普通高等教育“十一五”国家级规划教材

C++面向对象程序设计教程

游洪跃 伍良富 王景熙 李培宇 主编
彭骏 谭斌 副主编
向孟光 谢汶 主审

清华大学出版社

21世纪计算机科学与技术实践型教程

丛书主编 陈明

普通高等教育“十一五”国家级规划教材

C++面向对象程序设计教程

游洪跃 伍良富 王景熙 李培宇 主编
彭骏 谭斌 副主编
向孟光 谢汶 主审

清华大学出版社
北京

21世纪计算机科学与技术实践型教程
丛书主编 陈明

内 容 简 介

全书共分为8章。阐述了C++的特点和开发过程；面向对象程序设计技术、类的定义、对象的创建及访问，友元与静态成员等基本内容；模板编程方法，运算符重载；C++的继承机制及虚基类，多态性，输入输出流，C++中的其他主题。

本书可作为高等院校计算机及相关专业“C++面向对象程序设计”课程的教材，也可供其他从事软件开发工作的读者参考使用。同时，也适合初学程序设计或有一定编程实践基础、希望突破编程难点的读者作为自学教材。通过本书的学习，读者能迅速提高C++面向对象程序设计的能力。

本书取材新颖，内容丰富，可读性强。本书充分考虑了读者对书中部分内容的心理适应性，对于一些容易让读者产生畏惧心理的内容作了适当的处理。本书所有程序都在Visual C++ 6.0、Visual C++ 2005、Visual C++ 2005 Express、Dev-C++和MinGW Developer Studio开发环境中进行了严格的测试，在作者教学网站上提供了大量的教学支持内容。

图书在版编目（CIP）数据

C++面向对象程序设计教程/游洪跃等主编. —北京：清华大学出版社，2010.3
（21世纪计算机科学与技术实践型教程）
ISBN 978-7-302-22058-9

Ⅰ. ①C… Ⅱ. ①游… Ⅲ. ①C语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2010)第022785号

责任编辑：汪汉友
责任校对：李建庄
责任印制：孟凡玉

出版发行：清华大学出版社
http://www.tup.com.cn
社　　总　　机：010-62770175
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京市清华园胶印厂
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185×260　　**印　张**：18　　**字　　数**：443千字
版　　次：2010年3月第1版　　**印　　次**：2010年3月第1次印刷
印　　数：1～4000
定　　价：28.00元

《21世纪计算机科学与技术实践型教程》

编辑委员会

《21世纪计算机科学与技术实践型教程》

序

21世纪影响世界的三大关键技术：以计算机和网络为代表的信息技术；以基因工程为代表的生命科学和生物技术；以纳米技术为代表的新型材料技术。信息技术居三大关键技术之首。国民经济的发展采取信息化带动现代化的方针，要求在所有领域中迅速推广信息技术，导致需要大量的计算机科学与技术领域的优秀人才。

计算机科学与技术的广泛应用是计算机学科发展的原动力，计算机科学是一门应用科学。因此，计算机学科的优秀人才不仅应具有坚实的科学理论基础，而且更重要的是能将理论与实践相结合，并具有解决实际问题的能力。培养计算机科学与技术的优秀人才是社会的需要、国民经济发展的需要。

制定科学的教学计划对于培养计算机科学与技术人才十分重要，而教材的选择是实施教学计划的一个重要组成部分，《21世纪计算机科学与技术实践型教程》主要考虑了下述两方面。

一方面，高等学校的计算机科学与技术专业的学生，在学习了基本的必修课和部分选修课程之后，立刻进行计算机应用系统的软件和硬件开发与应用尚存在一些困难，而《21世纪计算机科学与技术实践型教程》就是为了填补这部分空白。将理论与实际联系起来，使学生不仅学会了计算机科学理论，而且也学会应用这些理论解决实际问题。

另一方面，计算机科学与技术专业的课程内容需要经过实践练习，才能深刻理解和掌握。因此，本套教材增强了实践性、应用性和可理解性，并在体例上做了改进——使用案例说明。

实践型教学占有重要的位置，不仅体现了理论和实践紧密结合的学科特征，而且对于提高学生的综合素质，培养学生的创新精神与实践能力有特殊的作用。因此，研究和撰写实践型教材是必需的，也是十分重要的任务。优秀的教材是保证高水平教学的重要因素，选择水平高、内容新、实践性强的教材可以促进课堂教学质量的快速提升。在教学中，应用实践型教材可以增强学生的认知能力、创新能力、实践能力以及团队协作和交流表达能力。

实践型教材应由教学经验丰富、实际应用经验丰富的教师撰写。此系列教材的作者不但从事多年的计算机教学，而且参加并完成了多项计算机类的科研项目，他们把积累的经验、知识、智慧、素质融合于教材中，奉献给计算机科学与技术的教学。

我们在组织本系列教材过程中，虽然经过了详细的思考和讨论，但毕竟是初步的尝试，不完善甚至缺陷不可避免，敬请读者指正。

本系列教材主编　陈明

2005年1月于北京

前　言

作者使用过数本 C++ 面向对象程序设计的教材，发现不少问题，C++ 教学的普遍结果是，学生学完了 C++，但却不会使用目前流行的 C++ 开发工具编写程序。而且不少教材都存在错误。例如某 C++ 语言经典教材在关于打开文件的代码中出现了类似如下的代码：

```
ofstream outFile;                                    // 定义文件变量
if (outFile.open("test.txt", ios::app))              // 以追加方式打开文件
{    // 打开文件失败
     cout <<"打开文件失败!" <<endl;
     exit(1);                                        // 退出程序
}
```

上面代码完全不能通过编译，原因是文件流类的成员函数 open()返回值类型为 void，出现这些错误的原因是作者想当然按照 C 语言类似函数 fopen()编写代码，没有上机测试所写代码，至使学生看完书后还不能上机编程或上机编程非常困难，实际上只要上机运行很容易就能发现类似的错误及错误的原因，可按如下方式进行修改：

```
ofstream outFile;                                    // 定义文件变量
outFile.open("test.txt", ios::app);                  // 以追加方式打开文件
if (outFile.fail())
{    // 打开文件失败
     cout <<"打开文件失败!" <<endl;
     exit(1);                                        // 退出程序
}
```

书籍中存在错误是在所难免的，但是这种潜在错误对读者的影响是难以估量的。由于这类教材的读者面太大，读者很难有机会发现这种错误，并会一直延续这种错误的观念，这类问题在一些教材中存在了十多年，甚至最近的最新版也依然存在。

传统的 C++ 教学都过于注重解释 C++ 语言本身，而忽视了其在具体环境中的使用指导，例如对于如下的类声明及相关代码：

```
#include <iostream>                          // 编译预处理命令
using namespace std;                         // 使用命名空间 std

// 声明复数数
class Complex
{
private:
// 数据成员
    double real;                             // 实部
    double image;                            // 虚部
```

```
public:
// 公有函数
    Complex(double r = 0, double i = 0): real(r), image(i){}          // 构造函数
    friend Complex operator+(const Complex &z1, const Complex &z2)    // 复数加法
    { return Complex(z1.real+z2.real, z1.image+z2.image); }
    ...
};
```

上面的类声明及相关代码在 Visual C++ 2005、Visual C++ 2005 Express、Dev-C++ 4.9.9.2 和 MinGW Developer Studio 2.05 都能正常通过运行，但在 Visual C++ 6.0 SP6 下会出现编译时错误，是 Visual C++ 6.0 的一个 Bug，在 Visual C++ 6.0 中可将：

```
#include <iostream>                    // 编译预处理命令
using namespace std;                   // 使用命名空间 std
```

改为：

```
#include <iostream.h>                  // 编译预处理命令
#include <stdlib.h>                    // 包含 system()的声明
```

这时才可正常运行，又比如对于输入运算符“>>”和输出运算符“<<”重载为类的友元函数时，采用标准头文件 iostrteam，在 Visual C++ 6.0 SP6、Visual C++ 2005、Visual C++ 2005 Express、Dev-C++ 4.9.9.2 和 MinGW Developer Studio 2.05 中都不能通过编译，只能在 Visual C++ 6.0 中采用传统的头文件 iostream.h 才能通过编译。但将输入运算符“>>”和输出运算符“<<”重载为普通函数时无任何编译问题。

可惜的是，作者还没有发现哪本教材对上面类似的具体编程环境进行详细指导，这类教材无形中大大增加了学生应用 C++ 的难度。

本书作者经过十多年教学及查阅大量参考资料后编写了本书，全书共分为 8 章。第 1 章阐述 C++ 的主要特点及 C++ 程序开发过程，还详细介绍了 C++ 在非面向对象方面的常用新特性。第 2 章介绍了面向对象程序设计技术，C++ 类的定义、对象的创建以及对象成员的访问，友元与静态成员等基本内容。第 3 章介绍了模板编程方法，并对模板容易出现的编程问题进行详细的讨论。第 4 章介绍了运算符重载，重点对不同 C++ 编译器使用运算符重载时的兼容性问题进行具体指导。第 5 章着重介绍了 C++ 的继承机制及虚基类。第 6 章介绍了多态性，重点介绍了虚函数和抽象类。第 7 章介绍了输入输出流，重点讨论标准输入输出流类、文件操作与文件流类。第 8 章对 C++ 中的其他主题进行了深入阐述，这些主题都是难点，但都不是重点。如果这些内容在前面的章节中加以讨论，对于学生就会因难度过大而较难进入面向对象的思维模式，当学生已具备面向对象的思维习惯以后，再来介绍这些典型问题应该比较合适。

对于初学者，考试时往往会感到茫然而不知所措，因此本书习题包括了选择题、填空题和编译题。这些题目选自于考试题，可供学生期末复习，也可供教师编写试题时参考，对教师还提供了习题参考答案。

本书在部分章节中还提供了实例研究，主要提供给那些精力充沛的学生深入学习与研究，这些实例包括对正文内容的应用（例如第 6.4 节中栈的实现实际上就是抽象类的一个典

型应用，第 7.5 节中的简单工资管理系统就是文件操作的应用)、读者深入学习时可能会遇到的算法(例如第 3.4 节中的快速排序)以及应用所学知识解决实际问题(例如第 7.5 节中的简单工资管理系统就是文件操作实现简单信息管理系统)，通过读者对实例研究的学习，可提高实际应用 C++ 面向对象程序设计的能力，当然有一定的难度，但应比读者的想象更易学习与掌握。

为了尽快提高读者的实际编程能力，本书各章提供了程序陷阱，包含了在实际编程时容易出现的问题，也包括了正文内容的深入讨论，还包括了对 C++ 编译环境中存在兼容性问题进行了实用而具体的指导，这部分内容不管对初学者还是长期编程的人都很有用。

现在谈谈有关 C++ 编译器的问题，在 C++ 之外的任何编程语言中，编译器都没有受到过如此的重视。这是因为 C++ 是一门非常复杂的语言，以至于编译器也难于构造，我们常用的编译器都不能完全符合 C++ 标准，如下介绍一些常用的优秀 C++ 编译器。

(1) Visual C++ 编译器。由微软开发，现在主要流行 Visual C++ 6.0、Visual C++ 2005 以及 Visual C++ 2005 Express，特点是集成开发环境用户界面友好，信息提示准确，调试方便，对模板支持最完善；Visual C++ 6.0 对硬件环境要求低，现在安装的计算机最多，但对标准 C++ 兼容只有 83.43%，Visual C++ 2005 与 Visual C++ 2005 Express 在软件提示信息上做了进一步的优化与改进，并且对标准 C++ 兼容达到了 98%以上的程度，但对硬件的要求较高；还有 Visual C++ 2005 Express 是一种轻量级的 Visual C++ 软件，对于编程爱好者、学生和初学者来说是很好的编程工具。微软在 2006 年 4 月 22 日正式宣布 Visual Studio 2005 Express 版永久免费。

(2) GCC 编译器。著名的开源 C++ 编译器。是类 UNIX 操作系统(例如 Linux)下编写 C++ 程序的首选，有非常好的可移植性，可以在非常广泛的平台上使用，也是编写跨平台、嵌入式程序很好的选择。GCC 3.3 与标准 C++ 兼容能够达到 96.15%。现已有一些移植在 Windows 环境下使用 GCC 编译器的 IDE(集成开发环境)，例如 Dev-C++ 与 MinGW Developer Studio，其中 Dev-C++ 是能够让 GCC 在 Windows 下运行的集成开发环境，提供了与专业 IDE 相媲美的语法高亮、代码提示和调试等功能；MinGW Developer Studio 是跨平台下的 GCC 集成开发环境，目前支持 Windows、Linux 和 FreeBSD；根据作者的实际使用，感觉使用 GCC 编译器的 IDE 错误信息提示的智能较低，错误提示不太准确，还有就是对模板支持较差，但对语法检查较严格，在 Visual C++ 编译器中编译通过的程序可能在 GCC 编译器的 IDE 还会显示有错误信息。

本书所有算法都同时在 Visual C++ 6.0、Visual C++ 2005、Visual C++ 2005 Express、Dev-C++ 和 MinGW Developer Studio 中通过测试。读者可根据实际情况选择适当的编译器，建议选择 Visual C++ 6.0。

教师可采取多种方式来使用本书作为讲授 C++ 面向程序设计，应该根据学生的背景知识以及课程的学时数来进行内容的取舍。为满足不同层次的教学需求，本教材使用了分层的思想，分层方法如下：没有加星号“*”及双星号“**”的部分是基本内容，适合所有读者学习；加有星号“*”的部分适合计算机专业的读者作为深入学习的选学部分；加有双星号“**”的部分适合于感兴趣的读者研究。

作者为本书提供了全面的教学支持，如果在教学或学习过程中发现与本书有关的任何问题都可以与作者联系：youhongyue168@gmail.com，作者将尽力满足各位的要求，并可能

将解答公布在作者的教学网站 http://cs.scu.edu.cn/~youhongyue 上。在教学网站上将提供如下内容。

(1) 提供书中所有例题在 Visual C++ 6.0、Visual C++ 2005、Visual C++ 2005 Express、Dev-C++ 和 MinGW Developer Studio 开发环境中的测试程序，今后还会提供当时流行的 C++ 开发环境的测试程序，还提供本书作者开发的软件包。

(2) 提供教学用 PowerPoint 幻灯片 PPT 课件。

(3) 向教师提供所有习题参考答案，对学生来讲，将在每学期期末(第 15 周～第 20 周)公布解压密码。

(4) 提供高级语言程序设计问答专栏。

(5) 提供至少 6 套 C++ 面向对象程序设计模拟试题及其解答，以供学生期末复习，也可供教师出考题时参考。

(6) 提供 C++ 面向对象程序设计相关的其他资料(例如 Dev-C++ 与 MinGW Developer Studio 软件、流行免费 C++ 编译器的下载网址)。

希望各位读者能够抽出宝贵的时间，将对本教材的建议或意见寄给作者，你的意见将是我们再版修订教材的重要参考。

何凯霖、姜琳、聂清彬、黄维、游倩、邹昌文、王文昌、周焯华、胡开文、沈洁、周德华、欧阳、文涛、文芝明和文波等人对本书做了大量的工作，包括编写部分章节，提供资料，调试算法，参与了部分内容的编写，在此特向他们表示感谢；作者还要感谢为本书提供直接或间接帮助的每一位朋友，由于你们热情的帮助或鼓励激发了作者写好本书的信心和写作热情。

本书的出版要感谢清华大学出版社编辑，由于他们为本书的出版倾注了大量热情，也由于他们具有前瞻性的眼光才让读者有机会看到本书。

尽管作者有良好而负责任的严格态度，并尽了最大努力，但由于作者水平有限，书中难免有不妥之处，因此，敬请各位读者不吝赐教，以便作者有一个提高的机会，并在再版时尽力采用你们的意见，尽快提高本书的质量。

作　者

2010 年 2 月

目　录

第 1 章　C++ 程序设计基础

1.1　C++ 的发展和主要特点

1.1.1　C++ 的发展

C++ 程序设计语言是由来自 AT&T Bell Laboratories 的 Bjarne Stroustrup 设计和实现的，C++ 最初的版本称作“带类的 C”，在 1980 年第一次投入使用；支持面向对象程序设计的语言特性在 1983 年被加入到 C++ 中；从此之后，面向对象设计方法和面向对象程序设计技术逐渐进入了 C++ 领域。在 1987 年至 1989 年，模板程序设计技术加进了 C++ 。

随着 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(American National Standard Institute，美国国家标准化组织)以及后来加入的 ISO(International Standards Organization，国际标准化组织)负责。1998 年正式发布了 C++ 语言的国际标准。

1.1.2　C++ 的特点

1. 一个更好的 C

C++ 兼容 C，因此 C 程序在 C++ 环境下也可运行，会 C 的程序员，可在 C 的基础上逐步加入 C++ 的新特性，这样学起来更容易，对于要解决实际问题的程序员而言，使用 C++ 比使用 C 进行程序设计更有乐趣。

2. 支持面向对象程序设计

C++ 通过类实现了封装性、承继性和多态性面向对象程序设计技术，使 C++ 支持面向对象程序设计。

3. 支持范型程序设计

对范型程序设计的支持在 C++ 的后期才作为一个明确、独立的目标来实现。在 C++ 中，通过模板简单而实用地实现了范型程序设计技术。

1.2　第一个 C++ 程序以及 C++ 程序开发过程

1.2.1　第一个 C++ 程序

C++ 程序的结构严谨，下面介绍十分著名的“Hello，World!”程序，此程序一般用作介绍各种语言的第一个程序，其功能是在屏幕上输出字符串“Hello，World!”。

例 1.1　在屏幕上输出“Hello，World!”。

```
//文件路径名:e1_1\main.cpp
#include<iostream>                    //编译预处理命令
using namespace std;                  //使用命名空间 std
```

```
int main()                              //主函数 main()
{
    cout<<"Hello, World!"<<endl;  //用 C++的方法输出一行

    system("PAUSE");                    //调用库函数 system(),输出系统提示信息
    return 0;                           //返回值 0, 返回操作系统
}
```

程序运行时屏幕输出如下：

```
Hello, World!
请按任意键继续...
```

为了使读者更好地理解，下面详细地剖析上面的程序。

1. 注释

上面程序的第一行如下：

```
//文件路径名:e1_1\main.cpp
```

这一行不是程序代码，它只是注释，告诉读者程序的文件路径名，位于“//”后面的文本都是注释。读者应养成给程序添加注释的习惯，在 C++ 程序中，可以使用 C 语言中“/*……*/”形式的注释，还可以使用以“//”开头的注释。

注意：以“//”开头的注释可以不单独占一行，可以出现在一行中的语句之后。编译系统将“//”以后到本行末尾的所有字符都作为注释。“//”开头的注释是单行注释，不能跨行。

2. 输出信息

例 1.1 中的 main()函数体包含了如下的语句：

```
cout<<"Hello, World!"<<endl;      //用 C++的方法输出一行
```

在 C++ 程序中，一般都用 cout 输出信息，它是C++ 用于输出的语句。cout 实际上是 C++ 系统预定义的对象名，称为标准输出流对象。为便于理解，将 cout 和“<<”实现输出的语句简称为 cout 语句。“<<”是“输出运算符”，“<<”应与 cout 配合使用，在上面的代码中将运算符“<<”右侧双引号内的字符串“Hello, World!”插入到输出流中，“<<”还有一个输出控制符 endl，用于表示换行，endl 也将被插入到输出流中，C++ 系统将输出流的内容输出到系统指定的设备(一般为显示器)上。除了可以用 cout 进行输出外，在 C++ 中也可以用 C 函数 printf()进行输出。

main()函数中还包含了调用 system()函数的语句：

```
system("PAUSE"); //调用库函数 system(),输出系统提示信息
```

system("PAUSE")函数调用将使程序暂停，以便用户观察执行结果。

3. 预处理命令和命名空间 std

cout 需要用到头文件 iostream。程序中如下的代码行：

```
#include<iostream>                  //编译预处理命令
```

是一个预处理命令，文件 iostream 的内容提供输入或输出时所需要的一些信息。这类文件都放在程序单元的开头，所以称为“头文件”(head file)。

注意：在 C 语言中所有的头文件都带后缀.h(如 stdlib.h)，按 C++ 标准要求，由系统提供的头文件不带后缀.h，用户自己编制的头文件可以有后缀.h。在 C++ 程序中也可以使用 C 语言编译系统提供的带后缀.h 的头文件，如“#include <stdio.h>”。

程序的如下代码：

```
using namespace std;                    //使用命名空间 std
```

表示使用命名空间 std。C++ 标准库中的类和函数是在命名空间 std 中声明的，程序中如果需要使用 C++ 标准库中的有关内容可用“using namespace std;”语句作声明，表示要用到命名空间 std 中的内容。

4. 定义 main()函数

下面的代码行定义了 main()函数：

```
int main()                              //主函数 main()
{
    cout<<"Hello, World!"<<endl;        //用 C++的方法输出一行

    system("PAUSE");                    //调用库函数 system( ),输出系统提示信息
    return 0;                           //返回值 0, 返回操作系统
}
```

所有 C++ 程序都由一个或多个函数组成，因为每个程序总是从这个 main()函数开始执行，所以每个 C++ 程序都必须有一个 main()函数。

定义 main()函数的第一行代码如下：

```
int main()                              //主函数 main()
```

这行代码是 main()函数的起始，其中的 int 表示 main()函数的返回值的类型，int 表示 main()函数返回一个整数值，执行完 main()函数后将返回给操作系统，它表示程序的状态。在下面的语句中，指定了执行完 main()函数后要返回的值：

```
return 0;                               //返回值 0, 返回操作系统
```

这个 return 语句结束 main()函数的执行，把值 0 返回给操作系统。main()函数通常用返回 0 表示程序正常终止，而返回非 0 值表示发生了异常，也就是在程序结束时，发生了不应发生的事情。

标准 C++ 要求 main()函数必须声明为 int 型。有的操作系统(如 Linux)要求执行一个程序后必须向操作系统返回一个数值。在目前使用的一些 C++ 编译系统并未完全执行 C++ 这一规定，如果主函数首行写成“void main()”也能通过，本书中的所有例题都按标准 C++ 规定写成“int main()”。

1.2.2 C++ 程序开发过程

一个程序从编写到最后得到运行结果要经历编写源程序、编译、链接和运行 4 个步骤，

下面分别加以介绍。

1. 编写源程序

程序是一组计算机系统能识别和执行的指令。每一条指令使计算机执行特定的操作，用高级语言编写的程序称为源程序(source program)。C++ 的源程序以.cpp 作为后缀(cpp 是 c plus plus 的缩写)。

2. 编译

编译工作是由编译器完成的。C++ 代码不能被机器直接识别，首先需要将 C++ 程序代码转换为机器代码。编译过程所做的工作就是把 C++ 程序代码翻译成机器认识的机器代码的过程，编译后得到的机器代码文件称为目标文件。

编译时将对源程序进行词法检查和语法检查。词法检查是检查源程序中的单词拼写是否有错，例如把 main 错拼为 mian。语法检查根据源程序的上下文来检查程序的语法是否有错，例如在 cout 语句中输出变量 x 的值，但是在前面并没有定义变量 x。编译时对文件中的全部内容进行检查，编译结束后显示出所有的编译出错信息。出错信息一般分为两种：

(1) 错误(error)：存在这类错误就不生成目标文件，必须改正后重新编译。

(2) 警告(warning)：一些不影响运行的轻微的错误。

3. 链接

经编译后得到的目标文件中的机器代码是相互独立的，需要链接器将它们组合在一起，此时要用系统提供的“连接程序(linker)”将一个程序的所有目标文件和系统的库文件以及系统提供的其他信息连接起来，最终形成一个可执行的二进制文件，它的后缀是.exe。

4. 运行

完成链接后将得到一个可执行文件，可以直接运行。运行后，就可以得到程序结果。图 1.1 描述了从编译到运行的整个过程。

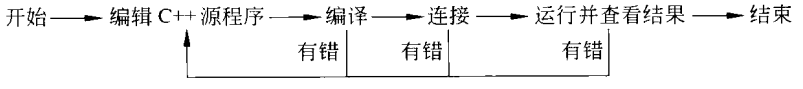

图 1.1　C++ 程序开发过程

注意：当前的 C++ 语言开发环境中都集成了以上 4 个步骤，大大方便了 C++ 语言的开发工作。附录 B 讨论了常用 C++ 语言开发环境的使用方法。

读者最好尽早在计算机上编译和运行 C++ 程序，以便加深对 C++ 程序的认识。只靠课堂和看书是难以真正掌握 C++ 的所有知识及其应用的。希望读者善于在实践中学习。

请读者至少选择一种(如能做到两种更好)C++ 编译系统，在该环境下输入和运行例题和习题中的程序。

1.3　C++ 在非面向对象方面的常用新特性

C++ 是从 C 发展而来的，C++ 对 C 引入了面向对象的新概念，同时也增加一些非面向对象的新特性，这些特性使 C++ 使用起来更方便、更安全，本节将讨论一些常用新特性。