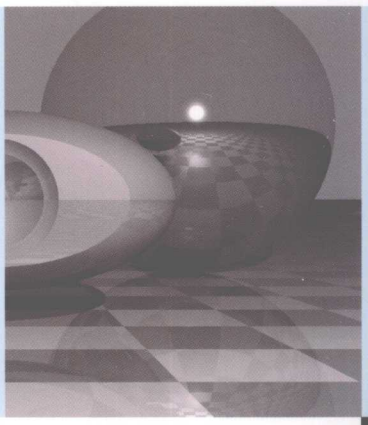
高等学校计算机教材

C++

面向对象实用教程



郑阿奇 主编 丁有和 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http: //www.phei.com.cn

高等学校计算机教材

C++面向对象实用教程

郑阿奇　主　编

丁有和　编　著

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书以读者学过 C 语言作为基础，系统地介绍 C++面向对象程序设计的基本概念和编程方法，包括 C++结构化程序设计、类和对象、数据共享和成员特性、继承和派生、多态、输入/输出流、模板和异常处理等。全书由教程、习题和实验部分组成。内容解释尽可能可视化，实例程序运行结果屏幕化，每一章都有综合应用实例，在同类书中具有一定特色。

本书可作为本科、高职高专计算机及相关专业 C++面向对象程序设计课程的教材，也可作为广大自学者的教材或参考书。

图书在版编目（CIP）数据

C++面向对象实用教程 / 郑阿奇主编.—北京：电子工业出版社，2009.2
（高等学校计算机教材）
ISBN 978-7-121-08190-3

Ⅰ. C… Ⅱ. 郑… Ⅲ. C 语言－程序设计－高等学校－教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2009）第 011288 号

策划编辑：赵云峰
责任编辑：左　雅
印　　刷：北京市顺义兴华印刷厂
装　　订：三河市双峰印刷装订有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：24　字数：614.4 千字
印　　次：2009 年 2 月第 1 次印刷
印　　数：5000 册　定价：33.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

目前，市场上以C语言为先修课程，系统介绍C++面向对象程序设计的书并不多，但以这种模式教学的学校并不少。我们以编写《C++实用教程》的经验为基础，结合教学实践，编写了这本《C++面向对象实用教程》。

本书基本思路如下：第1～3章在复习C语言的同时，介绍C++结构化程序设计；第4～9章系统介绍面向对象程序设计。除了第1章，每章都有综合应用实例，用于进一步消化前面介绍的内容。教程部分一般在讲解内容后紧跟示例，凡标有【例 Ex_Xxx】的均是一个完整的程序，且都上机调试通过。上机实验指导需要首先熟悉Visual C++ 6.0（SP6中文版）的编程环境，再训练C++结构化程序设计，为面向对象程序设计做好准备；然后分别训练面向对象程序设计各个方面的内容；综合应用实习对面向对象程序设计主要内容进行综合应用。

为了方便读者，本书还有以下特点：

（1）解释尽可能可视化，更便于理解。

（2）实例程序运行结果屏幕化，不可能出错，更便于准确把握。

（3）书中的源代码用底纹显示，对于C++语言的语法内容、运行结果、提示和讨论内容、图例等均采用具有立体阴影效果的方框来呈现，对于需要强调的文字内容则使用黑体来区分。另外，书中还用一些图标来修饰，如表示说明、提示和本书约定的内容，表示讨论的内容，表示该程序是Visual C++编译通过的。

本教程由电子工业出版社专门的平台华信教育资源网 http://www.huaxin.edu.cn（http://www.hxedu.com.cn）为读者提供服务，可免费下载教学课件，实例、实验和实习的源文件等。

本书由丁有和（南京师范大学）等编写，郑阿奇（南京师范大学）对全书进行统稿。其他很多同志对本书的编写提供了许多帮助，在此一并表示感谢！

参加本套丛书编写的有郑阿奇、梁敬东、顾韵华、王洪元、杨长春、丁有和、徐文胜、曹弋、刘启芬、殷红先、姜乃松、彭作民、张为民、郑进、王一莉、周怡君、刘毅等。

由于作者水平有限，不当之处在所难免，恳请读者批评指正。

编　者

目　录

第 1 章　C++基础 …… 1
1.1　从 C 到 C++ …… 1
1.2　C++程序创建 …… 2
1.3　C++程序结构 …… 4
1.3.1　main 函数 …… 4
1.3.2　头文件 …… 4
1.3.3　新头文件格式和名称空间 …… 5
1.3.4　注释 …… 6
1.4　C++简单程序设计 …… 6
1.4.1　数据和数据类型 …… 6
1.4.2　数据的基本输入/输出 …… 10
1.4.3　运算符和表达式 …… 12
1.4.4　基本语句 …… 16
1.4.5　编译预处理 …… 20
习题 …… 25
第 2 章　函数和作用域 …… 27
2.1　函数定义和调用 …… 27
2.1.1　函数定义 …… 27
2.1.2　函数的调用和声明 …… 28
2.1.3　值传递 …… 30
2.1.4　函数的默认形参值 …… 31
2.2　C++函数特性 …… 33
2.2.1　函数重载 …… 33
2.2.2　函数嵌套调用 …… 34
2.2.3　递归函数 …… 36
2.2.4　内联函数 …… 38
2.3　作用域和存储类型 …… 39
2.3.1　作用域 …… 39
2.3.2　域运算符 …… 41
2.3.3　存储类型 …… 42
2.4　名称空间 …… 47

2.4.1 名称空间的定义 47
2.4.2 名称空间的使用 49
2.5 综合应用实例：Fibonacci 数列 51
习题 53
第 3 章 指针和引用 56
3.1 指针的定义和操作 56
3.1.1 指针的定义和引用 56
3.1.2 指针的算术运算 58
3.1.3 const 指针 59
3.2 指针和数组 60
3.2.1 指针和一维数组 60
3.2.2 指针和二维数组 62
3.2.3 字符指针和字符串 64
3.3 指针和函数 65
3.3.1 指针作为函数的参数 65
3.3.2 返回指针的函数 69
3.3.3 指向函数的指针 70
3.3.4 带参数的 main 函数 72
3.4 动态内存和 void 指针 73
3.4.1 new 和 delete 运算符 74
3.4.2 void 指针 76
3.5 引用 76
3.5.1 引用的声明和操作 76
3.5.2 引用传递 80
3.5.3 返回引用 80
3.6 综合应用实例：josephus 问题 81
习题 83
第 4 章 类和对象 85
4.1 面向对象程序设计概念 85
4.2 类和对象 85
4.2.1 从结构到类 86
4.2.2 类的声明 87
4.2.3 对象的定义和成员的访问 90
4.2.4 类作用域和成员访问权限 91
4.3 构造函数和析构函数 95
4.3.1 构造函数 95
4.3.2 析构函数 100
4.3.3 new 和 delete 101

4.4 对象的使用 103
4.4.1 对象赋值和拷贝 103
4.4.2 浅拷贝和深拷贝 105
4.4.3 对象成员的初始化 108
4.4.4 const 对象 115
4.4.5 对象的生存期 115
4.5 综合应用实例：栈类设计 116
习题 121
第 5 章 数据共享和成员特性 125
5.1 静态成员 125
5.1.1 静态数据成员 125
5.1.2 静态成员函数 128
5.2 友元 131
5.2.1 友元概述 131
5.2.2 友元函数 132
5.2.3 友元类 135
5.3 成员其他特性 137
5.3.1 const 成员 137
5.3.2 mutable 成员 140
5.3.3 explicit 成员 141
5.4 this 指针 143
5.4.1 成员函数的效率 143
5.4.2 this 指针的实质 146
5.5 综合应用实例：栈类静态操作 148
习题 153
第 6 章 继承和派生 155
6.1 继承和派生概述 155
6.1.1 继承的概念 155
6.1.2 继承的特性 156
6.1.3 派生类的定义 157
6.2 继承方式 158
6.2.1 公有继承 158
6.2.2 私有继承 161
6.2.3 保护继承 164
6.2.4 不同继承方式的比较 164
6.3 派生类的构造和析构 165
6.3.1 构造和析构次序 166
6.3.2 派生类数据成员初始化 168

6.3.3 基类成员的访问 172
6.4 二义性和虚基类 172
6.4.1 二义性概述 172
6.4.2 二义性解决方法 176
6.4.3 虚基类和虚继承 177
6.5 兼容 180
6.5.1 赋值兼容规则 180
6.5.2 赋值兼容机理 181
6.6 综合应用实例：继承和组合类的设计 185
6.6.1 类间关系 185
6.6.2 设计实例 187
习题 190
第 7 章 多态 192
7.1 多态和虚函数 192
7.1.1 多态概述 192
7.1.2 虚函数机制 194
7.1.3 虚析构函数 197
7.1.4 纯虚函数和抽象类 199
7.2 运算符重载 201
7.2.1 运算符重载函数 201
7.2.2 运算符重载限制 203
7.2.3 友元重载 203
7.2.4 转换函数 206
7.3 典型运算符重载 207
7.3.1 赋值运算符的重载 208
7.3.2 自增自减运算符的重载 209
7.3.3 下标运算符重载 211
7.4 综合应用实例：简单链表类模型 213
习题 217
第 8 章 输入/输出流 219
8.1 概述 219
8.1.1 流和流类 219
8.1.2 标准流对象 220
8.1.3 提取和插入运算符重载 220
8.2 格式控制 222
8.2.1 设置输出宽度和填充字符 222
8.2.2 控制实数显示 223
8.2.3 左右对齐输出 224

8.3 使用输入/输出成员函数 225
8.3.1 输入操作的成员函数 225
8.3.2 输出操作的成员函数 227
8.3.3 流的错误处理 228
8.4 文件流 229
8.4.1 文件和文件流概述 229
8.4.2 文件流的使用方法 230
8.4.3 顺序文件操作 233
8.4.4 随机文件操作 234
8.5 标准 C++ string 类 239
8.5.1 string 构造和对象定义 239
8.5.2 string 类输入 239
8.5.3 string 的属性 241
8.5.4 string 常用操作 242
8.6 综合应用实例：文件操作 248
习题 252
第 9 章 模板和异常处理 254
9.1 函数模板 254
9.1.1 函数模板定义 254
9.1.2 函数模板实例化 255
9.1.3 函数模板具体化 258
9.1.4 函数模板重载 259
9.2 类模板 262
9.2.1 类模板的定义 262
9.2.2 类模板的实例化 264
9.3 标准模板库（STL） 265
9.3.1 迭代器（Iterator） 266
9.3.2 向量（vector）、链表（list）和双端队列（deque） 269
9.3.3 栈（stack）和队列（queue） 275
9.3.4 映像（map） 277
9.3.5 集合（set） 282
9.4 异常及其传统处理方法 285
9.4.1 判断函数返回值或形参 285
9.4.2 使用全局的标志变量 285
9.4.3 使用 exit 和 abort 286
9.5 使用 C++异常处理 286
9.5.1 try/throw/catch 结构 286
9.5.2 C++异常处理过程 288

9.5.3 嵌套异常和栈展开 290
9.6 综合应用实例：栈类模板设计 294
习题 297
实验部分 300
实验 1 认识 Visual C++ 6.0 中文版开发环境 300
实验 2 基本程序设计 1 312
实验 3 基本程序设计 2 317
实验 4 类和对象 326
实验 5 继承和派生 332
实验 6 多态 339
实验 7 输入/输出流 345
实验 8 模板和标准模板库 352
综合应用实习 356
附录 365
附录 A 常用 C++库函数及类库 365
附录 B ASCII 码表 367
附录 C 格式算子 369
附录 D 格式控制成员函数 370
附录 E 运算符优先级和结合性 371
附录 F 使用 typedef 372

第 1 章　C++基础

C++是在 C 语言的基础上逐步发展和完善起来的一门高级语言。

1.1　从 C 到 C++

C 语言是在 20 世纪 70 年代由 AT&T 贝尔实验室推行出来的，随着当时微型计算机的日益普及，出现了许多 C 语言版本，也出现了许多不一致的地方。为了改变这种情况，美国国家标准学会（ANSI）为 C 语言制定了一套 ANSI 标准，成为现行的 C 语言标准。

C 语言具有许多优点，比如语言简洁、灵活，运算符和数据结构丰富，具有结构化控制语句，程序执行效率高，同时具有高级语言和汇编语言的优点等。与其他高级语言相比，C 语言还具有可以直接访问物理地址的优点，与汇编语言相比又具有良好的可读性和可移植性。

但随着 C 语言应用的推广，C 语言存在的一些缺陷和不足也开始暴露出来了，并受到大家的关注。比如 C 语言对数据类型检查的机制比较弱；缺少支持代码重用的结构；随着软件工程规模的扩大，难以适应开发特大型程序。同时 C 语言毕竟是一种面向过程的编程语言，已经不能满足运用面向对象的方法开发软件的需要。**1980 年**，贝尔实验室的 Bjarne Stroustrup 为克服 C 语言本身存在的缺点，同时支持面向对象的程序设计，在 C 语言基础上研制出来一种通用的程序设计语言 C++。

为了使 C++具有良好的可移植性，1990 年，美国国家标准学会（American National Standards Institute，ANSI）设立了委员会（ANSI X3J16）专门负责制定 C++标准。很快，国际标准化组织（International Organization for Standardization，ISO）也成立了自己的委员会（ISO WG 21）。同年，ANSI 与 ISO 将两个委员会合并，统称为 **ANSI/ISO**，共同合作进行标准化工作。经过了长达 9 年的努力，C++的国际标准（ISO/IEC）在 1998 年获得了 ISO、国际电工技术委员会（International Electrotechnical Commission，IEC）和 ANSI 的批准，这是第一个 C++的国际标准 ISO/IEC 14882:1998，常称为 **C++ 98**、**标准 C++**或 **ANSI/ISO C++**。2003 年，发布了 C++标准第二版（ISO/IEC 14882:2003）。

本书以 C++标准第二版（ISO/IEC 14882:2003）为基础，但仍称之为 ANSI/ISO C++。

总之，从 1983 年 AT&T 的 Bell 实验室推出了它的 C++标准后，C++的应用已经广泛地深入到计算机的各个领域，并取得了很大的成功。对象的概念越来越多地被人们应用。面对越来越复杂的系统，越来越大型的软件，面向对象的程序设计从一个新的角度出发，力图通过使问题空间和解题空间保持更为有效的一致性，使得计算机软件更加有效和易于理解。作为一种成

功的面向对象程序设计语言，C++是人们对于客观世界进行清晰和准确描述的有效手段，C++使得软件开发者能方便地仿真客观世界。

同时，面向对象方法学的提出是程序设计思想的革命，很多专家学者都为了使得这种思想更完美实现而努力。不仅仅是软件在进行巨大变革，计算机硬件也受到这个思想的影响而发生变化。所以，C++也像当年的C语言一样，给软件设计带来一次新的巨大进步，它的应用会迅速地普及到计算机技术的各个领域，成为更有效的工具。

1.2 C++程序创建

使用C++高级语言编写的程序称为**源程序**。由于计算机只能识别和执行是由0和1组成的二进制指令，即**机器代码**，因而C++源程序是不能被计算机直接执行的，必须转换成机器代码才能被计算机执行。这个转换过程就是编译器对源代码进行**编译**和**连接**的过程，如图1.1所示。

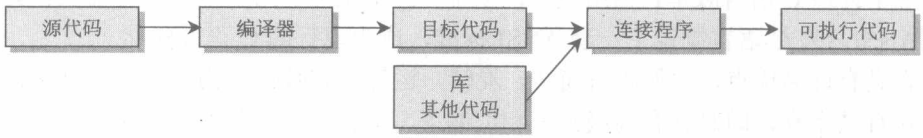


图1.1 C++程序创建过程

事实上，对于C++程序的源代码**编写**、**编译**和**连接**的步骤，许多C++编程工具软件商都提供了各自的C++集成开发环境（Integrated Devel opment Environment，IDE）用于程序的一体化操作。常见的有：Microsoft Visual C++、各种版本的Borland C++（如Turbo C++、C++ Builder等）、IBM Visual Age C++和bloodshed免费的Dev-C++等。这里仅来讨论用Visual C++ 6.0来创建C++程序的过程。

1. 创建工作文件夹

创建Visual C++ 6.0的工作文件夹“D:\C++程序”，以后所有创建的C++程序都在此文件夹下，这样既便于管理，又容易查找。在文件夹“D:\C++程序”下再创建一个子文件夹“第1章”用于存放第1章中的C++程序；第2章程序就存放在子文件夹“第2章”中，依次类推。

2. 启动Visual C++ 6.0

选择“开始”→“程序”→“Microsoft Visual Studio 6.0” →“Microsoft Visual C++ 6.0”，运行Visual C++ 6.0。第一次运行时，将显示如图1.2所示的“每日提示”对话框。单击[下一条]按钮，可看到有关各种操作的提示。如果在[启动时显示提示]复选框中单击鼠标，去除复选框的选中标记✔，那么下一次运行Visual C++ 6.0，将不再出现此对话框。单击[关闭]按钮关闭此对话框，进入Visual C++ 6.0开发环境。

3. 添加C++程序

（1）单击标准工具栏  上的“新建”（  ）按钮，打开一个新的文档窗口，在这个窗口中输入下列C++代码。

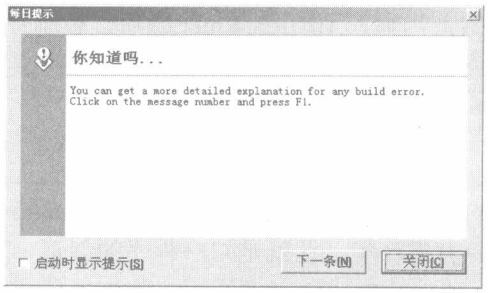


图 1.2 “每日提示”对话框

【例 Ex_Simple】一个简单的 C++程序

```
1   /* 第一个简单的 C++程序 */
2   #include <iostream.h>
3   int  main()
4   {
5       double  r,  area;                          // 定义变量
6       cout<<"输入圆的半径:";                     // 显示提示信息
7       cin>>r;                                    // 从键盘上输入变量 r 的值
8       area = 3.14159 * r * r;                    // 计算面积
9       cout<<"圆的面积为:"<<area<<"\n";           // 输出面积
10      return 0;                                  // 指定返回值
11  }
```

（2）选择“文件”→“保存”菜单，或按快捷键 Ctrl+S，或单击标准工具栏的按钮，弹出“保存为”文件对话框，将文件定位到“D:\C++程序\第 1 章”文件夹中，文件名指定为“Ex_Simple.cpp”（注意扩展名.cpp 不能省略，cpp 是“C Plus Plus”的缩写，即“C++”的意思）。

此时在文档窗口中所有代码的颜色都发生改变，这是 Visual C++ 6.0 的文本编辑器所具有的语法颜色功能，绿色表示注释，蓝色表示关键词等。

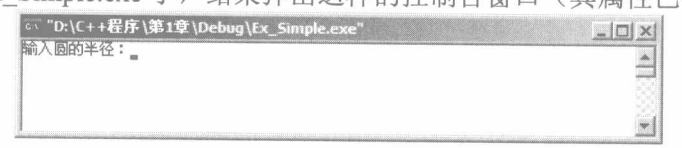
4. 编译和运行

（1）单击编译工具条上的生成工具按钮或直接按快捷键 F7，系统弹出一个对话框，询问是否为该程序创建默认的活动工作区间文件夹，单击[是]按钮，系统开始对 Ex_Simple 进行编译、连接，同时在输出窗口中显示编连的有关信息，当出现：

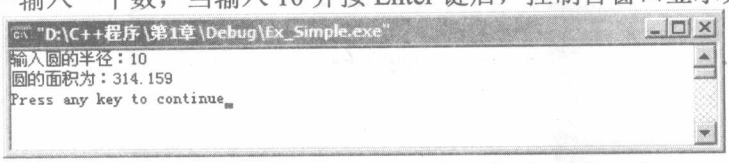
```
Ex_Simple.exe - 0 error(s), 0 warning(s)
```

表示 Ex_Simple.exe 可执行文件已经正确无误地生成了。

（2）单击编译工具条上的运行工具按钮或直接按快捷键 Ctrl+F5，就可以运行刚刚生成的 Ex_Simple.exe 了，结果弹出这样的控制台窗口（其属性已被修改过）。



此时等待用户输入一个数，当输入 10 并按 Enter 键后，控制台窗口显示如下。



其中，“Press any key to continue”是 Visual C++自动加上去的，表示 Ex_Simple 运行后，按任意键将返回到 Visual C++开发环境。这就是 C++程序的创建、编连和运行过程。

在以后的 C++程序运行结果中，本书不再完整显示其控制台窗口，也不再显示“Press any key to continue”，仅将控制台窗口中运行结果部分裁剪下来列出，并加以单线阴影边框，**本书作此约定。**

1.3 C++程序结构

一个程序是由若干个程序源文件组成的。为了与其他语言相区别，每一个 C++程序源文件通常是以.cpp（c plus plus，C++）为扩展名的，它是由编译预处理指令、数据或数据结构定义以及若干个函数组成。下面就以 Ex_Simple.cpp 的程序代码来分析 C++程序的组成和结构。

1.3.1 main 函数

代码中，main 表示**主函数**，由于每一个程序执行时都必须从 main 开始，而不管该函数在整个程序中的具体位置，因此每一个 C++程序或由多个源文件组成的 C++项目都必须包含一个且只有一个 main 函数。

在 main 函数代码中，“int main()”称为 main 函数的**函数头**。函数头下面是用一对花括号“{”和“}”括起来的部分，称为 main 函数的**函数体**。函数体中包括若干条语句（**按书写次序依次顺序执行**），每一条语句都由分号“;”结束。由于 main 函数名的前面有 int，它表示 main **函数的类型**是整型，须在函数体中使用关键字 return，用来将其后面的值作为函数的返回值。由于 return 语句运行后，函数体 return 后面的语句不再被执行，因此除非想要函数提前结束，否则 return 语句应写在函数体的最后。

main 函数体的第 1 条（**行号为 5**）语句是用来定义两个双精度实型（double）变量 r 和 area。第 2 条（**行号为 6**）语句是一条输出语句，它将双引号中的内容（即字符串）输出到屏幕上，cout 表示标准输出流对象（屏幕），“<<”是插入符，它将后面的内容插入到 cout 中，即输出到屏幕上。第 3 条（**行号为 7**）语句是一条输入语句，cin 表示标准输入流对象（键盘），“>>”是提取符，用来将用户输入的内容保存到后面的变量 r 中。最后一条（**行号为 9**）语句是采用多个“<<”将字符串和变量 area 的内容输出到屏幕中，后面的“\n”是换行符，即在内容输出后回车换行。

1.3.2 头文件

行号为 2 的语句的代码是 C++文件包含#include 的编译指令，称为**预处理指令**。#include

后面的 iostream.h 是 C++编译器自带的文件，称为 **C++库文件**，它定义了标准输入/输出流的相关数据及其操作。由于程序用到了输入/输出流对象 cin 和 cout，因而需要用#include 将其实现合并到程序中。又由于它们总是被放置在源程序文件的起始处，所以这些文件被称为**头文件**（Header File）。C++编译器自带了许多这样的头文件，每个头文件都支持一组特定的“工具”，用于实现基本输入/输出、数值计算、字符串处理等方面的操作。

在 C++中，头文件包含有两种格式。一是将文件名用尖括号“< >”括起来，用来包含那些由编译系统提供的并放在指定子文件夹中的头文件，这称为**标准方式**。二是将文件名用双引号括起来的方式，称为**用户方式**。这种方式下，系统先在用户当前工作文件夹中查找要包含的文件，若找不到再按标准方式查找（即再按尖括号的方式查找）。

1.3.3 新头文件格式和名称空间

ANSI/ISO C++不再采用早期 C/C++的头文件名形式，而是推荐使用无.h 扩展名的 C++头文件，并指定 std 名称空间。例如：

```
#include <iostream>
using namespace std;                    // 注意不要漏掉后面的分号
```

using 是一个在代码编译之前处理的指令。namespace 称为**名称空间**，它是 ANSI/ISO C++一个新的特性，用于解决在程序中同名标识存在的潜在危机。例如，程序中包含了两个头文件，而这两个头文件中都有一个已定义的函数 AFun，则编译不知道使用哪一个 AFun 函数。若在同一个作用域中，则还会出现“重复定义”的编译错误。采用名称空间，就可避免这些情况的发生。例如，若这两个头文件所定义的名称空间分别为 NA 和 NB（定义方法以后会讲到），则在程序调用时，只要使用作用域运算符“::”来指定相应的名称空间即可，如 NA::AFun(…)或 NA::BFun(…)。

又由于 iostream 是 ANSI/ISO C++标准组件库，它所定义的类、函数和变量均放入名称空间 std 中，因此需要在程序文件的开始位置处指定“using namespace std;”，以便能被后面的程序所使用。

事实上，cin 和 cout 就是 std 中已定义的流对象，若不使用“using namespace std;”，则还可有下列两种方式来指定。

第 1 种方式是在使用前用下列代码来指定。

```
using std::cout;                        // 指定以后的程序中可以使用 cout 对象
using std::cin;                         // 指定以后的程序中可以使用 cin 对象
```

第 2 种方式是在调用时指定它所属的名称空间，即如下述格式来使用。

```
std::cout<<"输入圆的半径:";             // ::是域作用运算符, 表示 cout 是 std 域中的对象
std::cin>>r;
```

显然，用下列两句代码来替代 C 语言风格的头文件包含#include <iostream.h>是一种最为简捷的做法，这也是本书所采用的方法。

```
#include <iostream>
using namespace std;
```

1.3.4 注释

程序 Ex_Simple 中的“/*...*/”之间的内容以及“//”开始一直到行尾的内容是用来注释的，它的目的只是为了提高程序的可读性，对编译和运行并不起作用。正是因为这一点，所注释的内容既可以用汉字来表示，也可以用英文来说明，只要便于理解就行。

一般来说，注释应在编程的过程中同时进行，不要指望程序编制完成后再补写注释。那样只会多花好几倍的时间，更为严重的是，时间长了以后甚至会读不懂自己写的程序。

通常，必要的注释应包含以下内容。

（1）在源文件头部进行必要的源程序的**总体注释**：版权说明、版本号、生成日期、作者、内容、功能、与其他文件的关系、修改日志等，头文件的注释中还应有函数功能简要说明。

（2）在函数的头部进行必要的**函数注释**：函数的目的/功能、输入参数、输出参数、返回值、调用关系（函数、表）等。

（3）**其他少量注释**：如全局变量的功能、取值范围等。千万不要陈述那些一目了然的内容，否则会使注释的效果适得其反。

需要说明的是，C++中的“/*...*/”是用来表示多行的注释，它是将由“/*”开头到“*/”结尾之间所有内容均视为注释，称为**块注释**。块注释的注解方式可以出现在程序中的任何位置，包括在语句或表达式之间。而“//”只能实现单行的注释，它是将“//”开始一直到行尾的内容作为注释，称为**行注释**。

1.4 C++简单程序设计

由于 C 语言是 C++的子集，因此 C++当然支持 C 语言中的**结构化程序设计**方式，这里就基本数据类型、运算符和表达式，以及结构化程序中的三种基本控制结构来阐述一下。

1.4.1 数据和数据类型

和 C 语言一样，C++的数据包括常量和变量两类，其类型也分为基本类型、派生类型以及复合类型三类。

1. 基本数据类型

C/C++的基本数据类型有字符型（char）、整型（int）和浮点型（float、double）三种。这些基本数据类型还可用 short、long、signed 和 unsigned 来修饰。表 1.1 列出了 C++中的基本数据类型，其字宽（以字节数为单位）和取值范围是在 Visual C++ 6.0 中的情况。

表 1.1　C++的基本数据类型

类 型 名	类 型 描 述	字　宽	范　围
char	字符型	1	–128~127
unsigned char	无符号字符型	1	0～255(0xff)
signed char	有符号字符型（与 char 相同）	1	–128～127
short [int]	短整型	2	–32 768～32 767
unsigned short [int]	无符号短整型	2	0～65 535(0xffff)
signed short [int]	有符号短整型（与 short int 相同）	2	–32 768～32 767
int	整型	4	–2 147 483 648～2 147 483 647
unsigned [int]	无符号整型	4	0～4 294 967 295(0xffffffff)
signed [int]	有符号整型（与 int 相同）	4	–2 147 483 648～2 147 483 647
long [int]	长整型	4	–2 147 483 648～2 147 483 647
unsigned long [int]	无符号长整型	4	0～4 294 967 295(0xffffffff)
signed long [int]	有符号长整型（与 long int 相同）	4	–2 147 483 648～2 147 483 647
float	单精度浮点型	4	3.4E–38～3.4E+38
double	双精度浮点型	8	1.7E–308～1.7E+308
long double	长双精度浮点型	8	1.7E–308～1.7E+308

注：表中[int]表示可以省略，即在 int 之前有 signed、unsigned、short、long 时，可以省略 int 关键字。

需要注意的是：

（1）C++还可以有**布尔型**（bool），即值为 true 或 false，事实上，在计算机内，编译系统将 true 表示成整数 1，false 表示成整数 0，因此也可把布尔型看成是一个整型。

（2）无符号（unsigned）和有符号（signed）的区别在于数值最高位的含义。对于 signed 类型来说，最高位是符号位，其余各位表示数值大小，而 unsigned 类型的各个位都用来表示数值大小，因此相同基本数据类型的 signed 和 unsigned 的数值范围是不同。例如，无符号字符型值的范围为 0～255，而有符号字符型值的范围为 −128～127。

（3）char、short、int 和 long 可统称为整型。默认时，char、short、int 和 long 本身是有符号（signed）的。

2. 常量

在程序运行过程中，其值不能被改变的量称为常量。常量可分为不同的类型，如 1、20、0、-6 为整型常量，1.2、-3.5 为浮点型常量，‘a’、‘b’为字符常量，“123”为字符串常量。常量一般从其字面形式即可判别。需要说明的是转义字符。

C/C++还可以用一个“\”开头的字符来表示特殊含义的字符常量。例如前例中“\n”，代表一个换行符，而不是表示字母 n。这种将反斜杠后面的字符转换成另外意义的方法称为转义表示法，“\n”称为转义字符。表 1.2 列出了常用的转义字符。

表 1.2　常用转义字符

字 符 形 式	含　义	字 符 形 式	含　义
\a	响铃	\’	单引号
\b	退格（相当于按 Backspace 键）	\”	双引号
\f	进纸（仅对打印机有效）	\\	反斜杠
\n	换行	\?	问号
\r	回车（相当于按 Enter 键）	\ooo	1 到 3 位八进制数所代表的字符
\t	水平制表（相当于按 Tab 键）	\xhh	1 到 2 位十六进制数所代表的字符
\v	垂直制表（仅对打印机有效）		