计算机及软件技术丛书

TURBO C++简明教程

顾铁成 顾其兵 潘金贵 等编著

陈世福 主审

南京大学出版社

计算机及软件技术丛书

TURBO C++简明教程

顾铁成　顾其兵　潘金贵　等编著
陈世福　　主审

南京大学出版社
1993·南京

(苏)新登字第 011 号

内 容 简 介

面向对象程序设计(即 OOP)是 90 年代流行的一种程序设计技术。在面向对象程序设计语言中,C++已开始逐渐被人们选用于教学和程序开发。在各种C++的实现中,Borland 公司开发的 Turbo C++是强有力的开发环境与C++语言和库的有机结合。它体现了该公司一贯向教育界提供高质量编译器的传统,颇受广大教师、学生和用户的欢迎。

本书主要是向熟悉C语言的读者介绍如何利用 Turbo C++来进行程序设计,采取了一种逐步深入的教学式过程,以C语言为起点,逐步使读者掌握 Turbo C++的使用。

本书可以用作C++程序设计初级教程课本,也可以用作高级程序设计技术、数据结构、软件方法学、命令式语言等课程的补充教材。本书简明而彻底地介绍了 Turbo C++中的程序设计过程。书中每一章都给出了一些经过仔细解释的程序,其中的主要程序都通过了上机测试。

计算机及软件技术丛书

TURBO C++简明教程

顾铁成　顾其兵　潘金贵　等编著

陈世福　　主审

*

南京大学出版社出版

(南京大学校内　邮编 210008)

江苏省新华书店发行　丹阳市横塘印刷厂印刷

*

开本:787×1092　1/16　印张:14.5　字数:371 千

1993 年 7 月第 1 版　　1993 年 7 月第 1 次印刷

印数:1—8000

ISBN 7-305-02244-6/TP·72

定价:9.50 元

《计算机及软件技术丛书》编委会

出版者的话

我国社会主义经济建设的蓬勃发展，极大地推动着社会信息化的进程，也促进了信息产业的发展。现在，计算机的应用已渗透到社会和生活的各个领域。作为社会信息化基础的计算机及软件技术，正为越来越多的人掌握和应用，计算机及软件技术也因此而不断更新、发展。

掌握计算机技术，是现代人特别是跨世纪的中青年人在当今激烈的社会竞争中制胜的基础，也是未来信息化社会对每个人的要求。然而，在我国，计算机基础教育尚欠普及，计算机特别是微型计算机及软件技术的应用和开发也还处在一个较低的层次。许多非专业人员希望能使用计算机，但面对纷繁的专业知识、众多的技术资料，视学习计算机的使用为畏途，专业人员面对软件技术的快速更新，目不暇接。为了让更多的人熟悉计算机技术，利用计算机服务于自己的管理、科研、教学工作，使我国的计算机及软件技术的应用和开发紧随国际潮流，普及和提高我国计算机应用和开发的水平，我们为此组织编写并陆续出版《计算机及软件技术丛书》。

本《丛书》将以应用为基础，兼顾普及与提高。组织科研、教学和应用开发第一线的专家、学者，结合国外计算机及软件技术的最新发展和趋向与国内的应用现状和方向，为初学者提供系统的入门读物，为专业人员介绍适合国情的最新实用技术，既有理论性、学术性强的专著、专论，也有普及性、实用性的教材、手册，以满足多层次读者的需要。

本《丛书》的编写将立足于现实，着眼于未来，力争反映国内外计算机及软件技术的最新动态和发展趋向，引导和帮助读者学习、吸收、掌握计算机的新理论、新技术和新成果。

我们将根据读者需要，不断充实、完善本《丛书》内容，同时诚恳欢迎读者对本《丛书》提出建议、批评，也热忱欢迎向本《丛书》赐稿。

南京大学出版社

《计算机及软件技术丛书》编委会

1993.4.

前　言

本书是关于 Turbo C++程序设计的一本教科书，意在向已经熟悉 C 语言的程序人员或学生介绍 Turbo C++程序设计。具体做法是以 C 为起点逐步深入，最后以 C++结束。

C++是 C 语言的一个有力的后继者。Turbo C++又是强有力的开发环境与 C++的有机结合。最初，C++是由贝尔实验室的 Bjarne Stroustrup 于 80 年代中期设计的，它在 C 语言的基础上增加了类(class)的概念。类是一种用以提供用户定义类型(亦称为抽象数据类型)的机制。通过类、继承性和运行时刻类型联编等机制，C++就能支持面向对象程序设计。目前，C 用得较多，但可以预料，不久的将来，C++会被广泛使用。

本书的特点是通过采用"剖析"方法来仔细分析 C++示例程序，从而简明而彻底地介绍用 C++进行程序设计的过程。利用剖析这种方法可以向读者解释、强调示例程序中新的、关键的内容。

本书可以用作 C++程序设计初级教程课本，也可以用作高级程序设计技术、数据结构、软件方法学、命令式语言等课程的补充教材。书中每一章都给出了一些经过仔细解释的程序。我们对多数程序和函数都进行了剖析。

所有主要的代码段都经过了测试。在本书中，我们从一开始就采用了软件专业人员普遍采用的一种一致而正确的编程风格。

本书的每一章是这样安排的：

- Turbo C++的考虑。Turbo C++是用于开发 C 语言和 C++语言软件的完整环境，它提供给用户的是一个相当友善的集成开发环境(Integrated Development Environment，简称 IDE)。Turbo C++基于 C++ 2.0 版，每一章的这部分内容主要介绍 Turbo C++的新颖而独特的特征，并解释与具体系统有关的或与较早的 C++语言实现不同的论题。
- 剖析。在每一章的剖析部分对那些特别能说明该章主题的程序加以分析。剖析类似一种对代码段的结构化分析，其目的在于向读者解释新的程序设计成分与术语。
- 小结。在每一章的结尾给出该章主要内容的一个简要列表。
- 练习。每一章的后面给出了一些练习，用来检查读者对语言的掌握程度。许多练习可以结合阅读课文来做，同时也对课文的内容加以了扩展。

从另外一个角度来看，本书的特点和内容包括：

- 由浅入深逐步演化的途径。C 程序人员很快就会得益于 Turbo C++程序设计。开始的几章介绍在 C 语言基础上增加的更强的类型机制、新的注解风格、按引用调用，等等。中间的一些章节介绍类的概念与用法。类是抽象数据类型和面向对象程序设计的基础。在介绍类时还是从 C 逐步过渡到 C++。后面的一些章节介绍了有关继承性和流 I/O 的高级使用。读者可以在课文中的任何一处停下来，并对所学的新内容试加应用。
- 示例教学。本书一开始就向读者介绍完整的样本程序，并推荐一种交互式的学习方法，即将例子与练习结合起来以鼓励实践。在解释编写工作代码的成分时，我们避免了过分的细节。每一章都有几个重要的示例程序，这些程序的主要成分都通过剖析方法加以解释。

· C++中的数据结构。本书着重介绍了一些计算机科学中常用的标准数据结构，如堆栈、安全数组、动态分配的多维数组、列表、树和串等，并给出了它们的实现。每章后的练习可以帮助读者更好地理解如何来实现与应用这些数据结构。书中给出的实现与设计软件的抽象数据类型途径是相一致的。

· 面向对象的程序设计。本书试图使读者逐渐适应面向对象的程序设计风格。第一章和第二章讨论C程序人员在转向C++和面向对象程序设计风格后可以得到哪些益处。这两章里还定义了有关面向对象概念的术语，并介绍了C++是如何支持这些概念的。第四章介绍类，它是设计模块化程序和实现抽象数据类型的基本机制。类变量即待操纵的对象(object)。第七章引入继承性和虚函数概念，这是面向对象程序设计模式中的两个关键元素。第八章将前面介绍过的各种技术综合起来，说明如何正确地书写包括各种成分的面向对象风格的程序。OOP的微妙之处来自一种柏拉图式的(Platonic)程序设计哲学，本书还将力图使读者能够逐渐开始欣赏这种思想与观点。

· C++ 2.0版与iostream.h。目前，C++正在发展和变化着。本书基于最近的标准：AT&T 2.0版及与其相关的库。书中的例子要用到iostream.h I/O库，它已开始取代较早的stream.h和stdio.h等库。我们力求保持与可能成为未来标准的语言的主流方面相一致，尽量避免C++中易出错或搞混的较艰涩的内容。

参加本书编写工作的主要人员有潘金贵、顾铁成和顾其兵。张汝平、高青、陈欣等也为本书的出版做了许多不可缺少的工作。潘金贵和顾铁成同志对全书进行了仔细的修改和统编。

本书承蒙南京大学计算机科学与技术系主任陈世福教授主审，在此谨表深切谢意。

由于水平、时间所限，不妥之处在所难免，欢迎广大读者批评指正。

需要本书中示例程序盘和Turbo C++系统盘的读者，可与编者联系获得有关信息。

编　者

1993年1月于南京

目　录

第一章　面向对象程序设计与 Turbo C++综述 …… 1

1.1　面向对象程序设计 …… 1
1.2　C++是一种更好的C …… 1
1.3　输出 …… 3
1.4　输入 …… 4
1.5　函数原型 …… 7
1.6　类与抽象数据类型 …… 8
1.7　重载 …… 10
1.8　构造函数与析构函数 …… 12
1.9　面向对象程序设计与继承性 …… 14
1.10　Turbo C++的考虑 …… 16
小结 …… 19
练习一 …… 20

第二章　C++：一种更好的C …… 23

2.1　注解的风格 …… 23
2.2　使用 inline 和 const 以避免使用预处理器 …… 24
2.3　声明 …… 25
2.4　void 的使用 …… 28
2.5　域分辨符:: …… 30
2.6　函数原型 …… 31
2.7　按引用调用和引用声明 …… 33
2.8　缺省参数 …… 34
2.9　函数的重载 …… 34
2.10　自由存储操作符 new 和 delete …… 35
2.11　其他 …… 38
2.12　Turbo C++的考虑 …… 39
小结 …… 41
练习二 …… 42

第三章　类 …… 49

3.1　集聚类型 struct …… 49
3.2　结构指针操作符 …… 51

3.3 一个例子:栈 …… 51
3.4 成员函数 …… 55
3.5 可见性:private 和 public …… 57
3.6 类 …… 58
3.7 静态成员 …… 60
3.8 嵌套类 …… 60
3.9 一个例子:同花牌 …… 61
3.10 Turbo C++的考虑 …… 65
小结 …… 67
练习三 …… 67

第四章 构造函数与析构函数 …… 71

4.1 带构造函数的类 …… 71
4.2 带析构函数的类 …… 74
4.3 一个例子:动态分配的串 …… 74
4.4 类 vect …… 78
4.5 类成员 …… 79
4.6 一个例子:单链表 …… 80
4.7 二维数组 …… 84
4.8 this 指针 …… 86
4.9 Turbo C++的考虑 …… 87
小结 …… 89
练习四 …… 90

第五章 操作符重载与类型转换 …… 96

5.1 传统的类型转换 …… 96
5.2 ADT 转换 …… 98
5.3 重载与函数选择 …… 99
5.4 友元函数 …… 102
5.5 重载操作符 …… 105
5.6 一元操作符重载 …… 105
5.7 二元操作符重载 …… 107
5.8 重载赋值和下标操作符 …… 109
5.9 Turbo C++的考虑 …… 113
小结 …… 115
练习五 …… 117

第六章 继承性 …… 121

6.1 派生类 …… 121

6.2 public,private 和 protected …… 122
6.3 派生安全数组类型 vect _ bnd …… 124
6.4 类型转换与可见性 …… 127
6.5 虚函数 …… 130
6.6 一个例子:类层次 …… 132
6.7 二叉树类 …… 137
6.8 Turbo C++的考虑 …… 140
小结 …… 141
练习六 …… 142

第七章 输入/输出 …… 147

7.1 输出类 ostream …… 147
7.2 格式化输出和 iomanip.h …… 148
7.3 用户定义的类型:输出 …… 150
7.4 输入类 istream …… 152
7.5 文件 …… 153
7.6 ctype.h 中的函数和宏 …… 156
7.7 流状态的使用 …… 157
7.8 Turbo C++的考虑 …… 159
小结 …… 160
练习七 …… 161

第八章 C++的高级特性 …… 164

8.1 多重继承性 …… 164
8.2 抽象基类 …… 167
8.3 缺省赋值和重载 new …… 169
8.4 指针操作符 …… 170
8.5 作用域和链接 …… 173
8.6 重复子 …… 174
8.7 一个例子:单词频率 …… 177
8.8 OOP:面向对象程序设计 …… 183
8.9 柏拉图主义:面向对象的设计 …… 183
8.10 Turbo C++的考虑 …… 184
小结 …… 185
练习八 …… 186
附录 A 操作符优先级和结合性 …… 189
附录 B Turbo C++语言导引 …… 190
附录 C Turbo C++的编辑命令 …… 219
主要参考资料 …… 221

第一章　面向对象程序设计与 Turbo C++综述

C++是 Bjarne Stroustrup 于 80 年代设计的。当时，有两个主要目标：

(1) C++要与传统的 C 兼容；

(2) 用 Simula 67 中的类构造来扩充 C。

类构造是对 C 中 struct 的扩充。早期的 C++在 B. Stroustrup 的"C++程序设计语言"[1]中加以了介绍。最近增加的内容见 B. Stroustrup 的"C++的演化：1985－1987"[2]。

本书是面向那些已经熟悉 C 语言的读者的，因而着重从两个方面来介绍作为 C 的后继语言的 C++语言。首先，作为通用的程序设计语言来说，C++因为有着一些新的特征而优于 C。其次，作为一种面向对象的程序设计语言来说，C++是成功的。本章我们将首先定义"面向对象"这个概念的含义，接着说明为什么 C++是一种更好的 C，然后概述 C++的主要语言成分和特性。

§1.1　面向对象程序设计

抽象数据类型(Abstract Data Type，简称 ADT)和面向对象程序设计(Object－Oriented Programming，简称 OOP)是一种强有力的新型程序设计途径。ADT 是对语言中已有类型的由用户定义的扩充，它由一组值以及作用于这些值上的一组操作构成。例如，在 C 语言中是没有复数类型的，但 C++语言提供了类构造来构造这个类型，并将它加入到已有的类型中去。对象是类变量，面向对象程序设计允许很容易地构造并使用 ADT。OOP 利用继承性(inheritance)机制来方便地从已有的用户定义类型派生出一种新类型，并允许程序人员通过用类构造对问题中对象的内容及行为进行编程来加以模型化。

C++语言中新颖的类构造提供了封装性(encapsulation)，从而可以方便地实现 ADT。封装性将类型的内部实现细节以及从外部可取接的、作用于该类型对象上的操作和函数封装在一起。对用到该类型的客户代码来说，实现细节可以是不可见的。例如，可以将栈实现为一个固定长度的数组，从外部可取接的对栈的操作有压入和弹出。将栈的内部实现改为一个链表并不影响从外部使用压入和弹出的方式，即栈的实现对其客户来说是隐蔽的。在本章中将介绍类构造是如何提供数据隐蔽的，在第三章和第六章中还要深入讨论。

§1.2　C++是一种更好的 C

因为 C++是基于 C 的，所以它保留了 C 语言的大部分内容和特性，包括丰富的操作符集、近乎直交的设计、简洁性和可扩充性。C++是一种可移植性极好的语言，现在已有许多 C++编译器在各种不同的机器与系统上运行。同时，C++编译器与已有的 C 程序是高度兼容的，这也是 C++最初的设计目标之一。与其他的一些面向对象的语言(如 Smalltalk)不同的是，C++是对已有的一种被广泛应用的语言扩充而成的。

与另一种典型的面向对象语言 Smalltalk 相比，用 C++进行程序设计不需要图形环境，并且 C++程序不会导致由于类型检查或废料收集而产生的运行时刻代价。

C++在一些重要的方面对 C 作了改进，特别是在支持强类型方面。现在由 ANSI C[4,5]所要求的函数原型其实是 C++的发明。一般来说，C++具有比 C 更强的类型规则，这使它成为一种更安全的语言。

C++是高级与低级语言成分的综合。正如 Stroustrup 所说的[1]，"C 语言处在机器这一级，而 C++处在问题领域这一级。"程序人员可以在接触到机器一级实现细节的同时根据与问题相适应的层次来编程。

根据不同的参数类型，可以重新定义(即重载)操作符。这种操作符重载支持新类型的实现，对这些类型可以透明地操作。像操作符一样，函数通常也以可被重载。

在 C++中，对预处理器的依赖减弱了。C 语言程序常常利用预处理器来实现常量与宏，但参数化的#define 宏又会导致不安全性。在 C++中，关键字 inline 指示编译器把一个函数当作宏来编译。这与通常的函数相比，可以在改善运行时刻性能的同时保持做类型检查。对预处器的依赖又被 const 类型修饰符进一步削弱了，这个修饰符说明了一个对象是只读的。

C++对 C 还有着许多其他方面的改进，例如，新增的用//符号来表示一行注解就深为程序人员所喜爱。另外还新增了一个很方便、很有用的 I/O 库 iostream.h，它取代了 stdio.h。操作符 new 和 delete 也提供了对自由存储区的很方便的取接。

在 C++中，抽象数据类型是通过 class(类)机制来实现的。程序人员可以通过类来控制底层实现的可见性，即公有部分是外部可取接的，而私有部分对外是隐蔽不可见的。数据隐蔽是面向对象程序设计的一个特色。类有成员函数，包括那些重载操作符的函数。成员函数使程序人员可为某一 ADT 设计出合适的功能。类还可以通过一种继承机制来定义，这种机制能增进代码共享与库的开发。继承性是面向对象程序设计的又一标志。

常常有人批评 C 是一种弱类型的、不安全的语言[6]。相反，C++是一种强类型的语言，它允许良定义的类型转换。可以这么说，C++允许程序人员构造在任意类型之间的转换函数。

在 C 语言中，对参数传递是不做类型与实参个数的检查的，这常常会导致编译器发现不了的错误。在 C++中，利用函数原型可以从参数个数和类型上对函数进行完全的检查。另外，C++也支持带可变类型与个数参数的函数。函数原型的增加对程序人员来说是个帮助，它使得编译器能做先前仅由 lint 提供的一些检查。C 中的数组模型是基于指针的、一维的，且不带越界检查。因为没有现成的多维数组，程序人员就必须花更多的时间和精力来利用一维存储映射函数对它们加以构造，这就是 C 的模式。还有，设计和使用动态数组也要花不少的力气。C++保持了 C 中的数组处理，而类又进一步提供了透明地实现通用数组的方法。多维数组、动态数组以及可作越界检查的数组都能在库中实现。

总的来说，C++的语义定义比 C 的要严格得多。例如，类型与类型转换都被更仔细地加以了实现。还有，C 对预处理程序的扩展性的需求也被削弱了：函数重载与 inline 可以替代带参数的宏；类型修饰符 const 足以表示多数命名的常量。C++中预处器仍具有的作用主要是文件包含与条件编译。

Turbo C++支持一个可运行于多数 DOS 和 OS/2 机器上的高级的集成开发环境。这个环境就是我们为什么要使用 Borland 公司的 Turbo C++的一条最好理由。相对过去非集成的批处理和工作站环境来说，它提高了编程效率。Borland 公司的 Turbo C++系统提供了一

个新颖的调试器、窗口管理器和工程管理器。

C++支持面向对象的程序设计风格，由此而带来的一个主要代价就是语言的复杂性增加了(考虑到C++的目标，这一点不足为奇)。这也是C++最主要的缺点之一。这种复杂性反映了对大量的新观念的需要，但也使得熟练地掌握C++语言变得非常困难。为了解决这个问题，本书采取了循序渐进的教学过程，逐步把C程序人员转化为熟练的C++程序人员。书中每一章都是对前一章中程序人员可能用到的观念的扩展，都要给出一些关于如何利用Turbo C++的新的建议。实际上，读者是从一个标准的C程序人员演进为一个面向对象的C++程序设计人员的。

总之，C++是一种更好的C，Turbo C++是一个更好的程序开发环境，具有高效性、经济性、完整性。使用C++可以使编程更加方便，编出的程序的安全性更好。新增的一些特性如表示单行注解的//，const与inline，用于存储管理的new和delete，以及按引用调用参数等使得在C++中的编程过程较C中的更为简化。更强的类型机制、函数原型等都增强了软件的安全性。

下面，我们要对Turbo C++程序设计语言作个综述，初步介绍如何用C++语言进行面向对象的程序设计。和本书的其他部分一样，这里也假定读者已掌握了C语言。本章给出了一系列的程序，并对各程序的主要成分作了仔细解释。各程序的复杂性逐渐增加，稍后的几小节中的例子将对面向对象程序设计的一些概念加以说明。这样的安排应能使读者对C++有个初步的感受。

我们对C++的每个特性都作了简要介绍，后面的一些章节将要对每个新的概念作进一步的解释。本章的例子给出了C++语言的关键特性的一些简单、直接的介绍，包括流I/O，操作符重载和函数重载，引用参数，类，构造函数与析构函数，以及继承性等。程序所基于的系统是Borland C++，它与AT&T C++ Release 2.0是兼容的。

面向对象程序设计是用类(class)构造来实现的。C++中的class构造是对C中struct的一个扩展。本章中稍后的一些例子要说明C++如何实现了OOP(面向对象程序设计)概念，如数据隐蔽，ADT，继承性，类型层次，等等。下一章中，还将说明C++的另外一些特性，以进一步阐述C++是一种更好的C。

§1.3 输　出

程序必须与外界通信才能有用。我们的第一个例子是这样一个程序，它将短语"Turbo C++ is an improved C."显示到屏幕上，我们称之为advert程序。完整的程序是：

```
// A first C++ Program illustrating output.
#indude<iostream.h>
main( )
{
  cout<<"Turbo C++ is an improved C. \n";
}
```

该程序在屏幕上显示：

Turbo C++ is an improved C.

对 advert 程序的剖析

```
//A first C++ program illustrating output.
```

· 双斜杠//是一种新的注解符号。该注解一直延续到行末。C 中原有的括号注解符号/* */仍然能用，用于多行注解。

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

· 头文件 iostream.h 为 C++引入了 I/O 设施。

```
cout<<"Turbo C++ is an improved C.\n";
```

· 这一句在屏幕上显示。标识符 cout 是标准输出流的名称。操作符<<将串"Turbo C++ is an improved C.\n"送至标准输出。按这种方式使用的输出操作符<<被称作是"放至"(put to)或"插入"(insertion)操作符。

可以将前一个程序重写如下：

```
//A first C++ program illustrating output.
#include<iostream.h>
main( )
{
  cout<<"Turbo C++ is an improved C."<<"\n";
}
```

虽然与它的第一个版本不一样，但产生的输出是一样的。每次在 cout 后使用<<时，显示从上次停下的位置继续。在本例中，回车换行符在第二个放置操作符之后被输出。

§1.4 输　　入

以下，我们将写一个把地球与月球之间距离的英里数表示转换成公里表示的程序 moon。按英里来表示的话，这个距离大约是 238857 英里。这是个整数。为将英里数转换为公里数，要乘上一个转换因子 1.609，这是一个实数。

我们的转换程序将利用能够存储整数值和实数值的变量。在C++中，所有变量都必须在使用前声明；但与 C 中不同的是，它们不必位于一个分程序的头上。声明可以与可执行语句混在一起，但变量的作用域仍是它们的声明所在的分程序。标识符的选择要能反映出它们在程序中的作用。这样可使得程序更可读。

```
//The distance to the moon converted to kilometers.
#include<iostream.h>
main( )
{
  const long int moon=238857;
  cout<<"The moon's distance from Earth is"<<moon;
  cout<<"miles. \n";

  long int moon_kilo;   moon_kilo=moon*1.609;
  cout<<"In kilometers this is"<<moon_kilo;
  cout<<"km. \n";
}
```

这个程序的输出为

```
The moon's distance from Earth is 238857 miles.
In kilometers this is 38420 km.
```

上面这些程序都假定 int 型占 4 个字节，但在某些机器上这种长度的整型应定义为 long。

对 moon 程序的剖析

```
const long int moon=238857;
```

· 关键字 const 是C++中新增的。它取代了预处器命令 define 的部分作用来创建命名的字面常量。用这个类型修饰符来将 moon 的不能被改变的初始值信息通知编译器。

```
cout<<"The moon's distance from Earth is"<<moon;
```

· C++的流 I/O 能够在不需要额外的格式化信息的前提下区别一系列的值。这儿 moon 的值将作为一个整数而显示出来。

```
long int moon_kilo;
moon_kilo=moon*1.609;
```

· 声明可以在可执行语句之后出现，这就使得对变量的声明与它们的使用更加接近。下面，我们来写一个将一系列值从英里数转换为公里数的程序 mi_km，并要求它是交互的，即用户键入一个以英里数表示的值，该程序在将此值转换为公里数表示后显示出来。

```
//Miles are converted to kilometers.
#include<iostream.h>
const float m_to_k=1.609;
inline int convert(int mi){return(mi*m_to_k);}
main( )
{
    int miles;
    do {
       cout<<"Input distance in miles:";
       cin>>miles;
       cout<<"\nDistance is"<<convert(miles)<<"km.\n";
    }while (miles>0);
}
```

这个程序用到了输入流变量 cin，通常它就是标准输入。输入操作符(input operator)>>又被称为“取自”(get from)或“抽取”(extraction)操作符，它将来自输入流的值赋给一个变量。本程序说明了输入和输出的用法。

对 mi_km 程序的剖析

```
const float m_to_k=1.609;
inline int convert(int mi){return(mi*m_to_k);}
```

· C++减弱了 C 对预处理器的传统的依赖。例如，对某些特别的常量，如转换因子 1.609，在 C++中不必一定用 define，而是直接赋给说明为常量的变量即可。新出现的关键字 inline 说明了如果可能的话，将函数按宏来编译。切记：inline 要少用，且应只用于短小的函数。另请注意在函数括号中是如何定义参数 mi 的。C++利用函数原型(function prototype)来定义并声明函数。这一点将在下一节解释。

```
do {
   cout<<"Input distance in miles:";
   cin>>miles;
   cout<<"\nDistance is"<<convert(miles)<<"km.\n";
}while(miles>0);
```

· 本程序反复提示用户输入一个以英里表示的距离值，在遇到 0 或负值时结束。标准输入流中的值被自动地转换为一个赋给 miles 的整型值。

§1.5 函数原型

C++中函数的语法引出了 ANSI C 中新增的函数原型语法。简单地说,就是把各参数的类型列在函数头的括号内。由于显式地列出了参数的个数与类型,C++就可以做强类型检查和赋值相容的类型转换。下面的例子说明了这些观点。

```
//A program illustrating function prototype
#include <iostream.h>
main()
{
    int add3(int,int,int);
    float average(int);
    int score_1,score_2,score_3,sum;

    cout<<"\nEnter 3 scores:";
    cin>>score_1>>score_2>>score_3;
    sum=add3(score_1,score_2,score_3);
    cout<<"\nTheir sum is "<<sum<<".\n";
    cout<<"\nTeir average is "<<average (sum)<<".\n";
    sum=add3(1.5*score_1,score_2,score_3);
    cout<<"\nThe weighted sum is "<<sum<<".\n";
    cout<<"\nTheir average is "<<average(sum)<<".\n";
}
int add3(int a, int b, int c)
{
    return(a+b+c);
}
float average(int s)
{
    return(s/3.0);
}
```

对 add3程序的剖析

```
int add3(int,int,int);
float average(int);
```

· 这些在 main 头部的声明都是函数原型,它们将每个在外部定义的函数的参数类型与个数告知编译器。参数列表中也可以包含变量名。因而