

SAMS

Teach Yourself

C++ in **21** Days

经典 C++ 教程
全美销量超过 25 万册

21天学通

C++

(第五版)

[美] Jesse Liberty Bradley Jones 著

李佩乾 杨小珂 译



人民邮电出版社
POSTS & TELECOM PRESS

21 天学通 C++ (第五版)

[美] Jesse Liberty Bradley Jones 著

李佩乾 杨小珂 译

人民邮电出版社

图书在版编目 (CIP) 数据

21 天学通 C++: 第五版 / (美) 利伯帝 (Liberty, J.), (美) 琼斯 (Jones, B.) 著; 李佩乾, 杨小珂译. —北京: 人民邮电出版社, 2005.9

ISBN 7-115-13692-0

I. 2... II. ①利...②琼...③李...④杨... III. C 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2005) 第 097601 号

版 权 声 明

Jesse Liberty, Bradley Jones: Sams Teach Yourself C++ in 21 Days, Fifth Edition

ISBN: 0672327112

Copyright © 2005 by Sams Publishing.

Authorized translation from the English language edition published by Sams.

All rights reserved.

本书中文简体字版由美国 Sams 出版公司授权人民邮电出版社出版。未经出版者书面许可, 对本书任何部分不得以任何方式复制或抄袭。

版权所有, 侵权必究。

21 天学通 C++ (第五版)

-
- ◆ 著 [美] Jesse Liberty Bradley Jones
 - 译 李佩乾 杨小珂
 - 责任编辑 李 际
 - ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
 - 邮编 100061 电子函件 315@ptpress.com.cn
 - 网址 <http://www.ptpress.com.cn>
 - 北京顺义振华印刷厂印刷
 - 新华书店总店北京发行所经销
 - ◆ 开本: 787×1092 1/16
 - 印张: 40.5
 - 字数: 1 317 千字 2005 年 9 月第 1 版
 - 印数: 1—5 000 册 2005 年 9 月北京第 1 次印刷

著作权合同登记号 图字: 01-2004-5456 号

ISBN 7-115-13692-0/TP · 4812

定价: 59.00 元

读者服务热线: (010) 67132705 印装质量热线: (010) 67129223

作者简介

Jesse Liberty 编著了大量有关软件开发的图书，其中包括 C++ 和 .NET 方面的畅销书。他是 Liberty Associates 公司 (<http://www.LibertyAssociates.com>) 的总裁，该公司致力于为客户提供编程、咨询和培训方面的服务。

Bradley Jones 是 Microsoft Visual C++ MVP，他身兼网站管理员、经理、编码大师、执行编辑等职，其主要精力放在众多软件开发网站和频道上，其中包括 Developer.com、CodeGuru.com、DevX、VBForums、Gamelan 以及 Jupitermedia 的其他网站。这些影响力在不断扩大的网站每月为 250 万开发人员提供信息。

他最擅长的是大 C 语言领域：C、C++ 和 C#，但在 PowerBuilder、VB、Java、ASP 和 COBOL I/II 开发以及各种现在没有人提起的古老技术方面也拥有丰富的经验。他还从事过咨询人员、分析员、项目负责人、联合发行人和作者等工作，他最近参与编著的图书包括《21 天学通 C#》和《21 天学通 C 语言（第六版）》等。他还是 Indianapolis .NET Developers Association（成员超过 700 的 INETA 集团的一家连锁公司）的联合创始人兼总裁。你经常可以在 CodeGuru.com 和 VBForums.com 论坛上看到他发表的帖子，他还每周在 CodeGuru 上发表一篇通信稿，其读者为数以万计的开发人员。

本书旨在帮助读者学习如何使用 C++ 进行编程。没有人仅在三周内就能学好一种严谨的编程语言，但本书每章的内容都可以在几小时内阅读完毕。

只需 21 天，读者就能学习诸如控制输入/输出、循环和数组、面向对象编程、模板和创建 C++ 应用程序等基本知识，所有这些内容都被组织成结构合理、易于理解的章节。每章都提供范例程序清单，并辅以范例输出和代码分析以演示该章介绍的主题。

为加深读者对所学内容的理解，每章最后都提供了常见问题及其答案以及测验和练习。读者可对照附录 D 提供的测验和练习答案，了解自己对所学内容的掌握程度。

本书针对的读者

通过阅读本书来学习 C++ 时，读者不需要有任何编程经验。本书从入门开始，既介绍 C++ 语言，又讨论使用 C++ 进行编程涉及的概念。本书提供了大量语法实例和详细的代码分析，它们是引导读者完成 C++ 编程之旅的优秀向导。无论读者是刚开始学习编程还是已经有一些编程经验，书中精心安排的内容都将让你的 C++ 学习变得既快速又轻松。

本书约定

提示：提供使读者进行 C++ 编程时更高效、更有效的信息。

注意：提供与读者阅读的内容相关的信息。

FAQ：对 C++ 语言的用法进行了深入剖析，澄清一些容易混淆的问题。

警告：提醒读者注意在特定情况下可能出现的问题或副作用。

应该：提供当前章介绍的基本原理的摘要。

不应该：提供一些有用的信息。

在程序清单中，在每行代码中都加上了行号；没有行号的代码行是前一行的续行（有些代码行太长，无法在一行中列出）。这种情况下，应将两行作为一行输入，不能将它们分开。

本书的范例代码

本书正文及附录 D 中的范例代码可从 Sams 网站下载, 其网址为 <http://www.sampublishing.com>。在文本框“Search”中输入本书英文版的 ISBN (0672327112), 单击 Search 按钮, 然后单击原版书名 (Sams Teach Yourself C++ in 21 Days, 5th Edition) 便可链接到可下载范例代码的页面, 点击 Downloads 即可下载。

第1周课程简介

第1章 绪论.....	2
1.1 C++简史.....	2
1.1.1 解决问题.....	3
1.1.2 过程化编程、结构化编程和面向对象编程.....	3
1.1.3 面向对象编程(OOP).....	4
1.1.4 C++和面向对象编程.....	4
1.2 C++的发展历程.....	5
1.3 应该先学习C语言吗.....	5
1.4 C++、Java和C#.....	5
1.5 微软的C++可控扩展.....	5
1.6 ANSI标准.....	5
1.7 编程准备.....	6
1.8 开发环境.....	6
1.9 创建程序的步骤.....	7
1.9.1 用编译器生成对象文件.....	7
1.9.2 用链接器生成可执行文件.....	7
1.10 程序开发周期.....	7
1.11 HELLO.cpp: 第一个C++程序.....	8
1.12 编译器初步.....	9
1.13 编译错误.....	10
1.14 小结.....	10
1.15 问与答.....	10
1.16 作业.....	11
1.16.1 测验.....	11
1.16.2 练习.....	11
第2章 C++程序的组成部分.....	12
2.1 一个简单程序.....	12
2.2 cout简介.....	13
2.3 使用标准名称空间.....	15
2.4 对程序进行注释.....	16
2.4.1 注释的类型.....	17
2.4.2 使用注释.....	17
2.4.3 有关注释的警告.....	18

2.5 函数	18	4.3.1 赋值运算符	40
2.5.1 使用函数	19	4.3.2 数学运算符	40
2.5.2 方法和函数	20	4.3.3 整数除法和求模	41
2.6 小结	20	4.4 赋值运算符与数学运算符的组合	42
2.7 问与答	20	4.5 递增和递减	42
2.8 作业	21	4.6 理解运算符优先级	44
2.8.1 测验	21	4.7 括号的嵌套	45
2.8.2 练习	21	4.8 真值的本质	45
第3章 使用变量和常量	22	4.9 if 语句	46
3.1 什么是变量	22	4.9.1 缩进风格	49
3.1.1 将数据存储在内中	22	4.9.2 else 语句	49
3.1.2 预留内存	22	4.9.3 高级 if 语句	51
3.1.3 整型变量的大小	23	4.10 在嵌套 if 语句中使用大括号	52
3.1.4 基本变量类型	24	4.11 使用逻辑运算符	54
3.2 定义变量	25	4.11.1 逻辑 AND 运算符	54
3.2.1 区分大小写	26	4.11.2 逻辑 OR 运算符	55
3.2.2 命名规则	26	4.11.3 逻辑 NOT 运算符	55
3.2.3 关键字	26	4.12 简化求值	55
3.3 一次创建多个变量	27	4.13 关系运算符的优先级	55
3.4 给变量赋值	27	4.14 再谈真和假	56
3.5 使用 typedef 来创建别名	28	4.15 条件运算符 (三目运算符)	56
3.6 何时使用 short 和 long	29	4.16 小结	57
3.6.1 unsigned 整型变量的回绕	30	4.17 问与答	58
3.6.2 signed 整型变量的回绕	30	4.18 作业	58
3.7 使用字符	31	4.18.1 测验	58
3.7.1 字符和数字	32	4.18.2 练习	59
3.7.2 特殊打印字符	32	第5章 组织成函数	60
3.8 常量	33	5.1 什么是函数	60
3.8.1 字面常量	33	5.2 返回值、参数和实参	61
3.8.2 符号常量	33	5.3 声明和定义函数	61
3.9 枚举常量	34	5.3.1 函数原型	62
3.10 小结	36	5.3.2 定义函数	62
3.11 问与答	36	5.4 函数的执行	64
3.12 作业	37	5.5 确定变量的作用域	64
3.12.1 测验	37	5.5.1 局部变量	64
3.12.2 练习	37	5.5.2 作用域为语句块的局部变量	65
第4章 创建表达式和语句	38	5.6 参数是局部变量	66
4.1 语句简介	38	5.6.1 全局变量	67
4.1.1 使用空白	38	5.6.2 有关全局变量的注意事项	69
4.1.2 语句块和复合语句	38	5.7 创建函数语句时的考虑因素	69
4.2 表达式	39	5.8 再谈函数实参	69
4.3 使用运算符	40	5.9 再谈返回值	70

5.10 默认参数.....	71	地方.....	101
5.11 重载函数.....	73	6.11 内联实现.....	102
5.12 函数特有的主题.....	76	6.12 将他类用作成员数据的类.....	104
5.12.1 内联函数.....	76	6.13 结构.....	107
5.12.2 递归.....	77	6.14 小结.....	108
5.13 函数的工作原理.....	81	6.15 问与答.....	108
5.13.1 抽象层次.....	81	6.16 作业.....	109
5.13.2 划分 RAM.....	81	6.16.1 测验.....	109
5.13.3 堆栈和函数.....	83	6.16.2 练习.....	109
5.14 小结.....	83	第 7 章 再谈程序流程.....	111
5.15 问与答.....	83	7.1 循环.....	111
5.16 作业.....	84	7.1.1 循环的鼻祖: goto.....	111
5.16.1 测验.....	84	7.1.2 为何避免使用 goto 语句.....	112
5.16.2 练习.....	84	7.2 使用 while 循环.....	112
第 6 章 理解面向对象编程.....	86	7.2.1 更复杂的 while 语句.....	113
6.1 C++是面向对象的吗.....	86	7.2.2 continue 和 break 简介.....	114
6.2 创建新类型.....	86	7.2.3 while(true)循环.....	116
6.3 类和成员简介.....	87	7.3 实现 do...while 循环.....	117
6.3.1 声明类.....	88	7.4 使用 do...while.....	118
6.3.2 有关命名规则的说明.....	88	7.5 for 循环.....	120
6.3.3 定义对象.....	88	7.5.1 高级 for 循环.....	122
6.3.4 类与对象.....	88	7.5.2 空 for 循环.....	124
6.4 访问类成员.....	89	7.5.3 循环嵌套.....	124
6.4.1 给对象而不是类赋值.....	89	7.5.4 for 循环中声明的变量的 作用域.....	126
6.4.2 类不能有没有声明的功能.....	89	7.6 循环小结.....	126
6.5 私有和公有.....	90	7.7 使用 switch 语句控制程序流程.....	128
6.6 实现类方法.....	94	使用 switch 语句来处理菜单.....	130
6.7 添加构造函数和析构函数.....	96	7.8 小结.....	133
6.7.1 默认构造函数和析构函数.....	96	7.9 问与答.....	133
6.7.2 使用默认构造函数.....	96	7.10 作业.....	134
6.8 const 成员函数.....	99	7.10.1 测验.....	134
6.9 接口与实现.....	99	7.10.2 练习.....	134
6.10 将类声明和方法定义放在什么			

第 1 周复习

第 2 周课程简介

第 8 章 理解指针.....	144	8.1.4 指针名.....	146
8.1 什么是指针.....	144	8.1.5 获取指针指向的变量的值.....	146
8.1.1 内存简介.....	144	8.1.6 使用间接运算符解除引用.....	147
8.1.2 获取变量的内存地址.....	144	8.1.7 指针、地址和变量.....	147
8.1.3 将变量的地址存储到指针中.....	145	8.1.8 使用指针来操纵数据.....	148

8.1.9 查看地址	149	9.15.1 测验	190
8.2 为什么使用指针	151	9.15.2 练习	191
8.3 栈和自由存储区(堆)	151	第 10 章 有关函数的高级主题	192
8.3.1 使用关键字 new 来分配内存	152	10.1 重载成员函数	192
8.3.2 使用关键字 delete 归还内存	152	10.2 使用默认值	194
8.4 再谈内存泄漏	154	10.3 在默认值和重载函数之间做出选择	196
8.5 在自由存储区上创建对象	154	10.4 默认构造函数	196
8.6 删除自由存储区中的对象	155	10.5 重载构造函数	196
8.7 访问数据成员	156	10.6 初始化对象	198
8.8 在自由存储区中创建成员数据	157	10.7 复制构造函数	199
8.9 this 指针	159	10.8 运算符重载	202
8.10 迷途指针	160	10.8.1 编写一个递增函数	203
8.11 使用 const 指针	162	10.8.2 重载前缀运算符	203
8.11.1 const 指针和 const 成员函数	162	10.8.3 运算符重载函数的返回类型	205
8.11.2 使用 const this 指针	164	10.8.4 返回无名临时对象	206
8.12 小结	164	10.8.5 使用 this 指针	207
8.13 问与答	165	10.8.6 重载后缀运算符	209
8.14 作业	165	10.8.7 前缀和后缀之间的差别	209
8.14.1 测验	165	10.8.8 重载双目数学运算符	210
8.14.2 练习	165	10.8.9 运算符重载中存在的问题	213
第 9 章 使用引用	167	10.8.10 对运算符重载的限制	213
9.1 什么是引用	167	10.8.11 重载什么	214
9.2 将地址运算符用于引用	168	10.8.12 赋值运算符	214
9.3 引用对象	170	10.9 处理数据类型转换	216
9.4 空指针和空引用	172	10.10 转换运算符	218
9.5 按引用传递函数参数	172	10.11 小结	219
9.5.1 使用指针让 swap() 管用	173	10.12 问与答	220
9.5.2 使用引用来实现 swap()	174	10.13 作业	220
9.6 理解函数头和原型	176	10.13.1 测验	220
9.7 返回多个值	176	10.13.2 练习	221
9.8 按引用传递以提高效率	179	第 11 章 面向对象分析及设计	222
9.8.1 传递 const 指针	181	11.1 建立模型	222
9.8.2 用引用代替指针	183	11.2 软件设计: 建模语言	222
9.9 何时使用引用和指针	185	11.3 软件设计: 过程	223
9.10 混合使用引用和指针	186	11.3.1 迭代式开发和瀑布式开发	223
9.11 返回指向不在作用域中的对象的 引用	186	11.3.2 迭代式开发过程	224
9.12 指针归谁所有	189	11.4 第 1 步: 概念化阶段——从愿景 开始	225
9.13 小结	190	11.5 第 2 步: 分析阶段——收集需求	225
9.14 问与答	190	11.5.1 用例	225
9.15 作业	190	11.5.2 应用分析	232
		11.5.3 系统分析	232

11.5.4 规划文档	233	13.1.3 护栏柱错误	277
11.5.5 可视化	233	13.1.4 初始化数组	277
11.5.6 可交付品	233	13.1.5 声明数组	278
11.6 第3步: 设计阶段	234	13.2 使用对象数组	279
11.6.1 什么是类	234	13.2.1 声明多维数组	280
11.6.2 转换	235	13.2.2 初始化多维数组	281
11.6.3 其他转换	235	13.3 指针数组	282
11.6.4 建立静态模型	236	13.4 指针算术	284
11.6.5 动态模型	241	13.5 在自由存储区声明数组	286
11.7 第4~6步: 实现、测试和交付	243	13.5.1 数组指针和指针数组	286
11.8 迭代	244	13.5.2 指针和数组名	287
11.9 小结	244	13.5.3 删除自由存储区中的数组	288
11.10 问与答	244	13.5.4 在运行阶段调整数组大小	288
11.11 作业	244	13.6 字符数组和字符串	291
11.11.1 测验	245	13.7 使用方法 strcpy() 和 strncpy()	292
11.11.2 练习	245	13.8 String 类	294
第12章 实现继承	246	13.9 链表和其他结构	299
12.1 什么是继承	246	13.10 创建数组类	300
12.1.1 继承和派生	246	13.11 小结	300
12.1.2 动物世界	247	13.12 问与答	300
12.1.3 派生的语法	247	13.13 作业	301
12.2 私有和保护	248	13.13.1 测验	301
12.3 构造函数和析构函数的继承性	250	13.13.2 练习	301
12.4 覆盖基类函数	256	第14章 多态	302
12.4.1 隐藏基类方法	258	14.1 单继承存在的问题	302
12.4.2 调用基类方法	259	14.1.1 提升	304
12.5 虚方法	261	14.1.2 向下转换	304
12.5.1 虚函数的工作原理	264	14.1.3 将对象添加到链表中	306
12.5.2 通过基类指针访问派生类的方法	265	14.2 多重继承	307
12.5.3 切除	265	14.2.1 多重继承对象的组成部分	310
12.5.4 创建虚析构函数	267	14.2.2 多重继承对象中的构造函数	310
12.5.5 虚复制构造函数	267	14.2.3 歧义解析	312
12.5.6 使用虚方法的代价	270	14.2.4 从共同基类继承	313
12.6 小结	270	14.2.5 虚继承	316
12.7 问与答	271	14.2.6 多重继承存在的问题	319
12.8 作业	271	14.2.7 混合(功能)类	320
12.8.1 测验	271	14.3 抽象数据类型	320
12.8.2 练习	271	14.3.1 纯虚函数	323
第13章 管理数组和字符串	273	14.3.2 实现纯虚函数	324
13.1 什么是数组	273	14.3.3 复杂的抽象层次结构	327
13.1.1 访问数组元素	273	14.3.4 哪些类是抽象的	330
13.1.2 在数组末尾后写入数据	275	14.4 小结	331
		14.5 问与答	331

14.6 作业	332
14.6.1 测验	332

14.6.2 练习	332
-----------------	-----

第2周复习

第3周课程简介

第15章 特殊类和函数.....344

15.1 在同一种类型的对象之间共享 数据: 静态成员数据	344
15.2 静态成员函数	348
15.3 函数指针	350
15.3.1 为什么使用函数指针	353
15.3.2 函数指针数组	356
15.3.3 将函数指针传递给其他函数	358
15.3.4 将 typedef 用于函数指针	360
15.4 成员函数指针	363
15.5 小结	367
15.6 问与答	367
15.7 作业	367
15.7.1 测验	367
15.7.2 练习	368

第16章 高级继承.....369

16.1 聚合	369
16.1.1 访问被聚合类的成员	375
16.1.2 控制对被聚合成员的访问	375
16.1.3 聚合的代价	376
16.1.4 按值传递导致复制	378
16.2 以继承方式实现和聚合/代理	381
16.3 私有继承	390
16.4 添加友元类	398
16.5 友元函数	406
16.6 友元函数和运算符重载	406
16.7 重载插入运算符	410
16.8 小结	414
16.9 问与答	414
16.10 作业	415
16.10.1 测验	415
16.10.2 练习	415

第17章 处理流.....418

17.1 流概述	418
17.1.1 数据流的封装	418
17.1.2 理解缓冲技术	419

17.2 流和缓冲区	420
17.3 标准 I/O 对象	420
17.4 重定向标准流	421
17.5 使用 cin 进行输入	421
17.5.1 输入字符串	422
17.5.2 字符串的问题	423
17.5.3 >> 的返回值	425
17.6 cin 的其他成员函数	425
17.6.1 单字符输入	425
17.6.2 从标准输入读取字符串	428
17.6.3 使用 cin.ignore()	430
17.6.4 查看和插入字符: peek() 和 putback()	431
17.7 使用 cout 进行输出	432
17.7.1 刷新输出	432
17.7.2 执行输出的函数	432
17.7.3 控制符、标记和格式化指令	433
17.8 流和 printf() 函数之比较	437
17.9 文件输入和输出	439
17.10 使用 ofstream	439
17.10.1 条件状态	440
17.10.2 打开文件进行输入和输出	440
17.10.3 修改 ofstream 打开文件时的 默认行为	441
17.11 二进制文件和文本文件	443
17.12 命令行处理	445
17.13 小结	448
17.14 问与答	448
17.15 作业	448
17.15.1 测验	448
17.15.2 练习	449

第18章 创建和使用名称空间.....450

18.1 简介	450
18.2 根据名称解析函数和类	450
18.2.1 变量的可见性	451
18.2.2 链接性	452
18.2.3 静态全局变量	453

18.3 创建名称空间.....	453	20.2.1 异常处理的组成部分.....	510
18.3.1 声明和定义类型.....	454	20.2.2 手工引发异常.....	512
18.3.2 在名称空间外定义函数.....	455	20.2.3 创建异常类.....	513
18.3.3 添加新成员.....	455	20.3 使用 try 块和 catch 块.....	516
18.3.4 嵌套名称空间.....	455	20.4 捕获异常的工作原理.....	517
18.4 使用名称空间.....	456	20.4.1 使用多条 catch 语句.....	517
18.5 关键字 using.....	457	20.4.2 异常层次结构.....	520
18.5.1 using 编译指令.....	457	20.5 异常中的数据及给异常对象命名.....	522
18.5.2 using 声明.....	459	20.6 异常和模板.....	529
18.6 名称空间别名.....	460	20.7 没有错误的异常.....	531
18.7 未命名的名称空间.....	460	20.8 关于代码蜕变.....	532
18.8 标准名称空间 std.....	461	20.9 bug 和调试.....	532
18.9 小结.....	462	20.9.1 断点.....	532
18.10 问与答.....	463	20.9.2 监视点.....	532
18.11 作业.....	463	20.9.3 查看内存.....	532
18.11.1 测验.....	463	20.9.4 查看汇编代码.....	532
18.11.2 练习.....	464	20.10 小结.....	533
第 19 章 模板	465	20.11 问与答.....	533
19.1 什么是模板.....	465	20.12 作业.....	533
19.2 创建模板定义.....	466	20.12.1 测验.....	534
19.2.1 使用名称.....	467	20.12.2 练习.....	534
19.2.2 实现模板.....	468	第 21 章 杂项内容	535
19.3 将实例化的模板对象传递给函数.....	471	21.1 预处理器和编译器.....	535
19.4 模板和友元.....	472	21.2 预处理器指令#define.....	535
19.4.1 非模板友元类和函数.....	472	21.2.1 使用#define 来定义常量.....	536
19.4.2 通用模板友元类和函数.....	475	21.2.2 将#define 用于检测.....	536
19.5 使用模板对象.....	479	21.2.3 预编译器命令#else.....	536
19.5.1 使用具体化函数.....	483	21.3 包含和多重包含防范.....	538
19.5.2 静态成员和模板.....	488	21.4 宏.....	538
19.6 标准模板库.....	491	21.5 字符串操纵.....	540
19.6.1 使用容器.....	492	21.5.1 字符串化.....	540
19.6.2 理解顺序容器.....	492	21.5.2 拼接.....	540
19.6.3 理解关联容器.....	499	21.6 预定义的宏.....	541
19.6.4 使用算法类.....	502	21.7 assert()宏.....	541
19.7 小结.....	505	21.7.1 使用 assert()进行调试.....	542
19.8 问与答.....	505	21.7.2 assert()与异常之比较.....	542
19.9 作业.....	505	21.7.3 副作用.....	543
19.9.1 测验.....	506	21.7.4 类的不变量.....	543
19.9.2 练习.....	506	21.7.5 打印中间值.....	548
第 20 章 处理错误和异常	508	21.7.6 宏与函数及模板之比较.....	549
20.1 程序中的各种错误.....	508	21.8 内联函数.....	549
20.2 异常的基本思想.....	509	21.9 位运算.....	550

21.9.1 “与”运算符.....	551	21.10.9 设置访问权限.....	557
21.9.2 “或”运算符.....	551	21.10.10 类定义.....	558
21.9.3 “异或”运算符.....	551	21.10.11 包含文件.....	558
21.9.4 “求补”运算符.....	551	21.10.12 使用 assert().....	558
21.9.5 设置位.....	551	21.10.13 使用 const.....	558
21.9.6 清除位.....	552	21.11 C++开发工作的下一步.....	558
21.9.7 反转位.....	552	21.11.1 从何处获得帮助和建议.....	558
21.9.8 位字段.....	552	21.11.2 相关的 C++主题: 受控 C++、 C#和 Microsoft 的.NET.....	559
21.10 编程风格.....	555	21.11.3 保持联系.....	559
21.10.1 缩进.....	555	21.12 小结.....	559
21.10.2 大括号.....	555	21.13 问与答.....	559
21.10.3 长代码行和函数长度.....	555	21.14 作业.....	560
21.10.4 格式化 switch 语句.....	556	21.14.1 测验.....	560
21.10.5 程序文本.....	556	21.14.2 练习.....	561
21.10.6 标识符命名.....	556		
21.10.7 名称的拼写和大写.....	557		
21.10.8 注释.....	557		

第3周复习

附录 A 二进制和十六进制.....	574	第 6 章.....	587
A.1 其他进制.....	574	第 7 章.....	590
A.2 不同进制之间的转换.....	575	第 8 章.....	591
A.2.1 二进制.....	575	第 9 章.....	592
A.2.2 为什么使用二进制.....	576	第 10 章.....	594
A.2.3 位、字节和半字节.....	576	第 11 章.....	598
A.2.4 什么是 KB.....	576	第 12 章.....	601
A.2.5 二进制数.....	576	第 13 章.....	602
A.3 十六进制.....	577	第 14 章.....	603
附录 B C++关键字.....	580	第 15 章.....	604
附录 C 运算符优先级.....	581	第 16 章.....	610
附录 D 答案.....	582	第 17 章.....	613
第 1 章.....	582	第 18 章.....	615
第 2 章.....	582	第 19 章.....	616
第 3 章.....	584	第 20 章.....	620
第 4 章.....	585	第 21 章.....	625
第 5 章.....	585	附录 E 链表简介.....	627

第 1 周课程简介

开始学习如何使用 C++ 进行编程之前，读者需要有几样东西：编译器、编辑器以及本书。即使没有 C++ 编译器和编辑器，读者仍可通过本书学习 C++，但如果能做些练习，将学到更多的知识。

学习编程的最好方法就是编写程序！在每章最后都有作业，其中包括测验和一些练习。请务必花时间回答所有的问题，并尽可能客观地为自己打分。后面的章节都是以前面的内容为基础的，因此继续阅读后面的内容之前，一定要完全理解当前的知识。

C 程序员的注意事项

C 程序员一定很熟悉本书前 5 章的内容，然而，如果要遵循 C++ 标准，将存在一些细微的差别。请务必浏览这些内容并完成练习，以确保能够快速进入第 6 章的学习。

本周内容简介

本周简要地介绍编程（尤其是 C++ 编程）所需的一些知识。第 1 章和第 2 章介绍有关编程和程序流程的基本概念；第 3 章介绍有关变量和常量以及如何在程序中使用数据的知识；第 4 章讨论程序如何根据运行时提供的数据和遇到的条件执行不同的分支；第 5 章介绍函数是什么以及如何使用它们；第 6 章介绍类和对象；第 7 章介绍更深入地探讨有关程序流程的知识。经过本周的学习后，读者将能够编写真正的面向对象的程序。

第 1 章 绪 论

欢迎使用本书！今天你将迈出成为高级 C++ 程序员的第一步。

本章将学习：

- 为何 C++ 是软件开发的标准？
- 开发 C++ 程序的步骤。
- 如何输入、编译和链接你的第一个 C++ 程序。

1.1 C++ 简史

第一代电子计算机诞生于二战期间，用来帮助计算武器弹道数据。从此以后，计算机语言经历了翻天覆地的变化。起初，程序员们使用最原始的计算机指令即机器语言来编程。这些指令是由 0 和 1 组成的很长的字符串。很快，人们就发明了汇编语言，将机器指令映射为人们可以阅读和易于处理的助记符，如 ADD 和 MOV。

随后出现了高级语言，如 BASIC 和 COBOL。这些语言让人们能够用一些类似于单词或句子的源代码来编程，如 Let I=100。这些指令再通过解释器和编译器转换为机器语言。

解释器翻译并执行程序，它读取程序指令（源代码），直接将其转换为操作。

编译器先将源代码转换为中间格式；这通常被称为编译，它生成目标文件。然后，编译器调用链接器，将目标文件组合成为可执行程序。

由于解释器按原样读取代码并当场执行代码，因此程序员使用起来比较方便。当前，大部分解释型程序被称为脚本，解释器被称为脚本引擎。

有些语言（如 Visual Basic 6）将解释器称为运行库；其他语言（如 Visual Basic .NET 和 Java）有另一个组件——虚拟机（Virtual Machine, VM）或运行库。它也是解释器，然而它不是源代码解释器——将人类可读的语言转换为依赖于计算机的机器码；而是对一种编译后的独立于计算机的语言（中间语言）进行解释和执行。

编译器增加了一个步骤：将人类能够理解的源代码编译成机器能够理解的目标代码。这个步骤看似不太方便，但由于将源代码转换为机器语言这样一个耗时的任务已经在编译阶段完成，因此编译型程序的运行速度非常快。由于转换工作已经完成，因此执行程序时无需做这样的工作。

诸如 C++ 等编译型语言的另一个优点是，可以向没有编译器的用户提供可执行程序。使用解释型语言时，必须有解释器才能运行程序。

C++ 通常是一种编译型语言，虽然存在一些 C++ 解释器。和众多其他编译型语言一样，C++ 以能够生成快速而功能强大的程序著称。

事实上，很多年来，计算机程序员的一个主要目标是编写能够快速执行的简短代码。程序必须短小，因为内存昂贵；同时必须快速运行，因为处理能力也很昂贵。随着计算机的体积越来越小、价格越来越便宜、速度越来越快，内存的价格不断下降，这些优先考虑的因素发生了变化。当前，程序员的时间费用已远远超过了大多数商用计算机的费用。精心编写和易于维护的代码成了编程的首要目标。易于维护是指当程序需求

发生变化时，无需花太大的代价就可以对程序进行扩展和改进。

注意：程序一词通常有两种含义：一种是指程序员编写的指令（源代码），另一种是指可执行软件。这种区分可能会引起巨大的混乱，因此本书尽可能将源代码和可执行文件区分开来。

1.1.1 解决问题

当前，程序员们面临的问题与20年前完全不同。在20世纪80年代，人们编写程序是为了管理和处理大量的原始数据。那时，编写代码的人和使用程序的人都是计算机专业人员。现在，越来越多的人开始使用计算机，其中大部分人对计算机和程序的工作原理知之甚少。人们通常只愿意将计算机作为一种工具来解决商务问题，而无心去钻研它。

具有讽刺意味的是，为了满足这些新用户易于使用的要求，程序变得越来越复杂。那种需要输入神秘的命令，结果却只看到一大串枯燥的数据的时代已一去不复返。当前的程序大都采用“用户友好界面”，其中包括多个窗口、菜单、对话框以及非常形象的标识，对此我们已经非常熟悉了。

随着万维网的发展，计算机跨入了一个新的市场渗透时代：使用计算机的人比任何时候都多，他们对计算机的期望也非常高。万维网的易用性也提高了人们的期望，他们期望程序能够充分利用万维网提供的信息。

在过去的几年中，应用程序的运行平台已扩展到其他设备，不再限于桌面PC，而被用于手机、个人数字助理（PDA）、平板PC和其他设备。

在本书第一版面世后的几年中，为满足用户的需求，程序员编写的程序变得越来越庞大，越来越复杂。因此，对有助于应对这种复杂性的编程技术的需求显得越来越迫切。

随着编程要求的改变，用于编写程序的语言和技术都得到了发展，以帮助程序员应对这种复杂性。虽然完整的发展历史令人神往，但本书只介绍其中的重要组成部分：从过程化编程到面向对象编程的转变。

1.1.2 过程化编程、结构化编程和面向对象编程

直到不久前，计算机程序还被看作是一系列处理数据的过程。过程（函数或方法）是指一组依次执行的指令。数据与过程是分离的，编程技巧是跟踪哪些函数调用了其他函数以及哪些数据被修改了。为避免发生，结构化编程应运而生。

结构化编程的主要思想是分而治之。可将计算机程序看作由一系列任务组成。任何过于复杂、无法简单描述的任务都将分解为一系列较小的子任务，直至每个任务都很小，很容易理解。

例如，计算公司职员的平均工资是一项较复杂的任务。然而，可将其划分成下面几个子任务：

- (1) 确定公司员工数。
- (2) 确定每个员工的工资。
- (3) 计算工资总额。
- (4) 将工资总额除以员工数。

计算工资总额还可分成以下几步：

- (1) 读取每个职员的记录。
- (2) 读取工资额。
- (3) 将工资额添加到工资总额中。
- (4) 读取下一个职员的记录。

读取每个职员的记录又可分为以下几步：

- (1) 打开职员文件。
- (2) 找到正确的记录。
- (3) 读取数据。

结构化编程仍是一种非常成功的、处理复杂问题的方法。然而，到20世纪80年代后期，它的许多不足逐渐暴露出来了。

首先，一种自然而然的愿望是，将数据（如职员记录）及其操作（排序、编辑等）看作一个整体。不幸