高职高专计算机系列教材

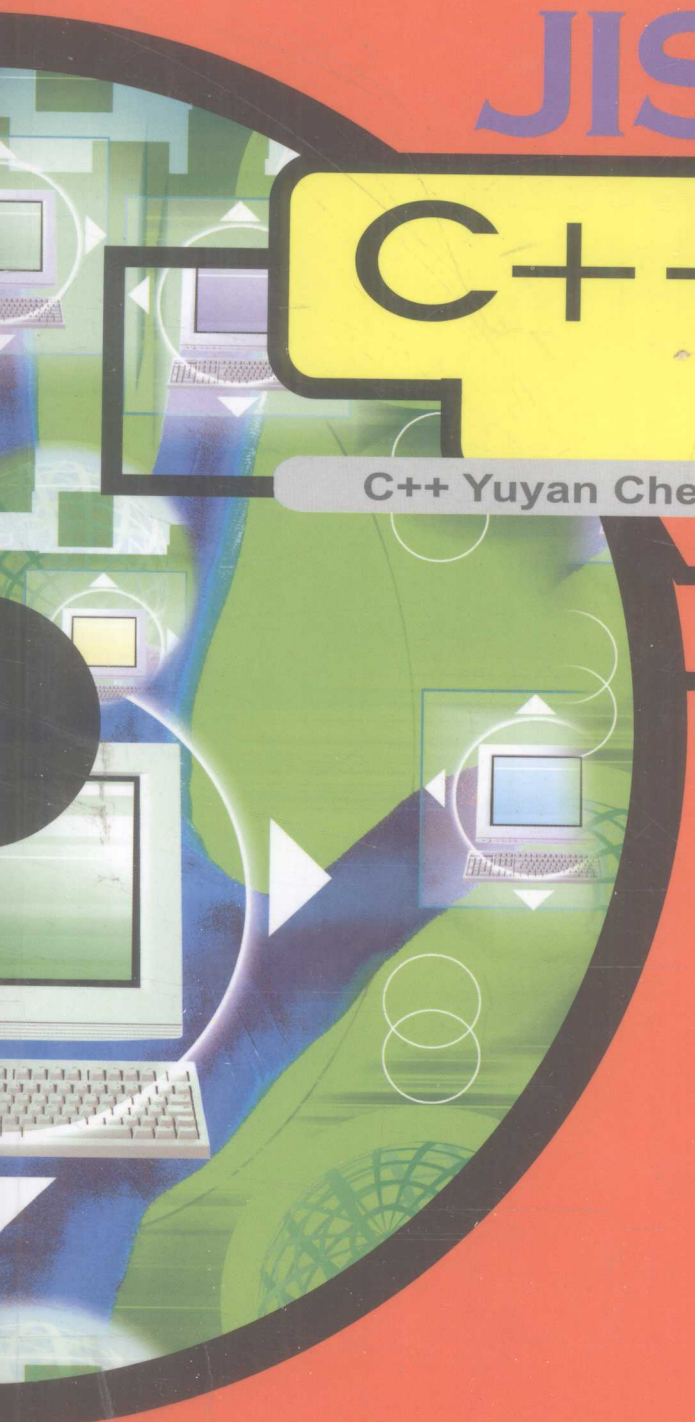
JISUANJI

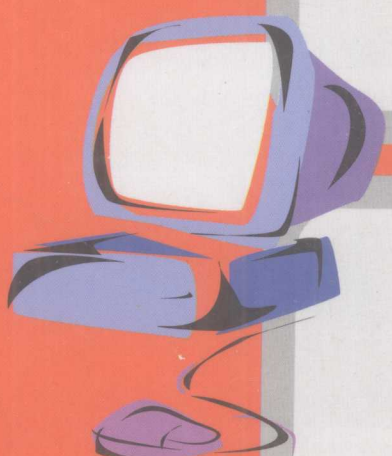
C++语言程序设计

C++ Yuyan Chengxu Sheji

主　编　田锋社

副主编　朱卫红





重庆大学出版社

C++语言程序设计

主　编　田锋社
副主编　朱卫红

重庆大学出版社

内 容 提 要

本书为高职高专计算机系列教材之一。全书内容包括：概述，基本数据类型、运算符及表达式，控制结构，数组，函数和作用域，编译预处理，指针，结构体与共用体，类与对象，继承与派生类，多态性和虚拟函数，文件和流类库。

本书供高职高专计算机专业作教材使用，也可供其他相关人员参考。

图书在版编目(CIP)数据

C ++ 语言程序设计/田锋社主编. —重庆：重庆大学出版社，2004.5
(高职高专计算机系列教材)
ISBN 7-5624-3105-1

Ⅰ.C... Ⅱ.田... Ⅲ.C 语言—程序设计—高等学校：技术学校—教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2004)第 028487 号

C ++ 语言程序设计

主　编　田锋社

副主编　朱卫红

责任编辑：曾令维　穆安民　　版式设计：曾令维

责任校对：任卓惠　　责任印制：张立全

*

重庆大学出版社出版发行

出版人：张鸽盛

社址：重庆市沙坪坝正街 174 号重庆大学(A 区)内

邮编：400030

电话：(023) 65102378　65105781

传真：(023) 65103686　65105565

网址：http://www.cqup.com.cn

邮箱：fxk@cqup.com.cn (市场营销部)

全国新华书店经销

重庆升光电力印务有限公司印刷

*

开本：787 × 1092　1/16　印张：14.25　字数：355 千

2004 年 5 月第 1 版　　2004 年 5 月第 1 次印刷

印数：1—5 000

ISBN 7-5624-3105-1/TP · 468　　定价：19.50 元

前言

随着计算机技术的迅速发展，面向对象程序设计异军突起，各种面向对象程序设计环境应运而生，程序设计从面向过程全面跨入面向对象。C ++ 语言就是在 C 语言的基础上发展起来的一种混合型的高级程序设计语言。它既具有独特的面向对象的特征，全面支持面向对象技术；同时又具有传统 C 语言的特点，支持面向过程的结构化程序设计。因此《C ++ 语言程序设计》已经是许多高等专科学院和高等职业技术学院开设的一门计算机教学的主干课程，根据高职高专计算机教育发展的需要，我们编写了这本《C ++ 语言程序设计》教材。

参加本书编写的编者都是在教学一线、具有多年从事程序设计教学的骨干教师。本书针对高职高专教育的具体特点，着力体现了高职高专院校最新的教学、教材改革思想，以突出培养高级应用型技术人才的教学目的。教材内容循序渐进、层次分明、重点突出、通俗易懂，具有较强的实用性。同时，教材编写时直接进入 C ++ 语言，没有阐述如何从 C 语言过渡到 C ++ 语言的过程及概念，也没有对两种语言进行过多的比较。这样做不仅克服了初学者的茫然，而且体现 C ++ 语言程序设计的完整性。因此本书不仅适用于有 C 语言基础的学生，也适用于没有任何程序设计语言基础的学生学习。本书着眼于读者编程能力的提高，书中的源程序均在 Visual C ++ 6.0 环境中进行了调试、运行。

本书可作为高职高专计算机及相关专业的《C ++ 语言程序设计》课程教材，也可以作为广大程序设计爱好者及工程技术人员的参考用书。为了帮助读者顺利学习本课程，我们另外编写了《C ++ 程序设计学习指南与上机指导》作为本书的配套教材，其内容包括：上篇（学习指南）内容是针对教材每一章内容编写有“本章小结”、“学习要点”、“例题解析”、“习题”几个部分。下篇（上机实训）内容是针对教材精心挑选了 14 个实验。

本书由田锋社担任主编，朱卫红、刘丽、梁艳、董纹颖、徐凯参加了编写。具体分工如下：田锋社编写了第1章、第5章、第9章、第10章、第11章。朱卫红编写了第7章、第12章；刘丽编写了第2章、第3章；梁艳编写第4章，董纹颖编写第8章，徐凯编写第6章。田锋社负责对全书进行修改、统稿。

由于作者水平有限，书中一定会有不少疏忽和不足之处，恳请专家和读者批评指正。

编　者

2004年3月

目录

第 1 章
C ++ 语言概述

随着计算机技术的迅速发展,面向对象程序设计异军突起,一跃成为程序设计的主流;各种面向对象程序设计环境应运而生,程序设计从面向过程全面跨入面向对象。C ++ 语言就是在 C 语言的基础上发展起来的一种混合型的高级程序设计语言。它既具有独特的面向对象的特征,全面支持面向对象技术;同时又具有传统 C 语言的特点,支持面向过程的结构化程序设计。在当前使用的程序设计语言中,C ++ 语言是使用最广泛的语言之一,深受广大编程人员的喜爱。

1.1 C ++ 语言概况及特点

1.1.1 C ++ 语言发展简史

C ++ 语言是在 C 语言的基础上扩充形成的一种语言,而 C 语言的历史可追溯到 1960 年由 Peter Naur 组织研制的 Algol60 程序设计语言,该语言是一种面向过程的高级语言,由于它离硬件较远,不宜用来编写系统程序。1963 年英国剑桥大学推出了一种 CPL 语言(combined programming language),它在 Algol60 语言的基础上更接近硬件一些;但它的规模较大,实现困难。1967 年剑桥大学的 Matin Richards 对 CPL 语言做了一些简化,推出了 BCPL 语言(Basic combined programming language)。同年 Nygaard & Dahl 对 Algol60 语言做了修改,先后推出了 Simula66 和 Simula67 两种版本的语言,该语言首次引进了对象和类的概念而成为了第一个面向对象的语言。1970 年美国贝尔实验室的 Ken Thompson 以 BCPL 语言为基础,并对其做了进一步的简化,设计出简单的、更接近硬件的 B 语言;同时使用 B 语言编写了第一个 UNIX 操作系统。1973 年贝尔实验室的 Dennis. M. Ritchie 在 B 语言的基础上设计出了 C 语言。

C 语言既保持了 B 语言精练、接近硬件的优点,又克服了 B 语言数据无类型等缺点。同时,D. M. Ritchie 和 Ken Thompson 合作,用 C 语言成功地改写了 UNIX 操作系统。1978 年 Brian W. Kernighan 和 Dennis M. Ritchie 推出了 UNIX 第七版中的 C 编译程序基础,合著了影响深远的名著《The C Programming Language》,这本书中介绍的 C 语言成为后来广泛使用的 C 语言版本的基础,即后来被称为“标准 C”的程序设计语言。1983 年美国国家标准化协会(ANSI)

根据 C 语言问世以来出现的各种版本的 C 语言,制定了新的标准,称为 ANSI C。1986 年贝尔实验室的 Bjarne Stroustrup 在 C 语言的基础上设计出了 C ++ 语言,它保持了 C 语言的所有优点并进行了增强,同时增加了面向对象的机制,成为了典型的面向过程和面向对象的混合语言。该语言非常适用于大型系统软件和应用软件的开发。随着可视化开发环境的出现,C ++ 实现的典型环境有 Visual C ++ (简称 VC ++)和 Borland C ++ (简称 BC ++)等。本书主要以 VC ++6.0 为编译环境介绍 C ++ 程序的设计。

1.1.2 C ++ 语言的特点

前已述及,C ++ 语言是一种既可面向对象又可面向过程的混合型程序设计语言;是在 C 语言的基础上发展起来的程序设计语言,它既适合于编写系统软件,也适合于编写应用软件。所以,它既具有 C 语言的面向过程的特点,又增加了许多面向对象程序设计的特点。具体归纳如下:

1)C ++ 语言表达简洁、紧凑,使用灵活、方便,关键字及控制语句较少;语法限制不太严格,对变量的类型使用比较灵活,程序设计自由度较大,因此,程序员应仔细检查程序的语法及结构的正确性,不能完全依靠编译系统查错,对所查出的错误要认真分析,有些错误可能是前面的某一语句引起。

2)C ++ 语言有丰富的运算符和数据类型,不仅可以实现各种线性数据结构,而且可以实现复杂的非线性数据结构。

3)C ++ 语言同 C 语言一样不仅具有高级语言的功能,还兼有低级语言的一些特征。允许程序员直接访问物理地址,能实现位操作,可以直接访问计算机硬件。正因为这一特点使它成为编写系统软件和应用软件最理想的语言之一。

4)C ++ 语言支持数据封装。在 C ++ 语言中,类是支持封装数据的工具,对象则是数据封装的具体实现。将数据和对该数据进行合法操作的函数封装在一起作为一个类的定义。同时封装还提供了一种对数据访问的严格控制机制;在 C ++ 类中,定义有三种不同访问权限的成员——私有成员、公有成员、保护成员,程序的其他部分不能直接作用于对象中的数据,对象间的相互作用只能通过明确的消息来进行,每个对象根据所收到消息的性质来决定需要采取的行动,以便响应这个消息。封装性避免了程序的许多维护问题,若一个基本数据类型的结构被修改,除类中访问该数据方法的代码外,程序的其他部分不受影响,这是因为基本数据在类的外部是不可见的。改变一个类的实现不影响使用这个类的程序,从而大大减小了应用程序出错的可能性。

5)C ++ 语言支持多态性,允许同名函数以不同的参数类型或不同的参数个数实现重载。允许一个相同的标识符或运算符代表多个不同实现的函数,用户可以根据需要定义标识符或运算符重载。同时,作为多态性的另一个方面,C ++ 支持动态联编。

6)C ++ 语言支持继承性,允许一个类可以根据需要生成派生类(单继承),也允许几个类共同派生出一个新类(多继承)。派生类继承了基类的所有特性,同时派生类自身还可以定义自己需要的但不包括在基类中的新方法。一个派生类的每个对象都包含有从基类那里继承来的成员及自己所特有的成员。由于支持继承性,无论是在程序编制阶段,还是对已有的程序进行扩充、维护、改写时,继承性都会带来极大的方便。

7)为了提高编程的灵活性,C ++ 允许使用友元破坏封装性。利用友元可以访问类的私有

成员。友元可以是定义在类外的函数,也可以是定义在类外的整个类。前者称为友元函数,后者称为友元类。

对C++语言的上述特点,读者也许现在还不能深刻地理解。这些特点将会在以后的章节中具体介绍,等学完C++语言以后,相信读者将会有比较深刻的体会。

1.1.3　C++程序结构

C++程序以函数作为基本单位,函数由“注释部分”和“说明部分”及“函数体”组成。“注释部分”是对程序的注解和说明,并不参加程序的编译和运行,它包括对该函数或某语句、某数据等的用途或其他文字说明。“说明部分”包括程序所包含的库函数和其他函数以及所定义的类、派生类、结构体、共用体或枚举类型等,同时也包括函数说明(函数类型、函数名、函数参数、参数类型等)。“函数体”一般包含该函数中所定义的变量和程序具体的执行部分。

例如:求一个实数的绝对值。

```
#include < iostream. h >        //文件包含,库函数
#include < math. h >            //文件包含,库函数
void main( )                    //函数说明
{                               //函数体
 double x,y;                    //变量定义
 cin >> x;                      //以下均为执行部分
 y = fabs( x) ;
 cout << "y = " << y << endl;
}
```

这是一个最简单的C++程序。由此可见,C++程序由前面介绍的三个基本部分组成。另外,每个语句后必须有一个分号“;”作为结束标志,一行允许书写几个语句,一个语句也可以分成几行书写,但是每个语句结束标志必须是分号。用“//”符号表示注释标志。花括号{ }内的是函数体;endl单词表示换行操作。程序的具体书写方式将在以后的章节中逐渐介绍。

1.2　C++的词法及词法规则

源程序的代码都是由一些合法的字符按一定的规则组成的单词组成。例如[ ]组成数组的下标,符号 << 及 >> 分别表示提取操作和插入操作,student_data代表一个合法的标识符等等。所以本节主要介绍C++语言中所使用的单词及字符集。

1.2.1　C++的字符集

字符是一些可以区分的最小的符号,而程序中的每个单词、标号、标识符等都由合法的字符组成,程序中不能使用非法的字符组成的单词(例如希腊字母等),因为计算机对这些字符没有相应的ASCII值。C++的字符集包括下列一些符号:

26个大小写的英文字母:　a~z及A~Z

阿拉伯数字:　0~9

特殊字符:空格 # ! % & ^ _(下划线) + - * / ~ > < \ |
. , ; ? ' " ( ) { } [ ] =

除上述的字符以外,C ++ 认为其他任何字符为非法字符。

1.2.2 单词及词法规则

单词就是词法记号,是由若干个字符组成的具有一定意义的最小词法单元。在 C ++ 中共有 6 种单词。

1)关键字

关键字是指系统已经预先定义的一些单词,也称为保留字(还包括设备字)。它们在系统中都有着不同的用途,程序员不能对这些单词重新进行定义。

C ++ 的关键字有: break, const, float, double, int, long, sizeof, static, do, switch, class, struct, union, while, char, short, unsigned, new, delete, signed, void, extern, static, register, auto, typedef, if, else, for, goto, case, default, return, continue 等等。

2)标识符

标识符是由程序员自己定义的一些单词,在程序中标识符一般表示程序中的一些实体。如函数名、变量名、常量名、对象名、类型名、语句标号等。在 C ++ 中,标识符由大小写的字母、数字和下划线组成,并且必须以字母或下划线开始。组成标识符的字符个数(标识符的长度)是任意的。但是值得注意的是不同的编译系统能够识别的标识符的长度不同,建议标识符的长度不要超过 32 个字符。

标识符中字母的大小写是有区别的。如 abc 和 aBc 代表不同的标识符。因此在实际使用中建议使用有意义的单词做标识符。如用 student_name 表示学生姓名,用 student_num 表示学号等。

程序员在定义标识符时不能使用关键字或保留字。

3)运算符

运算符是由 C ++ 字符集中的一个或多个字符组成的符号。C ++ 语言的运算符非常丰富,如“ + ”表示算术加法运算、“ = ”表示赋值运算等。下一章将详细介绍不同的运算符。

4)分隔符

分隔符是用来分隔程序的正文及单词的符号,也称为标点符号;它用来表示某个程序实体的结束和另一个程序实体的开始。C ++ 中,常用的分隔符如下:

空格:作为程序中单词和单词的分隔符。

逗号:用来分隔多个变量或多对象的;或作为函数多个参数之间的分隔符(逗号还可以作为运算符)。

冒号:用在 switch 语句中的 case < 整形常量表达式 > (或 default)关键字之后,也用来作为语句标号和语句之间的分隔符。

分号:表示 C ++ 语句的结束标志,也用在 for 语句中。

花括号{}:用在函数体的开始和结束、复合语句、类型(类、结构体、共用体、枚举)中等。

5)常量

常量是程序中使用符号直接表示的数据,C ++ 中有字符常量、字符串常量、数字常量等。

6)注释符

如前所述，C ++ 程序包括注释部分，它是对程序的注解和说明，不参加程序的编译和执行。在 C ++ 中使用“//”作为注释标志，从符号//开始直到该行结束系统均视为注释行。也可使用“/ * 注释内容 * /”作为注释标志。

1.3　C ++ 程序的开发环境

对于学习程序设计来说，不仅要学习程序设计语言及语法规则、程序设计方法，而且还要学习程序编辑、运行的环境。Windows 操作系统的出现为建立可视化的 C ++ 编程环境奠定了基础。目前使用较多的 C ++ 编程环境有 Borland C 公司推出的 Borland C ++ 和 Microsoft 公司推出的 Visual C ++ 及 IBM 公司推出的 Visual Age C ++ 。本书主要以 Visual C ++ 作为编程环境来介绍 C ++ 语言的具体规定及语法规则。

每一个 C ++ 程序都必须经过编辑—编译—链接—运行几个过程，才能实现进行程序设计、计算的目的，才能完成相应的任务。

1.3.1　编辑

编辑就是将 C ++ 源程序输入到计算机，保存后生成磁盘文件的过程。将源程序存入磁盘后生成的文件的扩展名为. cpp，在 Visual C ++（简称 VC ++）环境中可以利用“文件”菜单中的“新建”选项（图 1.1），在新建对话框中，选定“C ++ source file”并确定源程序的名称、存放地点，确定后可以方便地输入源程序；也可以利用 Win32 Console Application（Win32 控制台应用程序）建立工程，再添加 C ++ 源文件。在输入源程序时，可以利用“编辑”菜单中提供的编辑工具（具体的编辑过程详见配套的实训教材）。

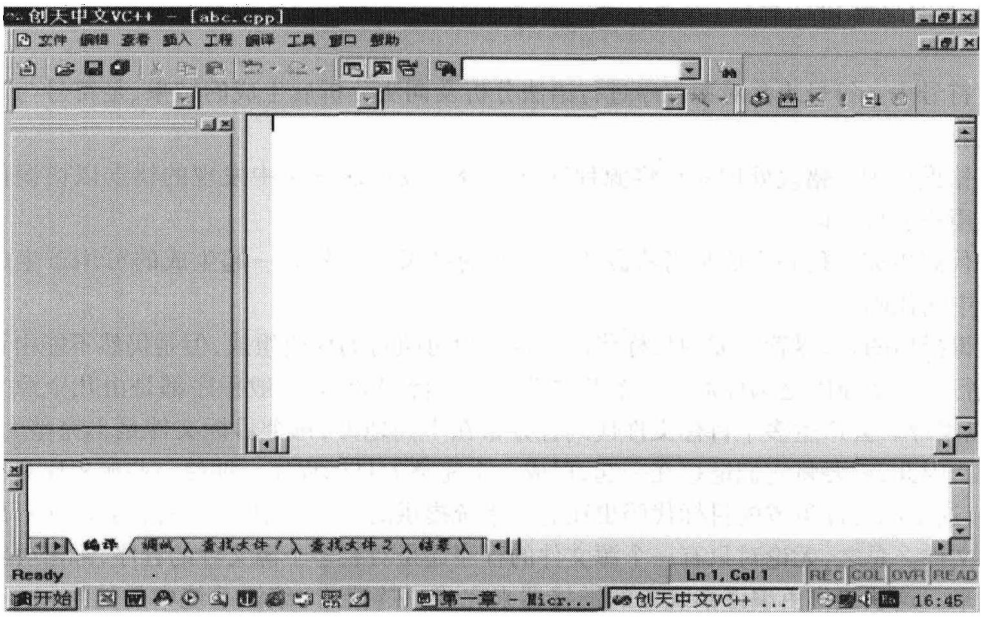


图 1.1　VC ++ 对话框

1.3.2 编译

当源程序编辑结束后,计算机系统并不认识这些源程序代码(.cpp),因此必须使用 C ++ 语言编辑器对其进行必要的翻译,这就是编译。编译器的功能就是将源程序代码(.cpp)首先转换成为机器代码形式(.obj);然后由链接器将目标代码进行链接,生成计算机可执行文件(.exe);同时在上述两个过程中,计算机系统还要负责检查源程序代码及目标代码的正确性。图 1.2 是编译的具体过程。

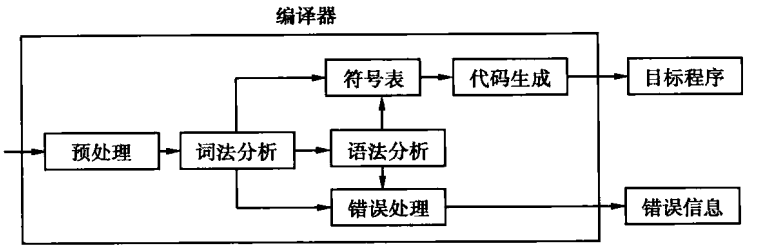


图 1.2 编译过程

1)预处理过程。预处理是在源程序进行编译前首先执行源程序中的预处理命令,是将有关符号进行直接替换的过程,不检查具体的语法问题,本书将在第 6 章予以介绍。如果程序中没有预处理命令,系统将直接进行下面的编译。

2)词法分析。词法分析主要是找出程序中的词法记号的有具体含义的单词,包括标识符、运算符、分隔符、关键字等,并检查是否有词法错误,若有将反馈给错误处理程序,若没有错误将生成相应的符号表,此阶段与程序其他部分基本无关。

3)语法分析。语法分析主要是对程序的语法结构进行分析,发现是否有不符合要求的语法错误,及时反映语法错误;若没有语法错误将生成符号表。此阶段基本上与机器的硬件无关。

4)符号表。符号表是对源程序进行语法分析及词法分析后生成的结果,是符号与它们属性的映射。

5)错误处理。错误处理就是将源程序词法分析及语法分析中发现的错误以错误信息的形式向用户进行反馈。

6)代码生成。代码生成是指将源程序分析的结果及符号表一起生成的可用于生成目标程序的中间代码。

值得指出的是,尽管生成的目标代码文件是由可执行的代码组成,但是仍然不能由计算机直接执行。主要原因是编译器对每个源文件分别进行编译,而一般程序都是由几个源文件组成,所以编译后会产生多个目标文件代码且分布在不同地方,每个目标文件是不完整的、彼此孤立的。因此,需要将它们链接在一起,形成一个完整的目标程序。即使一些源程序只有一个源文件,这个源文件生成的目标代码也还需要系统提供的库文件中的一些代码,才能形成完整的目标文件。总之,无论对只有一个源文件的程序还是对有多个源文件的程序,都需要将它们链接起来。这项工作由编译系统中的链接器来完成。链接器(编译系统中的链接程序)的功能就是将编译器生成的一个或多个目标代码文件和库文件中的有关代码进行链接,生成一个完整的可执行文件(.exe)。链接过程中,若发现链接错误,系统仍要向用户返回。

编译过程的具体操作详见本书配套的实训教材。

1.3.3 运行

经过上述的编辑、编译和链接后生成的文件为可执行文件，可以在计算机上直接运行。运行可执行文件的方法较多，在 VC ++6.0 系统下，通过选取“编译”菜单中的“执行”命令或“Ctrl + F5”便可实现。程序被运行后，计算结果将出现在屏幕上或生成磁盘文件，用户可以根据结果判断程序是否有算法或其他错误。

前已述及，程序在编译及链接过程中需要改正出现的所有致命错误和警告错误，这样才能保证生成没有错误的可执行文件。当然，程序中仅存在警告错误时，也会生成可执行文件，但一般要求改正警告错误后再执行可执行文件，以免造成计算结果错误。需要注意的是，由于 C ++ 语言语法限制不太严格，程序自由度较大，因此对于已经通过编译、链接和运行的程序，仍然不能认为是绝对正确的，仍然需要对程序进行必要的测试，以便确定其正确性。

程序测试的一般方法是通过让程序试运行一组简单的数据，来测试程序的逻辑以及各种语句有无错误。测试的数据应该结合具体实际，不仅包含输入数据，而且应确切知道程序执行后的预期结果。这里将涉及不同专业领域的判断和分析、计算（即程序所要处理的对象的专业）。不同的测试方法所对应的测试用例的设计方法各有不同。

1.4 程序设计的一般方法

随着计算机技术的迅速发展，程序设计在工程实践中的地位已变得越来越重要，绝大多数的工程问题涉及各种各样的程序设计（控制程序、计算程序、实时监控程序、检测程序等）。所以有人认为，在当今时代至少掌握一门程序设计语言是新世纪人才必须具备的业务素质之一。那么究竟什么是程序设计呢？从根本上说，程序设计是当人们要求计算机完成某项工作时，对计算机工作规则的具体描述；是人的智力克服客观问题的复杂性的过程。由于客观世界是复杂的，而人的智力是有限的，人的大脑不可能进行许多十分复杂的计算或判断。古语有云：“工欲善其事，必先利其器”。这就要求人们利用计算机这个工具将一个复杂的事物进行合理的描述，此即为程序设计方法。高级程序设计语言进步的主要动力也正源于人们对程序设计方法的不断探索。

目前，程序设计方法主要分为两大类，即面向对象的程序设计和面向过程的程序设计方法，本书主要介绍面向对象的程序设计。

1.4.1 面向对象程序设计方法

面向对象程序设计的最根本目的就是使程序员更好地理解和管理庞大而复杂的程序；是在结构化程序设计（面向过程）基础上进一步的抽象。最早的面向对象的设计语言有 Simula 和 Smalltalk。关于 Simula 语言已在 1.1.1 节中做了介绍，而 Smalltalk 语言是美国 Xeron 公司 Polo Alto 研究中心 1972 年为快速处理各种信息而在 Alto 个人机上研制的，后来有 Smalltalk 2.0及 Smalltalk8.0 等不同版本。

面向对象程序设计中最基本的概念是对象，一般意义上的对象指的是一个类（实体）的实