# 10.1 过程性编程和面向对象编程

面向对象编程（OOP）是一种特殊的、设计程序的概念性方法，C++通过一些特性改进了C语言，使得应用这种方法更容易。下面是最重要的OOP特性：

* 抽象；
* 封装和数据隐藏；
* 多态；
* 继承；
* 代码的可重用性。

为了实现这些特性并将它们组合在一起，C++所做的最重要的改进是提供了类。本章首先介绍类，将解释抽象、封装、数据隐藏，并演示类是如何实现这些特性的。本章还将讨论如何定义类、如何为类提供公有部分和私有部分以及如何创建使用类数据的成员函数。另外，还将介绍构造函数和析构函数，它们是特殊的成员函数，用于创建和删除当前类的对象。最后介绍this指针，对于有些类编程而言，它是至关重要的。后面的章节还将把讨论扩展到运算符重载（另一种多态）和继承，它们是代码重用的基础。

虽然本书前面偶尔探讨过OOP在编程方面的前景，但讨论的更多的还是诸如C、Pascal和BASIC等语言的标准过程性方法。下面来看一个例子，它揭示了OOP的观点与过程性编程的差别。

Genre Giants垒球队的一名新成员被要求记录球队的统计数据。很自然，会求助于计算机来完成这项工作。如果是一位过程性程序员，可能会这样考虑：

我要输入每名选手的姓名、击球次数、击中次数、命中率（命中率指的是选手正式的击球次数除以击中次数；当选手在垒上或被罚出局时，击球停止，但某些情况不计作正式击球次数，如选手走步时）以及其他重要的基本统计数据。之所以使用计算机，是为了简化工作，因此让它来计算某些数据，如命中率。另外，我还希望程序能够显示这些结果。应如何组织呢？我想我能正确地完成这项工作，并使用了函数。是的，我让main()调用一个函数来获取输入，调用另一个函数来进行计算，然后再调用第三个函数来显示结果。那么，获得下一场比赛的数据后，又该做什么呢？我不想再从头开始，可以添加一个函数来更新统计数据。可能需要和main()中提供一个菜单，选择是输入、计算、更新还是显示数据。如何表示这些数据呢？可以用一个字符串数组来存储选手的姓名，用另一个数组存储每一位选手的击球数，再用一个数组存储击中数目等等。这种方法太不灵活了，可以设计一个结构来存储每位选手的所有信息，然后用这种结构组成的数组来表示整个球队。

总之，采用过程性编程方法时，首先考虑要遵循的步骤，然后考虑如何表示这些数据（并不需要程序一直运行，用户可能希望能够将数据存储在一个文件中，然后从这个文件中读取数据）。

如果换成一位OOP程序员，又将如何呢？首先考虑数据——不仅要考虑如何表示数据，还要考虑如何使用数据：

我要跟踪的是什么？当然是选手。因此要有一个对象表示整个选手的各个方面（而不仅仅是命中率或击球次数）。是的，这将是基本数据单元——一个表示选手的姓名和统计数据的对象。我需要一些处理该对象的办法。首先需要一种将基本信息加入到该单元中的方法；其次，计算机应计算一些东西，如命中率，因此需要添加一些执行计算的办法。程序应自动完成这些计算，而无需用户干涉。另外，还需要一些更新和显示信息的方法。所以，用户与数据交互的方式有三种：初始化、更新和报告——这就是用户接口。

总之，采用OOP方法时，首先从用户的角度考虑对象——描述对象所需的数据以及描述用户与数据交互所需的操作。完成对接口的描述后，需要确定如何实现接口和数据存储。最后，使用新的设计方案创建出程序。