变量更新，再回过头来判断条件表达式是否成立，以决定是否再次执行循环体。若条件表达式为 false，则结束整个语句。

因此，for 语句与 while 语句一样，若条件表达式首次为 false，则循环体一次都不执行。

【例 5-2】 编程，定义使用 for 语句计算 1～n 的累加方法，然后计算 1～10 的累加。


ch5.4 例 5-2

代码如下：

```
public class Ex2 {
    public static int sum(int n) {                          //计算 1～n 的累加方法
        if(n<1) return 0;                                   //若 n 不是正整数,则返回 0
        int sum = 0;                                        //变量初值 0,因任意数加 0 不变
        for (int i = 1; i <= n; i++) {
            sum += i;                                       //加赋值运算相当于 sum = sum + i
            System.out.printf(" %d 的累加: %d\n", i, sum);  //可删除本语句
        }
        return sum;
    }

    public static void main(String[] args) {                //主方法
        System.out.printf("结果: %d", sum(10));             //调用方法计算 1～10 的累加
    }
}
```

运行结果与例 5-1 完全一样，如图 5-2 所示。

设 n 为正整数，1～n 才能累加，因此累加方法 sum 开头有一条 if 语句。变量 sum 初值也可设为 1，这时 for 语句的循环控制变量 i 初值要改为 2，少执行一次循环体。

【例 5-3】 定义计算 n 阶乘方法，然后调用该方法计算 10 的阶乘。


ch5.4 例 5-3

分析：n 阶乘等于 1×2×…×n，可在循环体中使用“乘赋值”进行运算。

代码如下：

```
public class Ex3 {
    public static long fact(int n) {                    //返回 long 型的 n 阶乘方法
        if(n<1) return 1;                               //若 n 不是正整数,则返回 1
        long f = 1;                                     //阶乘变量初值 1
        for(int i = 2; i<=n; i++) {                     //循环控制变量 i 始于 2
            f *= i;                                     //乘赋值运算相当于 f = f * i
            System.out.printf("%d 的阶乘:%d\n", i, f);   //可删除本语句
        }
        return f;
    }

    public static void main(String[] args) {           //主方法
        System.out.printf("结果:%d", fact(10));         //调用方法计算 10 阶乘
    }
}
```

程序运行结果如图 5-3 所示。阶乘方法内部 for 语句的循环控制变量 i 始于 2 而不是 1,循环体从 2 阶乘开始计算,可以少执行一次循环体。

图 5-3　10 的阶乘

如果不在阶乘方法内部输出中间结果,则删掉或注释掉 for 循环体调用 printf 方法的语句,这时程序运行只输出“结果: 3628800”。

注意:阶乘结果递增很快,采用 int 型整数只能存放 12 以内的阶乘,即使是 long 型也只能存放 20 以内的阶乘,超过了就会溢出,造成所存放的数据与实际不符。

for 语句补充说明如下:

(1) 循环变量若增 1 或减 1,则一般使用++或--运算符,否则使用+=或-=运算符。

(2) for 后面圆括号内三个部分都可省略,但分号不能省,即允许“for(; ;)”。中间部分的条件表达式为空表示 true,相当于 while(true)。

(3) 圆括号内可用逗号分隔第一和第三部分,即允许多个变量初始化及更新,但中间条件部分不能用逗号,若需要多个条件则可使用逻辑运算符 && 等构成组合条件,如:

```
for (int i = 1, j = 10; i<=j && i<=2; i++, j--) { … }
```

(4) 初始化部分声明的变量,是 for 语句的局部变量,只限于本语句内使用。

5.5　递归调用方法

设 n 为正整数,数学上使用 n! 表示 n 的阶乘,计算阶乘的数学公式如下:

$$n! = n \times (n-1)! \quad (若\ n>1)$$
$$n! = 1 \quad (若\ n=1)$$

若 n 大于 1，则 n 阶乘等于 n 乘以 n－1 的阶乘，于是可用递归调用方法计算阶乘。

所谓递归调用，就是在方法定义的内部直接或间接调用本方法。


ch5.5 例 5-4

【例 5-4】 定义求 n 阶乘递归方法，并调用该方法计算 10 的阶乘。

```
public class Ex4 {
    public static long f(int n) {                          //定义计算 n 阶乘递归方法 f
        if(n>1) {                                          //如果 n>1
            System.out.printf("f(%d) = %d×f(%d)\n", n, n, n-1);
            return n*f(n-1);                               //调用本方法 f 计算 n-1 阶乘
        } else {                                           //否则 n≤1
            System.out.printf("f(%d) = 1\n", n);
            return 1;                                      //返回阶乘值 1
        }
    }

    public static void main(String[] args) {              //主方法
        System.out.printf("结果：%d", f(10));               //调用 f 方法计算 10 阶乘
    }
}
```

运行程序，结果如图 5-4 所示。递归法包含递推和回归两个阶段，由于在递归方法 f 中编写了输出语句，于是运行结果清晰展示了递推过程。如果删掉 f 方法内部两个输出语句，则只显示“结果：3628800”。

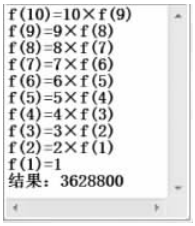

图 5-4　递归计算阶乘

由此可见，不用循环语句也能计算阶乘，因为递归方法隐含了循环功能。

注意：递归调用不能无限次地调用，为此递归调用的实参（如 n－1）比方法定义的形参（如 n）规模要小，并且小到一定程度时要有一个确定值（如 n≤1 时返回 1），只有这样程序运行才能正常终止。

【例 5-5】 计算 Fibonacci（斐波纳契）数列项。设 n 为正整数，有数学函数如下：

$$fib(n)=fib(n-1)+fib(n-2) \quad (若\ n\geqslant 3)$$

$$fib(n)=1 \quad (若\ n<3)$$

即前两项均为 1，从第 3 项开始，每项为前两项之和。


ch5.5 例 5-5

编程，定义计算斐波纳契第 n 项的递归方法，并调用方法计算若干项的值。

```
public class Ex5 {
    public static long fib(int n) {                        //计算斐波纳契第 n 项递归方法
        if(n>=3) { return fib(n-2)+fib(n-1);}              //若 n≥3，则调用本方法两次
        else if(n>=1) { return 1; }                        //否则，若 n 为 1 或 2，则返回 1
        else { return 0; }                                 //否则，若 n≤0 则返回 0
```

```
    }

    public static void main(String[] args) {                    //主方法
        System.out.printf("fib(%d) = %d\n", 6, fib(6));         //调用方法计算第 6 项
        System.out.printf("fib(%d) = %d\n", 7, fib(7));         //调用方法计算第 7 项
        System.out.printf("fib(%d) = %d\n", 8, fib(8));         //调用方法计算第 8 项
        System.out.printf("fib(%d) = %d\n", 9, fib(9));         //调用方法计算第 9 项
    }
}
```

程序运行结果如图 5-5 所示。

```
fib(6)=8
fib(7)=13
fib(8)=21
fib(9)=34
```

图 5-5　计算斐波纳契数列项

注意：使用递归编写方法，虽然代码简洁，但运行时资源消耗大，不可滥用。

试一试：能否不用递归方法计算斐波纳契数列项？

5.6　do-while 语句

for 和 while 循环语句都是首先判断条件，成立才执行循环体。do-while 语句刚好相反，它首先执行循环体，然后才判断条件。

do-while 语句简称 do 语句，语法格式如下：

```
do
    循环体
while ( 条件表达式 );
```

do 语句首先执行循环体，再判断条件表达式，若成立，则继续执行循环体，否则结束循环。因此，do 语句的循环体至少执行一次。do 语句流程图如图 4-5(b)所示。

ch5.6

注意：do 语句后面必须添加英文分号，否则有语法错误。

【例 5-6】 定义用 do 语句计算 1～n 的累加方法，再调用方法计算 1～10 的累加。

```
public static int sum(int n) {                          //使用 do 语句累加 1～n 方法
    if(n<1) return 0;                                   //若 n 不是正整数,则返回 0
    int sum = 0;
    int i = 1;
    do {
        sum += i;
        System.out.printf("%d 的累加: %d\n", i, sum);
        i++;
    } while (i<=n);                                     //也可把 i++置于 while(++i<=n)
    return sum;
}
```

上面只给出计算 1～n 累加方法，主方法与例 5-2 相同，略。

程序运行结果也与例 5-2 相同，如图 5-2 所示。

5.7 break 和 continue 语句

中断语句 break 能跳出 switch 语句，该语句还可跳出 while、for 和 do 循环语句。


ch5.7

继续语句 continue 只能置于循环语句内，用于结束本次循环，继续下一轮循环。当然下一轮循环是否执行还要看条件表达式是否成立。

【例 5-7】 在循环体内使用 break 和 continue 语句计算 20 以内的阶乘。

```
public class Ex7 {
    public static void main(String[] args) {                    //主方法
        long f = 1;                                              //阶乘变量
        int i = 1;                                               //递增变量
        do {
            f *= i;                                              //乘赋值
            System.out.printf("%d的阶乘：%d\n", i, f);
            if (++i <= 20) { continue; }                         //20 以内继续循环
            else { break; }                                      //否则终止循环
        } while( true );
    }
}
```

该例没有调用方法，而是直接在主方法内计算阶乘。程序运行结果如图 5-6 所示。

```
1的阶乘：1
2的阶乘：2
3的阶乘：6
4的阶乘：24
5的阶乘：120
6的阶乘：720
7的阶乘：5040
8的阶乘：40320
9的阶乘：362880
10的阶乘：3628800
11的阶乘：39916800
12的阶乘：479001600
13的阶乘：6227020800
14的阶乘：87178291200
15的阶乘：1307674368000
16的阶乘：20922789888000
17的阶乘：355687428096000
18的阶乘：6402373705728000
19的阶乘：121645100408832000
20的阶乘：2432902008176640000
```

图 5-6　20 以内的阶乘

5.8 多 重 循 环

前面例子循环语句的循环体比较简单，没有嵌入其他循环语句，属于单重循环。

如果要输出二维表格或平面图案，则需使用二重循环，即在循环语句的循环体内嵌入另

一循环语句。

ch5.8 例 5-8

内部循环语句的循环体,还可再嵌入循环语句,于是得到三重循环(对应立体图)。

二重以上的循环就是多重循环。

【例 5-8】 编程,使用二重循环,输出 6 行 3 列的表格。

```
public class Ex8 {
    public static void table(int row, int col) {                    //输出 row 行 col 列表格方法
        for (int i = 1; i <= row; i++) {                             //外层循环,i 控制行
            for (int j = 1; j <= col; j++) {                         //内层循环,j 控制列
                System.out.printf("%d 行 %d 列   ", i, j);           //输出 i 行 j 列单元格
            }
            System.out.println();                                    //行末换行
        }
    }

    public static void main(String[] args) {                        //主方法
        table(6, 3);                                                 //调用表格方法输出 6 行 3 列
    }
}
```

程序运行结果如图 5-7 所示,只输出单元格,不输出表格线。其中内层循环的 printf 语句共执行了 6×3 即 18 次。

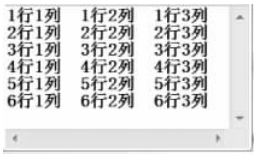

图 5-7 输出 6 行 3 列表格

图 5-8 输出倒三角形图案

ch5.8 例 5-9

【例 5-9】 编程,输出图 5-8 所示的 6 行倒三角形图案。

分析:图案由星号组成,但要布局为倒三角形,因此每行左边要插入相应的空格,即图案由空格和星号两种字符组成。而字符输出的顺序是先行后列,从上到下、从左到右,因此可用外循环从上到下控制各行,用两个内循环从左到右分别输出每行的空格和星号。

```
public class Ex9 {
    public static void triangle(int row) {                          //输出 row 行倒三角形方法
        for (int i = 0; i < row; i++) {                             //外循环,i 控制行
            for (int j = 0; j < i; j++) {                           //内循环 1,j 控制每行空格(列)数
                System.out.print(' ');                              //每次输出 1 个空格
            }
            for (int j = 0; j < 2 * (row - i) - 1; j++) {           //内循环 2,j 控制每行 * 数
                System.out.print('*');                              //每次输出 1 个星号
            }
```

```
            System.out.println();                    //行末换行
        }
    }

    public static void main(String[] args) {        //主方法
        triangle(6);                                 //调用方法输出 6 行倒三角形图案
    }
}
```

程序运行结果达到预期目标，如图 5-8 所示。

注意：3 种循环语句之间都可以相互嵌套，并且可以嵌套多层。

5.9 本章小结

有 3 种循环语句，语句之间可相互嵌套，其中 for 语句最精练，使用最多。

递归方法隐含循环结构，代码简明，但递归算法资源消耗大，不可滥用。

5.10 习 题 5

1. 设 int x=8;，则语句 while (x--＞4) { System.out.print('＊'); } 输出(　　)。

   A. ＊　　B. ＊＊　　C. ＊＊＊　　D. ＊＊＊＊

2. 设 int x, y;，则语句 x+=y; y=x-y; x-=y;功能是(　　)。

   A. 升序排列 x,y　　B. 降序排列 x,y　　C. 交换 x,y 值　　D. 不明确

3. 设 int i=5;，则 do { System.out.printf("i=%d",i); } while(i++＜6); 输出(　　)。

   A. i=5　　B. i=5i=6　　C. i=6　　D. i=5i=6i=7

4. 编程。使用循环结构计算：1+1/2+2/3+…+99/100。

5. 使用递归调用方法计算 1～100 的累加。

6. 试定义非递归方法计算斐波纳契(Fibonacci)数列项。

7. 计算 1～20 中除 5 和 15 以外所有奇数的平方，但若平方值超过 300，则终止。

8. 实行标准化考试，做对 1 道题得 2 分，做错 1 道题扣 1 分。某学生考完 50 道题得 82 分。请通过编程解答，该生做对多少道题？

提示：使用循环语句，穷举所有可能情况，并输出满足条件的答案。

9. 家长督促小孩做功课：做对 1 道题给 5 分，做错 1 道题扣 3 分。最终，小孩做完 20 道题后得 60 分。请编程解答，小孩做对几道题？

10. 有一条长阶梯，如果每步 2 阶，则最后剩 1 阶，每步 3 阶则剩 2 阶，每步 5 阶则剩 4 阶，每步 6 阶则剩 5 阶，只有每步 7 阶的最后才刚好走完。这条阶梯最少有多少阶？

11. 一球从 100 米高度自由落下，每次落地后反跳回原高度的一半，再落下。编程，计算它在第 10 次落地时共经过的距离，以及第 10 次反弹的高度。

12. 定义方法，使用二重循环输出由星号构成的直角三角形。

13. 使用二重循环输出 9 行下三角形状的乘法表。

14. 中国古典数学问题“百钱买百鸡”：鸡翁一值 5 文，鸡母一值 3 文，鸡雏三值 1 文。百文买百鸡，各买几何？请编程列出 100 个铜钱买 100 只鸡的各种买法，共有几种？

15. 计算输出 1～10 阶乘之和，即 1！＋2！＋…＋10！。

16. 编程。输出所有由数字 1、2、3、4 组成的三位数，要求互不相同且无重复位(如 123、124、132、134 等)，并统计这些三位数的总数。

17. 某人有 5 张 3 分和 4 张 5 分邮票，编程计算这些邮票中 1 张或若干张可得多少种不同邮资。要求输出“1 张邮票邮资分别是：5 分，3 分，(换行)2 张邮票邮资……”。

5.11 实训 5：累加、生成金字塔

ch5.11 实训 1

1. 编程，定义计算 1～n 累加的方法，运行时输入正整数 n，调用方法计算累加结果。运行界面如图 5-1(a)所示。

提示：累加方法参考例 5-2。部分代码参考如下：

```
public static int sum(int n) { … }                    //1～n 累加方法

public static void main(String[] args) {              //主方法
    …
    System.out.printf("结果：%d", sum(…));             //调用累加方法并输出计算结果
}
```

ch5.11 实训 2

2. 编程，定义生成 n 层金字塔的方法，运行时输入层数 n，调用方法生成 n 层金字塔图案。运行界面如图 5-1(b)所示。

提示：生成金字塔方法可参考例 5-9。