C++语言命令详解

（第二版）

覆盖最新的
ANSI/ISO
C++ 标准

- 最全！完整的库函数参考——可以在本书中查找标准库中的任何函数或类
- 使用C++面向对象的特征开发有效的、简洁的程序
- 对C++的关键字、运算符以及库名称通俗的解释

[美] Brian Overland 著

董梁 李君成 李自更 等译
董梁 审校

IDG BOOKS WORLDWIDE
M&T BOOKS
PHEI
電子工業出版社
Publishing House Of Electronics Industry
URL:http://www.phei.com.cn

C++语言命令详解

(第二版)

C++ IN PLAIN ENGLISH
(Second Edition)

[美]Brian Overland 著

董 梁 李君成 李自更 等译

董 梁 审校

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内容简介

本书使用清晰易懂的语言介绍了C++的语法规则以及使用C++进行程序设计的方法。全书由两大部分以及四个附录和一个词汇表构成，书的中间还有两个独立的章节。第一部分用九章的内容介绍了C++的基本概念以及C++程序的设计方法，这部分从C++最基本的概念讲起，覆盖了最新的C++标准中的所有概念并重点介绍了类及其有关的函数和运算符。第二部分可以用于C++编程的参考，分别详细介绍了数据类型、运算符、类型转换、关键字、预处理指令以及库函数和库类。四个附录分别介绍了C/C++的区别、ANSI C++的特征、标准异常以及前128个ASCII字符。在书中间的两个章节分别介绍了C++可以实现的功能以及C++成员的速查表。在书末尾有C++术语及概念的词汇表。

本书语言简洁清晰，完全覆盖了ANSI C++的所有内容，同时本书独特的编排方式使得本书的使用十分方便。本书既可以用作学习C++的标准教材，又可以成为高级程序员的有价值的参考书。

图书在版编目(CIP)数据

C++语言命令详解 /(美)奥弗兰(Overland, B.)著；李自更等译 .-北京：电子工业出版社，2000.1

书名原文：C++ in Plain English

ISBN 7-5053-5715-8

Ⅰ.C… Ⅱ.①奥…②李… Ⅲ.C语言-基本知识 Ⅳ.TP312

中国版本图书馆CIP数据核字(1999) 第74127号

书　　名：**C++语言命令详解(第二版)**
著　　者：[美]Brian Overland
译　　者：董　梁　李君龙　李自更　等译
审 校 者：董　梁
责任编辑：陈晓莉
特约编辑：郑文灏
印 刷 者：北京天竺颖华印刷厂
出版发行：电子工业出版社　URL：http://www.phei.com.cn
　　北京市海淀区万寿路173信箱　邮编 100036
经　　销：各地新华书店经销
开　　本：797×1092　1/16　印张：28.375　字数：681千字
版　　次：2000年3月第2次印刷
书　　号：ISBN 7-5053-5715-8/TP·2944　著作权合同登记号：图字：01-1999-2979
定　　价：48.00元

凡购买电子工业出版社的图书，如有缺页、倒页、脱页，所附磁盘或光盘有问题者，请向购买书店调换。若书店售缺，请与本社发行部联系调换。电话：68279077

译 者 序

十全十美的程序设计语言是不存在的。不同的程序设计问题需要用不同的方法来解决。为项目选择最好的语言是软件工程人员的任务。对于任何项目来说,这是首先要做出的选择。而且一旦开始编程就不可更换。对程序设计语言的选择也直接决定着项目的成败。在众多的程序设计语言当中,C语言无疑是最具有活力和生命力的编程语言之一。数十年的使用与改进以及C语言自身的特点,使得C语言已经为程序员广泛接受。

C语言是在十分低级的软件工具的基础上发展起来的(从BCPL到B,又由B到C)。C语言位于面向机器的低级语言的基础之上,但是仍位于许多高级语言之下。它既接近计算机,足以使用户充分控制应用实现中的细节,又不忽视硬件方面的细节。这就是为什么C语言同时被视为高级语言和低级语言的原因。C语言所具有的广泛的特性——从位运算到高级格式化的文件输入输出——以及平台独立的特性使得C语言在科学、工程、商业中被广泛应用。这也直接促成了UNIX系统的流行。

C语言本身的局限性(在现在看来,也是所有旧式编程语言的局限性)、操作系统的改进以及人们对世界的认识逐渐成熟,这些原因促成了C++的诞生。C++语言是C语言的的超集。C++继承了C语言的所有优点:处理软硬件接口的能力与灵活性、高效率的表达式以及平台独立性等等。C++改进了程序设计思路,将编程方式进化到面向对象进行程序设计这一新的层次。面向对象的编程方式是从人们对周围事物的认识方式中抽象出来的。它更接近于人的思维过程。所以C++语言的学习对于已经习惯标准过程化语言结构的程序员来说并不是一件很容易的事情,因为他们要学习的不仅是一门新的语言,而且要学习用新的方法来思考和解决问题。

C++语言代表着对以往语言之精华的发展和提炼。C++语言的一个关键设计目标就是与C语言兼容。正是由于这个原因,许多程序员发现过渡到C++语言比他们当初从其它语言,比如FORTRAN语言过渡到C语言更为简单。C++语言支持大规模的软件开发。由于它加强了类型检查,使得在编写C语言应用程序时曾出现过的许多副作用不复存在。从编程方式讲,C++语言是一门未来的语言。在学习了C++语言之后,再学习其它流行语言,譬如说Java语言,就会发现这些语言从C++中借用了许多东西,特别是面向对象的程序设计方法。

那么学习程序设计语言应该看什么样的书呢?特别是对那些初学者来说,应该怎样从头开始而又可以少走弯路呢?本书就是这样一本入门指导书,它依据制定这门语言规则的委员会的规定,覆盖了这门语言的所有语法并提供了用法的例子,而这些内容可以使初学者掌握语言的基础。另外在熟悉了这门语言之后,这本书还可以用做语言的速查手册。

在翻译过程中,译者发现本书不仅通俗易懂,而且它还方便用户的查找。本书以ANSI C++为基础,详细介绍了C++的语法并明确指出C++与C不同的地方。本书使用独特的表格形式,可以让用户按照所需要的功能查找可以使用的函数。又由于本书作者的背景,使得本书具有更大的价值。

全书的翻译工作主要由李自更、董梁、李洪,李君龙等人完成。在翻译过程中,译者力求忠实原文,但限于学识和水平,工作可能没有完成得很圆满,不妥之处,望读者指正。

译 者

1999年冬于北京

关于作者

Brian overland 作为软件测试员、程序员 /作家以及管理员在 Microsoft 工作了十年。他到达了对程序员 /作家来说可能是最高的技术高峰，而在 Microsoft，只有很少的人才能达到这个层次。他为 Microsoft 的计算机语言产品编写了样例代码，这些语言包括 C、Visual Basic、汇编语言以及 C++。作为 Visual Basic 最初开发的项目领导，他用 C /C++ 编写了自定义控制。他可能是唯一的，既是 Visual Basic 1.0 开发组成员又是 Visual C++ 1.0 开发组的成员的人。他发行的第一本书是“Lean and Mean Visual C++”。该书后来被翻译成波兰语和克罗地亚语。他的 Microsoft 手册被翻译成法语、德语等语言。

前　　言

* *

《C++语言命令详解》第一版的成功之处是源于使用通俗易懂的方式对 C++ 这种语言进行解释的需要。对 C++ 的前身 C 来说,这种需要也是存在的。在 C++ 中,不可能仅从名字中就知道诸如函数 scanf 或 strtok 是干什么的。当然,有许多函数是 BASIC 中经常使用的关键字,如 Input 以及 String。

事实上,人们特别需要对 C++ 进行通俗易懂的解释。C++ 高手们在公司里或在编程时嘟囔着奇怪的术语,如多态、实现、类层次等等。本书的目的在于用多种方式打破这些语言障碍。

《C++语言命令详解》的第二版覆盖了 C++ 语言的核心特征。在我编写完本书的第一版之后,越来越困绕我的是遗漏了仅在标准库中才出现的某些函数。已公布的语言核心中并没有包含这些库函数,但是它们相当重要。本书的第二版就是要填补这个漏洞;对每一个函数、类以及关键字,本书的第二版都进行了详尽的说明。

本书第二版的另一个目的是在以下的方面扩充了第一版的内容:

- 完善标准库以及 I /O 类

- 用通俗的语言解释关键字、运算符以及库名称。C 和 C++ 都迫切需要进行这样的解释。C /C++ 的函数名常常是难以理解的。

- 添加了最新的 ANSI 中规定的特征。

本书的第二版确实实现了这些目标。仅仅简单地对每一个特征进行语法总结可能会节约许多时间或节省许多空间。但是本书除了给出简单的总结之外,它对每个关键字以及函数都按如下方式进行了说明:相应的解释、该特征的用处,并给出了容易陷入的误区及相关技巧,最后还提供了样例代码。

类和模板库

大多数C++程序都充分使用了标准的I/O流类。在本书中也对这些类进行了说明,但是本书主要是对旧的I/O流类而不是对新的I/O流类进行解释。ANSI委员会引入了一个I/O类的新的扩展版本;幸好,只需要稍微改动,就可以像使用旧的类一样使用新类。

另外还要谈一谈标准模板库。尽管本书也讲到了模板,但因为模板库太大,所以本书没有足够的地方完全包含整个模板库。模板库是ANSI C++标准中非常有用的部分,并为本书增色许多。

C++与ANSI委员会

先有C,后有C++,然后才是ANSI C++。

在C++第一次出现时,每个人都想知道更多的东西。由于模板是许多其它高级语言的特征,所以许多人要求在C++中加入模板。后来,出现了不同版本的C++并且每个软件供应商都提供了不同的升级版本。

ANSI(美国国家标准研究所)的任务是设计每个C++提供者都必须遵守的C++标准。尽管人们还可以提供自己的升级版,但是ANSI的标准是每个C++编译器必须支持的标准特征集。一直到现在为止,该委员会还在工作,但是他们已经制定的标准目前仍然相当稳定(也就是说该委员会的任务完成了)。

ANSI委员会试图使用各种方式添加所有流行版本的特征。如果你早就使用了C++,你会发现自己很难使用这些新特征。在本书中,使用下面的图标提醒大家注意这些新特征:

●— ANSI

该图标指出了早期版本C++不支持而ANSI却引入的语法特征。

也可以参考附录B。附录B列出了ANSI C++的新特征。本书着重说明ANSI的特征。

本书的编排

第一部分,在第一版的基础之上,向新手介绍C++。如果你学过C,但没有学过C++,你需要认真看一看第四章或第五章。第一部分目的是培养对C++的兴趣以及如何用它来完成最基本的工作,特别是完成C++特有领域的工作:类、对象、虚函数以及操作符重载等。

第二部分的大部分是重新编写的，对每个关键字、操作符、命令、函数以及类进行了说明，并给出了样例代码而且在简短的标题中给出了其基本目的或操作的总结。

如果你知道想要查找的成员的名称，但是忘记了它是关键字还是库函数，可以查看本书中间的“ /* C++ 成员 A－Z”。该部分按字母顺序列出了所有的成员并给出了参考的章节。

如果你知道想要完成的任务但是不知道实现此任务所需要的关键字或函数的名称，可以查看本书的“ /* 通俗易懂 C++ ”部分。该部分列出了常用的任务以及实现这些任务的关键字或函数。

第二部分之后是 C++ 术语及概念的词汇表。在该部分读者可以了解 C++ 语言中的一些更深的概念并且可以查看一些常用术语的定义。

本书的约定

有时 C++ 使用复杂的语法，这反映这种语言的有效性和灵活性。为了方便，本书使用了一些约定。尽管过分使用某些约定可能并不明智，但是只要我认为使用它们能够简化某些事情，我就会使用这些约定。

主要的约定是：粗体的部分意味着你应该完整地输入粗体部分。斜体部分是占位符，表明你应该使用自己的内容替换它。例如，在下面的用法中，type 是你选择的 C++ 类型名称。因此，应该这样使用 sizeof(int)、sizeof(long)、sizeof(float)。

sizeof(*type* **)**

有时，也使用中括号来指明可选项。在下面的用法中，中括号意味着“C”可以加入也可以省略。当然，不应该输入中括号本身。

extern [**"c"**] *declaration* **;**

如果需要输入中括号，本书就使用粗体来显示它们。只有在涉及到数组以及 new 和 delete 操作符时才出现这种情况。例如，在下面的用法中，应该输入 new int [100]或 new long[56]。

new *type*[*size*]

最后，第十五章包括每个函数的声明。这些声明列出了函数的每个参数及其类型。在使用这些函数时，无须输入参数的类型，只输入参数即可。例如，如下的声明表明 abs 需要一个整形参数并返回一整形值：

int **abs(** *int num* **);**

下面是对此函数调用的一个例子：

```
int i=abs(-75);
```

其它图标

为使读者对不同的问题引起注意,除了前面提到的ANSI图标,本书还使用了其它几种图标。

●—注意

该图标标出一个与当前讨论有些不同的技术问题。C++中有许多and、if、but。为了正确描述它们而又不致离题太远,这段文字通常很简洁。

●—C/C++

该图标给出关于常用的语言特征的信息。C和C++都支持这些特征但在使用时两者的处理方式却不同。C有一些C++所不具有的限制,反之亦然。在将C代码转换为C++代码时,或者在从C程序员过渡到C++程序员时,要特别注意这些问题。

●—技巧

这段文字推荐另外一种可能更快或者更方便的方法来实现某个功能。这些技巧总是以建议的形式出现。

本书的读者

对新手来说,有时C++比其它语言更具有挑战性。甚至对向高级程序员过渡的人来说,在学习C++的某个新特征时,也需要认真地对待它们。通过使用通俗易懂的语言并结合使用例子、技巧以及说明等方式,本书使得使用C++编程的每一步都变及十分简单。

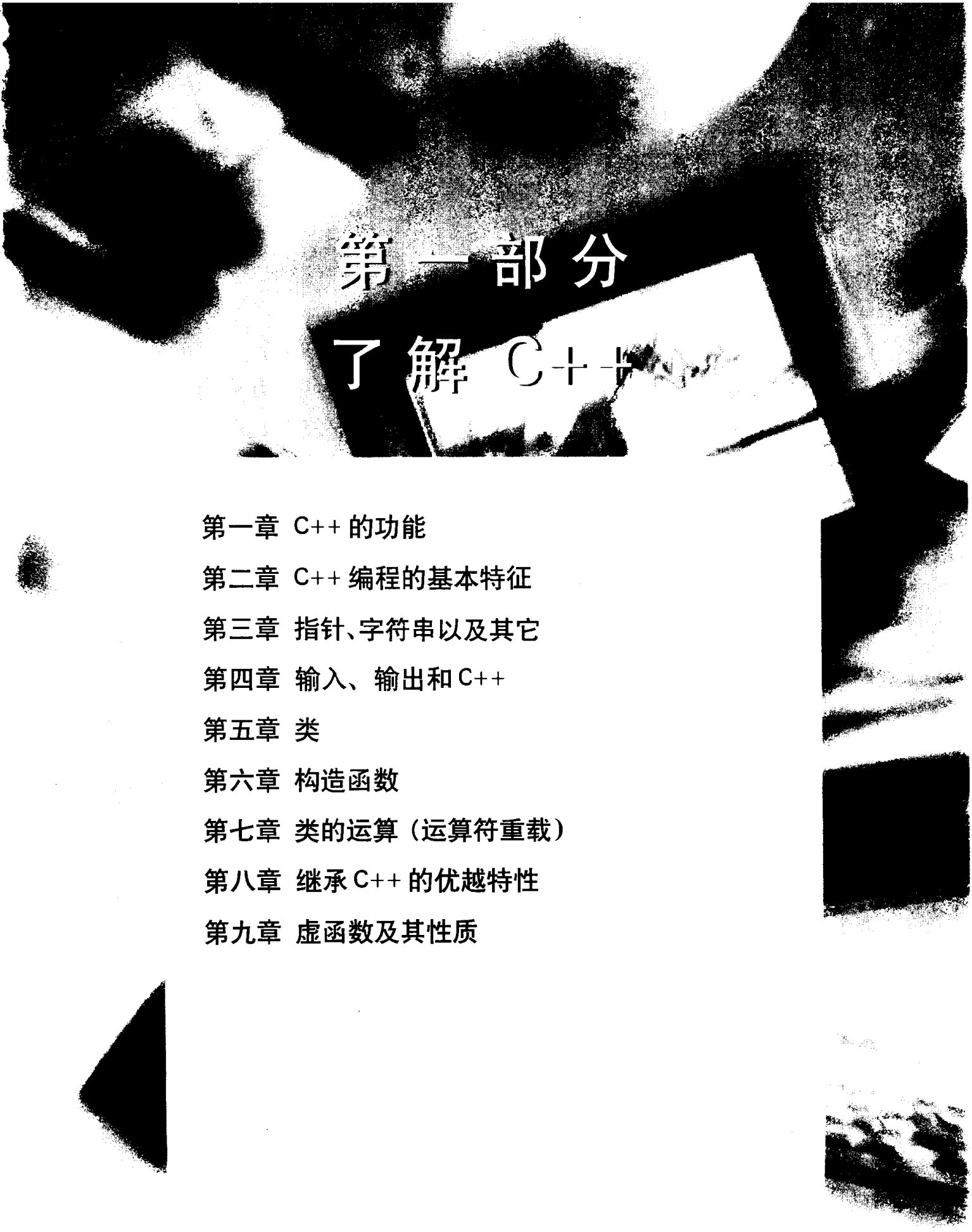
持之以恒,终有所尝。你将熟练地掌握这门奇妙的软件工具。一旦对C++有所了解,你就会感叹C++设计的灵巧以及它高度的逻辑性。对那些想要掌握未来技术的程序员来说,C++具有很大的吸引力。

如果你对本书有什么评价,请发e-mail给我:Briano2u@aol.com。我对用户的反馈总是十分感兴趣。另外,你可以在

http://www.idgbooks.com/extras/cipe/

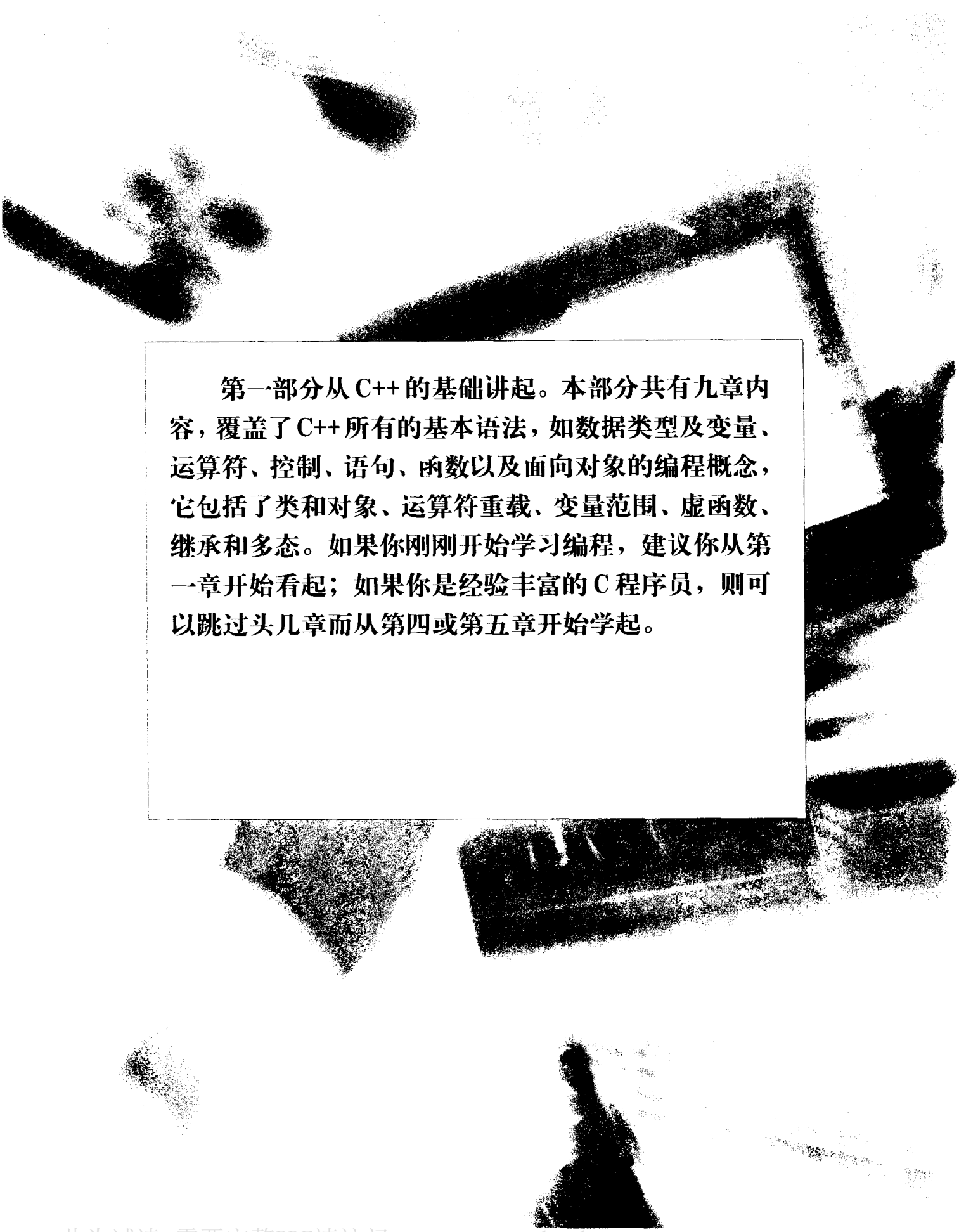
中找到样例程序文件的代码。

放松一下。只不过是C++而已,要记住它是奴隶而你是主人!



第一部分

了解 C++



第一部分从C++的基础讲起。本部分共有九章内容，覆盖了C++所有的基本语法，如数据类型及变量、运算符、控制、语句、函数以及面向对象的编程概念，它包括了类和对象、运算符重载、变量范围、虚函数、继承和多态。如果你刚刚开始学习编程，建议你从第一章开始看起；如果你是经验丰富的C程序员，则可以跳过头几章而从第四或第五章开始学起。

第　一　章

C++ 的功能

* *

在计算机的黑暗年代,程序员是机器的奴隶。开发者不得不用二进制代码(用 1 和 0 表示)编写所有的指令。二进制代码是计算机的母语。随着时代的发展,新的编程语言为程序员提供了表达算法的更好方式。计算机语言的改进意味着程序员可以将注意力从计算机的内部结构转移到程序的目的上来。

面向对象的编程方式使得软件的发展向前迈了一大步。尽管它的功能可能被夸大了,但是面向对象的编程方法确实提高了程序员的效率。一般来说,在面向对象的编程方式出现之前,最明显的编程结构是代码与数据之间的分离。这种分离精确地反映了计算机的内部工作方式,不过这并不是描述世界的理想方式。

而面向对象的编程方式却类似于人类的大脑。人类的大脑是单独的脑细胞所形成的巨大组合。如果借用计算机术语,每个细胞就是一个对象,它具有自己的基础材料(数据)以及可以控制的行为(代码)。没有必要将脑细胞的这些不同部分分离开,也就是说没有必要区分哪里是数据哪里是代码。更没有必要区分哪里存放着是大脑的所有数据,哪里存放着大脑的所有代码。

这个类比说明了面向对象的含义:它将对象作为基本的单位从而取代了传统的方式。对象由状态信息以及行为组成,并且每个对象都能象大脑中的细胞一样发出信号并可以对刺激作出响应。

使用面向对象的语言编写的程序不能自动地将现实模型化。从这一点来看它并不比其它程序优越。一个成功的程序是认真思考、详尽计划以及勤奋的结果,而不是语言选择的结果。并且,如果你想要使用面向对象的方法(越来越多的系统软件要求这种方法),那么象 C++ 这样的面向对象的语言的许多特征可以相当方便并十分有效地帮你达到目的。

为什么要使用 C++ 呢? C++ 可能不是使用最广泛的面向对象的语言;更多的人使用 VisualBasic,不过 VisualBasic 是否是面向对象的还是有争议的。但是 C++ 的争议更多。很明显,越来越多的程序员将 C++ 视为最完整的面向对象的语

言。有人争辩说Smalltalk或Eiffel已经完整地实现了面向对象,而C++可能设置了一些衡量其它语言的标准。对程序员来说,了解C++就像几年前了解C那样重要。今天任何一个大的、重要的开发项目通常都是使用C++来编写的。对开发管理员来说,C++是一门艺术。

对某些人来说,Java是刚刚出现的未来语言,至少对Internet中的程序是这样。但是应该了解到,Java有许多语法是从C++中借用去的。事实上,如果先学习了C++,Java学起来将十分容易。

在任何情况下"为什么要使用C++?"都是一个很有意义的问题。与C一样,答案很大程度上源于对功能和效率的双重需要。

C++的起源

C++是挪威编程人员的又一发明。在挪威先是出现了一个名叫Simula的语言,它也是最早使用类这个概念的语言(类是程序的单位,包含数据以及相关的函数)。类,像在本书中将要学到的那样,与对象这个概念紧密相关,类是一种对象类型。

Simula用于开发事件模型。尽管事件驱动模型与面向对象的模型在概念上并不一致,但二者有许多共同之处。这就是为什么Visual Basic、Windows、OS/2 Presentation Manager以及许多其它事件驱动的结构是面向对象的或者至少是基于对象的原因。因为面向对象的基础是可以响应消息的独立实体,所以它是实现事件驱动系统的很自然的方式。

1978年,在Cambridge大学的计算实验室,作为博士论文的一部分,Enter Bjarne Stroustrup需要编写用于分布式计算机系统的模拟器。Stroustrup发现使用Simula的类可以完整地描述网络中不同机器的接口。问题是对大规模的系统来说,Simula的效率太低。他需要将Simula面向对象的特征与某种语言,如C语言的能力和效率结合起来。

C++就是这么产生的。Stroustrup在C中使用了类并添加了Simula的数据类型创建了C++。"具有类的C"在成为今天使用的C++标准之前走了几次弯路。不过C++这种语言基本上仍然实现了Stroustrup最初将C与Simula的类结合的愿望。

从C到C++的转换

可能你以前听说过,C++是"更好的C"。这句话意味着C++是C的超集。说这句话的人认为可以使用C++编写所有用C编写的程序。另外,C++进行了一些改善,无须使用面向对象的方法就可以将这些改善添加到你的程序中。

上面陈述的第一部分只能说是大致正确。许多C程序无须修改就可以作为C++程序进行编译,但是对某些程序来说,由于存在一些语法区别而无法通过编译。本书的一个目标就是尽可能清晰地指出这些区别。(对这个问题的完整回答,请参考附录A)

C++重要的一点是它没有将面向对象强加在你身上。如果你是C程序员,应该清楚在转换到C++时,只需注意C++引入的为数不多的限制,就可以象从前一样正常工作。不必将任何东西重新编写为对象。只要花足够的时间,你就可以将C++的新功能逐步添加到你的程序中去。

类:对象的组织形式

尽管你可能想要了解C++的核心内容——面向对象,但是一开始,你还是需要花一些时间和精力来学习C++的基本知识。从概念上讲,对象只不过是内嵌有与之有关的函数的数据结构。(类是对象的类型;关于类,本书以后会经常提到。)看起来好象C++有相当多的运算符、关键字以及规则,并且它们大部分用于应付特殊情况。不过对象的概念本身却十分灵巧并相当简单。

传统程序的组织形式代表的是事物在计算机软件中的实现方式而不是代表事物在现实生活中实现方式。从技术上来说,计算机内存的所有内容都由数据构成,但是内存中存储的内容中最重要的部分是称为代码的特殊数据。代码由发送给处理器的指令构成:对两个数进行加、跳转到新地址等等。内存的其它部分由数据构成,正如人们所说的那样,数据是提供给用户或计算机使用的信息。

因为(与人类的大脑不同)计算机只有一个中央处理器,要由这个处理器来分别处理代码和数据,所以代码与数据在计算机内存中是截然分开的。除了偶尔跳到新的位置,中央处理器顺序地执行代码,由于这个原因,大段的代码不得不结合成代码段。传统程序的组织形式反映了这个事实。一般来说,代码可以使用函数图表来表示,由指令集合构成的每个函数作为一个代码块来执行。数据由记录(结构)、数组以及表格组成。

尽管有这么多的结构,最终的程序仍然是函数的层次图表和与之分离的数据结构的集合。这种编程语言在这两个部分之间没有加入联系(如图1-1)。

代码与数据的分离会产生问题吗?对简单的程序来说,这种分离不会有什么影响。但是程序员必须牢记哪个函数使用哪个数据结构,否则就会出现错误。

打开计算机,你就会明白计算机硬件本身并不适合传统的软件模式。计算机结构本身并没有反映代码与数据的分离!相反,计算机的主板是由一些独立的芯片组成,这些芯片通过电路在彼此之间发出或接收信号(如图1-2)。

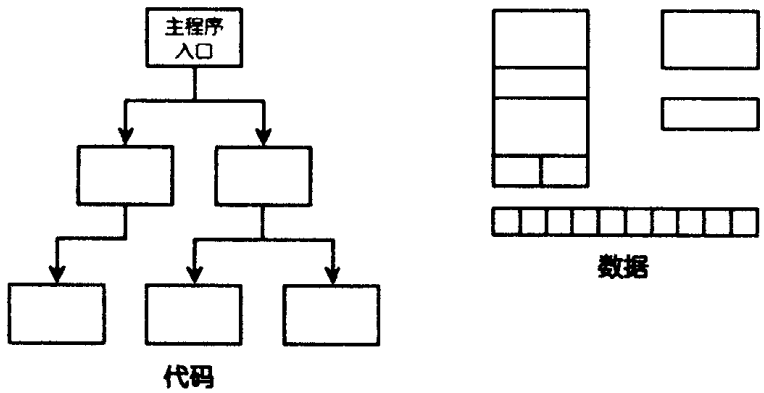


图 1-1 由传统程序组织形式分离代码与数据

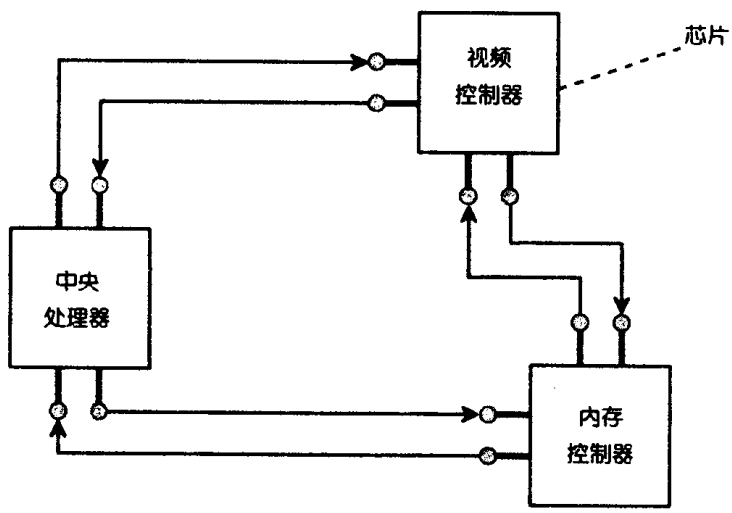


图 1-2 计算机箱内部的活动

在面向对象的编程方式中,这些芯片变成了对象。按照C++的术语,芯片(如奔腾)的制造及其模型是一个类;单独的一个芯片是一个对象。芯片的软件描述既不是纯代码的也不是纯数据的。与大脑细胞类似,一个芯片既有代码(行为)又有数据(状态信息)。

如果你开始编写一个描述计算机内部硬件的程序,你会发现对于程序的设计及编写来说,使用对象来表示每个芯片是相当有用的。以下就是面向对象的C++与C截然不同的所在:组成程序的基本单元不是代码或数据,而是对象的类,而类则是包含有数据与代码的一种新类型(图1-3)。

封装:方便的编程方式

图 1-3 中最重要的观点之一是黑匣子的概念。芯片仅通过特定的管脚进行通信;除了这些接口,每个芯片对其它的芯片来说是不可知的。没有任何一个芯片可以到达另一个芯片的内部并干涉其内部工作。另外,计算机的设计者也不需要知道芯片的内部电路,只要该芯片的输入输出如芯片制造者所说的那样工作正常即可。

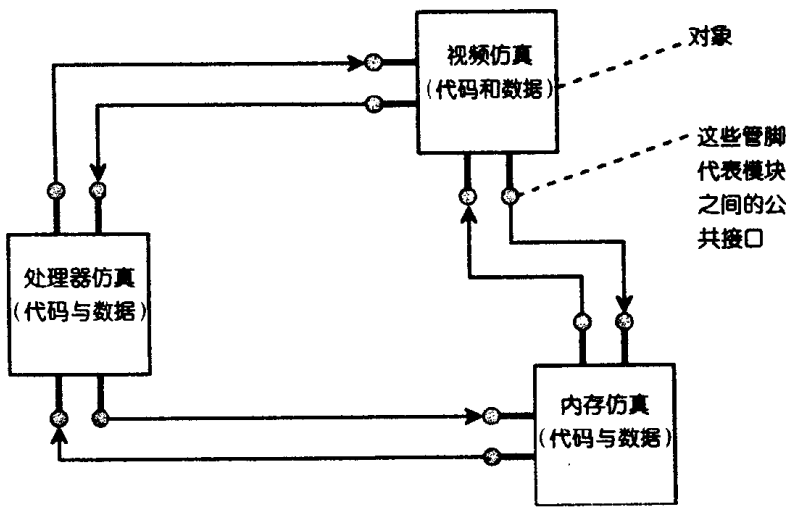


图 1-3
对象的
组织形式

黑匣子具有这种特征是必要的。由于每个芯片都有特定的状态指明芯片的运转情况,如果每个芯片都工作正常,系统就会正常工作。只要芯片使用同样的规定,就可以用新的芯片来代替旧的芯片而整个系统也会象以前一样运转正常。尽管这个新的芯片的内部结构与旧的芯片完全不同,你也无须担心。只要芯片与系统按照同样的方式相互作用,任何两个芯片都是可以互换的。

没有什么对互换性的要求比在软件开发中更大了。在一个典型的项目中,软件组要经常重写程序每个部分的内容。打比方说,为了查找错误、提高效率或者增加新功能,程序员要不断地用新的芯片代替旧的芯片。按照传统的编程方式,程序的各个部分没有完全明显分开,这样的程序常常会带来灾难性的后果。对程序任何部分的改变都可能会影响程序的其它部分。例如,Pete 现在的工作是使用 Camille 正在编写的程序的一些内容。但是一旦 Camille 对程序进行了某些修改,Pete 的所有假设就都是无效的,于是错误就发生了。

从某种程度来说,C 语言的功能也可以解决这个问题。可以在文件级别对不同模型(接口)之间的连接进行管理。例如,可以让 Pete 与 Camille 工作在不同的文件中并且在以后注明对所声明共享数据的限制。C 语言的某些功能(如关键字 extern

及 static)可以控制某个文件中的数据及函数是否对其它文件可见。

C++提供了在程序不同部分之间更好地共享数据及进行函数控制的方法。数据的保护部分不再限于文件级别,而是可以如你所愿——根据类任意扩大或缩小。在C++里,可以在类的级别使用各种私有函数或数据。函数和数据都被视作成员。这些私有成员对程序的其它部分是不可见的:不能对它们进行访问。对象的公有成员构成了对象的接口。这些公有成员构成了对象外部可见的"管脚"。程序的其它部分可以引用这些成员,前提是使用时这些成员没有发生改变。同时,你可以按照自己的意愿重新编写类的内部。

内部与接口之间明显的分离称为封装。封装意味着"保护某些事物的内部"。C++的封装方式不仅提供了更大范围的控制,而且使得在源代码中公有/私有的区别更加清晰。C++有助于表明哪一部分是类的接口,哪一部分是类的内部。

●—注意

在第一次学习C++及面向对象时,你可能也使用对象及类的可互换性这些术语。但是记住以下这些区别很重要:类是一种类型而对象是这种类型的一个实现。在第五章将会更多地强调这个区别。

多态:分散化控制

面向对象的系统的一个好处是每个对象都尽可能地独立。我可以给一个对象发送一个通用的信号,该对象能够正确地响应。我不必了解该对象是如何产生这些响应,甚至不必知道对象的确切类型。

如果使用硬件术语,可以这样说,我能够发送一个通用的打印命令而无须提前知道打印机的制造过程以及它的模型。只要每个人都遵守同样的通信协议,我的打印命令将一直起作用。一般来说,该命令的功能是:打印文档。而点阵打印机对该命令的响应方式与激光打印机的响应方式完全不同。

面向对象这一点意味着主函数或主循环知道的越少越好。如何执行命令的决定应该在对象内部产生。这时,对主函数来说要做的只是发送一个通用信号。

在第九章我将给出一个例子,用于说明程序是如何实现这一点的。该例子使用一组菜单命令。在用户选择一个命令时,主函数给菜单对象发送一个消息:Do_Command。按照命令的不同,程序按不同的方式进行响应(如图1-4)。

因为程序没有被限制在特定的命令集合中,所以这种方式十分有效。主函数所要做的只是给相应的菜单对象发送一个Do_Command命令,这样程序就会作出正确的响应。使用这种方式,在将来开发新的菜单命令时无须重写主函数,而且也可以动