高等学校21世纪计算机教材

C++ 程序设计

朱润酥　李令雄　编著

冶金工业出版社

高等学校21世纪计算机教材

C++程序设计

朱润酥　李令雄　编著

北　京

冶金工业出版社

2003

内 容 简 介

本书以循序渐进，深入浅出的方式引导读者步入面向对象程序设计的大门，本书从基本数据类型与基本控制结构开始，逐步过渡到数组函数、编译预处理类与对象、枚举类型、结构类型和类型别名、继承和派生类、友元与运算符重载、流类体系与文件操作等复杂机制和C++的高级知识。本书内容丰富、例题典型，即使你是一个从未接触过编程语言的新手，通过认真学习本书，也一定可以熟练地掌握C++这门语言。

本书强调理论与实践相结合，因此提供了大量完整的程序代码，要求读者上机编译通过。通过上机编译，可以杜绝一些小错误，提高自己的编程水平。

本书内容丰富、条理清晰、文笔流畅，并且每一章都附带精选的练习题和上机题，适合作为高等院校相关专业的教材和C++语言的培训用书，也可作为对计算机编程语言感兴趣的读者的自学教材。同时，本书也可供广大软件开发人员参考。

图书在版编目（CIP）数据

C++程序设计 / 朱润酥等编著. —北京：冶金工业出版社，2003.9
ISBN 7-5024-3344-9

I. C… II. 朱… III. C语言—程序设计
IV. TP312

中国版本图书馆CIP数据核字（2003）第072323号

出版人 曹胜利（北京沙滩嵩祝院北巷39号，邮编100009）
责任编辑 程志宏
广东惠阳印刷厂印刷；冶金工业出版社发行；各地新华书店经销
2003年10月第1版，2003年10月第1次印刷
787mm×1092mm 1/16； 23.5印张； 574千字； 368页； 1-3500册
30.00元
冶金工业出版社发行部 电话：（010）64044283 传真：（010）64027893
冶金书店 地址：北京东四西大街46号（100711） 电话：（010）65289081
（本社图书如有印装质量问题，本社发行部负责退换）

前　言

一、编写意图

编写本书的目的是为了给程序设计初学者提供一本清晰的入门教材。本书以面向对象程序设计为核心，并选用C++语言作为工具。

对于计算机初学者而言，学好程序设计的入门课程对于软件工程、编译原理、人工智能、自动化等后继课程有很大的帮助。

程序设计课程包括相辅相成的两个方面：程序设计方法和程序设计语言。自从1946年第一台电子计算机诞生以来，程序设计方法和程序设计语言不断在变化、在改进。20世纪70年代，结构化程序设计占主导地位；而20世纪80年代起，“面向对象”成为程序设计最热门、最实用的开发手段，并持续到现在。

二、有关本书

本书将从C++最底层的语法、语句入手，由实例作为主线，通过一步一步演示具体程序的方式，向读者介绍C++中的一些基本编程特性。当读者完成本书的学习和试验后，就能够掌握C++这门语言，为以后计算机知识的学习打下坚实的基础。

编写本书，是在C++程序设计上从繁到简的一次尝试，编者力图用比较简单的例子来介绍Visual C++的使用。通过本书，读者可以学习到C++的基本语法以及Visual C++的使用。而更重要的是，通过本书，读者可以领悟到学习C++的方法，能通过自学就能进一步掌握C++。

三、结构安排

本书共分12章，其结构如下：

第1章：C++概述。本章先介绍了C++的起源和特点，然后通过hello world程序和C++上机操作的介绍，让读者对C++语言有初步认识。

第2章：数据类型和表达式。本章介绍了常量和变量的定义与使用、关键字和标识符、数据类型、运算符和表达式以及简单的输入/输出等知识。

第3章：程序结构和流程控制语句。本章向读者介绍了C++的3种基本结构：顺序结构、分支结构和循环结构。接着讲述了分支语句（if语句，switch语句）、循环语句（while语句，do...while语句，for语句）、控制执行顺序的语句（break，continue，goto）的使用。

第4章：数组。本章分别介绍了一维数组、二维数组、字符数组的定义和使用以及字符串等知识。

第5章：函数。本章通过对函数的定义和调用的介绍，让读者进一步了解C++的程序组成的知识。

第6章：编译预处理。本章介绍了宏、文件包含和条件编译。

第7章：指针。指针是C++的重点和难点，本章通过对指针的概念、指针变量、数组与指针、指针数组和指向指针的指针以及new和delete运算符的介绍，使读者能够更好的掌握和使用指针。

第8章：类和对象。本章介绍了类与对象的基本概念和作用域、构造函数和析构函数、

内联函数和外联函数、对象指针、this 指针、对象数组和对象的生存期等内容。

第 9 章：枚举类型、结构类型和类型别名。本章主要介绍了枚举类型、结构类型的定义和使用以及类型别名的使用。

第 10 章：继承和派生类。本章主要介绍了继承与派生、多重继承、虚基类和静态成员。

第 11 章：友元与运算符重载。本章主要介绍了友元的概念、运算符重载、多态性与虚函数等 C++的高级内容。

第 12 章：流类体系与文件操作。本章主要介绍了 C++的流类体系以及文件操作的基本知识。

第 13 章：综合应用实例。本章通过几个综合应用实例，介绍了 C++程序设计的方法与思路，以巩固读者对 C++知识的全面理解和运用。

此外，书后还给出了各章习题的参考答案，以供读者参考。

四、本书特点

本书是编者程序设计经验的精华，选材从实际出发，语言通俗易懂，并兼顾了深度和广度。本书中所使用的实例均在计算机上调试通过，希望读者进一步上机去体会 C++程序设计技巧和精髓。在引入关键概念（例如引入类的概念）时，本书首先会介绍现实生活中的例子，再转入 C++中的概念，这种生动的叙述方式有助于读者对概念的深刻理解。学完课本内容后，本书丰富的课后习题将是读者提高自己的程序设计水平的有力保障。

五、编写方法

本书通过一个个实例向读者介绍 C++程序设计的知识，每一部分内容都先介绍基础知识，再通过例子来应用所介绍的知识。在每个例子中都会给出注释，并且详细讲解关键的代码。本书的实例全部使用 Visual C++ 6.0 编写，全部程序皆由编者上机通过。

六、使用范围

本书内容丰富、条理清晰、文笔流畅，并且每一章都附带精选的练习题和上机题，适合作为高等院校相关专业的教材和 C++语言的培训用书，也可作为对计算机编程语言感兴趣的读者的自学教材。同时，本书也可供广大软件开发人员参考。

本书由朱润酥、李令雄合力编写，并由朱润酥审校全部书稿。虽力图做好，但由于作者水平有限，不足之处在所难免，希望广大读者不吝赐教，以便使本书质量进一步得到提高。

读者在阅读本书的过程中，如果有好的意见或建议，可以发 E-mail 到 service@cnbook.net，也可登录网站：http://www.cnbook.net，在该网站的相关论坛进行讨论。

编　者

2003 年 8 月

目　录

第 1 章　C++概述

C++语言是一门目前使用最广泛的面向对象程序设计语言。随着时代的发展，结构化设计语言已经被面向对象设计语言所取代，面向对象设计语言具有封装、数据隐蔽、继承和多态 4 大特点，在本章中，将初步学习 C++这门语言。本章主要内容如下：

- C++的起源
- C++的特点
- 简单的 C++程序
- C++上机操作

1.1　C++的起源

随着计算机成本的普遍降低，程序员工作时间的成本往往超过了用于工作的计算机的成本，于是编写风格良好，易于维护的程序受到了普遍欢迎。程序设计方法主要经历了过程化程序设计（Procedural programming）、结构化程序设计（Structrued programming）、面向对象的程序设计（Object-oriented programming）3 个阶段。过程化的程序设计是在一个数据集合上进行的一系列操作。而结构化程序设计是系统地组织这些过程和能够处理大量数据的有效手段。

采用过程化程序设计的程序员发现：每一种相对于老问题的新方法都带来额外的开销。而采用结构化程序设计，当数据量增大时，数据与处理这些数据的方法之间的分离使得程序变得难以理解。

面向对象的程序设计的实质是把数据和处理这些数据的过程合并成为一个单独的“对象”，一个具有确定特性的自完备的实体。面向对象程序设计作为 90 年代软件设计领域的主导技术，已经受到越来越多的软件人员的重视。

C++是面向对象程序设计语言的代表，这也是我们要学习 C++的主要原因。需要说明的是，各种程序设计语言对于程序设计方法的支持程度是不同的，C++则是完全支持面向对象程序设计的语言。

这是因为 C++的出现主要归因于 C 语言，C 语言在 C++中作为子集保留下来。C 语言是在 1970 年由 Ken Thompson 设计的 B 语言发展而来的。

1980 年贝尔实验室的 Bjarne Stroustrup 为了解决（处理）仿真课题，编写了称为“带类的 C 语言”，1983 年将 C++ 名字定下来，并向研究小组之外发表，C++的名字强调了从 C 语言的演化特性，“++”是 C 的增量运算符，之所以没有叫 D 语言，是因为它是 C 语言的扩充，增加了面向对象的程序设计特征，于是，C++成为面向对象程序设计的主流语言。

目前，常用的 C++编译程序有 Borland C++、MC C++、Visual C++（本书采用此编译器）、GNU C++等。随着 Internet 的高速发展，网络应用软件需求越来越大，Internet 服务器大多使用 Unix 操作系统，而 Unix 操作系统与 C++有着天然的联系。而 Linux 操作系统的普及，使得在 Unix 操作系统开发软件不再是大型机专业人员特有的工具。因此，对于软件设计人员来说，使用 C++就意味着希望和成功。

1.2 C++的特点

1.2.1 对面向对象开发的支持

C++完全支持面向对象的程序设计，它包含了面向对象的开发所具有的 4 个重要特性：封装、数据隐蔽、继承和多态。

1. 封装和数据隐蔽

当一台计算机的光驱损坏后，我们并不需要用原料重新做一个，只需要按照这个组件的接口规范，找一个成品代替它就行了，即是说，买一个新的光驱就行了。这里，我们所关心的硬盘只是一把“钥匙”，只要符合“锁”要求的规范就可以了，我们并不需要明白它如何“开锁”的。也就是说，各个组件对我们而言是一个自包含的实体，是封装的。这种不需要知道封装单元如何工作，只需要直接使用的思想称为数据隐藏。

C++通过生成被称为类的用户定义的类型来支持封装和数据隐蔽。一个严格定义的类一旦生成后，就成为一个封装的实体，并可以作为一个整体被使用。而类内部的行为被隐藏起来，用户只需知道如何使用这种严格定义的类，而不需要知道它是如何工作的。

2. 继承

儿子拥有父亲的血型、爱好、特长，就称为儿子继承了父亲。例如小仲马、大仲马都是大作家，小仲马则是继承了他的父亲大仲马的才华。

C++中，继承指通过说明一个扩展已有类型的新类型，这个新的子类是从已有类型派生出来的，也被称为派生类型。

3. 多态

“萝卜白菜，各有所爱”，现实中，我们可以选择自己所需要的东西，这反映到 C++中，就称为多态。

多态性是指相同的消息为不同的对象接收到后，可能导致不同的动作。也就是说，一个多态性对象可以有多种形式。C++ 通过被称为函数多态与类多态的特性来支持这种不同类型有各自响应的消息。

此外，C++对面向对象程序设计中的其他一些主要特征的支持。

（1）C++类中包含私有、公有和保护成员。

C++类中可定义三种不同访问控制权限的成员。一种是私有（Private）成员，只有在类中说明的函数才能访问该类的私有成员，而在该类外的函数不可以访问私有成员；另一种是公有（Public）成员，类外面也可访问公有成员，成为该类的接口；还有一种是保护（Protected）成员，这种成员只有该类的派生类可以访问，其余的在这个类外不能访问。

（2）C++中通过发送消息来处理对象。

C++中是通过向对象发送消息来处理对象的，每个对象根据所接收到的消息的性质来决定需要采取的行动，以响应这些消息。响应这些消息是一系列的方法，方法是在类中使用函数来定义的，使用一种类似于函数调用的机制把消息发送到一个对象上。

（3）C++中允许友元破坏封装性。

类中的私有成员一般是不允许该类外面的任何函数访问的，但是友元可打破这条禁令，它可以访问该类的私有成员（包含数据成员和成员函数）。友元可以是在类外定义的函数，也

可以是在类外定义的整个类，前者称友元函数，后者称为友元类。友元打破了类的封装性，它是C++另一个面向对象的重要特性。

（4）C++允许函数名和运算符重载。

C++支持多态性，C++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，这就称标识符或运算符的重载，用户可以根据需要定义标识符重载或运算符重载。

（5）C++支持继承性。

C++中可以允许单继承和多继承。一个类可以根据需要生成派生类。派生类继承了基类的所有方法，另外派生类自身还可以定义所需要的不包含在父类中的新方法。一个子类的每个对象包含有从父类那里继承来的数据成员以及自己所特有的数据成员。

（6）C++支持动态联编。

C++中可以定义虚函数，通过定义虚函数来支持动态联编。

1.2.2 C++与C的比较

与C相比，C++又有哪些相同，哪些不同呢？

C++的一个设计目标就是通过支持面向对象的程序设计方法来增强C的功能。请注意“增强”一词的确切含义，C++语言完全设计为与C语言向后兼容：每一个合法的C语言程序就是一个合法的C++程序（实际上也有一些例外，但它们都不重要）。因此C++具有C的所有好的以及不好的设计特征。

与C相类似，C++是面向标识符的，并且是区分大小写字母的。编译程序将源代码拆分为单词组件而不理会它们在源代码中的位置，因此不必限定源代码中各个元素的书写位置（例如，在FORTRAN或COBOL中就要限定）。C++编译程序会忽略标记之间的所有空格，因此，程序员可以利用空格来调整源代码的格式，以提高可读性。区分大小写有助于避免命名冲突，但是如果程序员（或者维护人员）不重视（或者没有时间重视）诸如大小写等细节问题的话，就会出现错误。

与C相似，C++只有很少的数值型的内部数据类型，比其他的现代语言要少。为了增强对破坏的免疫力，其中一些基本的数据类型在不同的机器上具有不同的取值范围。程序员可以使用所谓的修饰符，以便将变量的合法取值范围改为某个机器可接受的范围，这样做会使事情变得更加复杂。

为了弥补内部数据类型的不足，C++支持将数据类型聚集为复合类型，包括数组、结构、联合以及枚举等类型。数据聚集还可以进一步合并为其他的聚集。这一特征也是从C中借用过来的。

C++支持一组标准的流控制结构，包括语句的顺序执行以及函数调用，语句的重复执行以及语句块（for，while，do循环）、判定语句（if，switch结构）、跳转（break，continue 还有go to语句）。这组控制语句与C的一样，但在使用for循环时有一些不同。

与C类似，C++语言是一种块结构语言，未命名的代码块可以嵌入到任意深度，在内层块中定义的变量在外层块中是不可见的。这使得编写内层块的程序员可以对局部变量任意地命名，而不必担心它会与编写外层块的程序员所定义的名字相冲突（以及需要协调）。

另一方面，一个C（以及C++）函数（即一个命名块）不能嵌套在其他函数内，因此函数的名字在程序中必须是惟一的，这是一个严重的限制。它增大了在开发过程中程序员之间的

协调压力并使得维护变得更加困难。C++通过引入类作用域，部分地更正了此问题。类方法（即在类中定义的函数）只要求在类中惟一，而不必在程序中惟一。

C++函数可以像C函数一样递归地调用。传统的语言不支持递归调用，因为递归算法代表了所有算法的一小部分。单单使用递归会浪费执行期间的时间和空间。然而，一些使用递归显得特别有效的算法确实得益于递归，因此，在现代程序设计语言中，递归是一个标准的特征（脚本语言除外）。

与C完全一样，C++的函数可以放在一个文件或多个源文件中。这些文件可以独立地进行编译和调试，这就使得不同的程序员可以独立地完成项目的不同部分。已编译的目标文件可以稍后连接起来，以生成可执行的目标文件。这对于实现大型项目的劳动力分工十分重要。

与C很类似，C++是一个强类型语言。例如，在表达式中或者向函数传递参数时，使用一个并非所期望的类型值是一个错误。目前，这是设计编程语言时普遍遵守的一个原则。很多要在运行时才表现出来的数据类型错误可以在编译时检查出来，于是节省了测试和调试的时间。

比C更进一步的是，C++又是一个弱类型语言（是的，它既是强类型语言，又是弱类型语言）。在表达式和函数调用中可以自动地进行数值类型之间的转换。这与现代语言设计有很大的差异，它容易使一些错误无法在编译时发现。另外，C++支持相关类之间的转换。一方面，这使得我们可以使用“多态”这一很好的编程技术；另一方面，这一特征阻止了编译程序发现因疏忽而导致的错误。

C++以三种目的继承了C对指针使用的支持：

（1）从调用函数向被调用函数传递参数。

（2）从计算机的堆中动态分配内存（以实现动态数据结构）。

（3）操纵数组及数组分量。所有使用指针的技术都容易出现错误，这些错误最难检查、局部化和更正。

与C很类似，C++是为了提高效率而提出的数组越界既不在编译时检查，也不在运行时检查。要由程序员来维护程序的完整性以及避免因非法使用下标而导致的内存破坏。这通常是C / C++程序错误的来源。

与C相类似，C++是为了编写简洁而紧凑的代码而设计的。它对标点符号以及运算符赋予了特定的含义，这些符号包括星号、加号、等号、花括号、中括号以及逗号等，这些符号在一个C++程序中代表了多种含义。它们的含义由所处的上下文决定，这使得学习和使用C++语言比学习和使用其他语言更难。

C++在C的基础上增加了一些新的特征。最重要的特征就是支持对象。C++将C的结构扩充为类，它们将数据和函数绑定为一个代码单元。类通过将数据表示限制在其边界内，使得数据成员在类的外部不可见，从而实现了信息隐藏。类通过提供由客户代码调用的访问函数（方法）来支持封装，使用类作用域来避免C++程序中的命名冲突。

类提供了用于设计的层次方法，使得高层次的类可以重用低层次的类。类复合以及类继承使得程序员可以实现现实世界中复杂的模型并且更容易操纵程序的组件。

C++还有许多其他帮助设计人员通过代码本身、而不必通过注释就可以表达设计意图的特征。然而，与C类似，C++语言是为一个有经验的程序员而设计的。编译程序不会试图去猜测程序员的意图，因为它假定程序员很清楚自己在做什么（但实际上，对于正在做什么，我

们并非总是很清楚的，不是吗？）。如果程序员不注意，一个C++程序就会相当复杂，且令读者望而生畏，因而难以修改和维护。类型转换、指针操纵、数组处理以及参数传递等都是C++程序中常见错误的来源。

1.3 简单的C++程序

下面我们来看一个最简单的C++程序——Hello World程序。

【程序清单1-1】

```
#include<iostream.h>                    //包含iostream.h头文件
void main()                             //主函数main()
{
    cout<<"Hello,world!\n";             //设计输出
}
```

与其他现代的语言一样，C++允许编写其形式与人的阅读习惯相同的源代码。C++编译程序将源代码转换为机器可以识别的目标代码。在程序执行时，机器语言的指令逐条执行，并产生结果。大多数计算都是对存储在计算机内存里的数值进行处理。在通常的用法中，可以将计算机内存看作是由含有数值的地址单元组成的数组。存储在这些地址单元的数值不能引用这些地址。这些地址只能用数字地址（在目标代码中）或者用符号名称（在源代码中）来引用。例如，一个C++程序中，可能含有以下的语句：

```
z = y + 1;
```

它指示计算机取出存储在名为y的地址单元中的数值，把该值加1（不改变地址单元y的内容），并将结果存放到标识为z的地址单元中。名为y和z的地址单元的真实地址在可执行代码中指定，而不是在源代码中指定。程序员为这些单元命名，但并不关心编译程序为每个名称分配了什么内存地址。而在实际的内存中，为整数、浮点数以及字符（文本）分配不同的位数以及字节数，并且它们的位模式在运行时有不同的处理方式。为了正确地生成可执行代码，编译程序必须理解程序员的意图。这就是为何在执行语句z = y + 1之前，必须告诉编译程序y和z是内存某地址单元的名称（而不是其他的含义，例如函数），以及这些名称对应的内存单元中所存放的数值属于double类型（C++中一种带有小数部分的数值类型）。

因此，程序员编写的大多数源代码要么定义了程序操纵的对象（这里是x, y, z，其他的是通过# include指令和# define指令指定的数据），要么表达了要使用这些对象执行什么操作（这里有加法、赋值以及向函数传递参数）。

C++程序的源代码可以是一个由文本编辑程序生成的普通文本文件，这些编辑程序可以是Unix上的Emacs或Vi、VMS上的Edt、PC或者Mac上的集成开发环境（Integrated Development Environment, IDE）。这里我们将它作为一个文件保存在硬盘上。通常，可以为源代码给定一个合适的名字，但对所用的文件扩展名是有限制的。根据编译程序的规定，源文件必须以文件扩展名.cc、.cpp或者.cxx来保存。使用其他的扩展名也可以，但不会太方便。如果使用标准的扩展名，那么只需指定源文件的名字，开发工具就会自动为文件加上扩展名。允许使用非标准的扩展名，但必须显式地指定它们。一个源文件可以定义若干个函数（第一个C++程序只有一个函数，其名字叫做main）。一个程序可以由若干个源文件组成（上述程序只有一个文件）。每个源文件必须被编译为对应的目标文件。大多数环境都要求在程序执行以前，要先将已编译的程序链接起来。

屏幕输出结果为图 1-1 所示。

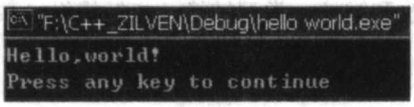

图 1-1

这个输出结果是由 Microsoft Visual C++编译程序的专业版 6.0 版本对程序编译执行后产生的。这是 Microsoft Development Studio 的一个组成部分，它在同一个软件包中集成了若干个开发工具。程序由 Development Studio 调用。输出的最后一行是由编译程序而不是由该程序产生的。否则，将会在程序结束时马上清除屏幕的信息，于是用户就不能够观察到程序的输出结果了。Microsoft 编译程序的早期版本没有增加这一信息，但也不会马上清除屏幕的信息,这项工作交给用户来完成。该程序也可以作为一个单独的应用程序在 DOS 提示符下运行，这时，就不会出现最后一行的信息。不同机器上的数值输出结果也有可能不同，这取决于对输出的数字位数的缺省设置。C++允许程序员显式指定不依赖于编译程序设置的输出格式，但这相当复杂，故不在此进行介绍。以后大家将会看到这样做的一些例子。

一个 C++程序给出了在所有的 C++程序中可能包含的以下成分：

（1）预处理程序。

（2）注释。

（3）关键字和标识符。

（4）语句。

（5）函数和函数调用。

下面我们逐个进行说明：

（1）预处理程序。#include 表示包含其他库，例如本程序包含的是 iostream 库，此库提供了输入输出库（library）函数，所以一般程序都需把 iostream 库包含进来。

在大多数其他语言中，编写的源文件就是编译程序在编译时所面对的源文件。但在 C++中却并不如此。在将源代码转换为可执行程序的过程中，编译程序并不是第一个工具。处理源代码的第一个工具是预处理程序。它是什么呢？其实，它是 C++从 C 那里继承下来的一个有趣的革新。它的目标是减少程序员在开发程序时编写的源代码的数量（或者在调试以及维护时阅读的源代码的数量）。

预处理程序处理源文件，并将处理结果传送给编译程序进行编译。预处理程序会忽略大多数的程序语句，并不做修改地将它们传送给编译程序。预处理程序只关注预处理程序指令（以及与它们相关的语句）。

预处理程序指令以'#'开头并占用一整行。不能在一个源文件行上写多于一条的指令。如果一行写不完一条指令，可以继续写在下一行，但前一行的末尾必须以一个特殊的转义符号'\'结束。符号'#'必须是一行的起始字符。

什么是 C++源代码的自由风格呢？正如前面所述，可以使用一种自己（而不是编译程序）认为合适的格式编写 C++代码，然而，预处理程序指令并不是 C++ / C 语言的一个部分，预处理程序也不是编译程序的一个部分。实际上，不使用预处理程序指令，即使是一个很简单的 C++程序也无法编写。但是，从理论上说，这些指令都不是语言的一个部分！实际上，是编译程序的供应商提供了预处理程序，但在理论上，编译程序和预处理程序没有任何联系。

最近，编译程序的供应商放松了以上的限制：'#' 号不一定是该行上的首字符，但它必须是第一个非空字符关于预处理程序，详细内容请参考第 6 章 编译预处理。

（2）注释。"//" 表示从开头直至该行行尾的内容称为注释（comment），用户用来说明或解释程序的功能。计算机运行程序的时候会跳过所有的注释，注释对于程序的运行效果不产生任何影响，但对于程序员而言，注释可提高程序的可读性和可理解性，因而对于程序质量有较大的意义。

C++提供了两类注释：块注释以及行尾注释。块注释以一个双字符符号'/*'开头，并以双字符符号'*/' 结束；行尾注释以双字符符号'//' 开头，并以行结束符（即：在源文件的下一个换行字符）结束。

双字符符号在 C++中是很常见的，它们大部分源于 C。使用双字符符号而不使用其他关键字表示操作并用于其他上下文中，是 C 的设计者为了设计一个只有 30 个关键字的语言而采取的方法之一（可见 C 是一个很小型的语言）。双字符符号的字符（C++中所有的双字符符号，不仅仅是注释）都必须连接输入书写。它们不能被空格符（或其他任何字符）隔开。

两种注释中的文本逻辑上等价于空格符，因而对编译程序来说逻辑上是不可见的。实际上，注释对于编译程序来说不可见，这是由于预处理程序在源代码文本被编译前，已经将注释删除。

许多程序员将块注释作为函数或者算法的重要部分的前言。在这些注释中，描述了算法的目的、要处理的输入数据、作为计算结果的输出数据以及为了完成任务所要调用的其他函数。

通常也会记录程序修改的历史：第一作者和第一版本的日期、其他作者和修改日期、每一次修改的目的等。这些块注释的格式因人而异。当然，最重要的是坚持使用一种统一的格式。更为重要的是要养成书写正确注释的习惯。更需要的是在修改了代码之后及时地修改注释，没有什么会比不正确的注释对维护造成的损坏更严重。

（3）关键字和标识符 （见第 2 章 2.2 关键字和标识符）。

（4）语句（见第 3 章 程序设计和流程控制语句）。

（5）函数和函数调用。

本程序只包含一个程序 - 主函数 main（）。

程序从 main（）的第一行开始执行，逐行执行 main（）中的每条语句。main（）前面的 void 表示 main（）函数是没有返回值的函数。如果定义为 int main（），则 main（）函数的返回值是 int 类型的函数（int 表示整数类型，关于类型定义在下一章有详细的介绍，关于函数与函数调用详细内容请参考第 5 章：函数。

"cout<<" 表示屏幕显示，加了 " " 号表示直接屏幕输出。与 cout<<相反的，cin>>则接受屏幕输入（关于输入输出详细内容请参照第 12 章：流类体系与文件操作）。

这是我们的第一个 C++程序，在这里并不需要读者很透彻地理解这个程序，通过以后的学习，自然会明白。

1.4 C++上机操作

1. 安装编译器

本书中采用 Microsoft Visual C++6.0 编译器（安装大约需 800M 空间，安装示意图略）。

2. 打开编译器

打开编译器的方法如图 1-2 所示。

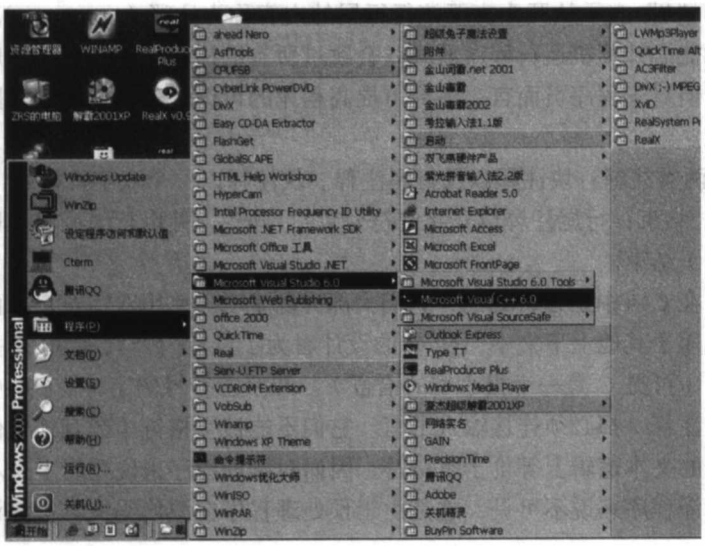

图 1-2

3. 新建一个 C++文件

新建一个 C++文件的方法如图 1-3 所示。

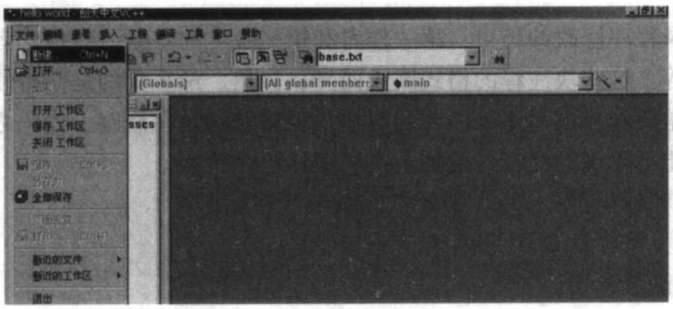

图 1-3

目前我们新建一个 C++ source 文件，并在右边的文件那里对该文件命名为“我的第一个程序”，如图 1-4 所示。

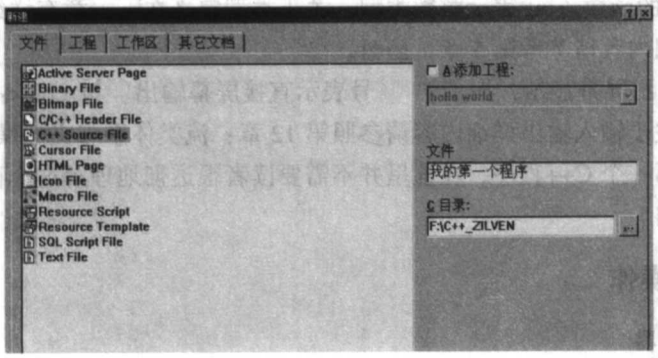

图 1-4