普通高等教育“十一五”规划教材

C++程序设计教程
（第二版）

张丽静　潘卫华
王　红　张锋奇　余晓晔　编著

中国电力出版社
http://jc.cepp.com.cn

内 容 提 要

本书为普通高等教育“十一五”规划教材。

本书由基础篇、提高篇、实用篇三部分组成，基础篇主要内容为C++语言基础知识及面向过程的程序设计；提高篇主要内容为数组、指针、结构、联合等复合数据类型及其应用；实用篇主要内容为面向对象程序设计的概念以及基于MFC的Windows应用程序设计。本书强调通过实例学编程，案例驱动的思想贯穿全书，通过大量的示例引导学生逐步熟悉程序设计。精选有趣、实用的例题讲解程序设计及调试方法，激发学生的编程兴趣，引导学生进入面向对象程序设计的大门。

本书可作为普通高等学校相关专业的教材，也可供程序设计人员阅读、参考。

图书在版编目（CIP）数据

C++程序设计教程 / 张丽静等编著. —2 版. —北京：中国电力出版社，2010.2

普通高等教育“十一五”规划教材

ISBN 978-7-5083-9936-2

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

中国版本图书馆 CIP 数据核字（2009）第 239997 号

中国电力出版社出版、发行

（北京三里河路 6 号 100044 http://jc.cepp.com.cn）

北京丰源印刷厂印刷

各地新华书店经售

*

2009 年 1 月第一版

2010 年 2 月第二版 2010 年 2 月北京第二次印刷

787 毫米×1092 毫米 16 开本 19 印张 464 千字

印数 3001—6000 册 定价 **30.50** 元

敬 告 读 者

本书封面贴有防伪标签，加热后中心图案消失

本书如有印装质量问题，我社发行部负责退换

版 权 专 有 翻 印 必 究

前　言

程序设计课程是高校非计算机专业计算机基础教学体系中的核心课程，通过该课程的学习，不仅培养了学生独立思考的习惯和利用计算机解决实际问题的能力，还为后续计算机课程的学习以及以后的工作打下良好的基础。随着社会和科学的发展，大学对该课程的教学质量、教学内容、教学模式及相应的教学环境、教材建设均提出了更高的要求。

《C++程序设计教程》第一版出版后，得到了读者的支持与肯定，同时也收到了大量的读者反馈。根据读者的意见和建议以及本书使用中的经验，我们在《C++程序设计教程》第一版的基础上，对内容进行了增加和调整，以使本书的结构更加合理、内容更加充实、例题更加丰富。本教材保持了第一版不以讲解高深难懂的理论为重点，而是强调通过实例学习编程的风格，案例驱动的思想贯穿全书，通过大量的示例引导学生逐步熟悉程序设计。精选有趣、实用的例题讲解程序设计及调试方法，激发学生的编程兴趣，引导学生进入面向对象程序设计的大门，使用浅显易懂的示例讲解 Windows 环境下面向对象的可视化编程。本书不仅是一本适合课堂教学的教材，也不失为一本难得的自学参考书。

本教材由基础篇、提高篇、实用篇三部分组成，基础篇主要内容为 C++语言基础知识及面向过程的程序设计，包括第 1 章～第 6 章；提高篇主要内容为数组、指针、结构、联合等复合数据类型及其应用，包括第 8 章～第 10 章；实用篇主要内容为面向对象程序设计的概念以及基于 MFC 的 Windows 应用程序设计，包括第 11 章、第 12 章。其中第 1 章～第 3 章、第 5 章由张丽静编写；第 4 章由张丽静、张锋奇编写；第 6 章由张锋奇编写；第 7 章、第 9 章由王红编写；第 8 章由余晓晔编写；第 10 章由潘卫华编写；第 11 章、第 12 章由张丽静、潘卫华编写。全书由张丽静教授任主编、潘卫华副教授任副主编，王振旗教授任主审。本书的编写也得到了教研室其他老师的支持，在此一并表示感谢。

由于作者的知识和写作水平有限，书中难免有不妥之处，恳请读者批评指导。

作　者

2009 年 11 月

第一版前言

为贯彻落实教育部《关于进一步加强高等学校本科教学工作的若干意见》和《教育部关于以就业为导向深化高等职业教育改革的若干意见》的精神，加强教材建设，确保教材质量，中国电力教育协会组织制订了普通高等教育"十一五"教材规划。该规划强调适应不同层次、不同类型院校，满足学科发展和人才培养的需求，坚持专业基础课教材与教育急需的专业教材并重、新编与修订相结合。本书为新编教材。

程序设计课程是高校非计算机专业计算机基础教学体系中的核心课程。通过该课程的学习，学生不仅能养成独立思考的习惯，更能提高利用计算机解决实际问题的能力，还为后续计算机课程的学习以及以后的工作打下良好的基础。随着社会和科学技术的发展，大学对该课程的教学质量、教学内容、教学模式及相应的教学环境、教材建设均提出了更高的要求。

结合软件开发技术的发展，以及计算机基础教育改革的要求，我们精心组织编写了《C++程序设计教程》一书，本书的编写人员都是教学一线的教师，从事程序设计教学多年，具有丰富的教学经验。针对学生普遍认为程序设计难学的特点，本教材不以讲解高深难懂的理论为重点，而是强调通过实例学编程，案例驱动的思想贯穿全书，通过大量的示例引导学生逐步熟悉程序设计。精选有趣、实用的例题讲解程序设计及调试方法，激发学生的编程兴趣，引导学生进入面向对象程序设计的大门。使用浅显易懂的示例讲解 Windows 环境下面向对象的可视化编程，使得本书不仅是一本适合课堂教学的教材，也不失为一本难得的自学参考书。

本教材由基础篇、提高篇和实用篇三部分组成。基础篇主要内容为 C++语言基础知识及面向过程的程序设计，包括第 1～6 章；提高篇主要内容为数组、指针、结构、联合等复合数据类型及其应用，包括第 7～9 章；实用篇主要内容为面向对象程序设计的概念以及基于 MFC 的 Windows 应用程序设计，包括第 10 章、第 11 章。

本书第 1～3 章、第 5 章由张丽静编写；第 4 章由张丽静、张锋奇编写；第 6 章由张锋奇编写；第 7 章由余晓晔、王红编写；第 8 章由王红编写；第 9 章由潘卫华编写；第 10 章、第 11 章由张丽静、潘卫华编写。全书由张丽静教授任主编、潘卫华副教授任副主编，王振旗教授任主审。本书的编写得到了学校和教研室其他老师的支持，在此一并表示感谢。

由于作者的知识和写作水平有限，书中难免有不妥之处，恳请广大读者批评指导。

作　者

2008 年 10 月

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第二篇 提 高 篇

第三篇 实 用 篇

第一篇 基 础 篇

第1章 概 述

有着良好工作习惯的人，在每天早晨睁开眼睛的时候，都会首先想想这一天需要做的事情，这些事情需要怎样做才能更好地得到解决。如果这些事情都可以交给计算机去完成，那我们的生活将会发生多么巨大的变化。事实上，我们的生活正朝着这个方向飞速前进，计算机在我们工作、生活的方方面面发挥着越来越重要的作用，帮助我们完成各种各样的工作。这里有一个问题：计算机是不是能像人一样自主地工作呢？答案是否定的。目前，计算机是按照人们预先规定的操作来进行工作的。

1.1 计 算 机 的 程 序

要使计算机能够完成人们预定的工作，就必须把要完成工作的具体步骤编写成计算机能够识别和执行的一条条指令。计算机执行这个指令序列后，就能完成指定的功能，这样的指令序列就是程序。编写这个指令序列的过程，就是程序设计。

1.2 程 序 设 计 语 言

在过去的几十年里，大量的程序设计语言被发明、被取代、被修改或组合在一起，到目前为止已经出现了超过2500种的编程语言，其中50多种为主流的编程语言，按照出现和被使用的时间先后，我们可以将程序设计语言分为机器语言、汇编语言、高级语言等。

1.2.1 机器语言

机器语言是计算机可以理解的唯一语言。这种语言包含特定计算机处理器的指令，这些指令以二进制编码表示，计算机能够直接识别和执行机器语言编写的程序。机器语言程序执行速度快、效率高，但是用机器语言编写程序是一件非常令人头疼的工作，二进制的编码指令难于记忆，而且不同的计算机使用的指令编码各不相同，无法编制通用的程序。所以，大多数程序是使用其他语言进行编写并转换为机器语言的。

1.2.2 汇编语言

在汇编语言中，所有的指令不再使用二进制编码的形式，而是以英文助记符的形式出现。系统可以借助于语言翻译程序将这些助记符转换为机器语言代码。虽然这些助记符比机器语言便于记忆和使用，而且程序执行的效率比较高，但是使用汇编语言编写程序和机器语言一样也有很强的硬件针对性，功能的实现需要使用基本指令编制复杂的程序，因此编写汇编语言的程序掌握起来比较困难，汇编语言的程序也不容易维护和修改。

1.2.3 高级语言

高级语言的出现进一步减轻了编程人员的工作强度，编写程序所需的命令已经接近自然语言和日常习惯。比如，使两个数相加，在机器语言中，需要执行多个步骤才能在存储器单元、运算器之间传递信息来完成加法操作，而使用高级语言则可直接表示为 a+b，很像数学中使用的代数公式。使用高级语言编写程序与机器语言不同的是，编程人员不用过多地考虑该程序将在何种内部结构的机器上使用，换句话说，就是用高级语言编写的程序具有通用性。但是，要想让计算机执行用高级语言编写的程序，就必须遵循一定的规则将程序准确地翻译为机器语言程序，“a＋b”经过翻译，相加两个数所必需的一组指令将以机器语言的形式给出并存入存储器中。任何一种高级语言都有和其对应的翻译程序，简单地讲，翻译程序的作用就是将高级语言编写的程序翻译成机器语言程序。

高级语言可以分为以下 4 类。

（1）过程化语言。

（2）函数式语言。

（3）声明式语言。

（4）面向对象语言。

表 1.1 是对高级语言分类的归纳，从表中可以知道，C++是一种面向对象的语言，C 是一种过程化语言。C 语言可以看成是 C++语言的一个子集，使用 C++语言既可以编写面向过程的程序又可以编写面向对象的程序。

表 1.1　　一些高级语言的小结

语言的名称	语言的类型	何时产生
Fortran	过程化语言	20 世纪 50 年代中期
Basic	过程化语言	20 世纪 60 年代中期
Lisp	函数式语言	20 世纪 50 年代后期
Prolog	声明性语言	20 世纪 70 年代早期
Pascal	过程化语言	20 世纪 70 年代早期
C	过程化语言	20 世纪 70 年代中期
Java	面向对象的语言	20 世纪 90 年代中期
C++	面向对象的语言	20 世纪 80 年代中期

学习 C++，既要会利用 C++进行面向过程的结构化程序设计，也要会利用 C++进行面向对象的程序设计。所以，本书既介绍如何用 C++设计面向过程的程序，也介绍如何用 C++设计面向对象的程序。

1.3 结构化程序设计

随着程序复杂度的提高和程序规模的增加，人们发现“随心所欲”的编程方法将会导致程序容易出错、程序结构杂乱无章、程序难以理解和修改等诸多问题。随着软件技术的发展，荷兰学者 Dijkstra 提出了“结构化程序设计”的思想，它规定了一套方法，使程序具有合理的结构，以保证和验证程序的正确性。这种方法要求程序设计者不能随心所欲地编写程序，

而要按照一定的结构形式来设计和编写程序。它的一个重要目的是使程序具有良好的结构，使程序易于设计、易于理解、易于调试修改，以提高设计和维护程序工作的效率。结构化程序设计可以归纳为“程序＝算法＋数据结构”，将程序定义为处理数据的一系列过程。这种设计方法的着眼点是面向过程的，特点是数据与程序分离。

结构化程序设计的核心是算法设计，基本思想是采用自顶向下、逐步细化的设计方法和单入单出的控制结构。自顶向下、逐步细化是指将一个复杂的任务按照功能进行拆分，形成由若干模块组成的树状层次结构，逐步细化到便于理解和描述的程度，各模块尽可能相对独立。而单入单出的控制结构是指每个模块的内部均用顺序、选择和循环 3 种基本结构来描述。

1.3.1　算法的概念

为解决一个问题而采取的方法和步骤，就称为“算法”。我们可以这样来描述什么是算法：算法是一系列解决问题的清晰指令，也就是说，能够对一定规范的输入，在有限时间内获得所要求的输出。

算法常常含有重复的步骤和一些比较或逻辑判断。如果一个算法有缺陷，或不适合于某个问题，执行这个算法将不会解决这个问题。不同的算法可能用不同的时间、空间或效率来完成同样的任务。

一个算法应该具有以下 5 个重要的特征：

（1）有穷性。一个算法必须保证执行有限步之后结束。

（2）确切性。算法的每一步骤必须有确切的定义。

（3）输入。一个算法有 0 个或多个输入，以刻画运算对象的初始情况。所谓 0 个输入是指算法本身要求的输入，但不包括初始条件。

（4）输出。一个算法有 1 个或多个输出，以反映对输入数据加工后的结果。没有输出的算法是毫无意义的。

（5）可行性。算法原则上能够精确地运行，而且人们用笔和纸做有限次运算后即可完成。

1.3.2　算法的表示

算法的表示方法有很多种，常用的有：自然语言、图形化表示的传统流程图或结构化流程图、伪代码和计算机语言。

1. 自然语言

用中文或英文等自然语言描述算法。用自然语言表示通俗易懂，但容易出现歧义性，在程序设计中一般不用自然语言表示算法。

2. 流程图

用图的形式表示一个算法，直观形象，易于理解，但不易于修改。如图 1.1 所示为常用的流程图符号。

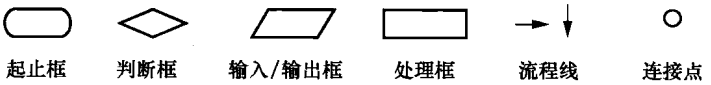

图 1.1　常用流程图符号

3. 伪代码

伪代码是用介于自然语言和计算机语言之间的文字和符号来描述算法。用伪代码写算法

时没有固定的、严格的语法规则，而且不用图形符号，因此书写方便、格式紧凑、易于修改，便于向计算机语言描述的算法（即程序）过渡。

4. 计算机语言

用某种计算机语言来描述算法，这就是计算机程序。

1.3.3 程序的基本结构

程序的流程可以用 3 种基本结构来表示，即顺序结构、选择结构和循环结构。对于一个算法，可以认为，无论其多么简单或多么复杂，都可由这 3 种基本结构组合构造而成。如图 1.2 所示为 3 种基本结构的程序执行流程。

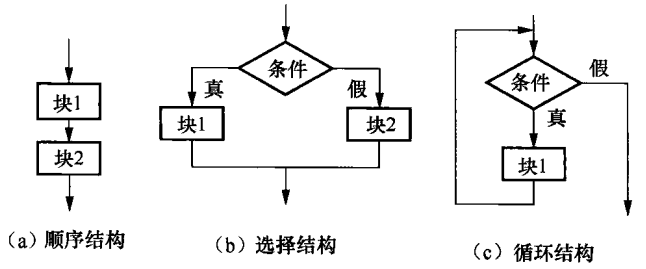

图 1.2 程序的 3 种基本结构

(a) 顺序结构；(b) 选择结构；(c) 循环结构

注意："块"在程序和算法中表现为一条指令或用"{"和"}"括起来的一组指令。一个块对外呈现一条指令的作用，其中的指令序列成为一个整体，要么一起执行，要么均不执行。后面将详细介绍这 3 种结构的程序设计。

1.4 面向对象的程序设计

面向过程程序设计的缺点根源在于数据与数据的处理相分离，而面向对象程序设计（Object Oriented Programming，OOP）方法正是克服这个缺点，同时吸收结构化程序设计思想的合理部分而发展起来的，这两种设计思想并非对立关系。面向对象程序设计思想模拟自然界认识和处理事物的方法，将数据和对数据的操作方法放在一起，形成一个相对独立的整体——对象（Object），对同类型对象抽象出共性，形成类（Class）。任何一个类中的数据都只能用本类自有的方法进行处理，并通过简单的接口与外部联系。对初学者的编程经历而言，理解构成 OOP 核心的类和对象可能会非常困难。有关面向对象的一些概念将在后面的章节中叙述。

1.5 C++ 语 言 的 发 展

C++语言于 20 世纪 80 年代由 Bjarne Stroustrup 在 Bell 实验室开发而成。该语言是作为对 C 语言的改进而开发的，C 语言也是在 Bell 实验室产生的。最初，C 语言是作为编写系统软件的程序设计语言设计出来的。例如，UNIX 操作系统就是使用 C 语言编写的。由于 UNIX 是一个非常成功的操作系统，被移植到许多计算机系统中，得到了广泛的应用。这样 C 语言也随着 UNIX 的发展而为更多的人所熟悉，现在大量的应用软件是使用 C 语言编写的。C++语言与 C 语言完全兼容，很多用 C 语言编写的应用程序都可以为 C++语言所用，这使得 C++

此为试读，需要完整PDF请访问：www.ertongbook.com

语言和面向对象技术很快得到推广。而由于C++语言面向对象的特性使其比C语言更加强大。由于C++语言与C语言的兼容，使它既支持面向对象程序设计，也支持面向过程程序设计，大家在学习当中应当注意面向对象和面向过程相结合来进行程序设计。

C++语言的开发环境有许多版本，国内较为流行的有 Microsoft 公司的 Visual C++、Borland 公司的 C++ Builder 等，本书所使用的是 Visual C++ 6.0。

1.6　C++ 的 数 据 类 型

数据是程序的必要组成部分，也是程序处理的对象。C++语言规定，程序中所使用的每个数据都属于某一种类型，在程序中用到的所有数据必须指定数据类型。数据类型是对程序所处理数据的一种“抽象”，通过类型对数据赋予一些约束，以便进行高效处理和语法检查。这些约束包括以下 3 个方面。

1. 取值范围

每种数据类型对应于不同的取值范围，即数据类型是数值的一个集合。

2. 存储空间大小

每种数据类型对应于不同规格的字节空间。

3. 运算方式

数据类型确定了该类数据的运算特性。

C++语言的数据类型极为丰富，包括基本类型、构造类型和指针等类型。图 1.3 给出了 C++数据类型的基本框架。

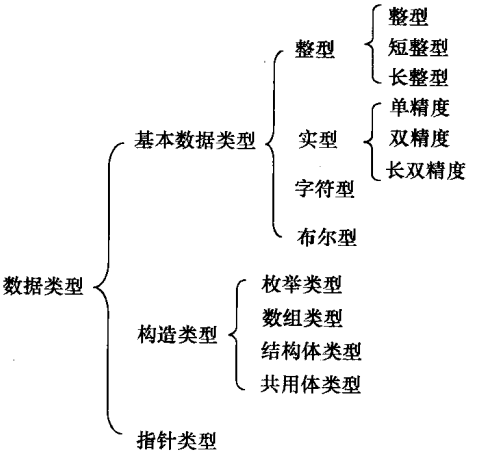

图 1.3　C++数据类型

本节只介绍基本数据类型，其他类型将在后叙章节中陆续介绍。

C++语言的基本数据类型有 4 种，整型、实型、字符型和布尔型。整型分为基本整型、短整型和长整型，整型和字符型又可分为有符号型和无符号型；其中无符号型是指存储单元中全部二进制位都用来存放数的本身，而不包括符号位；实型分为单精度、双精度和长双精度。

C++的基本数据类型见表 1.2。

表 1.2　　**C++的基本数据类型**

类　　型	类 型 标 识 符	字节	位数	取　值　范　围
整型	[signed] int	4	32	−2147483648～+2147483647
短整型	short [int]	2	16	−32768～+32767
长整型	long [int]	4	32	−2147483648～+2147483647
无符号整型	unsigned [int]	4	32	0～4294967295
无符号短整型	unsigned short [int]	2	16	0～65535
无符号长整型	unsigned long [int]	4	32	0～4294967295
单精度型	float	4	32	-3.4×10^{38}～$+3.4\times10^{38}$

续表

类 型	类型标识符	字节	位数	取值范围
双精度型	double	8	64	$-1.7\times10^{308}\sim+1.7\times10^{308}$
长双精度型	long double	16	128	$-3.4\times10^{-308}\sim+3.4\times10^{308}$
字符型	[signed] char	1	8	−128～+127
无符号字符型	unsigned char	1	8	0～255

说明：

（1）整型数据分为整型（int）、短整型（shor int）和长整型（long int）。在 int 前面加 long 和 short 分别表示长整型和短整型。C++每种类型的数据所占的字节数是一定的，一般在 16 位系统中，短整型和整型占两个字节，长整型占 4 个字节。在 32 位系统中，短整型占两个字节，整型和长整型占 4 个字节。

（2）整型数据的存储方式按二进制数形式存储。例如，十进制数 85 的二进制形式为 1010101，则在内存中的存储形式如图 1.4 所示。

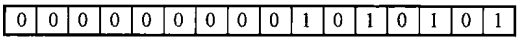

图 1.4 整数存储形式示意图

（3）在整型符号 int 和字符型符号 char 的前面，可以加修饰符 signed（表示“有符号”）或 unsigned（表示“无符号”）。如果指定为 signed，则数值以补码形式存放，存储单元中的最高位用来表示数值的符号；如果指定为 unsigned，则数值没有符号，全部二进制位都用来表示数值本身。例如短整型数占两个字节，有符号时，能存储的最大值为 $2^{15}-1$，即 32767，最小值为−32768；无符号时，能存储的最大值为 $2^{16}-1$，即 65535，最小值为 0。有些数据是没有负值的（如学号、货号、身份证号），可以使用 unsigned 类型，它存储正数的范围比用 signed 类型大一倍。

（4）实型（又称浮点型）数据分为单精度（float）、双精度（double）和长精度（long double）3 种。在 Visual C++ 6.0 中，从数据精度的角度来看，float 类型提供 6 位有效数字，double 类型则提供 15 位有效数字，两者的数值范围不同。从数据存储的角度看，float 类型分配 4 个字节，double 和 long double 则分配 8 个字节。

（5）表 1.2 中“类型标识符”一栏中，方括号 [] 包含的内容可以省略，如 short 和 short int 等效，unsigned int 和 unsigned 等效。

第 2 章　程序设计入门——程序的结构、屏幕输出和注释

学习编程的唯一方法就是自己编写程序。随着学习的深入，大家会发现，编程是听不会、看不会，而只能练会的。只有通过大量的上机训练，才能真正掌握程序设计。本章通过几个程序示例来说明 C++程序结构，讲述如何编写简单的文本输出程序和在程序中加入文字说明（注释的形式），介绍如何运行 C++的程序。

2.1　C++ 程 序 构 成

C++程序由若干程序行组成，每个程序行由 C++的一条或多条语句组成。下面通过例题说明 C++程序的构成。这里说的程序是指源程序，也叫"源代码"，"代码"来自于英文单词 Code 的直译，"代码"和"程序"在多数情况下含义相同，只是代码更倾向于表示程序片段。

【例 2.1】　编写程序，在屏幕上显示"Hello,this is my first C++ program."。

源程序：

```
#include <iostream>                                  //输入输出包含头文件
using namespace std;                                 //命名空间
void main( )                                         //main 函数，程序入口
 {                                                   //函数体开始
  cout <<"Hello,this is my first C++ program.";      //函数体。功能为在屏幕上输出字符串
 }                                                   //函数体结束
```

源代码编译、连接并运行后，在屏幕上显示：

```
Hello,this is my first C++ program.
```

下面分析程序代码。

1. 头文件

在 C++语言中，每一个程序单位叫做一个函数，一个 C++程序往往由若干个函数组成。这个程序首先用到了库函数。C++语言的设计者，将常用的一些功能模块编写成函数，并放在函数库中以供选用，用户设计程序时，就可以直接调用这些库函数。由于例 2.1 中用到输入/输出函数库中的函数，因此，程序中需要使用语句

```
#include <iostream>
```

该语句可使 C++预处理器从 iostream 函数库文件中读取代码，并将其与程序组合在一起，所有的代码（包括源代码和 iostream 库文件的代码）被编译成一个单独的二进制指令包。正是由于使用了这个特别的语句，程序才具有 C++系统的输入/输出功能。所以，该语句的作用是允许我们在程序中使用 cout 函数在屏幕上显示。以后凡在程序中用到输入/输出函数，都应当在程序中使用#include<iostream>语句。本书附录 B 中列出了一些常用库函数，如果需要更多的库函数，可以参考 C++库函数的相关书籍。

2. using namespace std;命名空间

在C++中，名称（name）可以是符号常量、变量、宏、函数、结构、枚举、类和对象等。在大规模程序设计中，程序员在使用各种各样的C++库时，标识符的命名难免发生冲突，就好像学校中遇到同名学生：A班中有张三，B班中也有张三，当A、B班一起上课时，就有名冲突问题。解决的简单办法就是两个张三分别命名为："A班张三"，"B班张三"。C++也是这样解决问题的。为了防止程序员自己又命名一个cout而造成冲突，则在每次使用cout时，就必须指明cout要参考的命名空间，即使用下面这种形式的语句（以本节的程序为例）：

```
std::cout<<"Hello,this is my first C++ program.";
```

该语句显式的指明使用标准库中cout。一般程序都要多次用到cout，只要在程序开始处使用语句using namepace std;，std在每次使用cout时就不需要用"std::"显式地说明了，语句可简写成：

```
cout<<"Hello, this is my first C++ program.";
```

关于语句using namepace std;，您只需要知道以上这些内容，不必详细深入地去了解命名空间。

还要说明的是，如果您在程序中没有使用标准的命名空间，即在程序开始没有写"using namepace std;"语句，区别是所使用的库函数不同，以本节程序为例，包含函数应写成如下形式：

```
#include<iostream.h>
```

3. main( )函数

在C++中不仅允许使用库函数，而且允许自己编写函数。main函数就是自己编写的函数，我们称它为主函数。每个C++程序必须有一个以main命名的主函数，一个C++的程序可以由若干个函数组成，但是，程序的执行都是从main函数开始的。

每一个函数至少有一对花括号。因此，C++程序的结构如下所示：

```
void main( )
{
  …
}
```

在第一行中main是函数名；void是一个关键字，表明main函数执行结束时对操作系统没有返回值（由操作系统来调用main函数）。main函数的小括号中为空，这表明操作系统没有传递给main函数任何信息。在此不详细讨论这一点。现在，读者应该记住该代码行的形式，因为以后所有的程序中都要用到它。

一对花括号中的是函数体，可以说函数从左"{"开始，到右"}"结束。函数体包含了C++的声明和语句。

4. cout<<"Hello,this is my first C++ program.";

这是函数体。由程序的运行结果知道，这行代码的作用是在屏幕显示：

```
Hello,this is my first C++ program.
```

即在屏幕上输出一串字符。C++没有专门的输出语句，用系统提供的cout函数实现输出，

使用 cout 和插入运算符“<<”将双引号之间的字符输出到屏幕。如果将双引号中的字符修改为“This is C++.”，则程序的运行结果是在屏幕显示：

```
This is C++..
```

所以，修改该行可以输出想输出的任何字符串。有关 cout 的具体使用将在后面详细讲述。

另外，程序行的右边都有以“//”开始的文字串，是对这一行程序的说明，称为 C++的注释，有关注释的内容在 2.3 节叙述。

2.2 C++ 的 语 法

编写 C++程序有一系列的规定，这些规定称为语法规则。编程人员必须严格遵守这些规则，程序编译时多数的语法错误都会被检测到。本书后面将详细介绍如何修改 C++语法错误，这里只讨论部分语法规则。随着课程的深入，我们将逐步学习更多的 C++语法规则。

1. 分号

例 2.1 中的程序行：cout<<"Hello,this is my first C++ program.";是 C++的可执行语句，它必须以分号结束。也就是说，分号是 C++语句的组成部分，它表明语句的结束。

2. 区分大小写

程序中 cout 和 COUT、Cout 或 CoUt 彼此的含义都是不同的。也就是说，C++程序区分字母的大小写。在例 2.1 中，除了双引号之间的字母外，其他字母都必须是小写。

3. 空格

C++程序中有许多标识符。标识符是 C++编译器不能拆分的最小元素。C++标识符可以是函数名（如 main）、C++的关键字（如 void）。所有的 C++的单词都应该连续书写。例如，语句行：

```
void ma in( )
```

是非法的，因为 main 中的字母 a 和 i 之间不允许有空格。也就是说，不允许在标识符中插入空格。但是，可以在标识符与标识符之间插入一个以上的空格，例如，语句：

```
void main( )
```

等价于

```
void    main( )
```

4. 间隔

在编译 C++程序时，C++编译器不受标识符之间多余空格的影响。因此，编写 C++代码的格式非常自由，在一行中可以写一条语句，也可以写多条语句；一条语句可以写在一行上，也可以在两行或两行以上。例 2.1 可以写成如下的两种形式。

形式 1：

```
#include <iostream.h>    void main( ) { cout <<"Hello,this is my first C++ program."; }
```

形式 2：

```
#include <iostream.h>  void
 main( ) {
cout <<
"Hello,this is my first C++ program."; }
```

显然，这种程序编写风格使程序变得很难理解，最好不要使用。

程序中空格的形式没有特别的要求，但编写格式一般要遵守一些特定的标准。本节的例题就说明了这个标准风格，就是采用缩进、在程序行中插入空格或在行与行之间插入空格。

总之，为了提高程序的可读性，通常编写代码时要做到一条语句占一行，程序中的花括号要对齐，代码指令中的自然中断也要加上空格或空行。要记住程序的可读性是很重要的，因为我们需要不断地修改程序。简洁而有组织的程序结构可使程序的可读性更好，从而易于维护与改进，这样的程序出错的可能性也会减小。

2.3 编 写 注 释

为什么程序中要有注释？因为具有丰富编程经验的人员，有时也不容易读懂一段程序或一个模块的功能，甚至很可能连自己编写的程序也很难读懂，或许别人要使用您编写的程序，但却很难读懂您的代码。还好，C++允许在程序代码中添加注释。注释的结构、注释在程序中的位置以及编写注释的风格是这一节中主要讲述的问题。

注释就是对程序模块的功能及其如何实现进行描述的文字说明。程序中的注释是非常重要的附加信息，它可以传递代码本身难以传递的信息。好的注释能减少错误发生的可能性，因为编程人员在修改程序时可以通过注释来读懂程序是如何工作的。

以下的程序说明注释的结构，该程序和例 2.1 功能相同，都是在屏幕上输出一行字符串。

【例 2.2】 说明注释结构的程序。

源程序：

```
//This is a single line comment.
#include <iostream.h>
void main( )
 {  /* This is a multiline
       comment */
   cout <<"Comment structure lesson.";      //End of line comment.
 }
//A comment can be written at the end of a program.
```

程序运行结果：

```
Comment structure lesson.
```

下面对这段程序中的注释进行说明。

（1）单行注释结构。如例题所示，单行的注释以“//”开始，两斜杠之间不允许留有空格，所有的注释必须在同一行，注释不执行任何操作，只是对程序的功能或该行代码的说明。

注释不必单独占一行。它可以放在 C++语句的后面，例如：

```
cout <<"Comment structure lesson.";    //在屏幕上输出一串字符
```

注意不能将注释放到 cout 语句的前面，如果这样，cout 语句将被认为是注释的一部分，从而 cout 语句不能被执行，如下所示：

```
//在屏幕上输出一串字符。  cout <<"Comment structure lesson.";
```

（2）多行注释结构。如例题所示，多行注释结构以“/*”开始，以“*/”结束，“/”和“*”之间不允许有空格，符号“/*”和“*/”必须成对出现，但可以不在同一行。注释文本可以

此为试读，需要完整PDF请访问：www.ertongbook.com