

SAMS

畅销全球的
经典C++教程

Sams Teach Yourself C++
in One Hour a Day
Seventh Edition

中文版
累计销量
超50000册

全面涵盖C++11新标准
帮助读者编写高效的C++应用程序

21天学通 C++ (第7版)

[美] Siddhartha Rao 著
袁国忠 译

 人民邮电出版社
POSTS & TELECOM PRESS

21天学通 C++ (第7版)

[美] Siddhartha Rao 著
袁国忠 译



图书在版编目 (CIP) 数据

21天学通C++ : 第7版 / (美) 罗奥 (Rao, S.) 著 ;
袁国忠译. — 北京 : 人民邮电出版社, 2012. 12
ISBN 978-7-115-29624-5

I. ①2… II. ①罗… ②袁… III. ①C语言—程序设计
IV. ①TP312

中国版本图书馆CIP数据核字 (2012) 第235664号

版权声明

Siddhartha Rao: Sams Teach Yourself C++ in One Hour a Day (7th Edition)

ISBN: 0672335670

Copyright © 2012 by Pearson Education, Inc.

Authorized translation from the English languages edition published by Sams.

All rights reserved.

本书中文简体字版由美国 Pearson 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 7 版)

-
- ◆ 著 [美] Siddhartha Rao
 - 译 袁国忠
 - 责任编辑 傅道坤
 - ◆ 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 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京昌平百善印刷厂印刷
 - ◆ 开本: 787×1092 1/16
印张: 29.5
字数: 870 千字 2012 年 12 月第 1 版
印数: 1-3 500 册 2012 年 12 月北京第 1 次印刷

著作权合同登记号 图字: 01-2012-5025 号

ISBN 978-7-115-29624-5

定价: 59.00 元

读者服务热线: (010)67132692 印装质量热线: (010)67129223

反盗版热线: (010)67171154

广告经营许可证: 京崇工商广字第 0021 号

内容提要

本书通过大量短小精悍的程序，详细而全面地阐述了 C++ 基本概念和技术以及 C++11 新增的功能，包括管理输入/输出、循环和数组、面向对象编程、模板、使用标准模板库以及 `lambda` 表达式等。这些内容被组织成结构合理、联系紧密的章节，每章都可在 1 小时内阅读完毕；每章都提供了示例程序清单，并辅以示例输出和代码分析，以阐述该章介绍的主题。为加深读者对所学内容的理解，每章末尾都提供了常见问题及其答案以及练习和测验。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。

本书是针对 C++ 初学者编写的，不要求读者有 C 语言方面的背景知识，可作为高等院校教授 C++ 课程的教材，也可供初学者自学 C++ 时使用。

作者简介

Siddhartha Rao 是全球领先的企业软件提供商 **SAP AG** 的技术专家。作为 **SAP Product Security India** 的负责人，其主要职责包括招募产品安全领域的专家以及制定软件开发最佳实践，以保持 **SAP** 软件的全球竞争力。作为一位 **Microsoft Visual C++ MVP**，他深信 **C++11** 有助于编写速度更快、更简洁、更高效的 **C++** 应用程序。

Siddhartha 酷爱旅游，不放过任何一次探索新文化的机会。例如，本书就是在 4 个不同的国度创作而成的，其中包括法国布列塔尼一个面朝大西洋的奇特村庄。他期待着您对这部全球之作提出宝贵的建议。

献词

谨将本书献给我的父母和我的妹妹，他们是我坚强的后盾。

致谢

当我为了编写本书而通宵达旦地忙碌时，是我的朋友承担起了我的后勤工作，对此深表谢意。谢谢本书的所有编辑人员，正是他们的勤劳付出，才让本书出现在各位读者的书架上。

前 言

对 C++ 来说，2011 是个很特别的年份。在这一年，C++11 终于获批成为新标准，它新增了一些可提高编程效率的关键字和结构，让您能够编写更优质的代码。本书旨在帮助您循序渐进地学习 C++11，其中的章节经过仔细编排，从实用的角度介绍这种面向对象的编程语言的基本知识。读者只需每天花 1 小时，在学完本书后，就能掌握 C++11。

学习 C++ 的最佳方式是动手实践。本书包含丰富的代码示例，有助于读者提高编程技能，请务必亲自动手尝试这些代码。这些代码片段都使用了（在本书编写时）最新版本的编译器进行了测试，具体地说是 Microsoft Visual C++ 2010 和 GNU C++ 编译器 4.6 版，它们都支持大量的 C++11 功能。

针对的读者

本书从最基本的 C++ 知识开始介绍，读者只需具备学习 C++ 的愿望及了解工作原理的好奇心即可；虽然具备一些 C++ 知识会有所帮助，但这并非必需的。本书也可供熟悉 C++ 但想了解 C++11 新增功能的读者参考；如果你是专业程序员，第 3 部分“学习标准模板库”可以帮助你创建更优质、更实用的 C++ 应用程序。

本书内容

读者可根据自己对 C++ 的熟练程度，阅读感兴趣的部分。本书包含 5 部分。

- 第 1 部分“基本知识”，引导读者编写一些简单的 C++ 应用程序，并介绍一些在 C++ 的未妥协类型安全变量的代码中最常见的关键字。
- 第 2 部分“C++ 面向对象编程基础”，介绍类的概念，您将学习 C++ 如何支持封装、抽象、继承和多态等重要的面向对象编程原则。第 9 章将介绍 C++11 新增的移动构造函数，而第 12 章将介绍移动赋值运算符。这些功能有助于避免不必要的复制步骤，从而提升应用程序的性能。第 14 章是一个跳板，助您编写功能强大的 C++ 通用代码。
- 第 3 部分“学习标准模板库”，将帮助您使用 STL `string` 类和容器编写高效而实用的 C++ 代码。您将了解到，使用 `std::string` 可安全而轻松地拼接字符串，您不再需要使用 C 风格字符串 (`char*`)。您可使用 STL 动态数组和链表，而无需自己编写这样的类。
- 第 4 部分“再谈 STL”，专注于算法，您将学习如何通过迭代器对 `vector` 等容器进行排序。在这部分，您将发现，通过使用 C++11 新增的关键字 `auto`，可极大地简化冗长的迭代器声明。第 22 章将介绍 C++11 新增的 `lambda` 表达式，这可极大地简化使用 STL 算法的代码。
- 第 5 部分“高级 C++ 概念”，阐述智能指针和异常处理等 C++ 功能。对 C++ 应用程序来说，这些功能并非必需的，但可极大地提高应用程序的稳定性和品质。在这部分的最后，简要地介绍了有助于编写杰出 C++ 应用程序的最佳实践。

本书体例

本书使用了下述提供更多信息的元素：

注意 提供与读者阅读的内容相关的信息。

C++11

突出 C++11 新增的功能。要使用这些功能，可能需要使用较新的编译器版本。

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。

提示 提供 C++ 编程最佳实践。

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。

本书使用不同的字体来区分代码和正文，全书都用特殊字体呈现代码、命令以及与编程相关的术语。

目 录

第1章 绪论	1	3.1.2 声明变量以访问和使用内存	15
1.1 C++简史	1	3.1.3 声明并初始化多个类型相同的变量	17
1.1.1 与C语言的关系	1	3.1.4 理解变量的作用域	17
1.1.2 C++的优点	1	3.1.5 全局变量	18
1.1.3 C++标准的发展历程	1	3.2 编译器支持的常见C++变量类型	19
1.1.4 哪些人使用C++程序	2	3.2.1 使用bool变量存储布尔值	20
1.2 编写C++应用程序	2	3.2.2 使用char变量存储字符	20
1.2.1 生成可执行文件的步骤	2	3.2.3 有符号整数和无符号整数的概念	20
1.2.2 分析并修复错误	2	3.2.4 有符号整型short、int、long和long long	21
1.2.3 集成开发环境	2	3.2.5 无符号整型unsigned short、unsigned int、unsigned long和unsigned long long	21
1.2.4 编写第一个C++应用程序	3	3.2.6 浮点类型float和double	21
1.2.5 生成并执行第一个C++应用程序	4	3.3 使用sizeof确定变量的长度	22
1.2.6 理解编译错误	4	3.4 使用typedef替换变量类型	24
1.3 C++11新增的功能	5	3.5 什么是常量	24
1.4 总结	5	3.5.1 字面常量	25
1.5 问与答	5	3.5.2 使用const将变量声明为常量	25
1.6 作业	6	3.5.3 使用constexpr声明常量	26
1.6.1 测验	6	3.5.4 枚举常量	26
1.6.2 练习	6	3.5.5 使用#define定义常量	27
第2章 C++程序的组成部分	7	3.6 给变量和常量命名	28
2.1 Hello World程序的组成部分	7	3.7 不能用作常量或变量名的关键字	28
2.1.1 预处理器编译指令#include	7	3.8 总结	29
2.1.2 程序的主体——main()	8	3.9 问与答	29
2.1.3 返回值	8	3.10 作业	30
2.2 名称空间的概念	9	3.10.1 测验	30
2.3 C++代码中的注释	10	3.10.2 练习	30
2.4 C++函数	10	第4章 管理数组和字符串	31
2.5 使用std::cin和std::cout执行基本输入输出操作	12	4.1 什么是数组	31
2.6 总结	13	4.1.1 为何需要数组	31
2.7 问与答	13	4.1.2 声明和初始化静态数组	32
2.8 作业	13	4.1.3 数组中的数据是如何存储的	32
2.8.1 测验	14	4.1.4 访问存储在数组中的数据	33
2.8.2 练习	14	4.1.5 修改存储在数组中的数据	34
第3章 使用变量和常量	15	4.2 多维数组	35
3.1 什么是变量	15	4.2.1 声明和初始化多维数组	36
3.1.1 内存和寻址概述	15		

4.2.2 访问多维数组中的元素	36	6.1.2 有条件地执行多条语句	62
4.3 动态数组	37	6.1.3 嵌套 if 语句	63
4.4 C 风格字符串	38	6.1.4 使用 switch-case 进行条件处理	66
4.5 C++字符串: 使用 std::string	40	6.1.5 使用运算符?: 进行条件处理	68
4.6 总结	41	6.2 在循环中执行代码	69
4.7 问与答	41	6.2.1 不成熟的 goto 循环	69
4.8 作业	42	6.2.2 while 循环	70
4.8.1 测验	42	6.2.3 do...while 循环	72
4.8.2 练习	42	6.2.4 for 循环	73
第 5 章 使用表达式、语句和运算符	43	6.3 使用 continue 和 break 修改循环的行为	75
5.1 语句	43	6.3.1 不结束的循环——无限循环	75
5.2 复合语句 (语句块)	44	6.3.2 控制无限循环	76
5.3 使用运算符	44	6.4 编写嵌套循环	78
5.3.1 赋值运算符 (=)	44	6.4.1 使用嵌套循环遍历多维数组	79
5.3.2 理解左值和右值	44	6.4.2 使用嵌套循环计算斐波纳契数列	80
5.3.3 加法运算符 (+)、减法运算符 (-)、 乘法运算符 (*)、除法运算符 (/) 和 求模运算符 (%)	44	6.5 总结	81
5.3.4 递增运算符 (++) 和递减 运算符 (--)	45	6.6 问与答	81
5.3.5 前缀还是后缀	45	6.7 作业	82
5.3.6 相等运算符 (==) 和不等 运算符 (!=)	47	6.7.1 测验	82
5.3.7 关系运算符	48	6.7.2 练习	82
5.3.8 逻辑运算 NOT、AND、OR 和 XOR	49	第 7 章 使用函数组织代码	83
5.3.9 使用 C++逻辑运算 NOT (!)、 AND (&&) 和 OR ()	50	7.1 为何需要函数	83
5.3.10 按位运算符 NOT (~)、 AND (&)、OR () 和 XOR (^)	53	7.1.1 函数原型是什么	84
5.3.11 按位右移运算符 (>>) 和 左移运算符 (<<)	54	7.1.2 函数定义是什么	85
5.3.12 复合赋值运算符	55	7.1.3 函数调用和实参是什么	85
5.3.13 使用运算符 sizeof 确定变量占用的 内存量	56	7.1.4 编写接受多个参数的函数	85
5.3.14 运算符优先级	57	7.1.5 编写没有参数和返回值的函数	86
5.4 总结	58	7.1.6 带默认值的函数参数	87
5.5 问与答	59	7.1.7 递归函数——调用自己的函数	88
5.6 作业	59	7.1.8 包含多条 return 语句的函数	89
5.6.1 测验	59	7.2 使用函数处理不同类型的数据	90
5.6.2 练习	59	7.2.1 函数重载	90
第 6 章 控制程序流程	60	7.2.2 将数组传递给函数	92
6.1 使用 if...else 有条件地执行	60	7.2.3 按引用传递参数	93
6.1.1 使用 if...else 进行条件编程	61	7.3 微处理器如何处理函数调用	94
		7.3.1 内联函数	94
		7.3.2 lambda 函数	96
		7.4 总结	97
		7.5 问与答	97
		7.6 作业	97
		7.6.1 测验	97
		7.6.2 练习	98
		第 8 章 阐述指针和引用	99
		8.1 什么是指针	99

8.1.1	声明指针	99	9.3.6	包含初始化列表的构造函数	133
8.1.2	使用引用运算符(&)获取变量的地址	100	9.4	析构函数	135
8.1.3	使用指针存储地址	101	9.4.1	声明和实现析构函数	135
8.1.4	使用解除引用运算符(*)访问指向的数据	102	9.4.2	何时及如何使用析构函数	135
8.1.5	将 sizeof()用于指针的结果	104	9.5	复制构造函数	137
8.2	动态内存分配	105	9.5.1	浅复制及其存在的问题	137
8.2.1	使用 new 和 delete 动态地分配和释放内存	105	9.5.2	使用复制构造函数确保深复制	139
8.2.2	将递增和递减运算符(++和--)用于指针的结果	107	9.5.3	有助于改善性能的移动构造函数	142
8.2.3	将关键字 const 用于指针	109	9.6	构造函数和析构函数的其他用途	143
8.2.4	将指针传递给函数	109	9.6.1	不允许复制的类	143
8.2.5	数组和指针的类似之处	110	9.6.2	只能有一个实例的单例类	144
8.3	使用指针时常犯的编程错误	112	9.6.3	禁止在栈中实例化的类	146
8.3.1	内存泄露	112	9.7	this 指针	147
8.3.2	指针指向无效的内存单元	112	9.8	将 sizeof()用于类	148
8.3.3	悬空指针(也叫迷途或失控指针)	113	9.9	结构不同于类的地方	150
8.4	指针编程最佳实践	114	9.10	声明友元	150
8.4.1	检查使用 new 发出的分配请求是否得到满足	115	9.11	总结	152
8.5	引用是什么	117	9.12	问与答	152
8.5.1	是什么让引用很有用	117	9.13	作业	153
8.5.2	将关键字 const 用于引用	118	9.13.1	测验	153
8.5.3	按引用向函数传递参数	119	9.13.2	练习	153
8.6	总结	119	第 10 章	实现继承	154
8.7	问与答	120	10.1	继承基础	154
8.8	作业	121	10.1.1	继承和派生	154
8.8.1	测验	121	10.1.2	C++派生语法	155
8.8.2	练习	121	10.1.3	访问限定符 protected	157
第 9 章	类和对象	122	10.1.4	基类初始化——向基类传递参数	159
9.1	类和对象	122	10.1.5	在派生类中覆盖基类的方法	160
9.1.1	声明类	123	10.1.6	调用基类中被覆盖的方法	162
9.1.2	实例化对象	123	10.1.7	在派生类中调用基类的方法	162
9.1.3	使用句点运算符访问成员	123	10.1.8	在派生类中隐藏基类的方法	164
9.1.4	使用指针运算符(>)访问成员	124	10.1.9	构造顺序	165
9.2	关键字 public 和 private	125	10.1.10	析构顺序	166
9.2.1	使用关键字 private 实现数据抽象	126	10.2	私有继承	167
9.3	构造函数	127	10.3	保护继承	169
9.3.1	声明和实现构造函数	128	10.4	切除问题	171
9.3.2	何时及如何使用构造函数	128	10.5	多继承	171
9.3.3	重载构造函数	130	10.6	总结	174
9.3.4	没有默认构造函数的类	131	10.7	问与答	174
9.3.5	带默认值的构造函数参数	133	10.8	作业	174
			10.8.1	测验	174
			10.8.2	练习	174
			第 11 章	多态	176
			11.1	多态基础	176

11.1.1	为何需要多态行为	176	类型识别	223
11.1.2	使用虚函数实现多态行为	177	13.3.3 使用 <code>reinterpret_cast</code>	225
11.1.3	为何需要虚构造函数	179	13.3.4 使用 <code>const_cast</code>	226
11.1.4	虚函数的工作原理——理解 虚函数表	182	13.4 C++类型转换运算符存在的问题	226
11.1.5	抽象基类和纯虚函数	184	13.5 总结	227
11.2	使用虚继承解决菱形问题	186	13.6 问与答	227
11.3	可将复制构造函数声明为虚函数吗	189	13.7 作业	228
11.4	总结	191	13.7.1 测验	228
11.5	问与答	192	13.7.2 练习	228
11.6	作业	192	第 14 章 宏和模板简介	229
11.6.1	测验	192	14.1 预处理器与编译器	229
11.6.2	练习	193	14.2 使用 <code>#define</code> 定义常量	229
第 12 章 运算符类型与运算符重载		194	14.3 使用 <code>#define</code> 编写宏函数	232
12.1	C++运算符	194	14.3.1 为什么要使用括号	233
12.2	单目运算符	195	14.3.2 使用 <code>assert</code> 宏验证表达式	234
12.2.1	单目运算符的类型	195	14.3.3 使用宏函数的优点和缺点	235
12.2.2	单目递增与单目递减运算符	195	14.4 模板简介	235
12.2.3	转换运算符	198	14.4.1 模板声明语法	235
12.2.4	解除引用运算符 (<code>*</code>) 和成员 选择运算符 (<code>-></code>)	199	14.4.2 各种类型的模板声明	236
12.3	双目运算符	202	14.4.3 模板函数	236
12.3.1	双目运算符的类型	202	14.4.4 模板与类型安全	238
12.3.2	双目加法与双目减法运算符	202	14.4.5 模板类	238
12.3.3	实现运算符 <code>+=</code> 与 <code>-=</code>	204	14.4.6 模板的实例化和具体化	239
12.3.4	重载等于运算符 (<code>==</code>) 和不等 运算符 (<code>!=</code>)	206	14.4.7 声明包含多个参数的模板	239
12.3.5	重载运算符 <code><</code> 、 <code>></code> 、 <code><=</code> 和 <code>>=</code>	207	14.4.8 声明包含默认参数的模板	240
12.3.6	重载复制赋值运算符 (<code>=</code>)	209	14.4.9 一个模板示例	240
12.3.7	下标运算符	211	14.4.10 模板类和静态成员	241
12.4	函数运算符 <code>operator()</code>	214	14.4.11 在实际 C++编程中使用模板	243
12.5	不能重载的运算符	219	14.5 总结	243
12.6	总结	219	14.6 问与答	244
12.7	问与答	220	14.7 作业	244
12.8	作业	220	14.7.1 测验	244
12.8.1	测验	220	14.7.2 练习	244
12.8.2	练习	220	第 15 章 标准模板库简介	245
第 13 章 类型转换运算符		221	15.1 STL 容器	245
13.1	为何需要类型转换	221	15.1.1 顺序容器	245
13.2	为何有些 C++程序员不喜欢 C 风格类型转换	222	15.1.2 关联容器	246
13.3	C++类型转换运算符	222	15.1.3 选择正确的容器	246
13.3.1	使用 <code>static_cast</code>	222	15.1.4 容器适配器	247
13.3.2	使用 <code>dynamic_cast</code> 和运行阶段		15.2 STL 迭代器	247

15.7 问与答	250	18.3 对 list 中的元素进行反转和排序	283
15.8 作业	251	18.3.1 使用 list::reverse()反转元素的 排列顺序	283
第 16 章 STL string 类	252	18.3.2 对元素进行排序	284
16.1 为何需要字符串操作类	252	18.3.3 对包含对象的 list 进行排序 以及删除其中的元素	286
16.2 使用 STL string 类	253	18.4 总结	290
16.2.1 实例化和复制 STL string	253	18.5 问与答	290
16.2.2 访问 std::string 的字符串内容	254	18.6 作业	291
16.2.3 拼接字符串	256	18.6.1 测验	291
16.2.4 在 string 中查找字符或子字符串	257	18.6.2 练习	291
16.2.5 截短 STL string	259	第 19 章 STL 集合类	292
16.2.6 字符串反转	260	19.1 简介	292
16.2.7 字符串的大小写转换	261	19.2 STL set 和 multiset 的基本操作	293
16.3 基于模板的 STL string 实现	262	19.2.1 实例化 std::set 对象	293
16.4 总结	262	19.2.2 在 set 或 multiset 中插入元素	294
16.5 问与答	262	19.2.3 在 STL set 或 multiset 中 查找元素	296
16.6 作业	263	19.2.4 删除 STL set 或 multiset 中的 元素	297
16.6.1 测验	263	19.3 使用 STL set 和 multiset 的优缺点	300
16.6.2 练习	263	19.4 总结	303
第 17 章 STL 动态数组类	264	19.5 问与答	303
17.1 std::vector 的特点	264	19.6 作业	304
17.2 典型的 vector 操作	264	19.6.1 测验	304
17.2.1 实例化 vector	264	19.6.2 练习	304
17.2.2 使用 push_back()在末尾插入元素	266	第 20 章 STL 映射类	305
17.2.3 使用 insert()在指定位置插入元素	267	20.1 STL 映射类简介	305
17.2.4 使用数组语法访问 vector 中的 元素	269	20.2 std::map 和 std::multimap 的基本操作	306
17.2.5 使用指针语法访问 vector 中的 元素	270	20.2.1 实例化 std::map 和 std::multimap	306
17.2.6 删除 vector 中的元素	271	20.2.2 在 STL map 或 multimap 中 插入元素	307
17.3 理解大小和容量	272	20.2.3 在 STL map 或 multimap 中 查找元素	309
17.4 STL deque 类	273	20.2.4 在 STL multimap 中查找元素	311
17.5 总结	275	20.2.5 删除 STL map 或 multimap 中的 元素	312
17.6 问与答	275	20.3 提供自定义的排序谓词	313
17.7 作业	276	20.3.1 散列表的工作原理	316
17.7.1 测验	276	20.3.2 使用 C++11 散列表 unordered_map 和 unordered_multimap	316
17.7.2 练习	276	20.4 总结	319
第 18 章 STL list 和 forward_list	277	20.5 问与答	319
18.1 std::list 的特点	277	20.6 作业	320
18.2 基本的 list 操作	277		
18.2.1 实例化 std::list 对象	277		
18.2.2 在 list 开头或末尾插入元素	279		
18.2.3 在 list 中间插入元素	280		
18.2.4 删除 list 中的元素	282		

20.6.1	测验	320	23.3.6	使用 <code>for_each()</code> 处理指定范围内的元素	350
20.6.2	练习	320	23.3.7	使用 <code>std::transform()</code> 对范围进行变换	352
第 21 章 理解函数对象		321	23.3.8	复制和删除操作	354
21.1	函数对象与谓词的概念	321	23.3.9	替换值以及替换满足给定条件的元素	356
21.2	函数对象的典型用途	321	23.3.10	排序、在有序集合中搜索以及删除重复元素	357
21.2.1	一元函数	321	23.3.11	将范围分区	359
21.2.2	一元谓词	325	23.3.12	在有序集合中插入元素	360
21.2.3	二元函数	326	23.4	总结	362
21.2.4	二元谓词	328	23.5	问与答	362
21.3	总结	330	23.6	作业	363
21.4	问与答	330	23.6.1	测验	363
21.5	作业	330	23.6.2	练习	363
21.5.1	测验	330	第 24 章 自适应容器：栈和队列		364
21.5.2	练习	330	24.1	栈和队列的行为特征	364
第 22 章 C++ lambda 表达式		331	24.1.1	栈	364
22.1	lambda 表达式是什么	331	24.1.2	队列	365
22.2	如何定义 lambda 表达式	332	24.2	使用 STL <code>stack</code> 类	365
22.3	一元函数对应的 lambda 表达式	332	24.2.1	实例化 <code>stack</code>	365
22.4	一元谓词对应的 lambda 表达式	333	24.2.2	<code>stack</code> 的成员函数	366
22.5	通过捕获列表接受状态变量的 lambda 表达式	334	24.2.3	使用 <code>push()</code> 和 <code>pop()</code> 在栈顶插入和删除元素	366
22.6	lambda 表达式的通用语法	335	24.3	使用 STL <code>queue</code> 类	367
22.7	二元函数对应的 lambda 表达式	336	24.3.1	实例化 <code>queue</code>	368
22.8	二元谓词对应的 lambda 表达式	337	24.3.2	<code>queue</code> 的成员函数	368
22.9	总结	339	24.3.3	使用 <code>push()</code> 在队尾插入以及使用 <code>pop()</code> 从队首删除	369
22.10	问与答	340	24.4	使用 STL 优先级队列	370
22.11	作业	340	24.4.1	实例化 <code>priority_queue</code> 类	370
22.11.1	测验	340	24.4.2	<code>priority_queue</code> 的成员函数	371
22.11.2	练习	340	24.4.3	使用 <code>push()</code> 在 <code>priority_queue</code> 末尾插入以及使用 <code>pop()</code> 在 <code>priority_queue</code> 开头删除	372
第 23 章 STL 算法		341	24.5	总结	373
23.1	什么是 STL 算法	341	24.6	问与答	373
23.2	STL 算法的分类	341	24.7	作业	374
23.2.1	非变序算法	341	24.7.1	测验	374
23.2.2	变序算法	342	24.7.2	练习	374
23.3	使用 STL 算法	343	第 25 章 使用 STL 位标志		375
23.3.1	根据值或条件查找元素	343	25.1	<code>bitset</code> 类	375
23.3.2	计算包含给定值或满足给定条件的元素数	345	25.2	使用 <code>std::bitset</code> 及其成员	376
23.3.3	在集合中搜索元素或序列	346			
23.3.4	将容器中的元素初始化为指定值	348			
23.3.5	使用 <code>std::generate()</code> 将元素设置为运行阶段生成的值	349			

25.2.1	std::bitset 的运算符	376	27.5	使用 std::fstream 处理文件	398
25.2.2	std::bitset 的成员方法	377	27.5.1	使用 open()和 close()打开和 关闭文件	398
25.3	vector<bool>	378	27.5.2	使用 open()创建文本文件并 使用运算符<<写入文本	399
25.3.1	实例化 vector<bool>	378	27.5.3	使用 open()和运算符>>读取 文本文件	399
25.3.2	vector<bool>的成员函数和 运算符	379	27.5.4	读写二进制文件	400
25.4	总结	380	27.6	使用 std::stringstream 对字符串 进行转换	402
25.5	问与答	380	27.7	总结	403
25.6	作业	381	27.8	问与答	403
25.6.1	测验	381	27.9	作业	403
25.6.2	练习	381	27.9.1	测验	403
第 26 章	理解智能指针	382	27.9.2	练习	404
26.1	什么是智能指针	382	第 28 章	异常处理	405
26.1.1	常规(原始)指针存在的问题	382	28.1	什么是异常	405
26.1.2	智能指针有何帮助	383	28.2	导致异常的原因	405
26.2	智能指针是如何实现的	383	28.3	使用 try 和 catch 捕获异常	406
26.3	智能指针类型	384	28.3.1	使用 catch(...)处理所有异常	406
26.3.1	深复制	384	28.3.2	捕获特定类型的异常	407
26.3.2	写时复制机制	385	28.3.3	使用 throw 引发特定类型的异常	408
26.3.3	引用计数智能指针	386	28.4	异常处理的工作原理	409
26.3.4	引用链接智能指针	386	28.4.1	std::exception 类	411
26.3.5	破坏性复制	386	28.4.2	从 std::exception 派生出自定义 异常类	411
26.4	深受欢迎的智能指针库	389	28.5	总结	413
26.5	总结	389	28.6	问与答	413
26.6	问与答	389	28.7	作业	413
26.7	作业	390	28.7.1	测验	414
26.7.1	测试	390	28.7.2	练习	414
26.7.2	练习	390	第 29 章	继续前行	415
第 27 章	使用流进行输入和输出	391	29.1	当今的处理器有何不同	415
27.1	流的概述	391	29.2	如何更好地利用多个内核	416
27.2	重要的 C++流类和流对象	391	29.2.1	线程是什么	416
27.3	使用 std::cout 将指定格式的数据 写入控制台	392	29.2.2	为何要编写多线程应用程序	417
27.3.1	使用 std::cout 修改数字的 显示格式	393	29.2.3	线程如何交换数据	417
27.3.2	使用 std::cout 对齐文本和设置 字段宽度	394	29.2.4	使用互斥量和信号量同步线程	418
27.4	使用 std::cin 进行输入	395	29.2.5	多线程技术带来的问题	418
27.4.1	使用 std::cin 将输入读取到基本 类型变量中	395	29.3	编写杰出的 C++代码	418
27.4.2	使用 std::cin:get 将输入读取到 char 数组中	396	29.4	更深入地学习 C++	419
27.4.3	使用 std::cin 将输入读取到 std::string 中	397	29.4.1	在线文档	419
			29.4.2	提供指南和帮助的社区	420
			29.5	总结	420

29.6 问与答	420	A.4 不同进制之间的转换	423
29.7 作业	420	A.4.1 通用转换步骤	423
附录 A 二进制和十六进制	421	A.4.2 从十进制转换为二进制	423
A.1 十进制	421	A.4.3 从十进制转换为十六进制	424
A.2 二进制	421	附录 B C++关键字	425
A.2.1 计算机为何使用二进制	422	附录 C 运算符优先级	426
A.2.2 位和字节	422	附录 D 答案	427
A.2.3 1KB 相当于多少字节	422	附录 E ASCII 码	456
A.3 十六进制	422		

第 1 章

绪 论

欢迎使用本书！通过阅读本章，您将迈出成为高级 C++ 程序员的第一步。

在本章中，您将学习：

- 为何 C++ 是软件开发的标准；
- 输入、编译和链接第一个 C++ 程序；
- C++11 新增的功能。

1.1 C++ 简史

编程语言旨在让人更容易使用计算资源。C++ 并非一种新语言，但仍被广泛采用，并在不断改进。2011 年，最新的 C++ 标准获得了 ISO 标准委员会的批准，名为 C++11。

1.1.1 与 C 语言的关系

C++ 最初由 Bjarne Stroustrup 于 1979 年在贝尔实验室开发，被设计为 C 语言的继任者。C 语言是一种过程型语言，程序员使用它定义执行特定操作的函数，而 C++ 是一种面向对象的语言，实现了继承、抽象、多态和封装等概念。C++ 支持类，而类包含成员数据以及操作数据的成员方法（方法类似于 C 语言中的函数）。其结果是，程序员需要考虑数据以及要用它们来做什么。一直以来，C++ 编译器都支持 C 语言，这具有向后与既有代码兼容的优势，但也存在缺点，那就是编译器非常复杂，因为随着 C++ 的发展，编译器既要实现所有的新功能，又要向程序员提供这种向后兼容的功能。

1.1.2 C++ 的优点

C++ 是一种中级编程语言，这意味着使用它既可以高级编程方式编写应用程序，又可以低级编程方式编写与硬件紧密协作的库。在很多程序员看来，C++ 既是一种高级语言，让他们能够开发复杂的应用程序，又提供了极大的灵活性，让开发人员能够控制资源的使用和可用性，从而最大限度地提高性能。

虽然有更新的编程语言面世，如 Java 以及其他基于 .NET 的语言，但 C++ 始终深受欢迎并在不断发展。较新的语言因提供了某些功能（如通过垃圾收集管理内存）让一些程序员钟爱有加，但在需要精确控制应用程序的性能时，他们还是会选择 C++。当前，常常使用 C++ 编写 Web 服务器，并使用 HTML、Java 或 .NET 编写前端应用程序。

1.1.3 C++ 标准的发展历程

经过多年的发展，C++ 得到了广泛接受和采纳，但存在多种不同版本，因为有很多不同的编译

器，它们风格各异。鉴于 C++ 广受欢迎，且不同的版本之间存在差异，这导致了众多互操作性和移植方面的问题，需要对其进行标准化。

1998 年，第一个 C++ 标准获得了 ISO 标准委员会的批准，这就是 ISO/IEC 14882:1998。2003 年进行了修订，即为 ISO/IEC 14882:2003。最新的 C++ 标准于 2011 年 8 月获批，其官方名称为 C++11 (ISO/IEC 14882:2011)，它包含一些雄心勃勃的改进。

注意

网上的很多文档仍称这个 C++ 标准为 C++0x。人们最初预期新标准将于 2008 或 2009 年获批，因此用 x 表示获批的年份。但提议的新标准到 2011 年才获批，因此将其称为 C++11。换句话说，C++11 是新的 C++0x。

1.1.4 哪些人使用 C++ 程序

无论你是谁，做什么工作（无论是经验丰富的程序员，还是将计算机用于特定目的的人），都可能经常会用到 C++ 应用程序和库。C++ 常用于开发操作系统、设备驱动程序、办公软件、Web 服务器、基于云的应用程序和搜索引擎，甚至用于编写新编程语言编译器。

1.2 编写 C++ 应用程序

当您在计算机上启动 Notepad 或 VI 时，实际上是命令处理器运行该程序的可执行文件。可执行文件是可运行的成品，应按程序员期望的那样做。

1.2.1 生成可执行文件的步骤

要创建可在操作系统中运行的可执行文件，第一步是编写一个 C++ 程序。创建 C++ 应用程序的基本步骤如下。

1. 使用文本编辑器编写 C++ 代码。
2. 使用 C++ 编译器对代码进行编译，将代码转换为包含在目标文件中的机器语言版本。
3. 使用链接程序链接编译器的输出，生成一个可执行文件（如 Windows 中的 .exe 文件）。

您在编程时创建的是文本文件，但微处理器无法处理这样的文件。在编译过程中，C++ 代码（通常包含在 .CPP 文本文件中）被转换为处理器能够处理的字节码。编译器每次转换一个代码文件，生成一个扩展名为 .o 或 .obj 的目标文件，并忽略这个 CPP 文件可能对其他文件中代码的依赖。解析这些依存关系的工作由链接程序负责。除将各种目标文件组合起来外，链接程序还建立依存关系，如果链接成功，则创建一个可执行文件，供程序员执行和分发。

1.2.2 分析并修复错误

大多数复杂应用程序很少能够一次通过编译并完美地运行，由众多程序员合作开发的应用程序尤其如此。无论使用什么语言（包括 C++）编写，庞大或复杂的应用程序都需要运行很多次，以分析问题和发现 Bug。修复 Bug 后，重新生成程序，再重复上述过程。因此，除编写、编译和链接等三个步骤外，开发过程通常还包括调试步骤。在这个步骤中，程序员使用工具（如监视点）和调试功能（如逐行执行应用程序）对应用程序中的异常和错误进行分析。

1.2.3 集成开发环境

很多程序员都喜欢使用集成开发环境（Integrated Development Environment, IDE）。集成开