参考《C 专家编程》

2.3.2 “有些运算符的优先级是错误的”

当C语言最初文献的作者告诉你“有些运算符的优先级是错误的”的时候，就像 Kernighan 和 Ritchie 在 *The C Programming Language* 第 3 页中所说的那样，你肯定会觉得确实存在问题。尽管如此，ANSI C 在修改运算符优先级方面并没有采取什么动作，这也毫不奇怪，因为如果对运算符的优先级作了修改，那么大量现有的代码就会出现问题。

但是，到底是哪些 C 运算符存在错误的优先级呢？答案是“当按照常规方式使用时，可能引起误会的任何运算符”。有些常常会给不注意的人带来麻烦的运算符见表 2-2。

表 2-2　　C 语言运算符优先级存在的问题

优先级问题	表 达 式	人们可能误以为的结果	实 际 结 果
.的优先级高于*。 ->操作符用于消除这个问题	*p.f	p 所指对象的字段 f (*p).f	对 p 取 f 偏移，作为指针，然后进行解除引用操作。 *(p.f)
[]高于*	int *ap[]	ap 是个指向 int 数组的指针 int(*ap)[]	ap 是个元素为 int 指针的数组 int *(ap[])
函数()高于*	int *fp()	fp 是个函数指针，所指函数返回 int。int(*fp)()	fp 是个函数，返回 int* int *(fp())

续表

优先级问题	表 达 式	人们可能误以为的结果	实 际 结 果
==和!=高于位操作符	(val & mask != 0)	(val & mask) != 0	val & (mask != 0)
==和!=高于赋值符	c = getchar() != ECF	(c = getchar()) != EOF	c = (getchar() != EOF)
算术运算高于移位运算符	msb << 4 + lsb	(msb << 4) + lsb	msb << (4 + lsb)
逗号运算符在所有运算符中优先级最低	i = 1 , 2	i = (1 , 2)	(i = 1) , 2

这些运算符中的大部分，如果坐下来好好想一下，就会变得明了。尽管有些涉及逗号的情况有时会让程序员歇斯底里。例如，当下面代码执行时：

```
i = 1, 2;
```

i 的最终结果将是什么？对，我们知道逗号运算符的值就是最右边操作数的值。但在这里，赋值符的优先级更高，所以实际情况应该是：

```
(i = 1), 2;  /* i 的值为 1 */
```

i 赋值为 1，接着执行常量 2 的运算，计算结果丢弃。最终，i 的结果是 1 而不是 2。

在多年前 Usenet 的一个公告中，Dennis Ritchie 解释了这些不正常的情况是如何由于历史的偶然而产生的。

建议：

Pascal 在布尔操作和算术操作进行混合计算时，要求在表达式里加上显式的括号，从而避免了这方面的种种问题。有些专家建议在 C 语言中记牢两个优先级就够了：乘法和除法先于加法和减法，在涉及其他的操作符时一律加上括号。我认为这是条很好的建议。