# 11.1 运算符重载

C++类特性丰富、复杂，功能强大。在第9章：您通过学习定义和使用简单的类，己踏上了面向对象编程之旅。通过定义用于表示对象的数据的类型以及（通过成员函数）定义可对数据执行的操作，您知道了类是如何定义数据类型的。我们还学习了两个特殊的成员函数——构造函数和析构函数，其作用是管理类对象的创建和删除。本章将进一步探讨类的特征，重点是类设计技术，而不是通用原理。您可能发现，本章介绍的一些特性很容易，而另一些很微妙。要更好地理解这些新特性，应使用这些示例进行练习。如果函数使用常规参数而不是引用参数，将发生什么情况呢？如果忽略了析构函数，又将发生什么情况呢？不要害怕犯错误，因为在解决问题的过程中学到的知识比生搬硬套而不犯错误时要多得多（然而，不要认为所有的错误就都会让人增长见识）。这样，您将更全面地了解C++是如何工作的以及它可以为我们做哪些工作。

本章首先介绍运算符重载，它允许将标准C++运算符（如=和+）用于类对象。然后介绍友元，这种C++机制使得非成员函数可以访问私有数据。最后介绍如何命令C++对类执行自动类型转换。学习本章和第12章后，您将对类构造函数和类析构函数所起的作用有更深入的了解。另外，您还将知道开发和改进类设计时，需要执行的步骤。

学习C++的难点之一室需要记住大量的东西，但在拥有丰富的实践经验之前，根本不可能全部记住这些东西。从这种意义上说，学习C++就像学习功能复杂的字处理程序或电子制表程序一样。任何特性都不可怕，但多数人只掌握了那些经带使用的特性，如查找文本或设置为斜体等。您可能在那里曾经学过如何生成替换字符或者创建目录，除非经常使用它们，否则这些技能可能根本与日常工作无关。也许，学习本章知识的最好方法是，在我们自己开发的C++程序中使用其中的新特性。对这些新特性有了充分的认识后，就可以添加其他C++特性了。正如C++创始人Bjarne Stroustrup在一次C++专业程序员人会上所建议的：“轻松地使用这种语言。不要觉得必须使用所有的特性，不要在第一次学习时就试图使用所有的特性。”

下面介绍一种使对象操作更美观的技术。运算符重载是一种形式的C++多态。第8章介绍了C++是如何使用户能够定义多个名称相同但特征标（参数列表）不同的函数的。这被称为函数重载或函数多态，旨在让您能够用同名的函数来完成相同的基本操作，即使这种操作被用于不同的数据类型（想象一下，如果必须对不同的物体使用不同的动词，如抬起左脚（lift\_lft），拿起汤匙（lift\_sp），英语将会多么笨拙）。运算符重载将重载的概念扩展到运算符上，允许赋予C++运算符多种含义。实际上，很多C++（也包括C语言）运算符已经被重载。例如，将\*运算符用于地址，将得到存储在这个地址中的值；但将它用于两个数字时，得到的将是它们的乘积。C++根据操作数的数目和类型来决定采用哪种操作。

C++允许将运算符重载扩展到用户定义的类型，例如，允许使用+将两个对象相加。编译器将根据操作数的数目和类型决定使用哪种加法定义。重载运算符可使代码看起来更自然。例如，将两个数组相加是一种常见的运算。通常，需要使用下面这样的for循环来实现：

for (int i = 0;i < 20; i++)

 evening[i] = sam[i] + janet[i]; // 一个一个元素相加

但在C++中，可以定义一个表示数组的类，并重载+运算符。于是便可以有这样的语句：

evening = sam+janet; // 两个数组对象相加

这种简单的加法表示法隐藏了内部机理，并强调了实质，这是OOP的另一个目标。

要重载运算符，需使用被称为运算符函数的特殊函数形式。运算符函数的格式如下：

operaor*op*{argument-list}

例如，operator+()重载+运算符，operator\*()重载\*运算符。op必须是有效的C++运算符，不能虚构一个新的符号。例如，不能有operator@()这样的函数，因为CH中没有@运算符。然而，operator[]()函数将重载[]运算符，因为[]是数组索引运算符。例如，假设有一个Salesperson类，并为它定义了一个operator+()成员函数，以重载+运算符，以便能够将两个Saleperson对象的销售额相加，则如果district2、sid和sara都是Salesperson类对象，便可以编写这样的等式：

district2 = sid + sara;

编译器发现，操作数是Salesperson类对象，因此使用相应的运算符函数替换上述运算符：

district2 = sid.operator.(sara);

然后该函数将隐式地使用sid（因为它调用了方法），而显式地使用sara对象（因为它被作为参数传递），来计算总和，并返回这个值。当然最重要的是，可以使用简便的+运算符表示法，而不必使用笨拙的函数表示法。

虽然C++对运算符重载做了一些限制，但了解重载的工作方式后，这些限制就很容易理解了。因此，下面首先通过一些示例对运算符重载进行阐述，然后再讨论这些限制。