第 3 章 运算符、表达式和语句

3.1 运算符与表达式

1. 算术运算符与算术表达式

（1）加与减运算符

实际上还有正与负运算符，虽然它们跟加与减运算符长得一模一样，但不是一回事。

a = -b

（2）乘、除与求余运算符

要特别注意两种除的表达差异（与 Python 有差异）。如果“/”两边的常量或变量都是整型的，那么结果也是整型的，否则结果就是实型的。

（3）无乘方运算符（Python 有）。

a^b 的值可以通过 Math.pow(a, b)求得。

（4）算术表达式

2. 自增与自减运算符（Python 无）

当它不与其他运算符一起合作时，无论它放在变量前还是放在变量后，效果都是一样的。但当它与其他运算符一起合作时，若它放在变量前则它先做，若它放在变量后则它后做。

a = 2 * ++b

a = 2 * b++

3. 关系运算符与关系表达式

注意不等于运算符只有“!=”，没有“<>”。

一个关系表达式中只能有一个关系运算符。

关系表达式的值不是 true 就是 false。

4. 逻辑运算符与逻辑表达式

数学表示	逻辑表达式	
a < -2 或 a > 2	a < -2 || a > 2	(a < -2) || (a > 2)
-2 ≤ a ≤ 2	a >= -2 && a <= 2	(a >= -2) && (a <= 2)
不是 a > b	!(a > b)	

“||”和“&&”称为短路逻辑运算符（手法实在是高）。

逻辑表达式的值不是 true 就是 false。

5. 赋值运算符与赋值表达式

（1）赋值运算符

（2）类型转换

应尽量让“=”左边被赋值变量的类型与“=”右边表达式求值后所得结果的类型一致。

（3）复合的赋值运算符

a = a + b → a += b

a = a - b → a -= b

a = a * b → a *= b

a = a / b → a /= b

a = a % b → a %= b

能使用复合的赋值运算符时应尽量使用。

（4）赋值表达式

赋值表达式有值，而赋值语句无值。赋值表达式实际上是为了能表示“连续赋值”而提出来的，之前只有赋值语句，在那个年代，一个赋值语句只能给一个变量赋值。

a = b = c = 1

设 a，b，c 的值分别为 1，2，3。a += b += c

6. 位运算符

执行速度快。

7. instanceof 运算符

用于判断某对象是否是某类或其子孙类的。

3.2 语句概述

1. 语句最后有一个分号。

2. 方法调用语句

无值方法的调用。

3. 表达式语句

表达式有值，而（表达式）语句无值。

4. 复合语句

语法上相当于一个语句，右花括号的右边不加分号。

5. 空语句

就一个分号，用作被转向点或循环体。

6. 控制语句

用于规定处理次序。

7. package（包）和 import（导入）语句

前者解决了在不同的包中可以出现同名的类的问题，后者解决了在一个类中可以使用不在同一个包中的其他类的问题。

3.3 if 语句

1. if 形式

例 1.

```
if (a > b)
    k = x;
```

2. if-else 形式

例 2.

```
if (a > b)
    k = x;
else
    k = y;
```

3. if-else if-else 形式

例 3.

```
if (a > b)
    k = x;
else if (c > d)
    k = y;
else
    k = z;
```

4. if 语句可以嵌套，此处 else 与前面离得最近的还没有匹配过的 if 匹配。

例 4.

```
if (a > b)
    if (c > d)
        k = y;
    else
        k = z;
```

例 5.

```
if (a > b) {
    if (c > d)
        k = y;
}
else
    k = z;
```

3.4 switch 语句

1. 形式

注意千万不要漏了 break 语句。

例 6. 编写程序，要求用户输入年份和月份，输出该年该月的天数。

```
switch (month) {
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
```

```
        case 10:
        case 12: days = 31;
                 break;
        case 4:
        case 6:
        case 9:
        case 11: days = 30;
                 break;
        case 2: if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0))
                    days = 29;
                else
                    days = 28;
    }
```

2. switch 语句可以嵌套。

3.5 循环语句

1. for 语句

（1）用于循环次数明朗的场合。

（2）圆括号内的初始化、条件和修正三个部分的内容应恰如其分，该放什么放什么。

（3）初始化和修正两个部分都可以有多个表达式，如果有多个表达式，之间用逗号分隔。

```
for (a = 1, b = 100; a <= b; a++, b -= 2)
```

2. while 语句

（1）用于循环次数不明朗的场合。

（2）上来先判断条件，使用时较为放心。

3. do-while 语句

（1）用于循环次数不明朗且至少循环一次的场合。

（2）上来先执行循环体，使用时要特别小心。

（3）循环体即使是一个单语句，也要用花括号括起来。将 while 直接放在右花括号的右边，这样看上去可以与 while 语句明显区分开。

4. 循环语句可以嵌套。

5. 任何一层循环的预备工作一定要放在该层循环之前，不能放在该层循环的循环体中。

p.45 例子 6，最好用 for 语句，用 while 语句和 do-while 语句都不太恰当。程序中，“item = item * (1.0 / i);”可以改为“item *= 1.0 / i;”、“item = item / i;”或“item /= i;”。

3.6 break 语句和 continue 语句

1. break 语句结束它直接所处的那层循环的循环体的全部执行，而 continue 语句只结束它直接所处的那层循环的循环体的当前一次执行。

2. break 语句只能往外跳一层。若要往外跳多层，使用 goto 语句解决（这是使用高级语言编程时，goto 语句唯一倡导使用的场合）。

pp.45-46 例子 7，程序中，两处“j / 2”最好改为“Math.sqrt(j)”。

3.7 for 语句与数组

新款 for 语句，用于循环变量依次取数组各个元素值的场合。

3.8 例题

例 1. 编写程序，求出并输出满足条件 $1^2+2^2+3^2+\cdots+n^2>10000$ 的最小正整数 n。

```
int sum = 0, i = 0;
while (sum <= 10000) {
    i++;
    sum += i * i;
}
System.out.println(i);
```

或

```
int sum = 0, i = 0;
do {
    i++;
    sum += i * i;
} while (sum <= 10000);
System.out.println(i);
```

例 2. 在闭区间[0，1]中，任何有理数都能在数列 0，1，1/2，1/3，2/3，1/4，2/4，3/4，1/5，2/5，3/5，4/5，1/6，…中找到。编写程序，按顺序输出该数列的前 1000 项（其中的分式按“分子/分母”的形式输出）。

```
int p = 1, q = 2, i;
System.out.print("0 1 ");
for (i = 3; i <= 1000; i++) {
    System.out.print(p + "/" + q + " ");
    if (p < q - 1)
        p++;
    else {
        p = 1;
        q++;
```

```
        }
    }
    System.out.println();
```

例 3. 编写程序，求出并输出分子、分母最小的分数 p/q，使|p/q-3.141593|<10^{-5}。

```
    int p = 1, q = 1;
    while (Math.abs((double)p / q - 3.141593) >= 1e-5) {
        if ((double)p / q < 3.141593)
            p++;
        else
            q++;
    }
    System.out.println(p + "/" + q);
```