Fetching contributors...

Cannot retrieve contributors at this time.

##**11.4** 非正式命名规则

大多数项目采用的都是类似于本节所讲的相对非正式的命名规则。

###与语言无关的命名规则的指导原则

下面给出用于创建一种与语言无关的命名规则的指导原则。

- 区分变量名和子程序名字 本书所采用的命名规则要求变量名和对象名以小写字母开始，子程序名字以大写字母开始：variableName 对 RoutineName()。
- 区分类和对象 类名字与对象名字 —— 或者类型与该类型的变量 —— 之间的关系会比较棘手。有很多标准的方案可用，如下例所示：

方案**1**：通过大写字母开头区分类型和变量

```
Widget widget;
LongerWidget longerWidget;
```

方案**2**：通过全部大写区分类型和变量

```
WIDGET widget;
LONGERWIDGET longerWidget;
```

方案**3**：通过给类型加 **“t_”** 前缀区分类型和变量

```
t_Widget Widget;
t_LongerWidget LongerWidget;
```

方案**4**：通过给变量加 **“a”** 前缀区分类型和变量

```
Widget aWidget;
LongerWidget aLongerWidget;
```

方案**5**：通过对变量采用更明确的名字区分类型和变量

```
Widget employeeWidget;
LongerWidget fullEmployeeLongerWidget;
```

每一种方案都有其优缺点。第一种方案是在大小写敏感语言如 C++ 和 Java 里常用的规则，但是有些程序员对仅依靠大小写区分名字感到不大舒服。的确，创建两个只有第一个字母大小写不同的名字所能提供的“心理距离”太短了，二者之间的视觉差异也太小。 在多语言混合编程的环境中，如果任一种语言是大小写不敏感的，则将无法抑制使用第一种命名方案。以 Microsoft Visual Basic 为例，Dimwidget as Widget 将会引发一处语法错误，因为 widget 和 Widget 会被当做同一种标识符看待。

第二种方案使类型和变量之间更加鲜明。然而，由于历史原因，在 C++ 和 Java 里面全部字母大写只用于表示常量，同时这种方案会与第一种方案一样面临混合语言环境的问题。

第三种方案可用于所有语言，但是很多程序员从审美的角度出发并不喜欢增加前缀。

第四种方案有时会用作第三种方案的备选项，但是它存在的问题是需要改变类的每个实例的名字，而不是仅仅修改类名。

第五种方案要求基于每个变量的实际情况做出更多的考虑。在大多情况下，有助于提高代码可读性。但是有时候，一个 widget 就是一个普通的 widget，在这种情况下，你会发现自己会想出一些并不鲜明的名字，如 genericWidget，它的可读性比较差。 简而言之，每一种可选方案都不是十全十美的。本书代码采用的是第五种方案，因为当不要求代码的阅读者熟悉一种不太直观的命名规则时，这种规则做是很容易理解的。

标识全局变量 有一种编程常见问题，那就是滥用全局变量。加入你在所有的全局变量名之前加上 g_ 前缀，那么程序员在督导变量 g_RunningTotal 之后就会明白这是个全局变量，并且予以相应对待。

标识成员变量 要根据名字识别出变量是类的数据成员。即明确表示该变量既表示局部变量，也不是全局变量。比如说，你可以用 m_ 前缀来标识类的成员变量，以表明它是成员数据。

标识类型变量 为类型建立命名规则有两个好处：首先它能够明确表明一个名字是类型名，其次能够避免类型名与变量名冲突。为了满足这些要求，增加前缀或者后缀是不错的方法。 C++ 的惯用方法是把类型名全部大写 —— 例如 COLOR 和 MENU。（这一规则适用于 typedef 和 struct,不适用于类名。）但是这样就会增加与命名预处理常量发生混淆的可能。为了避免出现这样的麻烦，你可以为类型名增加 t_ 前缀，如 t_Color 和 t_Menu。

标识具名常量 你需要对具名变量加以标识，以便明确在为一个变量赋值时你用的是另一个变量的值（该值可能变化），还是一个具名常量。在 VB 里，还有另外的可能，那就是该值可能是一个函数的返回值。 VB 不要求在函数调用时给函数名加括号，与之相反，在 C++ 里即使函数没有参数也要使用括号。 给常量命名的方法之一是给常量名增加 c_ 前缀。这会让你写出类似 c_RecesMax 或者 c_LinesPerPageMax 这样的名字来。 C++ 和 Java 里的规则是全部用大写，以及如果有可能，用下划线来分隔单词，例如 RECSMAX 或者 RECS_MAX，以及 LINESPERPAGEMAX 或者 LINES_PER_PAGE_MAX。

标识枚举类型的元素 与具名常量相同，枚举类型的元素需要加以标识 —— 以便表明该名字表示的是枚举类型，而不是一个变量、具名常量或函数。标准方法如下：全部用大写，或者为类型名增加 e_ 或 E_ 前缀，同时为该类型的成员名增加基于特定类型的前缀，如 Color_ 或者 Planet_。

在不能保证输入参数只读的语言里标识只读参数 有时输入参数会被意外修改。在 C++ 和 VB 这样的语言里，你必须明确是否希望把一个修改后的值返回给调用方子程序。在 C++ 里分别用 *、& 和 const 指明，在 VB 里分别用 ByRef 和 ByVal 指明。 在其他语言里，如果你修改了输入变量的取值，那么无论你是否愿意，它的值都会被返回。特别是当你传递对象的时候。举例来说，在 Java 里所有对象都是“按值”传递的，因此当你把一个对象传递给一个子程序的时候，该对象的内容就可以在被调用子程序中修改。 在这些语言里，如果你制定了为输入参数增加一个 const 前缀（或者 final、nonmodifiable 等）的命名规则，那么当你看到 const 前缀出现在赋值符号左边的时候，就会知道出现了错误。如果你看到 constMax.SetNewMax(..)，就会知道这里有大漏洞，因为 const 前缀表明了该变量是不应该被修改的。

格式化命名以提高可读性 有两种常用方法可以用来提高可读性，那就是用大小写和分隔符来分隔单词。例如，GYMNASTICSPOINTTOTAL 就要比 gymnasticsPointTotal 或者 gymnastics_point_total 难读的多。 C++ 、 Java 、 Visual Basic 和其他的编程语言允许混合使用大小写字符。另外， C++ 、 Java 、 Visual Basic 和其他的编程语言也允许使用下划线（_)作为分隔符。 尽量不要混用上述方法，那么会使代码难以阅读。如果你老老实实地坚持使用其中一种提高可读性的方法，你的代码质量一定会有所改善。人们曾就诸如变量名的第一个字母是不是应该大写的做法的价值展开了激烈的讨论，但是只要你和你的团队在使用上保持一致，那么大写小写就没太大区别。基于 Java 经验的影响，同时为了促进不同编程语言之间命名风格的融合，本书对首字母采用小写。

###与语言相关的命名规则的指导原则

应该遵循你所用语言的命名规则。对于大多数语言，你都可以找到描述其风格原则的参考书。下面将给出 C 、 C++ 、 Java 和 Visual Basic 的指导原则。

C 的命名规则 有很多命名规则特别适用于 C 语言。

- c 和 ch 是字符变量。
- i 和 j 是整数下标。
- n 表示某物的数量。

- p 是指针。
- s 是字符串。
- 预处理宏全部大写（ALL_CAPS）。这通常也包括 typedef。
- 变量名和子程序名全部小写（all_lowercase）。
- 下划线（_）用做分隔符：letters_in_lowercase 要比 lettersinlowercas 更具可读性。 这些都是属于一般性的、UNIX 风格或者 Linux 风格的 `C` 编程规则， `C` 编程规则在不同的环境下也会有所差异。开发 Microsoft Windows 应用的 `C` 程序员倾向于采用匈牙利命名法，并在变量名中混合使用大小写。在 Macintosh 平台下， `C` 程序员会倾向于在子程序的名字中混合使用大小写，这是因为 Macintosh 工具箱和操作系统子程序最初是为支持 Pascal 接口而设计的。

`C++` 的命名规则 以下是围绕着 `C++` 编程形成的命名规则。

- i 和 j 是整数下标。
- p 是指针。
- 常量、typedef 和预处理宏全部大写（ALL_CAPS）。
- 类和其他类型的名字混合大小写（）。
- 变量名和函数名中的第一个单词小写，后续每个单词的首字母大写 —— 例如，variableOrRoutineName。
- 不把下划线（_）用做分隔符，除非用于全部大写的名字以及特定前缀中（如用于标识全局变量的前缀）。 与 `C` 编程相比，上述规则还远没有形成标准，并且不同的环境也会形成不同的具体规则。

`Java` 的命名规则 与 `C` 和 `C++` 不同， `Java` 语言的风格约定从一开始就创建好了。

- i 和 j 是整数下标。
- 常量全部大写（ALL_CAPS）并用下划线（_）分隔。
- 类名和接口名中每个单词的首字母均大写，包括第一个单词 —— 例如，ClassOrInterfaceName。
- 变量名和方法名中的第一个单词小写，后续单词的首字母大写 —— 例如，variableOrRoutineName。
- 除用于全部大写的名字之外，不使用下划线（_）作为名字中的分隔符。
- 访问器子程序使用 get 和 set 前缀。

`Visual Basic` 的命名规则 `Visual Basic` 还没有固定的规则。下一节将就 `Visual Basic` 给出一份规则建议。

###混合语言编程的注意事项

在混合语言环境中编程时，可以对命名规则（以及各式规则、文档规则等）做出优化以提高主题的一致性和可持续性 —— 即使这意味着优化后的规则会与其中某种语言所用的规则相冲突。 在本书里，变量名均以小写开头，这符合 `Java` 的编程实践传统以及部分但并非全部的 `C++` 传统。本书把所有子程序名的首字母大写，这遵循了 `C++` 规则。在 `Java` 中所有的方法名都是以小写字母开始的，都是本书对所有语言的子程序名的首字母都大写，从而提高了整体可读性。

###命名规则示例

上述标准规则容易使我们忽略前几页里谈论过的有关命名的若干重要事项 —— 包括变量作用域（私有的，类的或者全局的）、类名、对象名、子程序名和变量名之间的差异等。 在命名规则的指导原则长度超过了几页之后，看上去就显得非常复杂。然而，它们没必要变得如此复杂，你可以按实际需求来加以应用。变量名包含了以下三类信息：

- 变量的内容（它代表什么）
- 数据的种类（具名常量、简单变量、用户自定义类型或者类）
- 变量的作用域（私用的，类的、包的或者全局的作用域） 根据上述指导原则，表 11-3、表 11-4 和表 11-5 给出了 `C` 、 `C++` 、 `Java` 和 `Visual Basic` 的命名规则。这些特殊规则并非是强制的，但是它们能帮你了解一份非正式的命名规则应包含哪些内容。

由于 `Visual Basic` 对大小写不敏感，因此需要采取一些特殊的规则来区分类型名和变量名。请见表 11-5。