##**11.1** 选择好变量名的注意事项

你可不能像给狗取名字那样给变量命名 —— 仅仅因为它听起来很可爱或者听上去不错。狗和狗的名字不一样，它们是不同的东西，变量和变量名就本质而言却是同一事物。这样一来，变量的好与坏就在很大程度上取决于它的命名的好坏。在给变量命名的时候需要小心谨慎。

下面举一个使用不良变量名的例子：

Java 示例：糟糕的变量名

```
X = X + XX;
XXX = fido + SalesTax( fido );
X = X + LateFee( X1, X ) + XXX;
X = X + Interest( X1, X );
```

这段代码究竟在做什么？ X1、XX 和 XXX 代表什么？ fido 又是什么意思？ 假如说有人告诉你这段代码基于一项余额和一组新的开销来计算一位顾客的支付总额，那么你应该使用哪个变量来为该顾客的那组新的花销打印账单呢？

下面是这些代码的另一种写法，它可以使上述问题回答起来非常容易：

Java 示例：良好的变量名

```
balance = balance + lastPayment;
monthlyTotal = newPurchases + SalesTax( newPurchases );
balance = balance + LateFee( customerID, balance ) + monthlyTotal;
balance = balance + Interest( customerID, balance );
```

从上述两段代码的比较可以看出，一个好的变量是可读的，易记的和恰如其分的。你可以通过应用多条原则来实现这些目标。

###最重要的命名注意事项

为变量命名时最重要的考虑事项是，该名字要完全、准确地描述出该变量所代表的事物。获得好名字的一种实用技巧就是用文字表达变量所代表的是什么。通常，对变量的描述就是最佳的变量名。这种名字很容易阅读，因为其中并不包含晦涩的缩写，同时也没有歧义。因为它是对事物的完全描述，因此不会和其他事物混淆。另外，由于这一名字与所表达的概念相似，因此也很容易记忆。

对于一个表示美国奥林匹克代表团成员数量的变量，你可能会把它命名为numberOfPeopleOnTheUsOlympicTeam。表示运动场中座椅数量的变量可能会命名为numberOfSeatsInTheStadium。表示某国代表团在现代奥运会上获得的最高分数的变量可能会命名为maximunNumberOfPointsInModernOlympics。表示当前利率的变量最好命名为 rate 或 interestRate，而不是 r 或 x。你明白了吧。

请留意上述这些命名所共有的两个特征。首先，它们都很容易理解。事实上它们根本不需要什么解释，因为你可以轻松地读懂它们。不过第二点，有些名字太长了 —— 长得很不实用。下面我很快就会降到变量名的长度问题。 表 11-1 中给出了更多变量名称的例子，其中有好的也有差的。

变量用途	好名字，好描述	坏名字，差描述
到期的支票累计额	runningTotal,checkTotal	written,ct,checks,CHKTTL,x,x1,x2
高速列车的运行速度	velocity,trainVelocity	velt,v,tv,x,x1,x2,train
当前日期	currentDay,todaysDate	cd,current,c,x,x1,x2,date
每页行数	linesPerPage	lpp,lines,1,x,x1,x2

currentDate 和 todaysDate 都是很好的名字，因为它们都完全而且准确地描述出了"当前日期"这个概念。事实上，这两个名字都用了非常直白的词。程序员们有时候会忽略这些普通词语，而它们往往却是最明确的。cd 和 c 是很糟的命名，因为它们太短，同时不具有描述性。current 也很糟，因为它并没有告诉你是当前什么。date 看上去不错，但是经过最后推敲它也只是个坏名字，因为这里所说的日期并不是所有的日期均可，而只是特指当前日期；而 date 本身并未表达出这层含义。x、x1 和 x2 永远是坏名字 —— 传统上用 x 代表一个未知量；如果不希望你的变量所代表的是一个未知量，那么请考虑取一个更好的名字吧。

名字应该尽可能地明确。像 x、temp、i 这些名字都泛泛得可以用于多种目的，它们并没有像应该的那样提供足够信息，因此通常都是命名上的败笔。

###以问题为导向

一个好记的名字反映的通常都是问题，而不是解决方案。一个好名字通常表达的是 "什么"（what），而不是 "如何"（how）。一般而言，如果一个名字反映了计算的某些方面而不是问题本身，那么它反映的就是 "how" 而非 "what" 了。请避免选取这样的名字，而应该在名字中反映出问题本身。

一条员工数据记录可以称作 inputRec 或者 employeeDate。inputRec 是一个反映输入、记录这些计算概念的计算机术语。employeeDate 则直指问题领域，与计算的世界无关。与此类似，对一个用于表示打印机状态的位域来说，bitFlag 就要比 printerReady 更具有计算机特征。在财务软件里，calcVal 的计算痕迹也要比 sum 更明显。

###最适当的名字长度

变量名的最佳长度似乎应该介于 x 和 maximumNumberOfPointsInModernOlympics 之间。太短的名字无法传达足够的信息。诸如 x1 和 x2 这样的名字所存在的问题是是，即使你知道了 x 代表什么，你也无法获知 x1 和 x2 之间的关系。太长的名字很难写，同时也会使程序员的视觉结构变得模糊不清。

`Gorla` 和 `Benander` 发现，当变量名的平均长度在 10 到 16 个字符的时候，调试程序所需花费的气力是最小的（1990）。平均名字长度在 8 到 20 个字符的程序也几乎同样容易调试。这项原则并不意味着你应该尽量吧变量名控制在 9 到 15 或者 10 到 16 个字符长。它强调的是，如果你查看自己写的代码时发现了很多更短的名字，那么 需要认真检查，确保这些名字含义足够清晰。

你可能已经通过 Goildlocks-and -the-Three-Bears(金发姑娘和三只小熊的经典童话，寓意权衡比较)的方法理解了如何为变量名命名，正如表 11-2 所示。

长度	命名
太长	numberOfPeopleOnTheUsOlympicsTeam
太短	n,np,ntm
正好	numTeamMembers,teamMemberCount

###变量名对作用域的影响

短的变量名总是不好吗？不，不总是这样。当你把一个变量名取得很短的时候，如 i,这一长度本身就对该变量做出了一些说明 —— 也就是说，该变量代表的是一个临时的数据，它的作用域非常有限。

阅读该变量的程序员应该会明白，这一数值只会用于几行代码之内。当你把变量命名为 i 的时候，你就是在表示"这是一个普通的循环计数器或者数组下标，在这几行代码之外他没任何作用"。

`W.J.Hansen` 所做的一项研究表明，较长的名字适用于很少用到的变量或者全局变量，而较短的名字则适用于局部变量或者循环变量（Shneiderman 1980）。不过，短的变量名常常会带来一些麻烦，因此，作为一项防御式编程策略，一些细心的程序员会避免使用短的变量名。

对位于全局命名空间中的名字加以限定词 如果你在全局命名空间中定义了一些变量（具体常量、类名等），那么请考虑你是否需要采用这种方式对全局命名空间进行划分，并避免产生命名冲突。在 C++ 和 C# 里，你可以使用 namespace 关键字来划分全局命名空间。

C++ 示例：使用 **namespace** 关键字来划分全局命名空间

```
namespace UserInterfaceSubsystem{
    ...
    // lots of declarations
    ...
}
namespace DatabaseSubsystem{
    ...
    // lots of declarations
    ...
}
```

如果你同时在 UserInterfaceSubsystem 和 DatabaseSubsystem 命名空间里声明了 Employee 类，那么你可以通过写 UserInterfaceSubsystem：：Employee 或者 DatabaseSubsystem：：Employee 来确定引用哪一个 Employee。在 Java 中，你也可以通过使用包来达到同样的目的。

在这些不支持命名空间或者包的语言里，你同样也可以使用命名规则来划分全局命名空间。其中一项规则要求为全局可见的类加上带有子系统特征的前缀。用户接口部分的雇员类可能命名为 uiEmployee，数据库部分的雇员类可能命名为dbEmployee，这样做能把全局命名空间的命名冲突降到最低。

###变量名中的计算值限定词

很多程序都有表示计算结果的变量：总额、平均值、最大值等等。如果你要用类似于 Total、Sum、Average、Max、Min、Record、String、Pointer这样的限定词来修改某个名字，那么请记住吧限定词加到名字的最后。

这种方法具有很多优点。首先，变量名中最重要的那部分，即为这一变量赋予主要含义的部分应当位于最前面，这样，这一部分就可以显得最为突出，并会被首先阅读到。其次，采纳了这一规则，你将避免由于同时在程序中使用 totalRevenue 和 revenueTotal 而产生的歧义。这些名字在语义上市等价的，上述规则可以避免将它们当作不同的东西使用。还有，类似 revenueTotal（总收入）、expenseTotal（总支出）、revenueAverage（平均收入）、expenseAverage（平均支出）这组名字的变量具有非常优雅的对称性。而从totalRevenue、expenseTotal、revenueAverage、averageExpense 这组名字中则看不出什么规律来。总之，一致性可以提高可读性，简化维护工作。

把计算的量放在名字最后的这条规则也有例外，那就是 Num 限定词的位置已经是约定俗成的。Num 放在变量名的开始位置代表一个总数：numCustomers 表示的是员工的总数。Num 放在变量名的结束位置代表一个下标：customerNum 表示的是当前员工的序号。通过 numCustomers 最后代表复数的 s 也能够看出这两种应用之间的区别。然而，由于这样使用 Num 常常会带来麻烦，因此可能最好的办法是避开这些问题，用 Count 或者 Total 来代表员工总数，用 Index 来指代某个特定的员工。这样，customerCount 就代表员工的总数，customerIndex 代表某个特定的员工。

###变量名中的常用对仗词

对仗词的使用要准确。通过应用命名规则来提高对仗词使用的一致性，从而提高其可读性。比如像 begin/end 这样的一组用词非常容易理解和记忆。而那些与常用语言相去甚远的词则通常很难记忆，有时甚至会产生歧义。下面是一些常用的对仗词：

- begin/end
- first/last
- locked/unlocked
- min/max
- next/previous
- new/old
- opened/closed
- visible/invisible
- source/target
- source/destination
- up/down