

<<21天学通C++>>

图书基本信息

书名：<<21天学通C++>>

13位ISBN编号：9787115296245

10位ISBN编号：7115296243

出版时间：2012-12

出版时间：人民邮电出版社

作者：Siddhartha Rao

页数：458

字数：870000

译者：袁国忠

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## 内容概要

《21天学通C++(第7版)》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，了解自己对所学习内容的掌握程度。

《21天学通C++(第7版)》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是一位Microsoft Visual C++ MVP，拥有在各种平台上编写驱动程序和应用程序的丰富经验。他受聘于SAP AG，当前为SAP Product Security India的负责人，主要职责包括制定软件开发最佳实践，以确保SAP软件的安全和全球竞争力。

## 书籍目录

## 第1章 绪论

## 1.1 C++简史

## 1.1.1 与C语言的关系

## 1.1.2 C++的优点

## 1.1.3 C++标准的发展历程

## 1.1.4 哪些人使用C++程序

## 1.2 编写C++应用程序

## 1.2.1 生成可执行文件的步骤

## 1.2.2 分析并修复错误

## 1.2.3 集成开发环境

## 1.2.4 编写第一个C++应用程序

## 1.2.5 生成并执行第一个C++应用程序

## 1.2.6 理解编译错误

## 1.3 C++11新增的功能

## 1.4 总结

## 1.5 问与答

## 1.6 作业

## 1.6.1 测验

## 1.6.2 练习

## 第2章 C++程序的组成部分

## 2.1 Hello World程序的组成部分

## 2.1.1 预处理器编译指令#include

## 2.1.2 程序的主体-main()

## 2.1.3 返回值

## 2.2 名称空间的概念

## 2.3 C++代码中的注释

## 2.4 C++函数

## 2.5 使用std::cin和std::cout执行基本输入输出操作

## 2.6 总结

## 2.7 问与答

## 2.8 作业

## 2.8.1 测验

## 2.8.2 练习

## 第3章 使用变量和常量

## 3.1 什么是变量

## 3.1.1 内存和寻址概述

## 3.1.2 声明变量以访问和使用内存

## 3.1.3 声明并初始化多个类型相同的变量

## 3.1.4 理解变量的作用域

## 3.1.5 全局变量

## 3.2 编译器支持的常见C++变量类型

## 3.2.1 使用bool变量存储布尔值

## 3.2.2 使用char变量存储字符

## 3.2.3 有符号整数和无符号整数的概念

## 3.2.4 有符号整型short、int、long和long long

## &lt;&lt;21天学通C++&gt;&gt;

- 3.2.5 无符号整型unsigned short、 unsigned int、 unsigned long和unsigned long long
- 3.2.6 浮点类型float和double
- 3.3 使用sizeof确定变量的长度
- 3.4 使用typedef替换变量类型
- 3.5 什么是常量
  - 3.5.1 字面常量
  - 3.5.2 使用const将变量声明为常量
  - 3.5.3 使用constexpr声明常量
  - 3.5.4 枚举常量
  - 3.5.5 使用#define定义常量
- 3.6 给变量和常量命名
- 3.7 不能用作常量或变量名的关键字
- 3.8 总结
- 3.9 问与答
- 3.10 作业
  - 3.10.1 测验
  - 3.10.2 练习
- 第4章 管理数组和字符串
  - 4.1 什么是数组
    - 4.1.1 为何需要数组
    - 4.1.2 声明和初始化静态数组
    - 4.1.3 数组中的数据是如何存储的
    - 4.1.4 访问存储在数组中的数据
    - 4.1.5 修改存储在数组中的数据
  - 4.2 多维数组
    - 4.2.1 声明和初始化多维数组
    - 4.2.2 访问多维数组中的元素
  - 4.3 动态数组
  - 4.4 C风格字符串
  - 4.5 C++字符串：使用std::string
  - 4.6 总结
  - 4.7 问与答
  - 4.8 作业
    - 4.8.1 测验
    - 4.8.2 练习
- 第5章 使用表达式、语句和运算符
  - 5.1 语句
  - 5.2 复合语句(语句块)
  - 5.3 使用运算符
    - 5.3.1 赋值运算符(=)
    - 5.3.2 理解左值和右值
    - 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(%)
    - 5.3.4 递增运算符(++和递减运算符(——))
    - 5.3.5 前缀还是后缀
    - 5.3.6 相等运算符(==)和不等运算符(!=)
    - 5.3.7 关系运算符

## &lt;&lt;21天学通C++&gt;&gt;

- 5.3.8 逻辑运算NOT、AND、OR和XOR
- 5.3.9 使用C++逻辑运算NOT(!)、AND(&&)和OR(||)
- 5.3.10 按位运算符NOT(~)、AND(&);、OR(|)和XOR(^)
- 5.3.11 按位右移运算符(>>)和左移运算符(<<)访问成员
- 9.2 关键字public和private
  - 9.2.1 使用关键字private实现数据抽象
- 9.3 构造函数
  - 9.3.1 声明和实现构造函数
  - 9.3.2 何时及如何使用构造函数
  - 9.3.3 重载构造函数
  - 9.3.4 没有默认构造函数的类
  - 9.3.5 带默认值的构造函数参数
  - 9.3.6 包含初始化列表的构造函数
- 9.4 析构函数
  - 9.4.1 声明和实现析构函数
  - 9.4.2 何时及如何使用析构函数
- 9.5 复制构造函数
  - 9.5.1 浅复制及其存在的问题
  - 9.5.2 使用复制构造函数确保深复制
  - 9.5.3 有助于改善性能的移动构造函数
- 9.6 构造函数和析构函数的其他用途
  - 9.6.1 不允许复制的类
  - 9.6.2 只能有一个实例的单例类
  - 9.6.3 禁止在栈中实例化的类
- 9.7 this指针
- 9.8 将sizeof()用于类
- 9.9 结构不同于类的地方
- 9.10 声明友元
- 9.11 总结
- 9.12 问与答
- 9.13 作业
  - 9.13.1 测验
  - 9.13.2 练习
- 第10章 实现继承
  - 10.1 继承基础
    - 10.1.1 继承和派生
    - 10.1.2 C++派生语法
    - 10.1.3 访问限定符protected
    - 10.1.4 基类初始化-向基类传递参数
    - 10.1.5 在派生类中覆盖基类的方法
    - 10.1.6 调用基类中被覆盖的方法
    - 10.1.7 在派生类中调用基类的方法
    - 10.1.8 在派生类中隐藏基类的方法
    - 10.1.9 构造顺序
    - 10.1.10 析构顺序
  - 10.2 私有继承

## &lt;&lt;21天学通C++&gt;&gt;

- 10.3 保护继承
- 10.4 切除问题
- 10.5 多继承
- 10.6 总结
- 10.7 问与答
- 10.8 作业
- 10.8.1 测验
- 10.8.2 练习
- 第11章 多态
- 11.1 多态基础
- 11.1.1 为何需要多态行为
- 11.1.2 使用虚函数实现多态行为
- 11.1.3 为何需要虚构造函数
- 11.1.4 虚函数的工作原理——理解虚函数表
- 11.1.5 抽象基类和纯虚函数
- 11.2 使用虚继承解决菱形问题
- 11.3 可将复制构造函数声明为虚函数吗
- 11.4 总结
- 11.5 问与答
- 11.6 作业
- 11.6.1 测验
- 11.6.2 练习
- 第12章 运算符类型与运算符重载
- 12.1 C++运算符
- 12.2 单目运算符
- 12.2.1 单目运算符的类型
- 12.2.2 单目递增与单目递减运算符
- 12.2.3 转换运算符
- 12.2.4 解除引用运算符(\*)和成员选择运算符(-&gt;)
- 12.3 双目运算符
- 12.3.1 双目运算符的类型
- 12.3.2 双目加法与双目减法运算符
- 12.3.3 实现运算符+=与(=
- 12.3.4 重载等于运算符(==)和不等运算符(!=)
- 12.3.5 重载运算符、=
- 12.3.6 重载复制赋值运算符(=)
- 12.3.7 下标运算符
- 12.4 函数运算符operator()
- 12.5 不能重载的运算符
- 12.6 总结
- 12.7 问与答
- 12.8 作业
- 12.8.1 测验
- 12.8.2 练习
- 第13章 类型转换运算符
- 13.1 为何需要类型转换
- 13.2 为何有些C++程序员不喜欢C风格类型转换

## &lt;&lt;21天学通C++&gt;&gt;

## 13.3 C++类型转换运算符

## 13.3.1 使用static\_cast

## 13.3.2 使用dynamic\_cast和运行阶段类型识别

## 13.3.3 使用reinterpret\_cast

## 13.3.4 使用const\_cast

## 13.4 C++类型转换运算符存在的问题

## 13.5 总结

## 13.6 问与答

## 13.7 作业

## 13.7.1 测验

## 13.7.2 练习

## 第14章 宏和模板简介

## 14.1 预处理器与编译器

## 14.2 使用#define定义常量

## 14.3 使用#define编写宏函数

## 14.3.1 为什么要使用括号

## 14.3.2 使用assert宏验证表达式

## 14.3.3 使用宏函数的优点和缺点

## 14.4 模板简介

## 14.4.1 模板声明语法

## 14.4.2 各种类型的模板声明

## 14.4.3 模板函数

## 14.4.4 模板与类型安全

## 14.4.5 模板类

## 14.4.6 模板的实例化和具体化

## 14.4.7 声明包含多个参数的模板

## 14.4.8 声明包含默认参数的模板

## 14.4.9 一个模板示例

## 14.4.10 模板类和静态成员

## 14.4.11 在实际C++编程中使用模板

## 14.5 总结

## 14.6 问与答

## 14.7 作业

## 14.7.1 测验

## 14.7.2 练习

## 第15章 标准模板库简介

## 15.1 STL容器

## 15.1.1 顺序容器

## 15.1.2 关联容器

## 15.1.3 选择正确的容器

## 15.1.4 容器适配器

## 15.2 STL迭代器

## 15.3 STL算法

## 15.4 使用迭代器在容器和算法之间交互

## 15.5 STL字符串类

## 15.6 总结

## 15.7 问与答



## &lt;&lt;21天学通C++&gt;&gt;

15.8 作业

第16章 STL string类

16.1 为何需要字符串操作类

16.2 使用STL string类

16.2.1 实例化和复制STL string

16.2.2 访问std::string的字符内容

16.2.3 拼接字符串

16.2.4 在string中查找字符或子字符串

16.2.5 截短STL string

16.2.6 字符串反转

16.2.7 字符串的大小写转换

16.3 基于模板的STL string实现

16.4 总结

16.5 问与答

16.6 作业

16.6.1 测验

16.6.2 练习

第17章 STL动态数组类

17.1 std::vector的特点

17.2 典型的vector操作

17.2.1 实例化vector

17.2.2 使用push\_back()在末尾插入元素

17.2.3 使用insert()在指定位置插入元素

17.2.4 使用数组语法访问vector中的元素

17.2.5 使用指针语法访问vector中的元素

17.2.6 删除vector中的元素

17.3 理解大小和容量

17.4 STL deque 类

17.5 总结

17.6 问与答

17.7 作业

17.7.1 测验

17.7.2 练习

第18章 STL list和forward\_list

18.1 std::list的特点

18.2 基本的list操作

18.2.1 实例化std::list对象

18.2.2 在list开头或末尾插入元素

18.2.3 在list中间插入元素

18.2.4 删除list中的元素

18.3 对list中的元素进行反转和排序

18.3.1 使用list::reverse()反转元素的排列顺序

18.3.2 对元素进行排序

18.3.3 对包含对象的list进行排序以及删除其中的元素

18.4 总结

18.5 问与答

18.6 作业

## &lt;&lt;21天学通C++&gt;&gt;

18.6.1 测验

18.6.2 练习

## 第19章 STL集合类

19.1 简介

19.2 STL set和multiset的基本操作

19.2.1 实例化std::set对象

19.2.2 在set或multiset中插入元素

19.2.3 在STL set或multiset中查找元素

19.2.4 删除STL set或multiset中的元素

19.3 使用STL set和multiset的优缺点

19.4 总结

19.5 问与答

19.6 作业

19.6.1 测验

19.6.2 练习

## 第20章 STL映射类

20.1 STL映射类简介

20.2 std::map和std::multimap的基本操作

20.2.1 实例化std::map和std::multimap

20.2.2 在STL map或multimap中插入元素

20.2.3 在STL map或multimap中查找元素

20.2.4 在STL multimap中查找元素

20.2.5 删除STL map或multimap中的元素

20.3 提供自定义的排序谓词

20.3.1 散列表的工作原理

20.3.2 使用C++11散列表unordered\_map和unordered\_multimap

20.4 总结

20.5 问与答

20.6 作业

20.6.1 测验

20.6.2 练习

## 第21章 理解函数对象

21.1 函数对象与谓词的概念

21.2 函数对象的典型用途

21.2.1 一元函数

21.2.2 一元谓词

21.2.3 二元函数

21.2.4 二元谓词

21.3 总结

21.4 问与答

21.5 作业

21.5.1 测验

21.5.2 练习

## 第22章 C++ lambda表达式

22.1 lambda表达式是什么

22.2 如何定义lambda表达式

22.3 一元函数对应的lambda表达式

## &lt;&lt;21天学通C++&gt;&gt;

- 22.4 一元谓词对应的lambda表达式
- 22.5 通过捕获列表接受状态变量的lambda表达式
- 22.6 lambda表达式的通用语法
- 22.7 二元函数对应的lambda表达式
- 22.8 二元谓词对应的lambda表达式
- 22.9 总结
- 22.10 问与答
- 22.11 作业
  - 22.11.1 测验
  - 22.11.2 练习
- 第23章 STL算法
  - 23.1 什么是STL算法
  - 23.2 STL算法的分类
    - 23.2.1 非变序算法
    - 23.2.2 变序算法
  - 23.3 使用STL算法
    - 23.3.1 根据值或条件查找元素
    - 23.3.2 计算包含给定值或满足给定条件的元素数
    - 23.3.3 在集合中搜索元素或序列
    - 23.3.4 将容器中的元素初始化为指定值
    - 23.3.5 使用std::generate()将元素设置为运行阶段生成的值
    - 23.3.6 使用for\_each()处理指定范围内的元素
    - 23.3.7 使用std::transform()对范围进行变换
    - 23.3.8 复制和删除操作
    - 23.3.9 替换值以及替换满足给定条件的元素
    - 23.3.10 排序、在有序集合中搜索以及删除重复元素
    - 23.3.11 将范围分区
    - 23.3.12 在有序集合中插入元素
  - 23.4 总结
  - 23.5 问与答
  - 23.6 作业
    - 23.6.1 测验
    - 23.6.2 练习
- 第24章 自适应容器：栈和队列
  - 24.1 栈和队列的行为特征
    - 24.1.1 栈
    - 24.1.2 队列
  - 24.2 使用STL stack类
    - 24.2.1 实例化stack
    - 24.2.2 stack的成员函数
    - 24.2.3 使用push()和pop()在栈顶插入和删除元素
  - 24.3 使用STL queue类
    - 24.3.1 实例化queue
    - 24.3.2 queue的成员函数
    - 24.3.3 使用push()在队尾插入以及使用pop()从队首删除
  - 24.4 使用STL优先级队列
    - 24.4.1 实例化priority\_queue类

## &lt;&lt;21天学通C++&gt;&gt;

24.4.2 priority\_queue的成员函数

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

24.5 总结

24.6 问与答

24.7 作业

24.7.1 测验

24.7.2 练习

第25章 使用STL位标志

25.1 bitset类

25.2 使用std::bitset及其成员

25.2.1 std::bitset的运算符

25.2.2 std::bitset的成员方法

25.3 vector

25.3.1 实例化vector

25.3.2 vector的成员函数和运算符

25.4 总结

25.5 问与答

25.6 作业

25.6.1 测验

25.6.2 练习

第26章 理解智能指针

26.1 什么是智能指针

26.1.1 常规(原始)指针存在的问题

26.1.2 智能指针有何帮助

26.2 智能指针是如何实现的

26.3 智能指针类型

26.3.1 深复制

26.3.2 写时复制机制

26.3.3 引用计数智能指针

26.3.4 引用链接智能指针

26.3.5 破坏性复制

26.4 深受欢迎的智能指针库

26.5 总结

26.6 问与答

26.7 作业

26.7.1 测试

26.7.2 练习

第27章 使用流进行输入和输出

27.1 流的概述

27.2 重要的C++流类和流对象

27.3 使用std::cout将指定格式的数据写入控制台

27.3.1 使用std::cout修改数字的显示格式

27.3.2 使用std::cout对齐文本和设置字段宽度

27.4 使用std::cin进行输入

27.4.1 使用std::cin将输入读取到基本类型变量中

27.4.2 使用std::cin:get将输入读取到char数组中

## &lt;&lt;21天学通C++&gt;&gt;

- 27.4.3 使用std::cin将输入读取到std::string中
- 27.5 使用std::fstream处理文件
  - 27.5.1 使用open()和close()打开和关闭文件
  - 27.5.2 使用open()创建文本文件并使用运算符<>读取文本文件
  - 27.5.4 读写二进制文件
- 27.6 使用std::stringstream对字符串进行转换
- 27.7 总结
- 27.8 问与答
- 27.9 作业
  - 27.9.1 测验
  - 27.9.2 练习
- 第28章 异常处理
  - 28.1 什么是异常
  - 28.2 导致异常的原因
  - 28.3 使用try和catch捕获异常
    - 28.3.1 使用catch(...)处理所有异常
    - 28.3.2 捕获特定类型的异常
    - 28.3.3 使用throw引发特定类型的异常
  - 28.4 异常处理的工作原理
    - 28.4.1 std::exception类
    - 28.4.2 从std::exception派生出自定义异常类
  - 28.5 总结
  - 28.6 问与答
  - 28.7 作业
    - 28.7.1 测验
    - 28.7.2 练习
- 第29章 继续前行
  - 29.1 当今的处理器有何不同
  - 29.2 如何更好地利用多个内核
    - 29.2.1 线程是什么
    - 29.2.2 为何要编写多线程应用程序
    - 29.2.3 线程如何交换数据
    - 29.2.4 使用互斥量和信号量同步线程
    - 29.2.5 多线程技术带来的问题
  - 29.3 编写杰出的C++代码
  - 29.4 更深入地学习C++
    - 29.4.1 在线文档
    - 29.4.2 提供指南和帮助的社区
  - 29.5 总结
  - 29.6 问与答
  - 29.7 作业
- 附录A 二进制和十六进制
  - A.1 十进制
  - A.2 二进制
    - A.2.1 计算机为何使用二进制
    - A.2.2 位和字节
    - A.2.3 1KB相当于多少字节

A.3 十六进制

A.4 不同进制之间的转换

A.4.1 通用转换步骤

A.4.2 从十进制转换为二进制

A.4.3 从十进制转换为十六进制

附录B C++关键字

附录C 运算符优先级

附录D 答案

附录E ASCII码

## 章节摘录

版权页： 23.4总结 本章介绍了STL中最重要、功能最强大的方面：算法。

在本章中，读者了解了各种类型的算法，并通过示例对其用法有更清晰的认识。

23.5问与答 问：诸如std::transform()等变序算法能否用于关联容器(如std::set)？

答：即使可以，也不应这样做。

应将关联容器的内容视为常量，这是因为关联容器在插入元素时进行排序，因此元素的相对位置不仅对find()等函数来说很重要，对容器的效率也很重要。

因此，不应将诸如std::transform()等变序算法用于STL set。

问：要将顺序容器的每个元素都设置为特定的值，可使用std::transform()吗？

答：虽然可以使用std::transform()，但使用fill()或fill\_n()更合适。

问：copy\_backward()是否会反转目标容器中元素的排列顺序？

答：不会。

STL算法copy\_backward()按相反的顺序复制元素，但不改变元素的排列顺序，即它从范围末尾开始复制到开头。

如果要反转集合中元素的排列顺序，应使用std::reverse()。

问：是否应对list使用std::sort()？

答：std::sort()可用于list，用法与用于其他顺序容器一样。

然而，list需要保持一个特殊特征：对list的操作不会导致现有迭代器失效，而std::sort()不能保证该特征得以保持。

因此，STL list通过成员函数list::sort()提供了sort算法。

应使用该函数，因为它确保指向list中元素的迭代器不会失效，即使元素的相对位置发生了变化。

问：为什么在将元素插入到有序范围中时，使用lower\_bound()或upper\_bound()等函数很重要？

答：这两个函数分别提供有序集合中的第一个位置和最后一个位置，将元素插入这些位置时不会破坏集合的有序状态。

23.6作业 作业包括测验和练习，前者帮助读者加深对所学知识的理解，后者提供了使用新学知识的机会。

请尽量先完成测验和练习题，然后再对照附录D的答案。

在继续学习下一章前，请务必弄懂这些答案。

23.6.1测验 1.要将list中满足特定条件的元素删除，应使用std::remove\_if()还是list::remove\_if()？

2.假设有一个包含ContactItem对象的list，在没有显式指定二元谓词时，函数list::sort()将如何对这些元素进行排序？

3.STL算法generate()将调用函数generator()多少次？

4.std::transform()与std::foreach()之间的区别何在？

23.6.2练习 1.编写一个二元谓词，它接受字符串作为输入参数，并根据不区分大小写的比较结果返回一个值。

2.演示STL算法(如copy)如何使用迭代器实现其功能——复制两个类型不同的容器存储的序列，而无需知道目标集合的特征。

#### 媒体关注与评论

对没有任何编程经验的新人和有其他语言编程经验的人来说，这是一本卓越的C++入门图书。  
——独立评论人



编辑推荐

畅销全球的C++经典入门教程全面涵盖C++11新标准帮助读者编写高效的C++应用程序中文版累计销售超50000册

#### 版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>