C++语言及其程序设计教程

张国峰　编著

電子工業出版社

C++语言及其程序设计教程

张国锋　编著

电子工业出版社

(京)新登字 055 号

内容简介

C++是在C的基础上发展起来的一种新型程序设计语言，它保持了C的简洁、高效和接近汇编语言的特点，消除了C中的一些不安全因素，更重要的是增加了类这种语言成分，以支持面向对象的程序设计。本书共分十五章和一个附录，系统和全面地介绍C++的各个语言特性，并介绍用C++进行程序设计的基本方法。

本书适合于大专院校师生、培训班师生和从事计算机软件开发和应用的人员使用。

CC+语言及其程序设计教程

张国锋 编著

责任编辑 王昌铭

电子工业出版社出版(北京万寿路)

电子工业出版社发行 各地新华书店经售

北京市燕山联营印刷厂印刷

*

开本：787×1092 毫米 1/16 印张：21.5 字数：550 千字

1992 年 12 月第 1 版 1992 年 12 月第 1 次印刷

印数：10100 册 定价：11.50 元

ISBN7-5053-1684-2/TP・369

前　言

C++是在C语言的基础上，吸收了BCPL、Simula 67和Alogl 68语言中的一些精华而逐渐发展起来的一种通用目的的程序设计语言。它是为适应九十年代开发和维护复杂的应用软件的需要而研制的，其目标是为程序员的程序开发活动提供一个优良的程序设计环境，以产生模块化程度高、重用性和维护性俱佳的程序。同时它非常强调代码的有效性和紧凑性，它是程序员的语言，允许程序员决定如何实现特定的操作。因此，自1983年贝尔实验室公布C++以来，C++在计算机界引起了广泛的重视，并在系统程序设计、计算机辅助设计、仿真、数据库和人工智能等领域中得到了广泛应用。

在程序设计方法方面，C++既支持传统的面向过程的程序设计方法，也支持新的面向对象的程序设计方法。因此，C++是一种混合语言。本书将系统和全面地介绍C++的各个语法特性，同时结合对C++的介绍，也介绍用C++进行程序设计的基础知识。

全书共分十五章和一个附录。第一章简要地回顾了C++的发展历史，然后介绍了几个简短的C++程序，使读者对C++的程序结构有个感性认识。第二章介绍C++的基本数据类型和表达式。第三章介绍C++的语句和函数，C++提供了函数和多种流程控制语句，以支持结构化程序设计。第四章介绍数组和指针，第五章介绍串以及与串有关的程序设计，这两章可以帮助读者掌握C++指针的概念以及和指针有关的程序设计方法；第五章还给出了一个数据分析实例程序，并通过这个程序介绍了有关结构化程序设计(面向过程的程序设计)的基本概念。第六章介绍C++的内联函数、重载函数、带缺省参数的函数和参数数目可变的函数，还介绍了C++程序的多文件组织方法和C++编译器常用的几个编译指令，递归程序设计也在这章介绍。第七章介绍结构，C++的结构将逻辑上相关的数据汇集在一起，使程序可以方便地处理复杂的数据。

以上几章介绍的C++的语法特性是C++对C进行改进之后所形成的语言特色，最初它们与传统的面向过程的程序设计方法有着较密切的联系。

C++最吸引人的地方在于它支持面向对象的程序设计方法。面向对象的程序设计方法是一种新的设计和构造软件的思维方法，该方法的目标是使程序员在计算机上解决问题的方式更加类似于人类活动。在C++中，面向对象和面向过程这两种程序设计方法是互补的，面向对象的程序设计方法影响着软件的整体结构，面向过程的程序设计方法影响着编写局部代码段的风格。

面向对象的程序设计方法要求语言必须具备抽象、封装、继承和多态性这几个关键要素。C++实现了这些要素，其关键语法特性是类。第八章讨论类以及和类有关的概念，在这章也介绍了面向对象程序设计的基本概念。第九章介绍对象的初始化，在C++中，类不但描述了数据和可对数据进行的操作，同时也定义了对象的初始化方式。第十章和第十一章讨论继承和多态性，继承增强了软件的可扩充性，并为代码重用提供了强有力的手段；多态性使程序员在设计程序时可以对问题进行更好的抽象，以设计出重用性和维护性俱佳的程序。第十二章对类的特性作进一步的讨论，这章主要介绍了友元、静态成员、转换函数和联合。第十三章介绍运算符重载，以及在进行运算符重载程序设计时应注意的问题。第十四章介绍C++系统的流库，流库是一个类库，它使程序员可以很容易地设计执行标准设备输入和输出、文件输入和输出以及内核输入和输出操作的程序，程序员也可以通过扩展类库使其适用于特殊目的或特殊设备的输入和输出操作，自定义操纵算子为程序员提供了扩展类库的另一种手段。

第十五章比较了C++和C的差异，本章的主要目的是帮助不懂C的读者能读懂C程序，另一

个目的是试图使具有C程序设计经验的读者对C++有个概貌性了解，C程序员可以先浏览一下这章的内容。附录A介绍了ASCII码表，并通过C++程序来解释一些常被人们混淆的概念，希望读者能先浏览一下这个附录所介绍的内容。

在本书写作过程中，作者参照了Borland C++和Zortech C++编译器所提供的文档资料中介绍的C++标准，这两个编译器都完整地实现了ANSI C和AT&T C++标准。书中的程序实例都是精选的，并在计算机上调试通过。阅读本书，要求读者对计算机的基础知识有所了解，并学习过一门高级计算机语言，本书可以作为学习C++的通用教材或参考书籍。

麦中凡教授审阅了本书的初稿，并提了许多宝贵意见，王坚、高照华、顾申懋、李大伟和徐希等同志参与了书稿的整理工作，作者对于他们给予本书的贡献表示深深的感谢。作者也非常感谢那些在本书写作过程中给予作者以鼓励和帮助的人们。对于本书所存在的错误和不足之处，作者殷切希望广大读者批评指正。

张国锋

1992年5月于北京航空航天大学

目　　录

第一章　C++ 概　　述

本章简要介绍C++的发展过程和特点，然后通过几个简单的程序介绍C++程序的结构和输入输出操作。

1.1　C++的起源和特点

C++语言是C语言的扩充。C是1972年由Dennis Richie在贝尔实验室设计的一个通用目的的程序设计语言，它的前身是B语言，而B语言又是在继承和发展了BCPL语言的基础上设计的。C最初用作UNIX操作系统的记述语言，由于UNIX的成功和广泛使用，也使C成为一种普遍使用的程序设计语言，目前在各种机型和各种操作系统上都运行有C编译器。C有以下几个突出的特点：

·C是结构化的程序设计语言。结构化程序设计要求程序的逻辑结构由顺序、选择和循环三种基本结构组成，C提供了编写结构化程序所需要的语句。另外，使用C也便于进行模块化程序设计，C程序由众多的函数组成，函数在C中是进行模块化程序设计的基本单位。

·C反映了设计者追求高效、灵活和强功能的设计思想。C是为了能够胜任系统程序设计的要求而开发的，因此有很强的表达能力，能够用于描述系统软件各方面的特性，用C编写的程序生成的机器代码质量也很高。

·C可以部分地取代汇编语言。C的第一个应用场合是开发UNIX操作系统，由于C的这个特点，使得UNIX只使用了很少的汇编代码，其余代码全部是用C写成的。

·C具有较高的可移植性。C的语句中没有依赖于硬件的输入输出语句，程序的输入和输出操作由系统提供的独立于C的库函数来完成，因此C本身不依赖于计算机的硬件，从而便于在硬件结构不同的机器间实现程序的移植。

·C提供了种类丰富的运算符和数据类型，因而极大地方便了程序设计，这也使得C较难被初学者学习和掌握。

C本身也存在着局限：

·C类型检查机制相对较弱，这使得程序中的一些错误不能在编译阶段由编译器检查出来，这些错误若是遗留到程序的运行阶段由程序员来检查，将是很困难的。

·C本身几乎没有支持代码重用的语言结构，因此一个程序员即使有很高的程序设计技巧，并严格尊循模块化程序设计方法，为一个应用程序编写的代码也很难重用于另一个程序中。

·C不适合于开发大型程序，当程序的规模达到一定的程度时，程序员就很难控制程序的复杂性。

为解决上述问题，并保持C的简洁、高效和接近汇编语言的特点，1980年，贝尔实验室的Bjarne Stroutstrup博士及其同事开始对C进行改进和扩充，最初称为“带类的C”，1983年取名为C++，以后又经过不断完善和发展，成为目前的C++。除了个别的例外情况，它将C作为它的子集，并引入了Simula 67、Algol 68和BCPL语言中的一些概念。除了使C++成为“更好的C”和对C的类型系统进行改进和扩充，使用户能定义更多更强有力的类型之外，C++对C的重要扩充是支持面向对象的程序设计。

C++保持与C兼容，这就使许多C代码不经修改就可以为C++所用，用C编写的众多的库函

数和实用软件可以用于C++中；另外，用C++编写的程序可读性更好，代码结构更为合理，可以更直接地在程序中映射问题空间的结构；更重要的是，C程序员仅需学习C++语言的新特征，就可以很快地用C++编写程序。

C++已被应用于程序设计的众多应用领域，它尤其适用于中等和大型的程序开发项目。有报告表明，C++已应用于C曾经使用过的所有场合，其效果比C要好得多，从开发时间、费用到形成的软件的可重用性、可扩充性、可维护性和可靠性等方面，都显示出C++的优越性。

1.2 C++程序

当你编写一个程序时，就是以一种程序设计语言所要求的规范在表达你的思想，这和人与人之间交流使用自然语言一样，程序员和计算机交流使用程序设计语言。C++有严格的语法规则，并表示确定的语义(即含义)。C++程序是用C++写成的，称为源程序，并以文件的形式保存在计算机中，这个文件称为源文件。计算机系统不能理解用C++编写的程序，所以，在这个程序运行以前，必须使用一个称为C++编译器的软件先对这个程序进行编译。C++编译器是个非常复杂的软件，它读入源程序文件，检查程序中的语法错误。当这些代码确实表示一个正确的C++程序时，编译器就将源程序翻译成目标代码，称为目标程序或目标模块，保存目标程序的文件称为目标文件。

目标文件由硬件级的计算机指令、符号信息和其它一些二进制信息组成。目标文件和源文件一样，也是不能由计算机直接执行的，必须经由连接器将目标文件和随同C++编译器一同提供的C++系统库连接在一起，目标文件中的符号信息就是供连接器使用的，经连接后的模块最终形成一个可执行的程序，这时你就可以让计算机执行这个程序。可执行程序保存在一个称为可执行文件的文件中，可以将几个目标文件和几个库连接在一起，形成一个可执行文件。

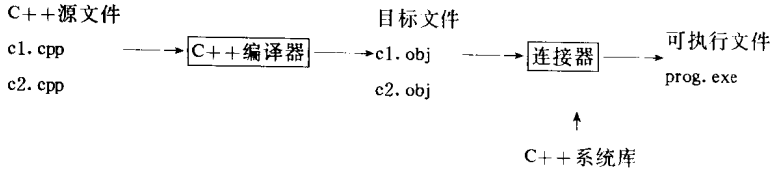

图1-1 C++程序的编译过程

图1-1说明了C++程序的编译和连接过程。在MS-DOS环境下，文件名是由1到8个字符组成的基本文件名和点号后跟1到3个字符组成的扩展名共同组成的，例如，下面是合法的文件名：

filename（八个字符组成基本名，无扩展名）

f1.cpp（由二个字符组成的基本名和三个字符组成的扩展名构成）

在C++系统中，源文件约定使用扩展名.cpp，另外还有一个称作头文件的文件，它约定使用扩展名.h或.hpp。现代的C++编译器可以兼容ANSI C，当一个源文件名的扩展名是.c时，编译器就按C的语法编译这个程序，这样，C++编译器就可以当作一个C编译器使用。C++系统指的是包括C++编译器、连接器、系统库、源文件编辑程序和其它辅助开发C++程序在内的各种软件的总称。

1.3 C++程序的结构

考虑下面这个简单的C++程序：

```
/* This is a sample C++ program that demonstrate the construction
```

```
        of C++ program file.  The file name is demo.cpp
*/
#include <iostream.h>
main()
  {    // main
  cout << "This is the start of something wonderful!" ;
  }    // main
```

编译和连接这个程序，当运行这个程序时，在屏幕上显示一行信息：

This is the start of something wonderful!

这条信息从屏幕的最左边缘开始显示。

程序中的第四行为

#include <iostream.h>

其中#include 是C++编译器的一个编译指令，iostream.h 是C++系统的一个系统文件，经常称其为头文件。这行的作用是指示C++编译器将文件 iostream.h 的内容插入到程序中#include 指令所在的这一行的后面，这使得程序可以使用在文件 iostream.h 中定义的标准输入和输出操作。只有这样，语句

cout << "This is the start of something wonderful!" ;

才正确。cout 称作标准输出流，在 iostream.h 中定义，它表示标准输出设备，一般指的是屏幕。<< 是一个运算符，它把它右边由双引号括起的内容在屏幕上显示出来。

通过运行这个程序，我们知道输出流 cout 将信息输往屏幕，但实现输出操作的标准输出流 cout 和运算符<<的代码是从哪里来的呢？这个程序首先交由C++编译器进行编译，在程序没有错误时，产生目标文件。如果程序有错误，会产生编译错误信息，你应依据这些错误信息修改源程序，并重新编译程序。当连接器工作时，连接器根据目标文件中保存的信息，从C++系统库中将实现 cout 和<<的代码加入到你的程序中，这样才形成一个可执行文件。cout 和<<只在 iostream.h 中作了说明，但未给出实现代码，所以连接过程是必需的。任何C++程序都要在连接阶段从系统库中补足程序中缺少的成份，如果不能补足这些缺少的成份，这个程序就是不完整的，连接器会告诉你这方面的错误。程序员在接到连接器给出的错误信息以后，就不能运行这个程序，这时需要修改程序中的错误，或检查一下所使用的C++系统是否完整。

许多程序要使用 iostream.h，有些程序还需要其它的头文件，不久我们就会遇到这些头文件。

在C++中，以分号结尾的句子称为语句。例如，在上面那个程序中，用于输出一条信息的语句就是以分号结束的。#include 不是C++的语句，它是一个编译指令，不以分号结束，在使用了编译指令#include 的行上不可以再有其它语句。

这个程序中以 main 开始的部分定义了一个函数，该函数规定了该程序的功能。main 是函数名，在函数名之后紧跟一对圆括号。所有的C++程序都必须有一个名为 main 的主函数。给定一个名为 main 的函数是程序员和C++系统之间的约定：程序执行的开始点是 main 函数中的第一条语句。

一个C++函数中的任何成份被括在一对花括号（"{"和"}"）中，在函数 main 的后面的右圆括号后紧跟一个左花括号，表示"这个函数从这里开始"，最后的右花括号表示"这个函数在这里结束"，花括号括起来的部分称作函数体，而函数名 main 和它后面的一对圆括号称为函数头。函数体

由一系列的C++语句组成，这些语句描述这个函数怎样实现它的功能(例如，怎样显示一条信息)。在上面这个例子中，函数main只包含一个简单的语句，它将一条信息送到标准输出流——屏幕上。

程序设计语言和人类所用的自然语言尚有一段距离，因此，为增强程序的可读性，程序员需要在程序中使用注释来描述程序的功能，为阅读程序的人员提供理解程序的线索。C++语言中使用两种方法标记注释。符号//告诉编译程序，//之后本行的所有内容都是注释；而符号/*和*/告诉编译器：在/*和*/之间的内容都是注释。//更适合于短的只占一行的注释，而/*…*/更适合于长的占用多行的注释。你可以根据你的偏好，倾向于使用一种注释，或两种都使用。例如，如果都使用/*…*/标记的注释，则标记函数main的开始和结束的注释可以用：

```
{     /*  main  */
      ......
}     /*  main  */
```

C++编译器跳过程序中的注释，不处理注释的内容。

C++程序文件的书写格式是自由的。在C++程序中，多个空格符被当作一个空格符看待，即和一个空格符一样，都起分隔单词的作用。在上面的程序中，通过恰当地使用空格，增强了程序的可读性。例如，cout的开始比上行的左花括号向后缩进了几列，这使得你一眼就可以看出，main函数的函数体由哪些语句组成。谁都不会认为下面这个正确的C++程序赏心悦目、易于阅读和理解：

```
#include <iostream.h>
main() { /* main */ cout << "All done !" ; }     /* main */
```

编译器可以正确地理解这个程序，但让人来阅读实在是太困难了。编写程序就象写文章一样，段落要清楚。

1.4 新行

下面这个程序在屏幕上显示两条信息，每条信息显示在单独的一行上：

```
/* This program writes two messages to the screen, say
   anything at all.  But make the messages appear on two
   separate lines.
*/
#include <iostream.h>
main()
{
cout << "This is the start of something wonderful!  \n" ;
cout << "And now we can say even more!" ;
}
```

运行这个程序，输出结果是：

```
This is the start of something wonderful!
And now we can say even more!
```

程序中，\n的作用是使第二条信息显示在另一行上(新行)，\n称为新行符。C++允许一个新行在所欲显示的正文的任何一点开始，例如，如果在“of”之后再增加一个新行符，这就在“of”之后

开始了一个新行。例如：

```
#include <iostream.h>
main()
{
        cout << "This is the start of\n something wonderful! \n" ;
        cout << "And now we can say even more!" ;
}
```

这使得"something wonderful!"也显示在单独的一行上。程序的输出是：

```
This is the start of
something wonderful!
And now we can say even more!
```

C++系统提供了一个操纵算子 endl，它的功能和新行符一样，开始一个新行，所以本节的第一个程序又可以写成：

```
#include <iostream.h>
main()
{
        cout << "This is the start of something wonderful!" ;
        cout << endl ;
        cout << "And now we can say even more!" ;
}
```

1.5 变量

变量是内存中的一个命名的存储单元，用于存放可由程序修改的值。任何变量必须在使用前进行说明。下面的程序用于说明如何在程序中使用变量，有关变量的更多概念在第二章讨论。这个程序将任何两个常数加在一起，显示其结果，然后在另一行上显示一条信息。

```
#include <iostream.h>
main()
{
        int result ;
        result = 5 + 10 ;
        cout << "the sum is " ;
        cout << result ;
        cout << endl ;
        cout << "All done!" ;
}
```

在这个程序中，语句

```
int result ;
```

是一个变量说明语句，它说明了一个整型变量 result。整型变量只能用于保存整数值。所谓整数值，就是不带有小数点的数值。

int 是 C++的关键字，在程序中用于说明整型变量。关键字是 C++的保留字，具有特定含义，通过使用关键字，程序员能够使 C++编译器理解程序中所表达的思想。程序员不能将关键字用于

别的目的，例如，不能将 int 用作变量名。

在这里，你可以简单地认为变量名是由大小写的英文字母构成的。

在程序中，语句

```
result = 5 + 10 ;
```

用于给变量 result 赋值，符号“=”称为赋值运算符，它把它右边的值赋给了左边的变量。这个语句的意思(语义)是将 5 和 10 相加，然后将结果赋给变量 result。所以这个语句执行以后，result 中保存的值是 15。语句

```
cout << result ;
```

用于显示变量 result 中保存的值。在运算符<<的右边放置一个变量名，就可以将这个变量中保存的数值显示出来。

在 C++程序中，象 5 和 10 表示一个固定的数值，在程序中它们的值不会被改变，我们称它们为常量，由于它们不带小数部分，所以更明确地称它们为整常量。

C++允许将连续几条输出语句并为一条语句，因此，上面的程序又可以写成：

```
#include <iostream.h>
main()
{
  int result ;
  result = 5 + 10 ;
  cout << "the sum is " << result << endl << "All done!" ;
}
```

这确实比前面的程序显得简洁，应该逐渐习惯这种表示法。

1.6 输入

上节的程序只能将 5 和 10 加在一起，显示一个唯一的结果，还不够灵活。如果在使用这个程序时，程序能够提示让用户输入要加的两个数，当用户输入两个数以后，程序将计算结果显示出来，这就灵活多了。下面是按这个要求修改后的程序：

```
#include <iostream.h>
main()
{
    cout << "Enter two numbers separated by spaces: " ;
    int a ;
    int b ;
    cin >> a ;
    cin >> b ;
    int result ;
    result = a + b ;
    cout << "The sum is " << result << endl << "All done!" ;
}
```

这个程序说明了两个整型变量 a 和 b，用来保存用户输入的数据。语句

```
cin >> a ;
```

和

```
cin >> b ;
```

将等待用户输入两个数，然后将这两个数分别存于变量 a 和 b 中。cin 称作标准输入流，象 cout 一

样，也是定义在头文件 iostream.h 中的，它表示标准输入设备，一般指键盘。运算符>>将键盘中输入的一个数，送到它右边的变量中保存起来。

这个程序运行后，在屏幕上你可以看到下面这条信息：

```
Enter two numbers separated by spaces: _
```

光标在冒号后面闪烁，等待你输入两个数。例如，你输入 10，按一下空格键，再输入 20，然后按回车键，这时程序就读入你所输入的数。10 送给 a，20 送给 b，然后开始执行以后的语句。语句

```
result = a + b ;
```

将 a 和 b 中保存的数值加在一起后赋给变量 result。程序最后一条输出语句显示一些信息和计算结果。这个程序执行完以后，在屏幕上可以看到这样几行：

```
Enter two numbers separated by spaces: 10 20<CR>
The sum is 30
All done!
```

其中 10 和 20 是用户输入的数据，<CR>表示用户按下了回车键(回车键产生的字符\n 是不可显示的，所以我们用<CR>表示这个字符)，后面的两行是程序输出的计算结果和表示程序正常结束的信息。

这个程序说明了三个变量，它们都遵循先说明后使用的原则，而且每个变量的说明都靠近它第一次使用的地方，这是使用 C++进行程序设计时应该注意的一种良好风格。因为若不这样做，当程序很长时，你得返回很远去查找某个变量是怎样说明的。可以将这个程序中连续的两条输入语句合并为一条，下面是修改后的程序：

```
#include <iostream.h>
main()
{
    cout << "Enter two numbers separated by spaces: " ;
    int a ;
    int b ;
    cin >> a >> b ;
    int result ;
    result = a + b ;
    cout << "The sum of " << a << " and " << b
    << " is " << result << endl
    << "All done!" ;
}
```

输出语句也做些变化，以便输出结果可读性更好。由于输出语句太长，可以将它分三行来写，但这还是一条语句，因为中间没有使用分号。C++是无格式的，这样书写程序不仅合法，而且容易被人阅读。

1.7 小结

本章简单地介绍了 C++的一些背景知识和特点，并通过几个简短的程序说明了 C++程序的结构，使读者对 C++程序的结构有个感性的认识，同时我们还介绍了变量的概念以及输入和输出操作。

练习

1－1　将本章给出的示例程序键入到计算机中，编译并运行这些程序。

1－2　下面程序的输出结果是什么？

```
#include <iostream.h>
main()
{
    cout << "This is the start of something wonderful!" ;
    cout << "And now we can say anything even more!" ;
}
```

1－3　程序常出现的错误是语法错误——程序的格式不符合C++的语法要求。考虑下面这个程序：

```
#include <iostream.h>
main()
  int i ;
  j = 5 + 20
  cout << i ;
}
```

编译这个程序，看一下编译器给出的错误信息。试着先修改第一个错误，然后看一下编译器给出的错误信息减少了多少。如果将程序中的#include编译指令去掉，结果又怎样呢？

1－4　当程序员使用以/*…*/标记的注释时，有时会遗漏与/*匹配的*/，考虑下面这个程序：

```
#include <iostream.h>
main()
{
  int i ;      /* this is a variable
  int j ;      /* this is another variable */
  i = 5 ;
  j = 10 ;
  cout << "i + j = " << i + j ;      /* output it
}
```

观察这个程序的编译错误信息，分析遗漏的*/对程序的影响。修改程序中的错误。

1－5　C++一般不允许在一个注释中再嵌套另一个注释，例如：

```
/*   this is a /* comment */ line */
```

试着将这条注释加到练习1－4的程序中，观察程序的编译情况。

1－6　编写一个计算三角形周长的程序：用户输入三角形的三条边长，程序输出计算结果。