参考《C 专家编程》 23 页

软件信条

一个微妙的 Bug

虽然规则作了修改，但微妙的 Bug 依然存在。在下面这个例子里，变量 d 比程序所需的下标值小 1，这段代码的目的就是处理这种情况。但 if 表达式的值却不是真。为什么？是不是有 Bug:

```
int array[] = { 23, 34, 12, 17, 204, 99, 16 };
#idefine TOTAL_ELEMENTS (sizeof(array)/sizeof(array[0]))

main( )
{
    int d = -1, x;
    /* ... */

    if(d <= TOTAL_ELEMENTS - 2)
        x = array[d+1];
    /* ... */
}
```

TOTAL_ELEMENTS 所定义的值是 unsigned int 类型（因为 sizeof()的返回类型是无符号数）。if 语句在 signed int 和 unsigned int 之间测试相等性，所以 d 被升级为 unsigned int 类型，-1 转换成 unsigned int 的结果将是一个非常巨大的正整数，致使表达式的值为假。这个 bug 在 ANSI C 中存在，而如果 K&R C 的某种编译器的 sizeof()的返回值是无符号数，那么这个 bug 也存在。要修正这个问题，只要对 TOTAL_ELEMENTS 进行强制类型转换即可:

```
if(d <= (int)TOTAL_ELEMENTS - 2)
```

小 启 发

对无符号类型的建议

尽量不要在你的代码中使用无符号类型，以免增加不必要的复杂性。尤其是，不要仅仅因为无符号数不存在负值（如年龄、国债）而用它来表示数量。

尽量使用像 int 那样的有符号类型，这样在涉及升级混合类型的复杂细节时，不必担心边界情况（如－1 被翻译为非常大的正数）。

只有在使用位段和二进制掩码时，才可以用无符号数。应该在表达式中使用强制类型转换，使操作数均为有符号数或者无符号数，这样就不必由编译器来选择结果的类型。