面向对象程序设计系列教材

Visual C++

与面向对象程序设计教程

刘路放 编著 冯博琴 审

高等教育出版社

面向对象程序设计系列教材

Visual C++与面向对象程序设计教程

刘路放　编著
冯博琴　审

高等教育出版社

内容提要

本书是根据教育部提出的高等学校计算机基础教学三层次要求组织编写的教材。

本书主要讲授如何应用 Visual C++进行面向对象和可视化编程。主要内容包括：Visual C++编程环境与 Visual C++程序设计基础，面向对象的概念和方法，图形用户界面程序设计等。本书结合应用开发实例，讲练结合，注重精讲多练，培养学生的程序设计和综合开发能力。书中配有丰富的例题和习题。

本书可作为高等学校计算机或相关专业的教材或参考书，也可供实际应用开发人员学习参考。本书同时配有教学辅助课件，供教师教学和学生自学使用。

图书在版编目(CIP)数据

Visual C++与面向对象程序设计教程/刘路放编著.
北京：高等教育出版社，2000.7
ISBN 7-04-007921-6

Ⅰ.V… Ⅱ.刘… Ⅲ.C 语言-程序设计-教材
Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2000)第 60396 号

Visual C++与面向对象程序设计教程
刘路放 编著 冯博琴 审

出版发行	高等教育出版社		
社　　址	北京市东城区沙滩后街 55 号	邮政编码	100009
电　　话	010-64054588	传　　真	010-64014048
网　　址	http://www.hep.edu.cn		
经　　销	新华书店北京发行所		
印　　刷	北京印刷集团有限责任公司 印刷二厂		
开　　本	787×1092　1/16	版　　次	2000 年 7 月第 1 版
印　　张	23.5	印　　次	2000 年 7 月第 1 次印刷
字　　数	570 000	定　　价	24.60 元

前　言

随着 Windows 操作系统的崛起，由传统的面向控制台的字符软件开发向面向窗口程序的可视化编程转化已成必然趋势。而 Visual C++正是 Windows 环境下最强大、最流行的程序设计语言之一。

Visual C++支持面向对象的程序设计方法（OOP——Object-Oriented Programming），支持 MFC（Microsoft Foundation Class）类库编程，有强大的集成开发环境 Developer Studio（其中包括了程序自动生成向导 AppWizard、类向导 Class Wizard 和各种资源编辑器，以及功能强大的调试器等可视化和自动化编程辅助工具）。Visual C++可用来开发各种类型、不同规模和复杂程度的应用程序，开发效率很高，生成的应用软件代码品质优良。这一切使得 Visual C++成为许多专业程序开发人员的首选。

然而，Visual C++一向有“难学”的名声，许多初学者视学习 Visual C++为畏途。究其原因，一方面是 Visual C++（包括 MFC 类库）的规模庞大，结构复杂，难于理出一条循序渐进的学习路线；另一方面是其 AppWizard 自动生成的程序专业化程度高，代码量大，结构复杂，以其为基础编写的例题程序难于为初学者理解和掌握。因此，目前的 Visual C++教科书多是为已有 C 语言或 C++语言编程基础的人准备的，起点较高。

本书是 Visual C++入门教科书，适用于本科类计算机及相关专业学生程序设计能力的培养。为了克服上述困难，使基础不高的初学者也能很快地掌握程序设计方法，我们在确定教学目标、设计教学模式、编写教程内容等方面进行了一系列革新探索，以现代教育理论为指导，多媒体教学手段为基础，提出了“精讲多练”的教学模式。使用“精讲多练”模式进行 Visual C++这类程序设计语言课程的教学，效果很好。

本教程的目标是使学生掌握使用 Visual C++设计应用程序的基本技能，了解面向对象的和结构化的程序设计方法，能够编写、调试和运行实用、规范、可读性好的 Visual C++程序。不像其他 Visual C++教材需要学习者具有一定的程序设计基础（如学过 C 语言或 C++语言），本书“从零开始”，不要求学生有程序设计方面的先修课程。但在学习本课程时，学生最好对计算机的使用有一定了解（如学习过“计算机文化基础”课程，了解 Windows 的使用，具有键盘操作和文件处理的基础）。

我们在设计本教程内容时，以面向对象和结构化程序设计思想贯穿全书，并以大量篇幅介绍了 Visual C++程序的调试技术和一些典型应用程序的设计思路，其中有些是作者在长期的编程和教学实践中摸索和总结出来的心得。

本教程共分 16 章，分别对应 16 个教学重点。这 16 个教学重点又可分为两组：前 8 章处理 C++的基本内容，包括控制结构、基本数据类型、表达式、函数、指针和引用，以及

类与对象的基本概念和封装、继承和多态性等面向对象程序设计的基础理论。在学习了这些内容之后，学生应能编写、调试和运行一般规模和难度的控制台类应用程序（如数值计算类程序），并对面向对象的和结构化的程序设计方法有所了解，为编写较大规模的应用程序打下基础。后 8 章处理 Windows 编程技术，包括消息传递机制、MFC 应用程序框架、设备环境、资源、文档/视图结构、对话框和控件等。在这一部分中，强调对基本概念的理解和掌握，以及在理解和掌握的基础上编写具有较复杂的窗口界面的 Windows 应用程序的能力。

为了便于教学，每章均按以下主题进行组织：

教学目标和学习要求 本书的特点是“精讲多练”，因此为教师和学生规定明确的教学和学习目标是非常重要的。

授课内容 是建议教师课堂讲授的内容。一般来说，授课内容是本章所有教学内容的“纲”，起着联系本章所有项目的作用。授课内容部分的份量按 2 学时组织。第一章的授课内容份量略轻，这是因为在第一章的授课时间中还应划分出部分时间用于介绍编辑、调试和运行应用程序项目的基本步骤（见第一章的“调试技术”）。

自学内容 “自学内容”和“授课内容”一起组成一章的基本教学内容。“自学内容”通常是“授课内容”的延伸和继续，由学生在课外时间自学。必须强调的是自学部分并非不重要，也不能省略。一般来说，教师应在授课时间内抽出 10～20 分钟对自学内容和调试技术略作导引，以便利学生自学。

调试技术 介绍 Developer Studio 集成开发环境的使用方法，以及如何调试、连接和运行 Visual C++应用程序项目。强调编程实践是本书的重要特色。第一章的调试技术中的部分内容可以在授课时间讲授，其他章的调试技术一般由学生自学，同时也可以作为学生上机的实验指导书，辅导教师在带学生上机时应对这些内容进行现场辅导。

程序设计举例 为了补充授课内容和自学内容部分的例题，我们设置了程序设计举例栏目。本栏目所有例题均与本章的授课、自学或调试技术等部分的内容密切相关，是学生学习和复习本章内容的重要参考资料。

上机练习题 每章均配有若干上机练习题目，供学生上机练习。“精讲多练”式教学方法的基本特点是上机时数较多，所以这部分的习题工作量较大，因此在上机时数不足的情况下可以酌情选做若干题目。

为了保证教学效果，在条件许可的情况下授课最好在多媒体教室进行。在这种情况下，每个教学单元（即每章）可使用连续的 4 课时，首先由教师讲解授课部分并对自学部分和调试技术等内容进行简短的指导（共 2 学时），然后学生即可在教师指导下上机练习（2 学时）。除此而外，如果能够提供一定数量的课外机时（如 20～30 小时）则更好。

近年来，我中心在计算机基础教育的理论和实践等方面进行了一系列探索和革新，其成果（“精讲多练”的教学模式是其中之一）荣获了 1997 年度国家级教学成果一等奖。这些成果都是在冯博琴教授的领导下完成的，本课程的建设也不例外。本教程的构思和编写得到了冯博琴教授的多方指导，并由他审核了书稿，在此向冯老师表示深深的谢意。在本书编写过程中，曾与李波、罗建军、卫颜俊、杨琦、吕军和张伟诸同事进行了多次交流，受益匪浅。以上同事还提供了一些有用的材料；杨琦同志为本书绘制了部分插图，

在此一并表示感谢。由于作者学识浅陋，编写时间仓促，书中错误在所难免。希望读者不吝指教。

编　者

2000 年 4 月于西安交通大学

计算机教学实验中心

第一章　Hello，C++!

[redacted]

本章介绍C++程序的基本结构以及在计算机上输入、编译、调试和运行C++程序的基本方法和步骤。

[redacted]

了解C++程序的基本特点，熟悉 Visual C++集成开发环境的基本使用方法。

[redacted]

1.1　软件开发与 C++语言

我们知道，使用计算机工作是通过相应的应用软件进行的，如用于文字处理的 Word，科学计算的 MATLAB，表格处理的 Excel 和各种数据库软件等，这些软件均由专业软件开发人员设计编写。一般来说，日常工作中遇到的任务多数都可以借助现成的应用软件完成。但有时仍需为具体问题自行开发相应的软件，特别是在工程应用领域中，解决各类项目中可能遇到大量的具体问题，使用通用的软件不仅效率低下，还可能无法完成任务。在这种情况下，自行编制具有针对性的应用软件可能是唯一的解决方法。

为计算机编写软件需要使用程序设计语言。目前可用的计算机语言很多，各有其特点。有些适用于开发数据库应用程序，有些适用于开发科学计算程序，有的简便易学，有的功能全面。在本课程中，我们介绍 Visual C++语言。

Visual C++语言支持面向对象的程序设计方法，适用于 Windows 应用程序的开发，可用于开发各类应用程序，功能十分强大，是目前最流行的程序设计语言之一。本教程以面向对象的程序设计方法为中心，本着简明、实用和循序渐进的原则，介绍使用 Visual C++开发应用程序的基本方法。

1.2 算法与程序

要编写用于解决应用问题的程序，首先必须确定解决问题的方案，也就是算法。例如，对于问题：给定两个正整数 p 和 q，如何求出其最大公因数？古希腊数学家欧几里德(Euclid)给出了一个著名的算法：

步骤 1：如果 $p<q$，交换 p 和 q；
步骤 2：求 p/q 的余数 r；
步骤 3：如果 $r=0$，则 q 就是所求的结果；
　　否则反复做如下工作：
　　令 $p=q$，$q=r$，重新计算 p 和 q 的余数 r，直到 $r=0$ 为止
　　则 q 就是原来的两正整数的最大公因数。

从原则上说，有了算法，人们就可以借助纸、笔和算盘等工具直接求解问题了。但如果问题比较复杂，计算步骤很多，则应通过编程，使用计算机解决。

例 1-1 使用欧几里德算法，编写一程序求解任意两正整数的最大公因数。

说明：根据 1.6.3 “用 Developer Studio 编写和调试简单 C++程序” 建立项目并调试和运行。

程序

```
// Example 1-1: 计算两个正整数的最大公因数
#include <iostream.h>
void main()
{   // 声明三个整型变量 p，q，r
    int  p，q，r;
    // 提示用户由键盘输入两个正整数
    cout << "Please input two integer numbers:" << endl;
    cin >> p >> q;
    // 如果 p < q，交换 p 和 q
    if((p<q)
    {   r = p;
        p = q;
        q = r;
    }
    // 计算 p 除 q 的余数 r
    r = p%q;
    // 只要 r 不等于 0，重复进行下列计算
    while(r != 0)
    {   p = q;
        q = r;
        r = p%q;
    }
```

```
    // 输出结果
    cout << "The maximum common divisor is " << q << "." << endl;
}
```

输入 输入两个整数，其间以空格分隔。如：

12 18

输出 The maximum common divisor is 6.

分析 可以看出，该程序的主体部分几乎与原算法完全相同，只是增加了一些 C++语言特有的内容。

程序的第 1 行是注解。注解以“//”开头，直到该行的末尾，用于说明或解释程序段的功能、变量的作用以及程序员认为应该向程序阅读者说明的任何内容。可以看到，在该程序中还有一些注解，用于说明程序的结构。在将 C ++程序编译成目标代码时所有的注解行都会被忽略掉，因此即使使用了很多注解也不会影响目标码的效率。恰当地应用注解可以使程序清晰易懂、便于调试，便于程序员之间的交流与协作，因此在自己编写的每个程序中都使用精心撰写的注解是一个良好的编程习惯。C++中还有一种注解格式，可参看 1.5.2。

第 2 行是编译预处理，有关内容将在 1.5.5 节介绍。

从第 3 行到最后一行是主函数。主函数是程序的主体部分，由其声明部分：

```
void main ()
```

和用一对花括号{}括起来的函数体构成。在函数体内，除了注解行以外，还有语句。C++的语句可分为声明语句和执行语句，声明语句用于声明程序中使用的变量、函数等的类型和参数，执行语句实际执行某功能。第 5 行就是一个类型声明语句：

```
int  p，q，r;
```

该语句声明在程序中使用了 3 个整型变量。关于类型和变量等概念，将在第三章中详细介绍。

一般的计算机程序总要包括三个基本内容：数据输入、计算和输出。上例中，程序的输入部分首先在计算机屏幕上显示一行提示信息(第 7 行)：

```
Please input two integer numbers:
```

然后等待使用者由键盘输入两个整数。用户输入的两个正整数由语句：

```
cin >> p>> q;
```

分别存入变量 p[①]和 q。

程序第 9 行及以下部分就是欧几里德算法的具体实现了。可以看出，C++程序类似自然语言，只是结构更加严谨。

程序中的最后一个语句是输出语句，输出语句将计算结果显示在计算机屏幕上。

1.3 输入、编译、调试和运行一个 C++程序

C++是一种编译语言，C++源程序需要经过编译、连接，生成可执行文件后方可运行。使

① 鉴于计算机语言程序的习惯和特点——所有的文字和符号都用正体，因此，为了保持上下文的一致和方便读者阅读，我们在对程序中的变量标识符进行说明时也用正体。

用C++开发一个应用程序大致要经过以下步骤：

1．首先要根据实际问题确定编程的思路，包括选用适当的数学模型。这方面的内容其实也是各应用学科的主要研究领域之一。

2．根据前述思路或数学模型编写程序。除了非常简单的问题可以直接写出相应的C++程序之外(在值得使用计算机解决的应用问题中，这种情况并不多)，一般都应该采用第二章中介绍的“逐步求精”的结构化程序设计方法来编程。

3．编辑源程序。首先应将源程序输入计算机，这项工作可以通过任何一种文本编辑器(如Visual C++集成环境中的文本编辑器)完成。输入的源程序一般以文件的形式存放在磁盘上(后缀为CPP)。

4．编译和连接。在设计高级语言(包括 C++)时充分考虑了人(程序员)的需要，源程序很接近人类的自然语言。因此，需要将源程序转换为计算机可直接执行的指令。就 C++而言，这项工作又分编译和连接两个步骤，编译阶段将源程序转换成目标文件(后缀为 OBJ)，连接阶段将目标文件连接成可执行文件(后缀为EXE)。

5．反复上机调试程序，直到改正了所有的编译错误和运行错误。在调试过程中应该精心选择典型数据进行试算，避免因调试数据不能反映实际数据的特征而引起计算偏差和运行错误。

6．运行。如果是自用程序，在调试通过以后即可使用实际数据运行程序，得到计算结果；如果是商品软件或受委托开发的软件，则运行由用户实施。

应该说明的是，如果要利用C++ 开发一个大型应用系统，例如管理信息系统、数据库应用系统、计算机辅助教学系统或者实时控制系统等，一般要比编写一个数值计算方面的应用程序复杂得多，上面介绍的开发步骤就显得过于简单了。这时要遵循软件工程的方法进行应用系统开发，例如采用面向对象的设计开发技术。这方面的内容已经超出本课程的范围，有兴趣的读者可以参看有关软件工程方面的资料。

1.4 C++语言的历史、特点、用途和发展

早期出现的高级程序设计语言大都面向某个具体的应用领域，如用于科学计算的FORTRAN 和 ALGOL 60、用于商业数据处理的 COBOL 以及用于人工智能编程的 LISP 等。但是对于最基本的系统软件，如操作系统和各种高级语言的编译程序来说，却缺乏一种合适的高级程序设计语言，因此通常还是直接使用机器指令代码或者汇编语言编写程序，效率很低且易于出错，根本无法适应规模越来越大的系统软件开发。究其原因，是当时的计算机硬件的基本性能，如处理速度、存储容量等相对来说还不很高，因此系统软件的主导设计思想是充分发挥系统硬件的能力，尽量提高软件的运行效率；而当时现有的程序设计语言与计算机硬件的实际情况脱节，经编译后生成的目标代码冗余量大、运行速度慢，也不适于用来编写作为应用软件基础的操作系统、编译程序等系统软件。

20 世纪 70 年代初，随着半导体集成电路技术的发展，计算机硬件的性能有了很大的提高，出现了每秒运算次数达百万次以上的大型计算机。为这种计算机配备的各种软件的规模也越来越大，结构越来越复杂，其生产组织和管理机制与极端注重代码运行效率的、以程序员个体劳动为基础的传统编程方式之间产生了尖锐的矛盾，终于导致了所谓的“软件危机”。其直接结果是出现了程序设计思想的革命，一改重视发挥机器效率的旧程序设计方法为重视人的因素，强调充分发挥程序员的效率，强调程序员之间合作交流的能力，即以提高程序可读性为主要目标的结构化程序设计方法。

C 语言正是在这种时代背景下诞生的。

1969 年，美国贝尔实验室的 Ken Thompson 为 DEC PDP-7 计算机设计了一个操作系统软件，这就是最早的 UNIX。接着，他又根据剑桥大学的 Martin Richards 设计的 BCPL 语言为 UNIX 设计了一种便于编写系统软件的语言，命名为 B。B 语言是一种无类型的语言，直接对机器字操作，这一点和后来的 C 语言有很大不同。作为系统软件编程语言的第一个应用，Ken Thompson 使用 B 语言重写了其自身的解释程序。

1972～1973 年间，同在贝尔实验室的 Denis Ritchie 改造了 B 语言，为其添加了数据类型的概念，并将原来的解释程序改写为可以直接生成机器代码的编译程序，然后将其命名为 C。1973 年，Ken Thompson 小组在 PDP-11 机上用 C 重新改写了 UNIX 的内核。与此同时，C 语言的编译程序也被移植到 IBM 360/370、Honeywell 11 以及 VAX-11/780 等多种计算机上，迅速成为应用最广泛的系统程序设计语言。

然而，C 语言也存在一些缺陷，例如类型检查机制相对较弱；缺少支持代码重用的语言结构等，造成用 C 语言开发大程序比较困难。

为了克服 C 语言存在的缺点，并保持 C 语言简洁、高效的特点，贝尔实验室的 Bjarne Stroustrup 博士及其同事开始对 C 语言进行改进和扩充，将“类”的概念引入了 C 语言，构成了最早的 C++语言(1983)。后来，Stroustrup 和他的同事们又为 C++引进了运算符重载、引用、虚函数等许多特性，并使之更加精炼，于 1989 年推出了 AT&T C++ 2.0 版。随后美国标准化协会以 AT&T C++ 2.0 为基础，制定了 ANSI C++标准。各软件商推出的 C++编译器都支持该标准，并有不同程度的拓展。

C++支持面向对象的程序设计方法(参看第七章)，特别适合于中型和大型的软件开发项目，从开发时间、费用到软件的可重用性、可扩充性、可维护性和可靠性等方面，C++均具有很大的优越性。同时，C++又是 C 语言的一个超集，这就使得许多 C 代码不经修改就可被 C++编译器编译通过。

早期的 C++编译器称为 Cfront，实际上只是一个预处理程序，负责将 C++程序转换为 C 程序，然后再由 C 语言编译程序继续编译。后来也出现了可以直接将 C++程序编译为目标代码的 C++编译程序。

随着 Windows 操作系统的出现，应用程序的外观和结构发生了巨大的变化，程序设计的环境也得到了很大的改善。为了适应 Windows 编程，各软件厂商纷纷推出了新型 C++编译器，Microsoft 公司的 Visual C++就是其中的佼佼者。Visual C++并不是一个单纯的编译器，而是一整套用于软件开发的集成环境(IDE)，其中包括了文本编辑、编译连接、调试、可视化界面设计和完备的在线帮助，甚至可直接通过 Internet 与 Microsoft 公司的站点连接，以得到技术支

持和产品更新升级。

1.5 C++程序的基本要素

1.5.1 标识符、关键词和标点符号

标识符是程序中变量、类型、函数和标号的名称。标识符由字母、数字和下划线“_”组成，第一个字符不能是数字。与 FORTRAN 和 BASIC 等程序设计语言不同，C++编译器把大写和小写字母当作不同的字符，这个特征称为“大小写敏感”。各种C++编译器对在标识符中最多可以使用多少个字符的规定各不相同，ANSI 标准规定编译器应识别标识符的前 6 个字符，而 Visual C++编译器允许在程序中使用长达 247 个字符的标识符！在标识符中恰当运用下划线，大、小写字母混用以及使用较长的名字都有助于提高程序的可读性。例如在一个人事管理软件中，用函数名 read_employee_file ()（读职工档案）、变量名 EmployeeCount（职工人数）等就要比用函数名 *f* ()、变量名 *n* 等清楚得多。

在 Visual C++中，有下列关键词：

asm, auto, bad_cast, bad_typed, bool, break, case, catch, char, class, const, const_cast, continue, default, delete, do, double, dynamic_cast, else, enum, except, extern, explicit, false, finally, float, for, friend, goto, if, inline, int, long, mutable, namespace, new, operator, private, protected, public, register, reinterpret_cast, return, short, signed, sizeof, static, static_cast, struct, switch, template, this, throw, try, type_info, typedef, typeid, union, unsigned, using, virtual, void, volatile, while, xalloc

关键词用于表示C++本身的特定成分，具有相应的语义。程序员在命名变量、数组和函数的名称时，不能使用这些标识符。

另外，C++还使用了下列 12 个标识符作为编译预处理的命令单词:

define, elif, else, endif, error, if, ifdef, ifndef, include, line, progma, undef

并赋予了特定含义。程序员在命名变量、数组和函数时也不要使用它们。

在 C++字符集中，标点和特殊字符有各种用途，从组织程序文本到定义编译器或编译的程序的执行功能。有些标点符号也是运算符(参见第三章基本数据类型与表达式中的有关内容)，编译器可从上下文确定它们的用途。C++的标点有：

[ ] ( ) { } * & , : = ; ... #

这些字符在C++中均具有特定含义。

1.5.2 注解

C++的注解有两种形式，一种以两个斜杠符“//”起头，直至行末，另一种是用斜线星号组合 “/*”和“*/”括起的任意文字(但不能再包含“/*”和“*/”，即注解不能嵌套)。后一种多用于注解篇幅多于一行的情况。注解可以出现在空白字符允许出现的任何地方，编译器把注解作为一个空白字符处理。

恰当使用注解可以使程序容易阅读。

1.5.3 源程序

一个 C++源程序由一个或多个源代码文件构成。C++源程序中包括命令、编译指示、声明、定义、注释和函数等内容。为了使程序的结构清晰，通常的做法是在一个源代码文件中放置变量、类型、宏和类等的声明(称为头文件，后缀为.H)，然后在另一个源代码文件中引用这些变量(称为源程序文件，后缀为.CPP)。采用这种方式编写的程序，很容易查找和修改各类声明。

1.5.4 编译预处理

将程序编译的过程分为预处理和正式编译两个步骤是C++的一大特点。在编译C++程序时，编译器中的预处理模块首先根据预处理命令对源程序进行适当的加工，然后才进行正式编译。

预处理命令均以符号“#”开头，且在一行中只能书写一条预处理命令(过长的预处理命令可以在使用续行标志“\”后续写在下一行上)，且结束时不能使用语句结束符(分号“;”)。

C++中有 3 种主要编译预处理命令：宏定义、文件包含和条件编译。下面先介绍前两种，条件编译将在第四章中介绍。

1. 宏定义

无参宏定义通常用来定义符号常数，其格式为:

```
#define 〈宏名〉 〈替换序列〉
```

其中<宏名>是一个标识符。为了和变量有所区别，习惯上在为无参宏定义起名时只使用大写字母。例如:

```
#define PI 3.14159
```

该宏定义命令为常量 3.14159 起了一个符号化的名字 PI。这样，在该宏定义命令之后的程序中，均可以使用符号常数 PI 表示数值 3.14159，例如:

```
s = PI*r*r;
```

使用符号常数的程序可读性好，易于阅读理解。另外，将程序中的常量用符号常数表示也有利于程序的调试和修改。例如，程序中分别使用了两个符号常数，它们的值碰巧相同:

```
#define MAX_NUMBER 80
#define LINE_LEN 80
```

以后如果发现需要将其中的一个修改为其他数值，则只须修改上面的宏定义即可。但是如果不使用宏定义，则需逐个查出程序中所有数值 80，一一判断是否需要修改，不但费事，而且容易出错。

应该说明的是，用宏命令定义的符号常数不是变量，像:

```
LINE_LEN = 100;
```

这样的用法是错误的。其实，宏定义命令的真正含义是要求编译程序在对源程序进行预处理时，将源程序中所有的符号名“MAX_NUMBER”和“LINE_LEN”(但不包括出现在注解与字符串中的“MAX_NUMBER”和“LINE_LEN”)分别替换为字符序列“80”。因此到了正式编译时，符号名“LINE_LEN”已经不存在了，语句“LINE_LEN = 100;”已经变为“80 = 100;”，由于赋值运算符的左边不是变量而是一个常数 80，显然这是一个错误的赋值表达式语句。

在定义宏时还可以加上参数，这就构成了带参数的宏:

#define 〈宏名〉(〈参数表〉) 〈带有参数的替换序列〉

例如:

```
#define max(a, b)    ((a)>(b)? (a): (b) )
```

带参数的宏的使用用法颇类似于函数。例如:

```
x = max(x, 10);
```

即如果变量 x 的值小于 10，则将常数 10 赋给 x，否则 x 的值保持不变。实际上，这个带参数的宏的确切含义为，通知编译程序中的预处理模块，在对应用程序进行处理时遇到形如 max() 的串时要进行转换，在转换时还要对参数进行代换处理。上述宏定义经过转换后变为:

```
x = ((x)>(10)? (x): (10));
```

在利用宏命令定义带参数的宏时，要注意宏名与括号之间不能有空格，且所有的参数均应出现于右边的替换序列中。

虽然带参数的宏很像函数，但这两者之间的区别是很明显的。编译程序在处理带参数的宏时，并不是用实参的值进行代入计算，而只是简单地将替换序列中的参数用相应的实参原样替换(按参数书写位置，从左到右一一对应)。因此，在书写带参数的宏时，要防止由于使用表达式参数带来的错误。例如定义了一个用于计算圆面积的宏:

```
#define circle_area(r)    r*r*3.14159
```

在计算:

```
s = circle_area(x+16)
```

就会出问题。将参数 x+16 代入上述宏定义中，有:

```
s = x+16*x+16*3.14159
```

就会发现运算的顺序显然有错误。如果重新定义这个宏:

```
#define circle_area(r)    ((r) * (r) *3.14159)
```

就没有问题了。总而言之，在定义带参数的宏时应将每个参数和整个替换序列都用括号括起来，以防止可能的计算错误。

取消宏定义命令#undef 用于撤消对一个符号名的定义。例如:

```
#define DEBUG  1
    // 在本程序段落中定义了一个值为 1 的符号常数 DEBUG
#undef DEBUG
```

如果没有#undef 命令，则宏定义起作用的范围为自#define 命令起至该源程序文件末尾。

2. 文件包含

编译预处理命令中的文件包含命令的功能，是将另一段 C++源程序文件嵌入到正在进行预处理的源程序中的相应位置上。文件包含命令的格式为:

```
#include ,〈文件名〉
```

或者

```
#include "文件名"
```

其中“文件名”是指被嵌入的 C++源程序文件的文件名，必须用双引号或者尖括号(即一个小于号和一个大于号)括起来。通过使用不同的括号可以通知预处理程序在查找嵌入文件时采用

不同的策略。如果使用了尖括号，那么预处理程序在系统规定的目录(通常是在系统的 include 子目录)中查找该文件。如果使用双引号，那么编译预处理程序首先在当前目录中查找嵌入文件，如果找不到则再去由操作系统的 path 命令所设置的各个目录中去查找。如果仍然没有查找到，最后再去上述规定的目录(include 子目录)中查找。

原来的源程序文件和用文件包含命令嵌入的源程序文件在逻辑上被看成是同一个文件，经过编译后生成一个目标文件，如图 1.1 所示。

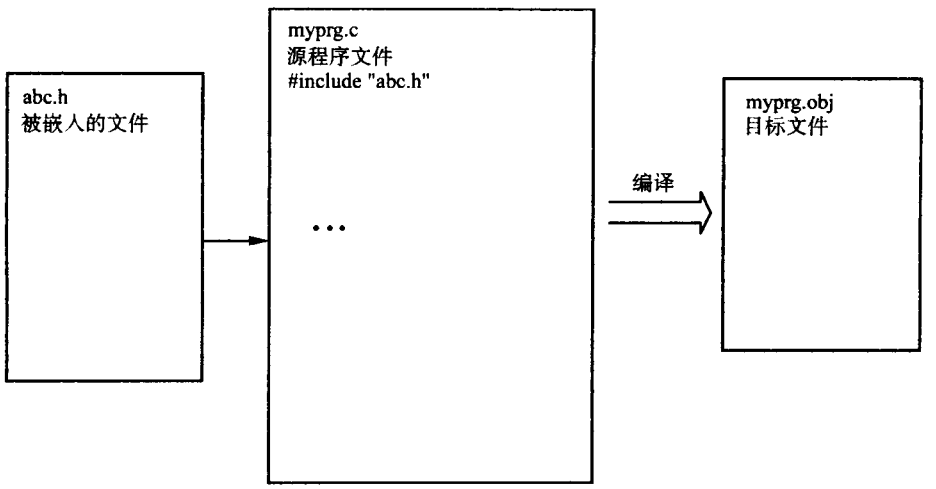

图 1.1 文件包含

1.5.5 输入与输出

程序通常会要求用户提供一些信息(如数据)，这一过程称为程序的输入。程序通常总是要向用户发出一些信息(如计算结果等)，这一过程称为程序的输出。C++程序的输入操作可由 cin 对象来完成，而输出操作则可由 cout 对象来完成。

cin 的基本用法为：

cin >>V_1>>V_2>>…>>V_n;

其中“>>”称做提取运算符，V_1，V_2，…，V_n 都是变量。这个语句的意思是，程序暂时中止执行，等待用户从键盘上输入数据。用户输入了所有的数据后，应以回车键表示输入结束，程序将用户键入的数据存入各变量中，并继续执行下面的语句。例如，例 1-1 中的第 8 行：

cin >> p >> q;

就是要求用户分别为变量 p 和 q 各输入一个数据。在输入时，应注意用空格或 tab 键将所输入的数据分隔开。

应说明的是，当用户响应 cin 的要求输入数据时，必须注意所输入数据的类型(C++的数据类型将在第三章中介绍)应与接受该数据之变量的类型相匹配，否则输入操作将会失败或者得到的将是一个错误的数据。

cout 的基本用法具有如下的一般形式：

cout << E_1 << E_2 << …… << E_m;

其中“<< ”称做插入运算符，E_1，E_2，…，E_m 都是表达式(第三章中介绍)。这个语句的意思是，将各表达式的值输出(显示)到屏幕上当前光标位置处。在输出时，要注意恰当使用字符串和新行符 endl，提高输出信息的可读性。

调试技术

1.6 Visual C++的集成开发环境

Visual C++软件包包含了许多独立的组件，如编辑器、编译器、连接器、实用程序生成器、调试器，以及各种各样为开发 Windows 环境下的 C/C++程序而设计的工具，其中最重要的是一个名为 Developer Studio 的集成开发环境。Developer Studio 把所有的 Visual C++工具结合在一起，集成为一个由窗口、对话框、菜单、工具栏、快捷键及宏组成的和谐系统，通过该集成环境，程序员可以观察和控制整个开发进程。

图 1.2 显示了一个典型的 Developer Studio 主窗口。

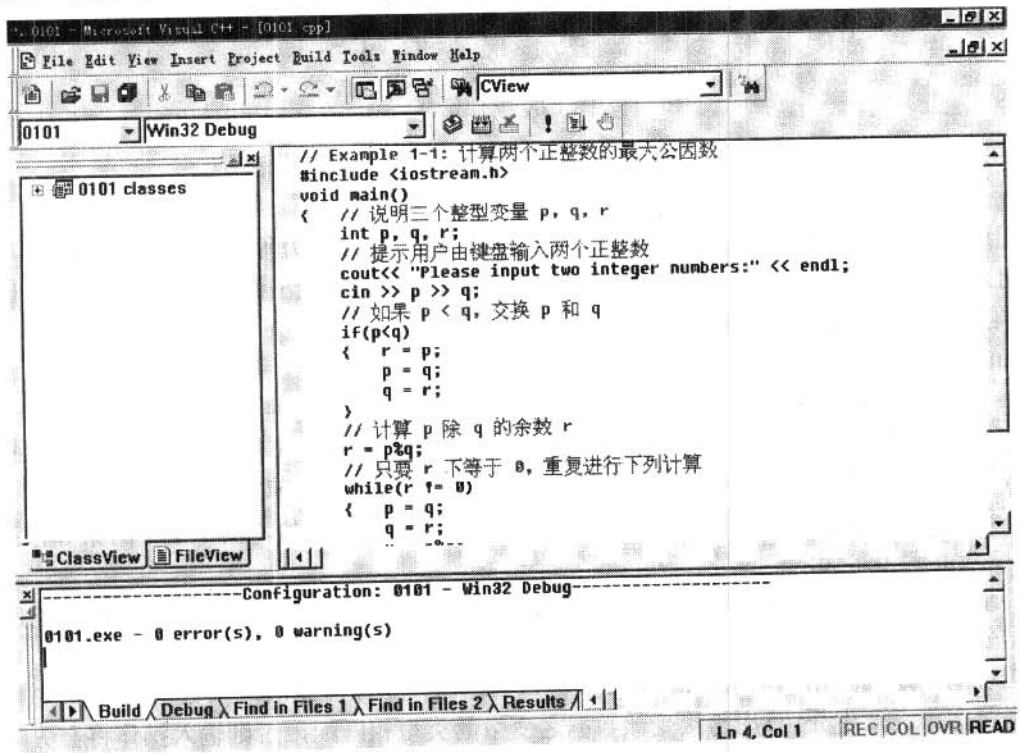

图 1.2 典型的 Developer Studio 窗口

从图 1.2 中可以看出，Developer Studio 主窗口可以分为几个部分：窗口顶部是菜单和工

具栏，左面的一个子窗口是工作区窗口，工作区窗口的右面是编辑子窗口。最下面是输出子窗口。值得注意的是，上述各种部件，包括子窗口、菜单栏和工具栏的位置不是一成不变的，可以根据个人的喜好重新安排。

1.6.1 菜单和工具栏

Developer Studio 中有一个 Menu Bar(菜单栏，通常位于开发环境窗口的顶部)，其中的菜单项有 File (文件处理)、Edit(编辑功能)、View(查看)、Insert(插入)、Project(项目管理)、Build(编译)、Tools(工具)、Window(窗口)和 Help(帮助)等，分别对应一个下拉子菜单。

除菜单栏外，开发环境中还有几个工具栏，一般均放在开发环境的顶部，菜单栏的下方，如 Standard(标准工具栏，用于文件管理、编辑和查看等)，Wizard Bar(向导工具栏)和 Build MiniBar(建立工具栏，用于编译、连接等)。工具栏上有常用命令的图标。一般来说，工具栏上的命令在菜单中均有对应选项，但工具栏使用更方便，只要用鼠标左键点击工具栏中的相应图标即可调用相应的功能。

开发环境的各种菜单栏和工具栏均为停靠式，可以用鼠标拖动改变它们的位置，也可使用菜单选项 Tools/Customize...(表示菜单栏的 Tools 子菜单中的 Customize...选项，下同)中的 Toolbars 选项打开或关闭相应的工具栏，或修改工具栏的内容。

除此之外，Developer Studio 的所有部分几乎都可响应鼠标右键单击而弹出一个上下文相关菜单。甚至当 Developer Studio 没有打开窗口时，在空白区右击鼠标也会弹出一个菜单，其中含有使窗口可见和调整工具栏是否可见的命令。在工具栏上除标题栏外的任何地方右击鼠标，同样可以弹出菜单。在使用集成环境工作时试一试鼠标右键，还会发现许多其他的快捷方式。例如，通过在工具栏或菜单栏上不是按钮或菜单名的地方单击并保持鼠标按键被按下，就可以把它们拖动到屏幕上新的地方。

1.6.2 Developer Studio 窗口

除了各种对话框外，Developer Studio 显示两种类型的窗口，即文档窗口和停靠窗口。文档窗口是一般的带边框子窗口，其中含有源代码文本或图形文档。窗口子菜单中列出了在屏幕上是以平铺方式还是以层叠方式显示文档窗口的命令。所有其他的 Developer Studio 窗口，包括工具栏和菜单栏，都是停靠式窗口。开发环境有两个主要的停靠窗口——Workspace(工作区)窗口和 Output(输出)窗口，另外还有一个 Debugger(调试器)停靠窗口，只在调试过程中显示。

停靠窗口可以固定在 Developer Studio 用户区的顶端、底端或侧面，或者浮动在屏幕上任何地方。停靠窗口，不论是浮动着的或是固定着的，总是出现在文档窗口的上面。这样，就保证了当焦点从一个窗口移到另一个时，浮动的工具栏一直都是可见的。但这也意味着，文档窗口偶尔会看起来像消失了似的。例如，如果你正在文本编辑器中编辑源代码，此时打开一个占据整个 Developer Studio 用户区的停靠窗口，源代码文档就会消失，它隐藏在新窗口之下。解决方法是要么关了覆盖的窗口，要么把它拖到不挡眼的地方去。

1.6.3 用 Developer Studio 编写和调试简单 C++程序

下面以例 1-1 为例，说明使用 Visual 集成开发环境编写和调试简单 C++程序的步骤。