##**12.5** 标准前缀

对具有通用含义的前缀标准化，为数据命名提供了一种简洁、一致并且可读性好的方法。有关标准前缀最广为人知的方案是匈牙利命名法，该方案由一组用于指导变量的子程序命名（而不是指导如何给匈牙利人取名）的详细原则组成，并且曾经一度被广泛用于 `Microsoft Windows` 编程。尽管目前匈牙利命名法已经不再得到广泛使用，但是使用简洁准确的缩写词的基本命名标准理念却仍然具有价值。 标准化的前缀由两部分组成：用户自定义类型（UDT）的缩写和语义前缀。

###自定义类型缩写

UDT 缩写可以标识被命名对象或变量的数据类型。UDT 缩写可以被用于表示像窗体、屏幕区域以及字体一类的实体。UDT 缩写通常不会表示任何由编程语言所提供的预置数据类型。

UDT 用很短的编码描述，这些编码是为特定的程序创建的，并且经过标准化以在该程序内使用。这些编码是为特定的程序创建的，并且经过标准化以在该程序内使用。这些编码有助于用户理解其代表的实体，如用 wn 代表窗体，scr 代表屏幕区域。表 11-6 列出了一份 UDT 示例，你可能会在开发文字处理程序时用到它们。

当你使用 UDT 的时候，你还要按与 UDT 同样的缩写去定义编程语言的数据类型。这样，如果你有表 11-6 所列出的那些 UDT，你就会看到下面这样的数据声明：

```
CH chCursorPosition;
SCR scrUserWorkspace;
DOC docActive;
PA firstPaActiveDocument;
PA lastPaAcLiveDocument;
WN wnMain;
```

同样，这些例子是与文字处理程序相关的。要把他们用于你自己的项目，你需要为环境中最常用的那些 UDT 创建 UDT 缩写。

###语义前缀

语义前缀比 UDT 更进一步，它描述了变量或者对象是如何使用的。语义前缀与 UDT 不同，后者会根据项目的不同而不同，而前者在某种程度上对于不同的项目均是标准的。表 11-7 列出了一组标准的语义前缀。

语义前缀可以全用小写，也可以混合使用大小写，还可以根据需要与 UDT 和其他语义前缀结合使用。例如，文档中的第一段应该命名为 pa，以表明它是个段落，还要加上 first 以强调它是第一个段落：即 firstPa。一组段落的下标可以命名为 iPa；cPa 是相应的计数值，段落的总数量；firstPaActiveDocument 和 lastPaActiveDocument 表示当前活动文档中的第一个和最后一个段落。

###标准前缀的优点

除了具备命名规则所能提供的一般意义上的优点外，标准前缀还为你带来了另外的一些好处。由于很多字都已经标准化了，因此你在一个程序或者类内需要记忆的名字更少了。

标准前缀能够更为准确地描述一些含义比较模糊的名字。min、first、last 和 max 之间的严格区别就显得格外有用了。

标准化的前缀使名字变得更加紧凑。例如，你可以用 cpa 而不是 totalParagraphs 表示段落总数。你可以用 ipa 表示一个段落数组的下标，而不是用 indexParagraphs 或者 paragraphsIndex。

最后，在你用的编译器不能检查你所用的抽象数据类型的时候，标准前缀能帮助你准确地对类型做出判断：paReformat = docReformat 很可能不对，因为 pa 和 doc 是不同的 UDT。

标准前缀的主要缺陷是程序员在使用前缀的同时忽略给变量起有意义的名字。如果ipa 已经能非常明确地表示一个段落数组的下标，你们程序员就不会主动地去想类似于 ipaActiveDocument 这样有意义的名字。为了提高可读性，应该停下来为数组下标起一个具有描述性的名字。