

异步图书
www.epubit.com.cn

SAMS

畅销全球的
轻量级C++经典教程

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

中文版
累计销量
70000册

涵盖C++14和C++17新标准

21天学通 C++ (第8版)

[美] Siddhartha Rao 著
袁国忠 译

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

21天学通 C++ (第8版)

[美] Siddhartha Rao 著
袁国忠 译

人民邮电出版社
北京

图书在版编目 (C I P) 数据

21天学通C++ : 第8版 / (美) 悉达多·饶
(Siddhartha Rao) 著 ; 袁国忠译. -- 北京 : 人民邮电
出版社, 2017.9

ISBN 978-7-115-46588-7

I. ①2… II. ①悉… ②袁… III. ①C语言—程序设
计 IV. ①TP312.8

中国版本图书馆CIP数据核字(2017)第189196号

版权声明

Siddhartha Rao: Sams Teach Yourself C++ in One Hour a Day(8th Edition)

ISBN: 0789757745

Copyright © 2017 by Pearson Education, Inc.

Authorized translation from the English languages edition published by Pearson Education, Inc.

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。

-
- ◆ 著 [美] Siddhartha Rao
 - 译 袁国忠
 - 责任编辑 傅道坤
 - 责任印制 焦志炜

 - ◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
三河市海波印务有限公司印刷

 - ◆ 开本: 787×1092 1/16
印张: 34.25
字数: 1008 千字 2017 年 9 月第 1 版
印数: 1-3 000 册 2017 年 9 月河北第 1 次印刷
- 著作权合同登记号 图字: 01-2016-9627 号
-

定价: 79.00 元

读者服务热线: (010)81055410 印装质量热线: (010)81055316

反盗版热线: (010)81055315

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、C++14 和 C++17 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、列表初始化、`lambda` 表达式、自动类型推断等。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 E 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 SE 负责安全响应的副总裁。C++的发展让他深信，您能编写速度更快、更简洁、更强大的 C++应用程序。Siddhartha 酷爱旅游，还是山地自行车运动的狂热爱好者；他期待着您对本书的反馈。

献辞

谨以本书献给我的父亲，您永远是我的灵感来源。

致谢

感谢家人对我无尽的支持，感谢我的妻子 Clara，感谢为本书贡献颇多的编辑人员。

前言

对 C++ 来说，2011 年和 2014 年都是很特别的年份。C++11 做了重大的改进，新增了一些可提高编程效率的关键字和结构，而 C++14 做了增量式改进，完善了 C++11 引入的新特性。

本书旨在帮助您循序渐进地学习 C++，其中的章节经过深思熟虑，从实用的角度介绍这种面向对象编程语言的基本知识。读者只需每天花 1 小时，就能掌握 C++。

学习 C++ 的最佳方式是动手实践。本书包含了丰富的代码示例，有助于提高编程技能，请读者务必亲自动手尝试这些代码。这些代码片段都使用了最新版本的编译器进行了测试，具体地说是 Microsoft Visual C++ 和 GNU C++ 编译器，它们都支持大量的 C++14 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，但这并非必须。本书也可供熟悉 C++ 但想了解其新增功能的读者参考。如果您是专业程序员，第三部分（学习标准模板库）将有助于您创建更优质、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。C++11 和 C++14 引入的新概念散落在本书各个相关的章节中。本书包含如下 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，并介绍一些常见的 C++ 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，帮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（STL）”，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 风格的字符串 `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 `sort` 操作。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，还展望了下一个 ISO 标准——C++17 有望引入的新特性。

本书使用的约定

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。
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	不应该
提供当前章介绍的基本原理的摘要。	提供一些有用的信息。

示例代码

本书的示例代码可从 www.epubit.com/book/details/4780 下载。

目录

第 1 章 绪论	1	3.1.2 声明变量以访问和使用内存	17
1.1 C++简史	1	3.1.3 声明并初始化多个类型相同的变量	19
1.1.1 与 C 语言的关系	1	3.1.4 理解变量的作用域	19
1.1.2 C++的优点	1	3.1.5 全局变量	20
1.1.3 C++标准的发展历程	2	3.1.6 命名约定	22
1.1.4 哪些人使用 C++程序	2	3.2 编译器支持的常见 C++变量类型	22
1.2 编写 C++应用程序	2	3.2.1 使用 bool 变量存储布尔值	23
1.2.1 生成可执行文件的步骤	2	3.2.2 使用 char 变量存储字符	23
1.2.2 分析并修复错误	2	3.2.3 有符号整数和无符号整数的概念	24
1.2.3 集成开发环境	3	3.2.4 有符号整型 short、int、long 和 long long	24
1.2.4 编写第一个 C++应用程序	3	3.2.5 无符号整型 unsigned short、unsigned int、unsigned long 和 unsigned long long	25
1.2.5 生成并执行第一个 C++应用程序	4	3.2.6 选择正确的数据类型以免发生溢出错误	25
1.2.6 理解编译错误	5	3.2.7 浮点类型 float 和 double	26
1.3 C++新增的功能	5	3.3 使用 sizeof 确定变量的长度	26
1.4 总结	5	3.4 使用 auto 自动推断类型	28
1.5 问与答	6	3.5 使用 typedef 替换变量类型	29
1.6 作业	6	3.6 什么是常量	30
1.6.1 测验	6	3.6.1 字面常量	30
1.6.2 练习	6	3.6.2 使用 const 将变量声明为常量	30
第 2 章 C++程序的组成部分	8	3.6.3 使用 constexpr 定义常量表达式	31
2.1 Hello World 程序的组成部分	8	3.6.4 枚举	32
2.1.1 预处理器编译指令#include	9	3.6.5 使用#define 定义常量	34
2.1.2 程序的主体——main()	9	3.7 不能用作常量或变量名的关键字	34
2.1.3 返回值	10	3.8 总结	35
2.2 名称空间的概念	10	3.9 问与答	36
2.3 C++代码中的注释	11	3.10 作业	37
2.4 C++函数	12	3.10.1 测验	37
2.5 使用 std::cin 和 std::cout 执行基本输入输出操作	14	3.10.2 练习	37
2.6 总结	15	第 4 章 管理数组和字符串	38
2.7 问与答	15	4.1 什么是数组	38
2.8 作业	15	4.1.1 为何需要数组	38
2.8.1 测验	16	4.1.2 声明和初始化静态数组	39
2.8.2 练习	16	4.1.3 数组中的数据是如何存储的	39
第 3 章 使用变量和常量	17		
3.1 什么是变量	17		
3.1.1 内存和寻址概述	17		

4.1.4 访问存储在数组中的数据	40	第 6 章 控制程序流程	71
4.1.5 修改存储在数组中的数据	41	6.1 使用 if...else 有条件地执行	71
4.2 多维数组	43	6.1.1 使用 if...else 进行条件编程	72
4.2.1 声明和初始化多维数组	44	6.1.2 有条件地执行多条语句	73
4.2.2 访问多维数组中的元素	44	6.1.3 嵌套 if 语句	74
4.3 动态数组	45	6.1.4 使用 switch-case 进行条件处理	77
4.4 C 风格字符串	46	6.1.5 使用运算符?:进行条件处理	80
4.5 C++字符串: 使用 std::string	48	6.2 在循环中执行代码	81
4.6 总结	50	6.2.1 不成熟的 goto 循环	81
4.7 问与答	50	6.2.2 while 循环	83
4.8 作业	50	6.2.3 do...while 循环	84
4.8.1 测验	51	6.2.4 for 循环	86
4.8.2 练习	51	6.2.5 基于范围的 for 循环	88
第 5 章 使用表达式、语句和运算符	52	6.3 使用 continue 和 break 修改循环的 行为	90
5.1 语句	52	6.3.1 不结束的循环——无限循环	90
5.2 复合语句(语句块)	53	6.3.2 控制无限循环	91
5.3 使用运算符	53	6.4 编写嵌套循环	93
5.3.1 赋值运算符(=)	53	6.4.1 使用嵌套循环遍历多维数组	94
5.3.2 理解左值和右值	53	6.4.2 使用嵌套循环计算斐波纳契 数列	95
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(%)	53	6.5 总结	96
5.3.4 递增运算符(++和递减运算符 (--)	54	6.6 问与答	96
5.3.5 前缀还是后缀	55	6.7 作业	97
5.3.6 相等运算符(==)和不等运算符 (!=)	56	6.7.1 测验	97
5.3.7 关系运算符	56	6.7.2 练习	97
5.3.8 逻辑运算 NOT、AND、OR 和 XOR	58	第 7 章 使用函数组织代码	99
5.3.9 使用 C++逻辑运算 NOT(!)、AND (&&)和 OR()	59	7.1 为何需要函数	99
5.3.10 按位运算符 NOT(~)、AND(&）、 OR()和 XOR(^)	63	7.1.1 函数原型是什么	100
5.3.11 按位右移运算符(>>)和左移 运算符(<<)	64	7.1.2 函数定义是什么	101
5.3.12 复合赋值运算符	65	7.1.3 函数调用和实参是什么	101
5.3.13 使用运算符 sizeof 确定变量占用 的内存量	67	7.1.4 编写接受多个参数的函数	101
5.3.14 运算符优先级	68	7.1.5 编写没有参数和返回值的函数	103
5.4 总结	69	7.1.6 带默认值的函数参数	103
5.5 问与答	69	7.1.7 递归函数——调用自己的函数	105
5.6 作业	70	7.1.8 包含多条 return 语句的函数	106
5.6.1 测验	70	7.2 使用函数处理不同类型的数据	107
5.6.2 练习	70	7.2.1 函数重载	107
		7.2.2 将数组传递给函数	109
		7.2.3 按引用传递参数	110
		7.3 微处理器如何处理函数调用	111
		7.3.1 内联函数	112
		7.3.2 自动推断返回类型	113
		7.3.3 lambda 函数	114
		7.4 总结	115

7.5 问与答	116	9.2 关键字 public 和 private	147
7.6 作业	116	9.3 构造函数	150
7.6.1 测验	116	9.3.1 声明和实现构造函数	150
7.6.2 练习	116	9.3.2 何时及如何使用构造函数	151
第 8 章 阐述指针和引用	118	9.3.3 重载构造函数	152
8.1 什么是指针	118	9.3.4 没有默认构造函数的类	154
8.1.1 声明指针	119	9.3.5 带默认值的构造函数参数	155
8.1.2 使用引用运算符 (&) 获取变量的地址	119	9.3.6 包含初始化列表的构造函数	156
8.1.3 使用指针存储地址	120	9.4 析构函数	157
8.1.4 使用解除引用运算符 (*) 访问指向的数据	122	9.4.1 声明和实现析构函数	157
8.1.5 将 sizeof() 用于指针的结果	124	9.4.2 何时及如何使用析构函数	158
8.2 动态内存分配	125	9.5 复制构造函数	160
8.2.1 使用 new 和 delete 动态地分配和释放内存	125	9.5.1 浅复制及其存在的问题	160
8.2.2 将递增和递减运算符 (++和--) 用于指针的结果	127	9.5.2 使用复制构造函数确保深复制	162
8.2.3 将关键字 const 用于指针	129	9.5.3 有助于改善性能的移动构造函数	166
8.2.4 将指针传递给函数	130	9.6 构造函数和析构函数的其他用途	166
8.2.5 数组和指针的类似之处	131	9.6.1 不允许复制的类	167
8.3 使用指针时常犯的编程错误	133	9.6.2 只能有一个实例的单例类	167
8.3.1 内存泄露	133	9.6.3 禁止在栈中实例化的类	169
8.3.2 指针指向无效的内存单元	133	9.6.4 使用构造函数进行类型转换	171
8.3.3 悬浮指针 (也叫迷途或失控指针)	134	9.7 this 指针	172
8.3.4 检查使用 new 发出的分配请求是否得到满足	135	9.8 将 sizeof() 用于类	173
8.4 指针编程最佳实践	137	9.9 结构不同于类的地方	175
8.5 引用是什么	137	9.10 声明友元	176
8.5.1 是什么让引用很有用	138	9.11 共用体: 一种特殊的数据存储机制	178
8.5.2 将关键字 const 用于引用	139	9.11.1 声明共用体	178
8.5.3 按引用向函数传递参数	140	9.11.2 在什么情况下使用共用体	178
8.6 总结	140	9.12 对类和结构使用聚合初始化	180
8.7 问与答	141	9.13 总结	183
8.8 作业	142	9.14 问与答	183
8.8.1 测验	142	9.15 作业	184
8.8.2 练习	142	9.15.1 测验	184
第 9 章 类和对象	144	9.15.2 练习	184
9.1 类和对象	144	第 10 章 实现继承	185
9.1.1 声明类	145	10.1 继承基础	185
9.1.2 作为类实例的对象	145	10.1.1 继承和派生	186
9.1.3 使用句点运算符访问成员	146	10.1.2 C++ 派生语法	186
9.1.4 使用指针运算符 (->) 访问成员	146	10.1.3 访问限定符 protected	188
		10.1.4 基类初始化——向基类传递参数	190
		10.1.5 在派生类中覆盖基类的方法	192
		10.1.6 调用基类中被覆盖的方法	194
		10.1.7 在派生类中调用基类的方法	194

10.1.8	在派生类中隐藏基类的方法	196	12.3.5	重载运算符<、>、<=和>=	245
10.1.9	构造顺序	198	12.3.6	重载复制赋值运算符(=)	248
10.1.10	析构顺序	198	12.3.7	下标运算符	250
10.2	私有继承	200	12.4	函数运算符operator()	253
10.3	保护继承	202	12.5	用于高性能编程的移动构造函数和 移动赋值运算符	254
10.4	切除问题	205	12.5.1	不必要的复制带来的问题	254
10.5	多继承	205	12.5.2	声明移动构造函数和移动赋值 运算符	254
10.6	使用final禁止继承	207	12.6	用户定义的字面量	258
10.7	总结	208	12.7	不能重载的运算符	260
10.8	问与答	208	12.8	总结	261
10.9	作业	208	12.9	问与答	261
10.9.1	测验	208	12.10	作业	261
10.9.2	练习	209	12.10.1	测验	261
第11章	多态	210	12.10.2	练习	261
11.1	多态基础	210	第13章	类型转换运算符	262
11.1.1	为何需要多态行为	210	13.1	为何需要类型转换	262
11.1.2	使用虚函数实现多态行为	212	13.2	为何有些C++程序员不喜欢C风格 类型转换	263
11.1.3	为何需要虚构造函数	213	13.3	C++类型转换运算符	263
11.1.4	虚函数的工作原理——理解 虚函数表	217	13.3.1	使用static_cast	263
11.1.5	抽象基类和纯虚函数	220	13.3.2	使用dynamic_cast和运行阶段 类型识别	264
11.2	使用虚继承解决菱形问题	222	13.3.3	使用reinterpret_cast	267
11.3	表明覆盖意图的限定符override	225	13.3.4	使用const_cast	267
11.4	使用final来禁止覆盖函数	226	13.4	C++类型转换运算符存在的问题	268
11.5	可将复制构造函数声明为虚函数吗	227	13.5	总结	269
11.6	总结	230	13.6	问与答	269
11.7	问与答	230	13.7	作业	270
11.8	作业	231	13.7.1	测验	270
11.8.1	测验	231	13.7.2	练习	270
11.8.2	练习	231	第14章	宏和模板简介	271
第12章	运算符类型与运算符重载	232	14.1	预处理器与编译器	271
12.1	C++运算符	232	14.2	使用#define定义常量	271
12.2	单目运算符	233	14.3	使用#define编写宏函数	274
12.2.1	单目运算符的类型	233	14.3.1	为什么要使用括号	276
12.2.2	单目递增与单目递减运算符	234	14.3.2	使用assert宏验证表达式	276
12.2.3	转换运算符	236	14.3.3	使用宏函数的优点和缺点	277
12.2.4	解除引用运算符(*)和成员选择 运算符(->)	238	14.4	模板简介	278
12.3	双目运算符	239	14.4.1	模板声明语法	278
12.3.1	双目运算符的类型	240	14.4.2	各种类型的模板声明	279
12.3.2	双目加法与双目减法运算符	240	14.4.3	模板函数	279
12.3.3	实现运算符+=与-=	242	14.4.4	模板与类型安全	281
12.3.4	重载等于运算符(==)和不等运算 符(!=)	243			

14.4.5	模板类	281	16.6.2	练习	313
14.4.6	声明包含多个参数的模板	282	第 17 章 STL 动态数组类		314
14.4.7	声明包含默认参数的模板	283	17.1	std::vector 的特点	314
14.4.8	一个模板示例	283	17.2	典型的 vector 操作	314
14.4.9	模板的实例化和具体化	284	17.2.1	实例化 vector	314
14.4.10	模板类和静态成员	286	17.2.2	使用 push_back() 在末尾插入元素	316
14.4.11	参数数量可变的模板	287	17.2.3	列表初始化	317
14.4.12	使用 static_assert 执行编译阶段检查	290	17.2.4	使用 insert() 在指定位置插入元素	317
14.4.13	在实际 C++ 编程中使用模板	290	17.2.5	使用数组语法访问 vector 中的元素	319
14.5	总结	291	17.2.6	使用指针语法访问 vector 中的元素	320
14.6	问与答	291	17.2.7	删除 vector 中的元素	321
14.7	作业	291	17.3	理解大小和容量	322
14.7.1	测验	291	17.4	STL deque 类	324
14.7.2	练习	292	17.5	总结	326
第 15 章 标准模板库简介		293	17.6	问与答	326
15.1	STL 容器	293	17.7	作业	327
15.1.1	顺序容器	293	17.7.1	测验	327
15.1.2	关联容器	294	17.7.2	练习	327
15.1.3	容器适配器	294	第 18 章 STL list 和 forward_list		328
15.2	STL 迭代器	295	18.1	std::list 的特点	328
15.3	STL 算法	295	18.2	基本的 list 操作	328
15.4	使用迭代器在容器和算法之间交互	295	18.2.1	实例化 std::list 对象	328
15.5	选择正确的容器	297	18.2.2	在 list 开头或末尾插入元素	330
15.6	STL 字符串类	298	18.2.3	在 list 中间插入元素	331
15.7	总结	298	18.2.4	删除 list 中的元素	333
15.8	问与答	299	18.3	对 list 中的元素进行反转和排序	334
15.9	作业	299	18.3.1	使用 list::reverse() 反转元素的排列顺序	334
第 16 章 STL string 类		300	18.3.2	对元素进行排序	335
16.1	为何需要字符串操作类	300	18.3.3	对包含对象的 list 进行排序以及删除其中的元素	337
16.2	使用 STL string 类	301	18.3.4	C++11 引入的 std::forward_list	340
16.2.1	实例化和复制 STL string	301	18.4	总结	341
16.2.2	访问 std::string 的字符内容	303	18.5	问与答	342
16.2.3	拼接字符串	305	18.6	作业	342
16.2.4	在 string 中查找字符或子字符串	306	18.6.1	测验	342
16.2.5	截短 STL string	307	18.6.2	练习	342
16.2.6	字符串反转	309	第 19 章 STL 集合类		343
16.2.7	字符串的大小写转换	310	19.1	简介	343
16.3	基于模板的 STL string 实现	311			
16.4	总结	312			
16.5	问与答	312			
16.6	作业	313			
16.6.1	测验	313			

19.2 STL set 和 multiset 的基本操作	344	21.5 作业	384
19.2.1 实例化 std::set 对象	344	21.5.1 测验	384
19.2.2 在 set 或 multiset 中插入元素	345	21.5.2 练习	384
19.2.3 在 STL set 或 multiset 中查找元素	347	第 22 章 lambda 表达式	385
19.2.4 删除 STL set 或 multiset 中的元素	348	22.1 lambda 表达式是什么	385
19.3 使用 STL set 和 multiset 的优缺点	352	22.2 如何定义 lambda 表达式	386
19.4 总结	354	22.3 一元函数对应的 lambda 表达式	386
19.5 问与答	355	22.4 一元谓词对应的 lambda 表达式	387
19.6 作业	355	22.5 通过捕获列表接受状态变量的 lambda 表达式	388
19.6.1 测验	355	22.6 lambda 表达式的通用语法	390
19.6.2 练习	355	22.7 二元函数对应的 lambda 表达式	391
第 20 章 STL 映射类	356	22.8 二元谓词对应的 lambda 表达式	392
20.1 STL 映射类简介	356	22.9 总结	394
20.2 STL map 和 multimap 的基本操作	357	22.10 问与答	394
20.2.1 实例化 std::map 和 std::multimap	357	22.11 作业	395
20.2.2 在 STL map 或 multimap 中插入元素	358	22.11.1 测验	395
20.2.3 在 STL map 或 multimap 中查找元素	361	22.11.2 练习	395
20.2.4 在 STL multimap 中查找元素	363	第 23 章 STL 算法	396
20.2.5 删除 STL map 或 multimap 中的元素	363	23.1 什么是 STL 算法	396
20.3 提供自定义的排序谓词	365	23.2 STL 算法的分类	396
20.4 基于散列表的 STL 键-值对容器	368	23.2.1 非变序算法	396
20.4.1 散列表的工作原理	368	23.2.2 变序算法	397
20.4.2 使用 unordered_map 和 unordered_multimap	368	23.3 使用 STL 算法	398
20.5 总结	372	23.3.1 根据值或条件查找元素	398
20.6 问与答	372	23.3.2 计算包含给定值或满足给定条件的元素数	400
20.7 作业	372	23.3.3 在集合中搜索元素或序列	401
20.7.1 测验	373	23.3.4 将容器中的元素初始化为指定值	403
20.7.2 练习	373	23.3.5 使用 std::generate() 将元素设置为运行阶段生成的值	405
第 21 章 理解函数对象	374	23.3.6 使用 for_each() 处理指定范围内的元素	406
21.1 函数对象与谓词的概念	374	23.3.7 使用 std::transform() 对范围进行变换	407
21.2 函数对象的典型用途	374	23.3.8 复制和删除操作	409
21.2.1 一元函数	374	23.3.9 替换值以及替换满足给定条件的元素	412
21.2.2 一元谓词	378	23.3.10 排序、在有序集合中搜索以及删除重复元素	413
21.2.3 二元函数	380	23.3.11 将范围分区	415
21.2.4 二元谓词	381	23.3.12 在有序集合中插入元素	417
21.3 总结	383	23.4 总结	419
21.4 问与答	384		

23.5 问与答.....	419	第 26 章 理解智能指针.....	441
23.6 作业.....	419	26.1 什么是智能指针.....	441
23.6.1 测验.....	420	26.1.1 常规(原始)指针存在的 问题.....	441
23.6.2 练习.....	420	26.1.2 智能指针有何帮助.....	442
第 24 章 自适应容器: 栈和队列.....	421	26.2 智能指针是如何实现的.....	442
24.1 栈和队列的行为特征.....	421	26.3 智能指针类型.....	443
24.1.1 栈.....	421	26.3.1 深复制.....	443
24.1.2 队列.....	422	26.3.2 写时复制机制.....	445
24.2 使用 STL stack 类.....	422	26.3.3 引用计数智能指针.....	445
24.2.1 实例化 stack.....	422	26.3.4 引用链接智能指针.....	445
24.2.2 stack 的成员函数.....	423	26.3.5 破坏性复制.....	445
24.2.3 使用 push() 和 pop() 在栈顶插入和 删除元素.....	424	26.3.6 使用 std::unique_ptr.....	447
24.3 使用 STL queue 类.....	425	26.4 深受欢迎的智能指针库.....	449
24.3.1 实例化 queue.....	425	26.5 总结.....	449
24.3.2 queue 的成员函数.....	426	26.6 问与答.....	449
24.3.3 使用 push() 在队尾插入以及使用 pop() 从队首删除.....	427	26.7 作业.....	450
24.4 使用 STL 优先级队列.....	428	26.7.1 测试.....	450
24.4.1 实例化 priority_queue 类.....	428	26.7.2 练习.....	450
24.4.2 priority_queue 的成员 函数.....	429	第 27 章 使用流进行输入和输出.....	451
24.4.3 使用 push() 在 priority_queue 末尾 插入以及使用 pop() 在 priority_queue 开头删除.....	430	27.1 流的概述.....	451
24.5 总结.....	432	27.2 重要的 C++ 流类和流 对象.....	452
24.6 问与答.....	432	27.3 使用 std::cout 将指定格式的数据 写入控制台.....	453
24.7 作业.....	432	27.3.1 使用 std::cout 修改数字的显示 格式.....	453
24.7.1 测验.....	432	27.3.2 使用 std::cout 对齐文本和设置 字段宽度.....	455
24.7.2 练习.....	432	27.4 使用 std::cin 进行输入.....	455
第 25 章 使用 STL 位标志.....	433	27.4.1 使用 std::cin 将输入读取到基本 类型变量中.....	455
25.1 bitset 类.....	433	27.4.2 使用 std::cin:get 将输入读取到 char* 缓冲区中.....	456
25.2 使用 std::bitset 及其成员.....	434	27.4.3 使用 std::cin 将输入读取到 std::string 中.....	457
25.2.1 std::bitset 的运算符.....	434	27.5 使用 std::fstream 处理文件.....	458
25.2.2 std::bitset 的成员方法.....	435	27.5.1 使用 open() 和 close() 打开和关闭 文件.....	459
25.3 vector<bool>.....	437	27.5.2 使用 open() 创建文本文件并使用 运算符<<写入文本.....	460
25.3.1 实例化 vector<bool>.....	437	27.5.3 使用 open() 和运算符>>读取文本 文件.....	460
25.3.2 vector<bool>的成员函数和 运算符.....	438	27.5.4 读写二进制文件.....	461
25.4 总结.....	439		
25.5 问与答.....	439		
25.6 作业.....	439		
25.6.1 测验.....	439		
25.6.2 练习.....	440		

27.6	使用 <code>std::stringstream</code> 对字符串进行转换	463	29.4.1	支持在 <code>if</code> 和 <code>switch</code> 中进行初始化	481
27.7	总结	464	29.4.2	保证复制得以避免	482
27.8	问与答	464	29.4.3	避免内存分配开销的 <code>std::string_view</code>	482
27.9	作业	465	29.4.4	类型安全的共用体替代品 <code>std::variant</code>	483
27.9.1	测验	465	29.4.5	使用 <code>if constexpr</code> 有条件地编译代码	483
27.9.2	练习	465	29.4.6	改进的 <code>lambda</code> 表达式	484
29.4.7	在构造函数中使用类型自动推断功能	484	29.5	更深入地学习 C++	484
29.5.1	在线文档	485	29.5.1	在线文档	485
29.5.2	提供指南和帮助的社区	485	29.5.2	提供指南和帮助的社区	485
29.6	总结	485	29.6	总结	485
29.7	问与答	485	29.7	问与答	485
29.8	作业	485	29.8	作业	485
第 28 章	异常处理	466	附录 A	二进制和十六进制	486
28.1	什么是异常	466	A.1	十进制	486
28.2	导致异常的原因	466	A.2	二进制	486
28.3	使用 <code>try</code> 和 <code>catch</code> 捕获异常	467	A.2.1	计算机为何使用二进制	487
28.3.1	使用 <code>catch(...)</code> 处理所有异常	467	A.2.2	位和字节	487
28.3.2	捕获特定类型的异常	468	A.2.3	1KB 相当于多少字节	487
28.3.3	使用 <code>throw</code> 引发特定类型的异常	469	A.3	十六进制	487
28.4	异常处理的工作原理	470	A.4	不同进制之间的转换	488
28.4.1	<code>std::exception</code> 类	472	A.4.1	通用转换步骤	488
28.4.2	从 <code>std::exception</code> 派生出自定义异常类	473	A.4.2	从十进制转换为二进制	488
28.5	总结	474	A.4.3	从十进制转换为十六进制	489
28.6	问与答	474	附录 B	C++关键字	490
28.7	作业	475	附录 C	运算符优先级	491
28.7.1	测验	475	附录 D	ASCII 码	492
28.7.2	练习	475	附录 E	答案	495
第 29 章	继续前行	477			
29.1	当今的处理器有何不同	477			
29.2	如何更好地利用多个内核	478			
29.2.1	线程是什么	478			
29.2.2	为何要编写多线程应用程序	479			
29.2.3	线程如何交换数据	479			
29.2.4	使用互斥量和信号量同步线程	480			
29.2.5	多线程技术带来的问题	480			
29.3	编写杰出的 C++ 代码	480			
29.4	C++17 有望引入的新特性	481			

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++ 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，但被广泛采用，仍在不断改进。本书编写期间，国际标准组织（ISO）批准的最新版 C++ 标准为 2014 年 12 月发布的 C++14。

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，旨在作为 C 语言的继任者。但不同于 C 语言，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成员数据的成员方法。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，很多 C++ 编译器都支持 C 语言。

注意 要学习 C++，并不要求您具备 C 语言编程方面的知识和经验。如果您的终极目标是学习 C++ 等面向对象编程语言，并不需要先学习 C 语言等过程性语言。

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，但在需要控制应用程序的资源使用和性能时，他们还是会选择 C++。当前，在分层架构中，常常使用 C++ 编写 Web 服务器，并使用 HTML、Java 或 .NET 编写其他组件。

1.1.3 C++标准的发展历程

经过多年的发展，C++被众多不同的平台接受和采纳，这些平台使用不同的编译器。鉴于不同的编译器之间存在差异，这导致了众多互操作性和移植方面的问题，因此需要对 C++进行标准化，让编译器厂商能够遵循标准的 C++语言规范。

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。从此以后，C++标准发生了翻天覆地的变化，极大地提高了 C++语言的可用性，并扩展了对标准库的支持。编写本书期间，获得批准的最新标准为 ISO/IEC 14882:2014，俗称 C++14。

注意 并非所有流行的编译器都会迅速而全面地支持最新标准，因此虽然从学术角度看，最好了解最新标准新增的功能，但就编写良好的 C++应用程序而言，这些新增功能并非是必不可少的。

1.1.4 哪些人使用 C++程序

使用 C++编写的应用程序、操作系统、Web 服务、数据库和企业软件多如牛毛，因此无论您从事什么工作，使用计算机来做什么，都可能正在使用 C++编写的软件。除软件工程师外，C++还常被物理学家和数学家用来从事研究工作。

1.2 编写 C++应用程序

当您在 Windows 系统上启动 Notepad 或在 Linux 系统上启动 Terminal 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. 使用链接器链接编译器的输出，生成一个可执行文件（如 Windows 中的 .exe 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执行的字节码。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，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。整个过程也被称为构建可执行文件。

1.2.2 分析并修复错误

大多数应用程序很少能够一次通过编译并按预期运行。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，以便通过测试来找出代码中的错误，即 bug。修复 bug 后，重新生成程序，再重复测试过程。因此，除编写、编译和链接等 3 个步骤外，软件开发过程通常还包括调试步骤。在这个步骤中，程序员对代码中的错误进行分析和修复。优秀的开发环境提供了帮助调试的工具和功能。