

这种新形式的注释是为了解决旧形式注释存在的潜在问题。假设有下面的代码：

```
/*
    希望能运行。
*/
x = 100;
y = 200;
/* 其他内容已省略。 */
```

接下来，假设你决定删除第 4 行，但不小心删掉了第 3 行（*/）。代码如下所示：

```
/*
    希望能运行。
y = 200;
/*其他内容已省略。 */
```

现在，编译器把第 1 行的/*和第 4 行的*/配对，导致 4 行代码全都成了注释（包括应作为代码的那一行）。而//形式的注释只对单行有效，不会导致这种“消失代码”的问题。

一些编译器可能不支持这一特性。还有一些编译器需要更改设置，才能支持 C99 或 C11 的特性。

考虑到只用一种注释风格过于死板乏味，本书在示例中采用两种风格的注释。

4. 花括号、函数体和块

```
{
    ...
}
```

程序清单 2.1 中，花括号把 main() 函数括起来。一般而言，所有的 C 函数都使用花括号标记函数体的开始和结束。这是规定，不能省略。只有花括号（{}）能起这种作用，圆括号（()）和方括号（[]）都不行。

花括号还可用于把函数中的多条语句合并为一个单元或块。如果读者熟悉 Pascal、ADA、Modula-2 或者 Algol，就会明白花括号在 C 语言中的作用类似于这些语言中的 begin 和 end。

5. 声明

```
int num;
```

程序清单 2.1 中，这行代码叫作声明（*declaration*）。声明是 C 语言最重要的特性之一。在该例中，声明完成了两件事。其一，在函数中有一个名为 num 的变量（*variable*）。其二，int 表明 num 是一个整数（即，没有小数点或小数部分的数）。int 是一种数据类型。编译器使用这些信息为 num 变量在内存中分配存储空间。分号在 C 语言中是大部分语句和声明的一部分，不像在 Pascal 中只是语句间的分隔符。

int 是 C 语言的一个关键字（*keyword*），表示一种基本的 C 语言数据类型。关键字是语言定义的单词，不能做其他用途。例如，不能用 int 作为函数名和变量名。但是，这些关键字在该语言以外不起作用，所以把一只猫或一个可爱的小孩叫 int 是可以的（尽管某些地方的当地习俗或法律可能不允许）。

示例中的 num 是一个标识符（*identifier*），也就是一个变量、函数或其他实体的名称。因此，声明把特定标识符与计算机内存中的特定位置联系起来，同时也确定了存储在某位置的信息类型或数据类型。

在 C 语言中，所有变量都必须先声明才能使用。这意味着必须列出程序中用到的所有变量名及其类型。

以前的 C 语言，还要求把变量声明在块的顶部，其他语句不能在任何声明的前面。也就是说，main() 函数体如下所示：

```
int main() //旧规则
{
    int doors;
    int dogs;
    doors = 5;
    dogs = 3;
    // 其他语句
}
```

C99 和 C11 遵循 C++ 的惯例，可以把声明放在块中的任何位置。尽管如此，首次使用变量之前一定要先声明它。因此，如果编译器支持这一新特性，可以这样编写上面的代码：

```
int main() // 目前的 C 规则
```