普通高等教育“十二五”规划教材

C++面向对象程序设计（第二版）

张 俊 主 编
张彦铎 主 审

教材资源网址：
http://www.51eds.com

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

普通高等教育“十二五”规划教材

C++面向对象程序设计

（第二版）

张　俊　主编

张彦铎　主审

中国铁道出版社

CHINA RAILWAY PUBLISHING HOUSE

内 容 简 介

本书以面向对象程序设计思想和方法为主线，以 C++语言为载体，并基于标准模板库（STL），全面系统地讲述了面向对象程序设计的重要思想、主要方法和基本特征、C++语言的基本概念、基本语法和编程方法。

本书在第一版的基础上进行了全面改进，延续了在内容叙述和程序设计等方面的风格和特色，充实了一些重要章节的内容。全书关于 C++语言的基本体系完整，关于面向对象思想方法的结构论述清晰，关于语言和方法综合应用的设计合理。用内容丰富、难易适度的例题阐述知识点，所设计的例题丰富、难易适度，强调重要概念的掌握以及程序分析和设计能力的训练；以 C++语言标准模板库为主线贯穿全书，注重反映 C++语言的新规范、新技术和新发展。与本书配套的《C++面向对象程序设计习题与实验指导》，提供了本书各章的练习题、实验指导以及 STL，供学生参考。

本书以培养 C++语言和面向对象程序设计、分析能力以及计算机综合应用能力为目的，适合作为高等院校计算机科学与技术及相关专业的教材，也可供读者自学使用和参考。

图书在版编目（CIP）数据

C++面向对象程序设计 / 张俊主编. — 2 版. — 北京：中国铁道出版社，2012.8

普通高等教育“十二五”规划教材

ISBN 978-7-113-14631-3

Ⅰ. ①C… Ⅱ. ①张… Ⅲ. ①C 语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2012)第 173019 号

书　　名：C++面向对象程序设计（第二版）

作　　者：张俊　主编

策　　划：杨　勇　　**读者热线：400-668-0820**

责任编辑：吴宏伟　彭立辉

封面设计：付　巍

封面制作：刘　颖

责任印制：李　佳

出版发行：中国铁道出版社（100054，北京市西城区右安门西街 8 号）

网　　址：http://www.51eds.com

印　　刷：北京新魏印刷厂

版　　次：2008 年 8 月第 1 版　2012 年 8 月第 2 版　2012 年 8 月第 2 次印刷

开　　本：787mm×1092mm　1/16　印张：25.5　字数：619 千

印　　数：5 001～8 000 册

书　　号：ISBN 978-7-113-14631-3

定　　价：45.00 元

第二版前言

FOREWORD >>>

本书第一版于2008年8月出版，经过几年的使用和教学实践，得到许多反馈意见，在中国铁道出版社的支持下，于2012年2月进行了改版工作。本书延续了第一版内容叙述和程序设计等方面的风格和特色，同时对各章节内容进行了较大幅度的修订，主要包括：

（1）重新编写了模板、继承与派生、虚函数与多态性等几章的内容。增加了关于模板概念、函数模板和类模板应用的讲述，以及基于类模板实现的链表相关的内容。修订了第一版中关于继承性和多态性的讲述，使得一些概念更为清晰，同时对继承性的典型应用进行了总结提炼。虽然多重继承不太常用，但是作为C++语言所拥有的功能之一，仍有其独特之处，因此进行了介绍。为了使概念及其应用论述得更加清晰和有目的性，还增加了虚函数和多态性的讲述。基于相同的目的，对其余各章都进行了不同程度的内容调整。

（2）对信息系统进行面向对象分析和设计是基础和前提性的工作，也是面向对象程序设计之前的任务。因此，本书将第一版中第2章关于面向对象概述的内容进行了充实、合并之后，作为本书的第2章。这样修改之后，教材体系结构显得更加紧凑（从第一版的11章缩为第二版的10章），内容安排更为充实合理（先有面向对象思想的概述，接着描述建立对象模型的方法，最后自然过渡到C++语言中类的定义）。

（3）完善了每章的例题，增强了例题描述的目的性。调整之后的例题各有其侧重点，在构建本书完整的知识体系结构中，各司其责，相得益彰。同时，对每个例题要表达的思想方法、观点等进行了清晰和重点突出的论述。

本书共分10章：第1章对C++语言中基于过程程序设计的内容进行了回顾和总结，并融入了大量STL函数和容器。第2章讲述类与对象的定义，包括类定义的语法、构造函数、析构函数等。第3章继续讲述类定义中常用的几个关键字的用法，理解这几个关键字的用法对于编写良好的C++程序具有重要作用。第4章单独讲解运算符重载，这是封装性的继续和深入。第5章讲述模板，这是C++语言非常有用的特性之一，也是理解和运用标准模板库（STL）的基础。第6章对STL进行概览式讲述，其中不乏重要内容的总结。第7章讲述了继承性与派生，对继承性的C++实现和典型应用进行了提炼。第8章讲述多态性以及C++虚函数在实现多态性时的应用。第9章详细讲述了文件流以及C++对文件流的实现和应用。第10章讲述异常处理，对于提高程序稳健性具有重要作用。

为更好地掌握本书的内容，同时出版了《C++面向对象程序设计习题与实验指导》作为本书的辅导用书。

经过修订，本书体系更加完整，结构更加合理，能够更好地服务于不同层次的教学，同时具有更为适宜的知识内容，包含更为清晰的概念和方法。编者衷心期望通过本教材的使用，能够在教学中注重以“实例导向”来讲解面向对象思想方法和C++语言，注意减少理论教学的比例，更加注重程序设计和软件开发实践，从而更有利于实践能力和综合应用能力的培养。

在本书改版过程中，得到了江世宏老师、王庆春老师、李晓林老师的关心和指导，在此表示衷心的感谢。同时，感谢张彦铎教授、王海晖教授、王忠教授、吕涛老师的热情支持与帮助。

特别需要感谢的是中国铁道出版社的编辑，他们严谨的作风令人非常愉快和亲切。编者对他们出色的工作表示高度赞赏和感谢！同时，对在本书出版过程中从事各项工作的人员表示诚挚的感谢。

本书凝结了编者多年来在C++语言和面向对象程序设计教学中的实践经验，并修订数次，但由于编者水平有限，其中难免存在疏漏和不足之处，恳请读者批评指正。

编　者

2012年5月

第一版前言

面向对象程序设计作为一种主流的程序设计思想和方法，因其能够更好地对现实世界中的各种数据、概念及其特征和相互联系进行真实的建模和抽象，使得程序设计与实体行为能够更加接近。此外，基于面向对象程序设计思想和方法，能够更好地组织和管理大型程序项目，并有利于继承发展程序设计领域中的各种杰出的智慧和闪亮的思想，例如各种程序库的出现和基于模式的设计。这些都极大地促进、推动了面向对象程序设计方法及其语言在各个领域的广泛应用。

C++语言是当今最流行的一种高级程序设计语言。它完全兼容 C 语言，既支持结构化的程序设计方法，也支持面向对象的程序设计方法。与其他程序设计语言相比，C++语言在运行效率、语法及语义、组件及类库、代码与资源等方面都有着显著的优越性。因此，学好 C++，很容易在一个较高平台上架设强大、易用的应用软件。

本书综合考虑了“关于进一步加强高等学校计算机基础教学意见”中 C++语言程序设计基础的大纲要求，以及 CC2001、中国计算机科学与技术学科教程和计算机学科专业规范中关于程序设计基础、算法和复杂性、程序设计语言、软件工程、数值科学计算等领域中的相关知识单元要求，并结合多年来在面向对象程序设计和 C++语言教学实践中的经验编写而成。

本套教材分为《C++面向对象程序设计》和《C++面向对象程序设计习题与实验指导》。《C++面向对象程序设计》教材以 C++语言为载体，结合 C++语言发展中的新特性、新技术，重点讲授面向对象程序设计的思想和方法。《C++面向对象程序设计习题与实验指导》是与《C++面向对象程序设计》配套的教材，包括测试习题、试验指导、相关程序库及算法参考等三部分。本套教材以面向对象和 C++程序设计的实践指导为主，重在培养学生的分析、设计、抽象和综合应用能力。

本教材的内容特点在于：体系编排完整，内容结构合理，例题适度丰富。本书对于面向对象和 C++程序设计中的重要内容，如引用与函数、数组、指针与字符串、结构、类和对象、运算符重载、模板、STL、继承与派生、虚函数与多态性、I/O 流、异常处理等都有着详细的讲解，同时强调重要概念，各章节所选择的例题贴合重点，同时具有较强的背景和完整性。

作为本教材的一个重要特色，其编排体系作了重要尝试，也是迄今为止，较鲜见于国内同类教材中。这个鲜明的特色是：面向应用，强调实践，以 C++语言标准库 STL 应用为主线贯穿全教材。在设计例题、选择习题时，以 STL 算法和容器的应用为主题，同时注重反映 C++语言的新规范、新技术和新发展。实践证明，这种安排有利于在一个较高起点和较高平台上迅速培养学生的应用能力。本教材中的示例程序全部在 Visual C++ 2005 上调试通过，程序代码符合 C++标准。

本书共分为 11 章，内容包括：

第 1 章 C++语言基础：对 C++程序设计中的基础知识、基本概念、基本运算和基本结构和数据类型进行了讨论。同时基于 STL 中的相关内容，对基本概念、基本运算和基本结构进行了新的阐述和诠释。

第 2 章 面向对象概述：介绍了面向对象程序设计的基本思想和方法以及重要概念，同时，基于统一建模语言 UML，介绍了面向对象设计和分析的基础知识。

第 3 章 类与对象的定义：介绍了 C++语言中类定义的语法，重点讨论了构造函数、析构函数、默认构造函数、转换构造函数、复制构造函数等重要概念。这是本书的重点之一。本章最后讨论了类的复合关系及其应用，并讨论了指向成员的指针及其在 STL 中的应用。

第 4 章 类的几个主题：结合 C++语言中几个关键字 this、const、new/delete、friend 和 static 的用法，讨论了 this 指针、const 成员函数、动态内存分配及其在构造、复制、赋值、析构中的应用，讨论了友元、static 成员的用法。

第 5 章 运算符重载：介绍了运算符重载常用的两种形式，重点讨论了包括算术运算符、赋值运算符、复合运算符、关系运算符、增量/减量运算符、流插入运算符/流提取运算符、下标运算符等常用运算符的重载，并结合函数调用运算符的重载讨论了函数对象的概念及应用。这是本书的重点之一。

第 6 章 模板：讨论了函数模板和类模板的概念及其应用，并适当讨论了在 STL 中常用的模板新技术。

第 7 章 标准模板库 STL：介绍了 STL 中关于容器、迭代器、算法等基本概念，重点讨论了函数对象、算法、容器及常见迭代器的应用。

第 8 章 继承与派生：介绍了继承的基本概念，讨论了不同继承方式中的访问权限控制，以及派生类对象的构造语法及其顺序等，并讨论了赋值兼容规则、虚基类等内容。

第 9 章 虚函数与多态性：介绍了面向对象程序设计中虚函数、抽象类等基本概念，讨论了纯虚函数、抽象基类等概念在多态性中的应用。

第 10 章 C++的 I/O 流：介绍了 C++中关于流的概念，重点讨论了标准 I/O 流的成员函数及运算符的用法、文件 I/O 流的操作及常见的文件操作，讨论了流格式控制的不同方式、字符串 I/O 流和流错误状态。

第 11 章 异常处理：介绍了异常的概念，结合 C++标准库中的异常类及其应用，讨论了 C++中关于异常处理的方法以及常见规则。

教学安排建议：课堂教学授课 40 学时，课内实验 24 学时，课外实验 40 学时。由于学时有限，在组织教学时重点应放在重要概念、主要思想、基本算法和常用结构等方面，可根据学生的特点适当取舍，部分内容可安排自学。

本书第 1、3、4、5、7、11 章由张俊编写，第 2、6、8、9 章由吕品编写，第 10 章由张彦铎编写。在本书编写过程中，得到了江世宏、王庆春、李晓林老师的热情指导，他们根据自己丰富的教学经验提出了大量宝贵的意见，在此表示衷心的感谢！同时感谢王海晖、何成万、赵彤洲、王邯、吕涛、姬涛等老师的热情支持！

本书在编排内容上采用了一些新的尝试，贯穿于其中的教学方法和思想也还有待改进和提高。虽然经过了多年的教学实践，但是由于计算机科学与技术的迅速发展，以及编者水平有限，本教材编写中难免存在错误和不足之处，我们诚恳期待并接受读者的批评和指正，以供今后进一步完善。

编 者

2008 年 6 月

目 录

CONTENTS >>>

第1章 C++语言基础

C++语言对 C 语言作了重要的改进和扩展，具有丰富的功能和重要的特性。C++语言完全兼容结构化程序设计，并引入了主流的面向对象程序设计思想和方法，同时它还支持基于模板的泛型程序设计。标准模板库（STL）的加入和 boost 等程序库的出现，也极大丰富了 C++语言。作为一种高度抽象和形式化的计算机语言，C++更加强调基于类型计算的思想，数据、类型、计算三者紧密相关。

本章首先概述了 C++语言的主要基础知识，包括数据类型、命名空间、运算符及常用运算、控制结构等，同时阐述了类型计算的基本思想，加入了 STL 的三大类基本运算函数对象。然后详细讲解了引用的机制及其在函数中的应用。最后讨论了数组、指针及字符串和结构类型的应用，并分别引入了 STL 的常用区间算法、容器 vector、字符串 string 和链表 list。

通过本章的学习，应加深理解数据类型及其计算这一基本思想；掌握引用类型及其在函数中的应用；熟练应用数组、指针、字符串和链表及其 STL 类型；定义并应用结构类型，理解结构化程序设计思想。

1.1 程序设计基础

1.1.1 数据类型及其运算

1. 数据类型

数据泛指可以用计算机处理、计算的各类信息。C++是一种强类型语言，为实现计算处理，需要先把复杂数据抽象为计算机可理解的模型，并且用这个模型描述出数据的主要属性。数据类型（data type）就是这样一种模型，它实现了对现实世界各类复杂数据的抽象和模拟，描述了数据内在的属性，提供了数据运算的各种能力。数据类型都会定义各自的类型标识符。例如，为了实现计数功能，并能够对数据进行汇总、比较等运算，C++语言对类似-2、-1、0、1、2 等形式的数据进行抽象，定义了 int 类型来表示这类数据，它对应于数学中整数的概念。有了 int 数据类型，就可以定义整型数据，在内存中进行存储、赋值、初始化等基本运算，并进行加减乘除等算术运算，以及大小比较等关系运算。因此，基于类型计算的基本思想是：首先针对实例数据进行抽象简化，保留该数据的主要属性和基本运算；然后通过类型系统把这些属性和运算固化为一种数据类型；最后通过应用该类型的数据来体现该类型所具有的功能。学习、定义任何一种数据类型都应该把这个思路作为切入点。

具体来说，任何数据类型 T 需要具有 3 种作用，或需要定义 3 个功能：

（1）限定数据的取值集合及其表示范围；

（2）规定该类型数据占用内存空间的大小；

（3）定义该数据类型可进行的运算集合或者合法操作。

数据类型的这3种作用与计算机系统及所用程序语言紧密相关，第一种功能受限于系统所用的字符集，第二种功能则密切相关于系统所用的软件、硬件平台，第三种功能取决于系统所定义或支持的运算。例如，C++中定义的双精度浮点数类型 double 主要用于描述数学中的实数类型，在32位平台（这也是本书所有程序的平台）上，double类型数据占用8字节长度的内存空间，因而表示数据的范围也宽至[2.2250738585072014e-308，1.7976931348623157e+308]，其合法的二元算术运算包括加（+）、减（-）、乘（*）、除（/），而没有取余（%）运算。又如，C++中定义的另外一种数据类型 bool，只能取两个值 false 和 true，bool 类型数据所占用的内存空间只有1字节，它可进行的运算有关系运算、逻辑运算，也包括因为具有整型数据的属性而可以进行的部分算术运算。

C++语言除了提供了 bool、char、int、float 和 double 等基本数据类型之外，还提供了数组、指针、字符串等数据类型，也允许通过关键字 typedef 定义各种复合数据类型，支持通过关键字 struct 和 class 定义各种新的数据类型。当定义一种新的数据类型时，需要定义数据的组织方式，还需要定义对数据的操作，或者数据之间的运算，并力求使得该自定义数据类型具有与基本数据类型相同的能力。例如，标准模板库 STL 以概念（concept）的形式对数据类型提出了各种需求，要求某数据类型可赋值（assignable）、可默认构造（default constructible）、可比较相等性（equality comparable）、可序性（ordering）。可赋值是指该数据类型能够通过赋值运算符“=”或者复制构造来产生值的副本，副本和原始值的等价性是该概念的必然结果。可默认构造是指该数据类型 T 允许以 T()方式构造变量或对象，即在构造对象时不指定任何初始值。可比较相等性是指该数据类型能够用运算符“==”比较变量（或对象）的相等性。可序性是指能够通过某种次序（一般是 operator <）给出一种排列。C++所有基本数据类型都是这4个概念的模型。

2. 基本数据类型

基本数据类型是指已经内置于系统中的数据类型，对它们的各种运算和操作已经得到系统“天然的”支持。C++中的基本数据类型分为3类：整型（integral）、浮点型（floating）和 void 类型。

整型泛指可对应于整型数据的数据类型，包括 char、bool、short、int、long 等类型，以及可用 unsigned/signed、short/long 等修饰符限制而产生的数据类型。其中，char 型数据因为在 C++中存储为对应的 ASCII 码，因而被认为是整型类型，bool 类型的两个值 false 和 true 分别对应于0和1，因而也被认为是整型。各种整型类型因其占用系统内存空间的大小不同，所表示数据的范围也不同。整型的运算包括算术运算、赋值运算、增量/减量运算、关系运算、逻辑运算、位运算等。

浮点型对应于数学中的实数概念，主要包括 float、double 类型，以及用 long 修饰而产生的 long double 数据类型。除了某些运算（例如取余%、位运算等）之外，浮点数可以进行整型数据的大多数运算和操作。

void 类型不能直接用于定义变量（或对象），常用类型标识符 void*定义指针类型，表示通用指针，即该类型指针可以容纳其他任意类型的指针类型数据。该关键字也常用于函数，用于函数参数时，表示空参数列表，即不带任何参数；用于函数的返回类型时，表示该函数不返回任何值。对 void*类型可以施加的运算主要是指针的常用运算。

3. typedef 与类型标识符

任何数据都有所属类型，任何类型都需要一个唯一的类型标识符来进行“称呼”。例如，类型标识符 char 是对字符型数据的一种表示，看到这个“名称”就想起字符数据及其相关属性。类型标识符 int 则是对整型数据的说明，有了这个标识符就可以定义该类型的变量，从而进行一些计算。所有基本数据类型都由系统定义并提供类型标识符。

但是，一些在基本数据类型的基础上复合而成的数据类型，例如数组、指针、字符串、引用、函数等则没有提供统一的标识符，那是因为它们可以任意定义而无法由系统提供显式的名称。以

int 类型数组为例，int a[1]、int b[2]、int c[3]是不同类型的数据，变量 a、b、c 的类型无法由系统提供。同样，对于枚举、联合、结构、类等机制来说，在定义的时候都只能由用户自定义一个类型标识符。

对于复合数据类型来说，关键字 typedef 提供了定义类型标识符的能力。对语句 int a[1];来说，它只是定义了一个变量 a，但是 a 所属的类型则没有显式地提供。若写成如下两个语句：

```
typedef int A[1];          //定义类型 A，它是“只有一个 int 元素的数组类型”
A a;                       //定义变量 a，它是 A 类型的变量
```

则非常符合我们的习惯：先有类型，再有变量。同样，对于指针变量 p 的定义：char ** p;则没能显式地给出变量 p 所属的类型，但若写成如下形式，则类型和变量的概念非常清晰。

```
typedef char** P;          //定义类型 P，它是“指向二级 char 指针的指针类型”
P p;                       //定义变量 p，它是 P 类型的变量
```

应用同样的方法来思考函数原型 void f(int);，标识符 f 只是一个函数类型的变量，至于它的类型，则可以用下列语句说明：

```
typedef void F (int);      //定义类型 F，它是“带有一个 int 参数、无返回值的函数类型”
F f;                       //定义变量 f，它是 F 类型的变量
```

因此，函数也有类型，通常我们只是定义了函数（函数类型的变量），而忽略了对其类型的分析，这一点在学习函数指针时尤为重要。

阅读本节后，读者要形成一个习惯：时常分析并分辨“C/C++语言中标识符表示的是类型还是变量？”，并考虑“若只是变量定义，则它的类型是什么？”。总之，本书想重点强调：“先有类型，再有变量”，这一“类型为本”的思想方法是非常重要的。C++本身是一个强类型语言，面向对象方法更加强化了 C++的类型系统。所有要处理的数据都必须归为一类，通过在该类型中定义主要属性和各种运算，从而达到处理该类数据的目的。从这个角度来说，本书正是围绕“类型及其计算”而展开：定义一种类型，提供计算能力（包括基本运算和自定义运算）。在学习一种新的数据类型、定义一种新的数据类型的时候，必须要牢记“类型—变量—计算”这一中心思想。在实践中，我们认识问题、处理问题都必须遵从这一方法，并养成“类型为本”的思维习惯。

4．基本运算

数据类型能够进行的运算，必须通过该类型的变量（或对象）体现出来。通过定义该数据类型的变量（或对象），并对该变量（或对象）施加该数据类型所允许的操作，从而体现出该数据类型的作用及其价值。所有数据类型都应该能够进行诸如定义（define）、赋值（assign）、初始化（initialize）等基本运算，同时还应包括诸如输入/输出的自定义运算。

为构造变量（或对象），并让该变量（或对象）存活且具有明确的值，需要用到定义、赋值、初始化等运算。

（1）定义：生成数据类型的一个实例。定义某类型的变量（或对象）时，会根据该数据类型要求的内存组织方式给该变量（或对象）分配内存，一个拥有了内存的变量（或对象）即开始“存活”，可以进行后续运算。如果用符号 T 表示某一基本数据类型，则定义其变量（或对象）的语法格式如下：

```
T t;
```

定义变量（或对象）需要为其分配内存，并进行构造。

（2）赋值：为了让已经定义好的 T 类型的变量（或对象）t 具有同类型数值 val，可以采用赋值运算来完成。为变量（或对象）赋值的语法格式如下：

```
t = val;
```

赋值运算是二元运算，要求右操作数是与左操作数同类型的常量、变量或者表达式。赋值运算需要用数据类型定义赋值运算符（“=”）。

（3）初始化：如果在定义变量（或对象）的同时给定初始值，则称对该变量（或对象）初始化。初始化的方式可以用两种形式进行，它们之间主要的区别是所用的运算符：赋值运算符“=”和函数调用运算符“()”，以“=”完成的初始化称为“复制初始化”，以“()”完成的初始化称为“直接初始化”。

【例 1.1】变量的定义、赋值和初始化。

```
#01     int main()
#02     {
#03         int x, y;                                    //定义两个变量
#04         x = 1; y = x;                                //变量赋值
#05
#06         int a = 2, b = x;                            //复制初始化
#07         int c(3), d(a);                              //直接初始化
#08     }
```

变量 a 和 b 以复制初始化的方式分别初始化为 int 常量 2 和变量 x，变量 c 和 d 则以直接初始化的方式分别初始化为 int 常量 3 和变量 a。

对于上述变量（或对象）的初始化方式，需要强调如下几点：

第一，比较复制初始化和直接初始化，以圆括号运算符实现的直接初始化方式更能说明：初始化是对变量（或对象）的“构造”行为，这种“构造”是通过函数调用的方式实现的，因为圆括号是函数调用运算符，是函数调用的标志。当构造变量（或对象），调用函数所需要的参数不只一个时（例如 T　t(2,3)），这种方式更适合对象的初始化过程。

第二，复制初始化容易给人一种错觉：这是赋值行为。例如，对变量 a 和 b 的初始化，虽然形式上是通过“=”完成，但一定不要认为它们是“赋值”行为，这两者是差异很大的行为。

第三，赋值和复制初始化的相同点在于：两者的结果都会使得赋值运算符的左操作数具有与右操作数相同的值。两者的区别在于：赋值时，变量（或对象）已经存在，而复制初始化时变量（或对象）正在生成。在本书第 2 章讲到构造函数时，会更加清晰地揭示两者的差别：赋值和初始化是通过完全不同的函数实现的。但是在实现过程中，这两个函数的相同点会使得它们具有相似的功能及程序代码，两者的不同点会使得它们在处理内存资源时具有不同的操作。

关于变量（或对象），还有一个重要的概念——声明，需要注意如下几点：

（1）声明是为了使用，所有变量（或对象，甚至标识符）必须先声明，然后才能使用。

（2）声明只需表明变量（或对象）的类型和名字，其目的是为了宣告该变量（或对象）的存在，而不是如定义一样，是为了使得该变量（或对象）开始“存活”。

（3）声明不一定是定义，两者的区别在于：声明可以多次，但定义只能一次；定义时会分配存储空间，而声明时不会。如果声明同时也定义，才可以对变量（或对象）初始化。

（4）声明可以采用两种方式进行：定义性声明；以 extern 实现的声明。所谓“定义性声明”，大多数情况下，定义完一个变量（或对象），就意味着已经声明了该变量（或对象）。所谓“以 extern 实现的声明”，当需要引用定义在文件作用域中的变量（或对象）时，则在引用之前，要以 extern 声明该变量（或对象）。典型的应用如：当变量 a 定义在文件 file1 中时，如果另外一个文件 file2 中需要引用变量 a，则在引用点之前要以 extern 语句声明变量 a 的类型和名字。

借助于本书第 1 个例程，在这里想强调：C++中 main()函数最为常用的标准形式一般如下：

```
int main() {}
```

虽然函数返回类型为 int，但是函数体中并没有用 return 返回值（VC++ 6.0 编译器警告：main()函数应该返回一个值，或者采用 void 返回类型），这是由于 C++隐式地在该函数末尾添加了语句 return 0;，因此本书中所有程序（在 VC++ 2005 中实现）都会采用 main()函数的这种标准形式。

除了上述基本运算，C++基本数据类型的一些属性也可以简单地得出。C++基本类型所占系统内存空间的大小也可以通过 sizeof 运算符求得，它们所能表示数据的范围可通过多种方式获得。