宋晓霞 李勇 仝庆华 编著

C++程序设计

兵器工业出版社

C++程序设计

宋晓霞　李　勇　仝庆华　编著

兵器工业出版社

内 容 简 介

C++以其高效实用的特点在国内外得到广泛使用,它既可进行过程化程序设计,也可进行面向对象程序设计。它深受编程人员的青睐,许多高校已经开设了C++程序设计课程。尽管市面上的C++教材层出不穷,但其内容或者太深,或者单一,或者太泛,适合作为大学教材和自学教材的较少,鉴于此情况,作者结合多年教学和开发经验编写了此书。

本书共分11章,前3章介绍了C++的基础知识;第4章介绍了支持面向过程设计的函数;第5~7章介绍了C++较复杂的数据类型及相关知识;第8~10章介绍了支持面向对象程序设计的相关内容;第11章较详细地介绍了C++的输入输出流。

本书在内容选择上力求体现C++的特点,内容充实,例题丰富,概念清晰,循序渐进,易于学习;是作者总结多年教学和开发实践的经验写成的,适合用作大学计算机专业和非计算机专业的程序设计课程教材,也非常适合读者自学使用。

图书在版编目(CIP)数据

C++程序设计/宋晓霞,李勇,仝庆华编著.—北京:兵器工业出版社,2006.4

ISBN 7-80172-660-X

Ⅰ.C... Ⅱ.①宋...②李...③仝... Ⅲ.C语言—程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字(2006)第039524号

出版发行:兵器工业出版社
发行电话:010-68962596,68962591
邮　　编:100089
社　　址:北京市海淀区车道沟10号
经　　销:各地新华书店
印　　刷:北京市登峰印刷厂
版　　次:2006年4月第1版第1次印刷
印　　数:1—2000

责任编辑:莫丽珠
封面设计:李　晖
责任校对:郭　芳
责任印制:赵春云
开　　本:787×1092　1/16
印　　张:15.5
字　　数:379千字
定　　价:30.00元

前　言

C++是目前世界上功能最强大的程序设计语言之一，是当前进行软件开发的主流程序设计语言。它既可进行过程化程序设计，也可进行面向对象程序设计。用C++程序设计语言开发的程序执行效率高，并且C++软件开发环境越来越友好，极大地提高了软件开发的效率，因此越来越多的人开始学习和使用C++。

C++是最有价值的程序设计语言之一。读者在掌握了C++之后，不仅可以写出具有专业水准的高性能程序，而且能够更容易地学习和使用其他高级程序设计语言。然而C++概念繁多，语法较复杂，涉及内容广泛，往往使初学者望而生畏，难以掌握。针对这种情况，本书从C++的基础知识开始，由浅入深，逐步介绍C++语言的核心内容。

本书作者在课内和课外做了大量调研，分析了学生在学习过程中遇到的困难，结合教育教学理论中学生的认知规律，设计了读者易于学习的教材体系。比如在每章的安排上，以学生更为容易接受的思路叙述问题，以前的绝大部分教材都是以“提出概念——解释概念——实例验证”思路安排章节，使得学生对于为什么会出现新知识、新内容感到费解。本书力求克服这些弊端。

在结构上，每章开始前通过实例或叙述说明为什么要学习新的知识，让学生带着问题和兴趣去学习这些知识，然后通过解决问题检查学生对新知识的掌握程度。

在语言上，将复杂的概念用通俗简洁的语言来描述，并配有大量实例，便于学生理解和掌握。

在例题后面安排了思考题，以拓宽学生的思路，使其能够达到举一反三的目的。

另外，为配合读者巩固所学知识，每章末都附有相当数量的习题。

学好任何程序设计语言，包括C++，最好的方法是自己动手编写程序。因此，学生在学完每一章之后，应该试着运行章节中的示例程序，以便在进入下一章之前，确保已经理解了这些程序设计思想。学生还可以按照书中给出的思考问题或自己动手更改示例程序中的一些代码，并观察修改后程序的运行结果。编写的程序越多，程序设计的学习效果越好。

本书由宋晓霞、李勇和仝庆华编写。宋晓霞编写了第1章，7～11章；李勇编写了第2～4章；仝庆华编写了第5～6章。最后由宋晓霞统一修改、整理和定稿。在这里作者对一切曾经鼓励、支持、帮助过我们的领导、专家、朋友、家人和读者，表示诚挚的谢意。

由于作者水平有限，本书难免有不妥甚至错误之处，真诚希望广大读者批评指正。

宋晓霞

2006年3月于山西大同大学

目　录

第1章　C++语言概述

1.1　程序设计语言的发展

程序是用能够被计算机理解的一种语言编写的语句的集合。语言是交流的工具，计算机程序设计语言是人与计算机之间交互的工具。随着计算机应用范围和规模的发展，程序设计语言和方法经历了由低级到高级、由单模式到多模式的发展历程：机器语言、汇编语言、高级语言、面向对象的程序设计语言。

1.1.1　机器语言

机器语言（Machine Language）是用于特定计算机自身的语言，由计算机的硬件设计定义。不同的机器有不同的语言，因此任何计算机只能直接理解本身的机器语言。机器语言通常是由一系列0和1组成的二进制代码构成的指令序列，是唯一直接面向机器的语言，也是最低级的程序设计语言。

用机器语言编写程序，就是从特定的机器指令系统中选择合适的指令，组成一个指令序列，让计算机一次一个地执行最基本的操作。用机器语言编写的程序可以被机器直接理解和执行，不需要任何翻译，但由于它们有不直观，可读性差，容易出错等缺点，只能被少数专业人员使用。

1.1.2　汇编语言

随着计算机使用越来越普及，用机器语言编程对大多数程序员来说显然太慢、太繁琐。程序员不再去直接理解一系列数字组成的指令序列，而是用类似英文缩写的助记符来表示计算机的基本操作，这些助记符便构成了汇编语言（Assembly Language）。

比如对于某CPU指令系统有机器指令：10 000 000——表示加操作，用汇编语言则可以写成：ADD A，B，这种代码对于程序员可以一目了然，但计算机却无法“理解”，必须将其翻译为相应的机器语言。因此，系统通过汇编程序将汇编语言源程序转换为机器能识别的机器语言，然后再去执行。

汇编语言是与机器有关的语言，执行速度快，然而即使是最简单的任务，也需要许多条指令才能完成，因此，程序员不仅要考虑算法，熟悉硬件结构，还要进行内存分配，劳动强度很大，程序编写效率很低，所以常用于编写系统软件中对执行速度要求很高的部分。

1.1.3　高级语言

随着汇编语言的出现，计算机的使用迅速推广，然而即使是最简单的任务，也需要许多条指令才能完成。为了提高编程效率，人们开发了高级语言（High－level Language）。在高级语言中，用一条简单的语句就可以完成大量任务，比如高级语言中的加法运算用我们熟悉的数学运算符（+）表示。一般用高级语言编写的源程序要经过编译程序翻译成机器能识

别的二进制目标代码，然后将相关模块连接生成可执行文件，最后才能运行。

由于高级语言与人们习惯的语言比较接近，它的使用较大幅度降低了编程的劳动强度，提高了编程的效率，受到了广大使用者的青睐。第一种高级语言 FORTRAN 于 1954 年问世，接着出现了不同风格、不同用途、不同规模的近百种高级语言，有 ALGOL，COBOL，LISP 等。以 FORTRAN 为代表的高级程序设计语言是计算机发展史上的一场巨大变革。20 世纪 60 年代出现了软件危机，结构化程序设计技术对于软件危机做出了自己的历史性贡献；20 世纪 70 年代到 80 年代是程序设计语言发展的高峰期，产生了大量新的程序设计语言。这一时期高级语言迅猛发展，并在整个 20 世纪 70 年代的软件开发中占绝对统治地位。

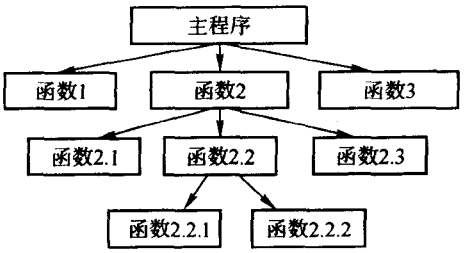

图 1－1　面向过程的程序结构

面向过程的高级语言采用了模块分解与功能抽象和自顶向下、分而治之的方法，从而有效地将一个较复杂的程序系统设计任务分解成许多易于控制和处理的子程序，便于开发和维护。面向过程的高级语言范型的主要特征是：程序由函数定义与函数调用组成，即：程序 = 函数 + 函数调用。图 1－1 所示为一个典型的面向过程的高级语言程序结构，每一个函数表示函数的定义，而箭头的指向则表示函数的调用关系。

1.1.4　面向对象的程序设计语言

随着计算机科学的发展和应用领域的不断扩大，对计算机技术的要求越来越高，结构化程序设计语言和结构化分析与设计已无法满足用户需求的变化，如大型软件的维护和数据重用长期得不到解决，这就需要新的语言机制来打破这些限制，于是面向对象技术开始浮出水面。

面向对象程序设计（Object Oriented Programming，简称 OOP）模拟人类习惯的解决问题的方法，用对象分析取代功能分析，即把程序分解成许多对象，不同对象之间通过发送消息向对方提出服务要求，接受消息的对象主动完成指定功能，程序中所有对象分工协作，共同完成整个程序的功能。实际使用中我们将要求解的问题分解为一系列对象，然后围绕每个对象建立数据和函数，因此每个对象都可看成数据和方法的封装体。C ++ 程序把操作类数据的函数称为方法，对象之间可以通过发送和接收消息建立联系，准确点说是通过对象的函数建立联系，而数据作为程序开发中的基本元素，不允许它们在系统中自由流动，这样可以保护数据不会被外界的函数意外的改变。常为这些数据和函数提供共同的独立内存空间，这样这些数据和函数可以作为模板以便在需要时创建类似模块的拷贝。面向对象程序设计的范型：程序 = 对象 + 消息，消息可通过各对象间的函数进行传递。图 1－2 所示为面向对象程序设计中的数据和函数的组织结构。

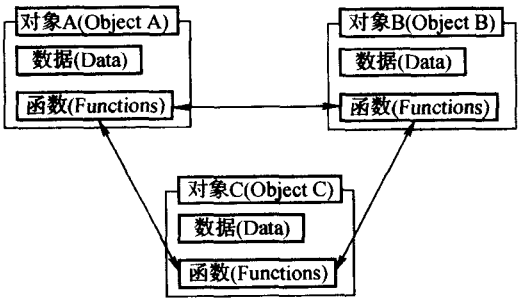

图 1－2　面向对象的程序结构

由于面向对象程序设计有封装、重用、派生等优点，它不仅可以提高程序员的编程效率，而且可以在提高软件质量的同时降低其维护费

用，很适合编写大型程序，因此受到程序设计者或用户的青睐，正在成为当代程序设计的主流。

1.2 C++程序设计语言的发展

图1-3示出了C++家族的发展演化过程：C++是从C语言演变而来的，而C语言又是从两个编程语言BCPL和B演变而来的，BCPL是Martin Richard于1967年开发的，用于编写操作系统软件和编译器。Ken Thampson在他的B语言中大量采用BCPL的特性，并用B语言在DEC PDP-7计算机上生成了UNIX操作系统的早期版本，由于BCPL和B都是“无类型”语言，每个数据项在内存中占一个“字”长，如果要将数据项作为整数或实数处理，编程的工作量会很大。1972年贝尔实验室的Dennis Ritchie开发了一种新型程序设计语言——C语言。

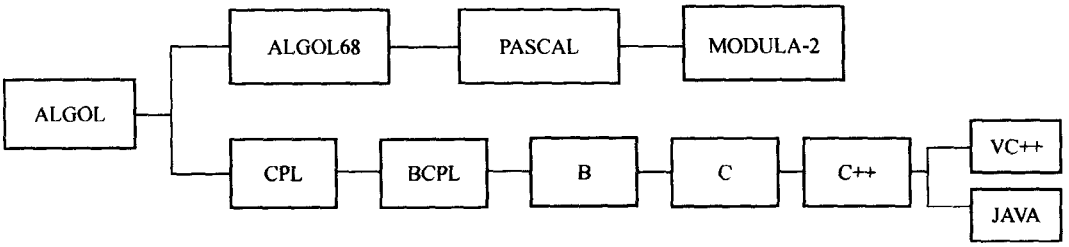

图1-3 C++的发展演化过程

C++是C语言的扩展，是20世纪80年代初由贝尔实验室的Bjarne Stroustrup开发的。C++的许多特性是从C语言中派生的，但更重要的是，它提供了面向对象编程的功能。最初的版本被称作“带类的C（C with classes）”，当时它只支持系统程序设计和数据抽象技术。1983年支持面向对象技术设施被加入C++；之后，面向对象设计方法和面向对象程序设计技术就逐渐进入了C++领域。在1985年C++第一次投入商业市场。在1987~1989年间，支持范型程序设计的语言设施也被加进了C++。随着若干独立开发的C++实现产品的出现和广泛应用，正式的C++标准化工作在1990年启动。标准化工作由ANSI以及后来加入的ISO负责。1998年正式发布了C++语言的国际标准。

1.3 C++程序设计语言的特点

C++是一种多范型程序设计语言，不仅可以用面向过程方式来编写程序，还可以用面向对象方式或兼用两种方式编写程序。这样做的好处是对每个问题能够提供最合适的解决方案，同时带来的缺点是使得语言过于庞大、复杂。

关于C++的特点如下：

（1）C++是面向过程的，具有结构化和模块化等结构化程序设计的优点。

（2）C++与C完全兼容。C++是对C的扩充，是C的超集。

（3）C++增加了面向对象的机制，具有面向对象的优点：

① 易于设计和代码重用——使用对象增强了重用设计和代码的能力，代码的可重用性是C ++最具威力的特性之一。

② 可靠性增强——对象库测试好以后，使用它将增强程序的可靠性。

③ 容易理解——对象的使用帮助程序员把精力集中在理解系统的关键构件，同时设计者只需关注系统的一小部分，由于提供了框架使设计者把注意力集中在程序对对象的操作、对象需保存的信息以及其他系统主要构件。

④ 抽象性增强——抽象使设计者和程序员看到全局，暂时忽略细节，从而能更容易理解系统的元素。抽象是面向对象设计的核心思想，抽象的艺术在于抓住一个系统最核心的特性。

⑤ 封装性增强——封装把对象的所有部分放到一个包里，不需知道类的细节，只要在程序中使用类，而类将带来所有需要的部分。

⑥ 信息隐藏——信息隐藏是程序把函数过程，或对象看成“黑箱”的能力，使用它实现指定的操作而不需知道内部的运转。例如我们可以很好地驾驶汽车而不需要知道发动机、传送系统和燃料系统内部如何工作。

实际应用中，编写面向过程程序与编写面向对象程序的区别：

① 在过程式编程语言中，编程是面向操作的，在面向对象编程中，编程是面向对象的。

② 在过程式编程语言中，编程单位是函数，在面向对象编程中，编程单位是类，通过类可以最终实例化（即生成）对象。

③ 在面向对象编程中，软件打包成类时，这些类变成组件，可以在以后的软件系统中复用。事实上利用对象技术，今后大多数软件都可以通过组合“标准化、可互换组件”的类而生成。对于每个已建类都是宝贵的软件资源，可以用于加速和提高以后的软件开发速度。

④ 面向对象程序需要较少的代码，因为问题中的许多部分都可以使用已存在的库代码。

C ++不仅可以用面向过程方式来编写程序，还可以用面向对象方式或兼用两种方式编写程序。为了挖掘C ++的强大功能，在解决每个问题前必须考虑面向对象的方法。在编写第一行程序之前，必须尽力地花时间去思考系统的设计、系统的基类以及期望由系统基类所派生出来的派生类。如果没有考虑到面向对象和模块化，C ++这种工具的面向对象强大之处就将变得毫无用处。

1.4 简单的C ++程序

为了使读者能早点认识C ++程序，并在程序中认识C ++语言的重要特性，下面先介绍两个程序，并对程序每行进行详细的解释。

【例1-1】 输出一行字符：“Welcome to study C ++!”。

```
//the first program in C ++
#include <iostream. h >
 int main( )
 {
 cout <<"Welcome to study C ++! \n";
```

```
    return 0; //indicate that program ended successfully
}
```

程序运行结果：

```
welcome to study C++!
```

思考：如何改变行文字的输出形式？

① 比如如何分多行输出？

② 如果将输出内容改为"Welcome to C++!"，该如何修改？

第1行：以//开头，表示该行其余部分是注释语句（Comment）。程序员插入注释语句用来说明程序和提高程序的可读性。注释语句还可以帮助其他人阅读和理解程序。在运行程序时注释语句并不使计算机采取任何操作。C++编译器忽略注释语句，不产生任何机器语言目标码。注释语句"the first program in C++"只是描述该程序的用途。以//开头的说明语句称为单行注释语句（single - - line comment），因为该注释语句在行尾结束（注意：C++程序员也可以用C语言注释语句样式，即注释语句以/*开头，以*/结束）。

第2行：#include <iostream.h>是条预处理指令，是发给C++预处理器的消息。预处理器先处理以#开头的一行语句之后再编译程序。这一行预处理指令告诉预处理器要在程序中包括输入/输出流头文件iostream.h的内容，应该在任何使用C++式输入/输出流从键盘输入数据或向屏幕输出数据的程序中包括这个文件。注意，最新C++标准指定iostream.h和其他标准头文件不需要后缀.h，写成iostream即可。我们在本书余下部分继续使用旧式头文件，因为许多编译器还不支持最新的ANSI/ISO C++标准。

第3行：int main()是每个C++程序都包含的语句。main后面的括号表示main是个程序基本组件，称为函数（function）。C++程序包含一个或几个函数，其中有且只有一个main函数。即使main不是程序中的第一个函数，C++程序都从main函数开始执行，并且终止于main函数。main左边的关键字int表示main返回一个整数值。我们将在介绍函数时深入介绍函数返回值的含义。目前只要在每个程序的main函数的左边包括关键字int即可。

第4行左花括号{应放在每个函数体开头，对应右花括号}应放在每个函数体结尾（第7行）。

第5行：cout <<"welcome to study C++! \n"，让计算机在屏幕上打印引号之间的字符串，整个行称为语句，每条语句应以分号结束。C++中的输出和输入是用字符流完成的，执行上述语句时，将引号中的字符流"Welcome to study C++!"发送到标准输出流对象，通常cout将其输出到屏幕。运算符<<称为流插入运算符，将运算符右边的值插入输出流中。右操作数字符通常按引号中的原样直接打印。执行这个程序时，要注意字符\n，不在屏幕中打印。反斜杠\n称为转义字符，表示要输出特殊字符。字符串中遇到反斜杠时，下一个字符与反斜杠组合，形成转义序列。转义序列\n表示换行符，使光标（即当前屏幕位置的指示符）移到屏幕中下一行开头。

第6行：return 0；放在每个main函数末尾。main函数末尾使用return语句时，是退出函数的一种方式。现在只要记住：main前面加int，同时在main函数的最后一条语句加上return 0；即可。

【例1-2】 一个包含类的简单C++程序。

```
//输出一个人的数据
```

```
#include <iostream.h>
class man
{   private:
    int age;
protected:
   int wage;
public:
  int height;
  void set(){age=20; wage=500; height=160;  }   //类内定义函数
  void show();  };
void man::show()      //类外定义函数显示数据
{   cout<<"The man's data:\n";
    cout<<"private age: "<<age<<"\n";
    cout<<"protected wage: "<<wage<<"\n";
    cout<<"public height: "<<height<<"\n";    }
int main()
{   man man1;
    man1.set();
    man1.show();
    return 0;  }
```

程序运行结果：

```
The man's data:
private age: 20
protected wage: 500
public height: 160
```

第3～13行定义了一个类，类名man，该类中包含两种成员：数据成员和函数成员。C++中把这些成员封装在一起组成一个类。类是面向对象程序设计的重要概念，是对具有相同属性和行为的一个或多个对象的抽象。其实类可以看成一种特殊的自定义的数据类型。

该类中的数据成员包含：age，wage，height；而它们有不同的访问权，age是私有访问（Private），wage是保护访问（Protected Wage），height是公有访问（Public）。在定义数据成员时在其前面加上不同的访问权符可以实现数据的封装，具有public访问属性的成员，可被与该类对象处在同一作用域中的任何函数使用；具有private访问属性的成员，只能被它所在类中的成员函数及该类的友元函数使用；具有protected访问属性的成员，只能被它所在的类、该类子类的成员函数、友元函数使用。一般把数据成员定义为私有的，不允许外界使用，以实现信息隐藏；成员函数定义为公有的，用来提供类与外界的接口。这样把数据和处理数据的函数紧密地联系起来放在对象内，使得处理数据的函数都集中在一个局部区域内，从而实现封装。利用封装机制，使用访问权符把由对象操纵的数据隐藏起来，然后提供一些公有的成员函数作为外界访问这些数据的界面。类体现了数据的封装性和信息隐藏。封装机制提高了软件的可维护性。

该类中的成员函数包括：定义在类内的 set( ) 和程序中的第 14 ~ 21 行在类外定义函数 show( ),成员函数可以定义在类内，也可以定义在类外；类内定义的成员函数无论有否 inline 关键字，都作为内置函数处理。类外定义的成员函数，必须在类内声明其函数原型，并作为普通函数处理。在类外定义成员函数与定义普通函数的区别在于：成员函数必须以作用域运算符“::”指明其所属的类。从功能上将成员函数划分为构造函数、析构函数和普通成员函数。详细的内容将在后续章节介绍。

程序第 22 ~ 26 行是主函数的定义，主函数中有 3 个语句，用来调用对象的成员函数，实现输出相关信息。其中第 23 行 man man1，定义了一个 man 类的对象 man1；第 24 行 man1. set( ),调用成员函数 set( ),但要用作用域运算符指明属于哪个对象；第 25 行 man1. show( ),调用成员函数 show( ),也指明该函数属于对象 man1。对象是类的实例，即用类定义的变量，可以像定义变量那样用类的名字去定义它的对象。

即使对以上说明不能完全理解的话，也不要紧，第 7 章会做详尽的介绍。

1.5 C ++ 程序的上机步骤

前一节已经介绍了两个简单的C ++ 程序，问题是否真的解决需要实践的检验。面对一个没有经过实验验证的程序，哪怕是最好的程序员都不能说它是完全正确的。

程序运行的具体过程可以参考图 1 -4，下面叙述一个程序从编写到上机运行要经历的步骤：

第 1 步：编辑（Edit）

程序是用能够被计算机理解的一种语言编写的语句的集合。所谓编辑就是用C ++ 语言在编辑器中编写源程序，C ++ 的源程序以 . cpp（c plus plus 的简写）作为后缀名。源程序写好后，通常存入文件中，这个文件叫做源代码文件。

第 2 步：编译预处理（Preprocess）

顾名思义，编译预处理是指在对程序进行通常的编译之前，先对程序的一些特殊命令进行预处理。通常需要进行编译预处理的特殊命令指以“#”号开头的那些命令，通常包含宏定义、文件包含、条件编译等。

第 3 步：编译（Compile）

由于计算机只能识别0 和 1 组成的二进制序列，而不能识别和执行C ++ 语言编写的源程序。当我们人类面对不能理解的语言时，就会去向翻译求助；对于用C ++ 语言编写的程序，机器无法识别，我们用一种称为编译器的软件将源程序翻译成机器能识别的二进制目标代码，帮助机器理解程序。现在通常使用的是菜单驱动系统，那么编译和运行程序的处理只不过是在适当的菜单选项上敲击一下。编译是以源程序为单位进行的，作用是对源程序进行词法检查和语法检查，编译成功后会生成目标文件（. obj），如果有多个源程序文件就会生成多个目标文件。

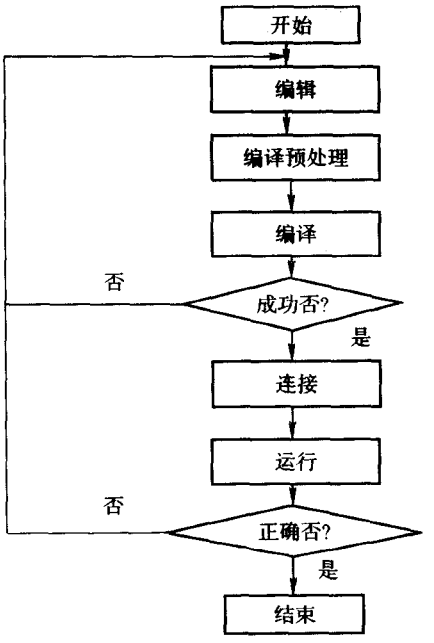

图 1 -4　C ++ 程序运行过程

第4步：连接（Link）

连接器将一个程序的所有目标文件和系统库文件以及程序引用支持例程连接在一起，最终形成一个可执行的二进制文件，以 .exe 为后缀名。我们把它叫做可执行代码，可直接运行。

第5步：运行（Run）

运行最终形成的可执行的二进制文件，得到运行结果。

第6步：运行结果分析

如果经测试程序的运行结果有问题，需要对程序进行修改，修改后再重复上面的编译、连接、运行步骤，直到程序运行正确为止。

本书所有的程序都是在 Visual C ++6.0 环境下调试的。

【小结】

本章介绍了程序设计语言及C ++语言的发展过程，阐述了C ++语言的特点：C ++是一种多范型程序设计语言，不仅可以用面向过程方式来编写程序，还可以用面向对象方式或兼用两种方式编写程序。通过简单的C ++程序实例，使读者对C ++程序结构有一定的认识，最后介绍了程序的上机步骤。

习　题

1. 简要叙述程序设计语言的发展过程及发展过程中每个阶段的语言特点。
2. 简要回答C ++程序设计语言的特点及发展历程。
3. 考虑用多种方法在屏幕上输出以下文字，并按照上机步骤在你自己的C ++环境下调试自己编写的程序：

“ I am a student, I love China!”

4. 根据你自己的理解，总结C ++程序的结构。

第2章　C++数据运算

2.1　数据类型

由程序操纵的实际的值叫做数据，数据可以表达成多种不同的形式。数据可由数字和字符组成。在C++程序中，不同的数据类型以不同的方式存取和处理。下面给出C++可以使用的数据类型：

本章我们将介绍基本数据类型，数组、枚举、结构体和共用体这些构造类型将在第4章详细介绍，指针和引用将在第5章详细介绍，而类类型将在第6章以后进行全面阐述，空类型就是无值型，为了保证语法的完整性而设置。

C++并没有统一规定各类数据的精度、数值范围和在内存中所占的字节数，各C++编译系统根据自己的情况做出安排。这里简短地介绍这些数据类型在计算机中如何存储。表2-1列出了基本数据类型在16位计算机上的存储需求和数值范围。请记住存储规定会因不同的计算机而异。如果在一台IBM或IBM兼容PC机上工作，那么存储规定将与此大致相

同，因此本教材用到的数据也以表 2 - 1 为准。

表 2 - 1　数据类型及存储需求

数据类型	C ++ 中的声明	存储字节	允许值范围
整数	int	2	-32 768 ~ 32 767
短整数	short int	2	-32 768 ~ 32 767
长整数	long int	4	-2 147 483 648 ~ 2 147 483 647
无符号整数	unsigned int	2	0 ~ 65 535
无符号短整数	unsigned short int	2	0 ~ 65 535
无符号长整数	unsigned long int	4	0 ~ 4 294 967 295
字符型	char	1	-128 ~ 127
单精度型	float	4	$3.4 \times 10^{-38} \sim 3.4 \times 10^{38}$
双精度型	double	8	$1.7 \times 10^{-308} \sim 1.7 \times 10^{308}$
长双精度型	long double	10	$3.4 \times 10^{-4\,932} \sim 1.1 \times 10^{-4\,932}$

C ++ 中数据分为常量和变量，它们分别属于以上这些类型。常量是在程序执行过程中其值不能发生改变的量，变量是在程序执行过程中其值可以发生改变的量。

2.2 常　　量

在程序中，不同类型的数据既可以是常量，也可以是变量。常量是在程序的运行期间其值不发生改变的量。按照字面能否看出又将常量分为直接常量和需要定义的常量。直接常量通常包括整型常量、实型常量、布尔常量、字符常量和字符串常量等；需要定义的常量包括 C 语言中用#define 来定义的常量（称为宏常量）和C ++ 语言中用 const 来定义的常量（称为 const 常量）。

2.2.1　为什么需要常量

如果不使用常量，直接在程序中填写数字或字符串，将会有什么麻烦?

① 程序的可读性（可理解性）变差。程序员自己会忘记那些数字或字符串是什么意思，用户则更加不知它们从何处来、表示什么。

② 在程序的很多地方输入同样的数字或字符串，难保不发生书写错误。

③ 如果要修改数字或字符串，则会在很多地方改动，既麻烦又容易出错。

2.2.2　直接常量

直接常量就是通常说的常数，由于可以从字面上直接看出，因此又称为“字面常量”或“文字常量”。当一个数值，例如 10，出现在程序中时，它被称为文字常量，称之为“文字”是因为我们只能以它的值的形式指代它，称之为“常量”是因为它的值不能被改变。每个文字都有相应的类型。例如，0 是 int 型，而 3.141 59 是 double 型的文字常量。文字常量是不可寻址的，尽管它的值也存储在机器内存的某个地方，但是我们没有办法访问它们的地址。

整数文字常量可以被写成十进制、八进制或者十六进制的形式（这不会改变该整数值的位序列）。例如，十进制的 20 可以写成下面三种形式中的任意一种：

十进制形式：20

八进制形式：024