



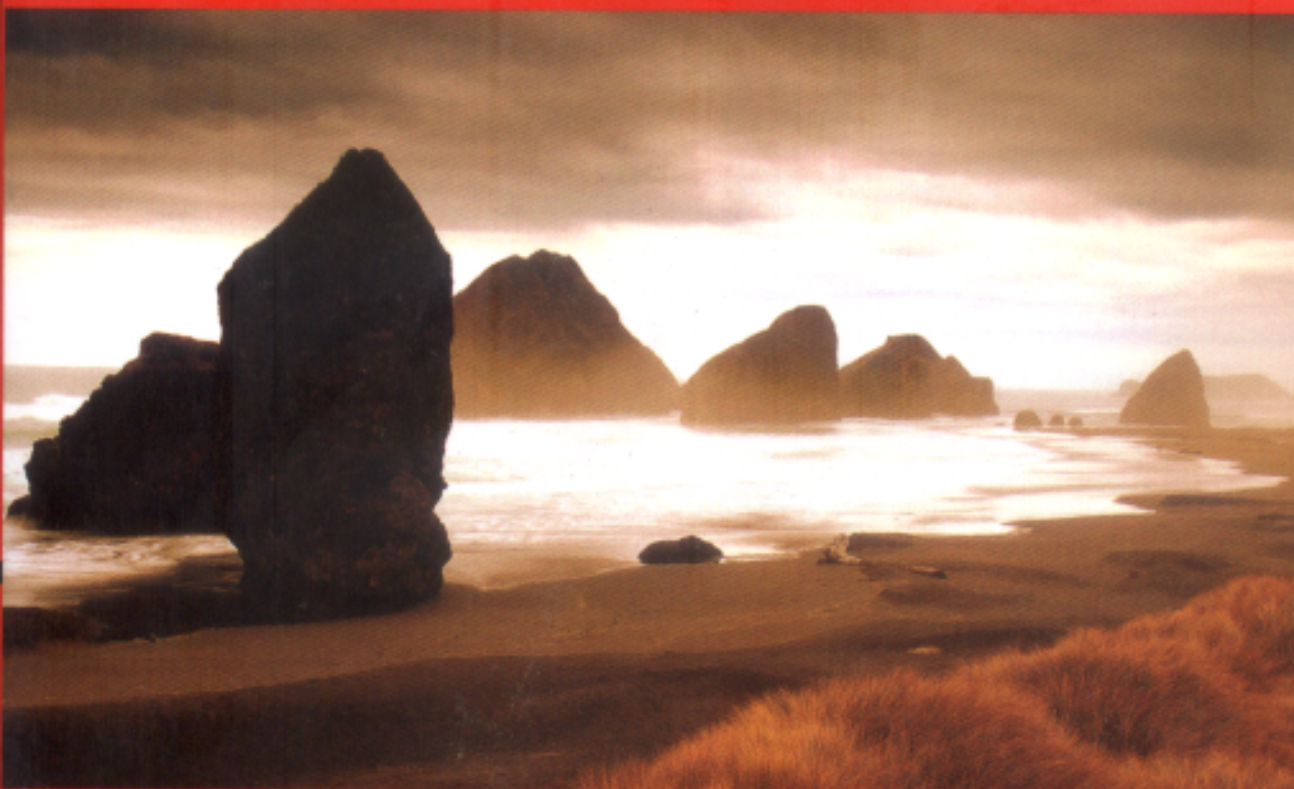
深入 C++ 系列

# C++ Primer

第三版

## 中文版

Stanley B. Lippman 著  
Josée Lajoie 译  
潘爱民 张丽



Addison-Wesley



中国电力出版社

[www.infopower.com.cn](http://www.infopower.com.cn)

# 前言

本书第二版和第三版之间的变化非常大。其中最值得注意的是，C++已经通过了国际标准化，这不但为语言增加了新的特性，比如异常处理、运行时刻类型识别（RTTI）、名字空间、内置布尔数据类型、新的强制转换方式，而且还大量修改并扩展了现有的特性，比如模板（template）、支持面向对象（object-oriented）和基于对象（object-based）程序设计所需要的类（class）机制、嵌套类型以及重载函数的解析机制。也许更重要的是，一个覆盖面非常广阔的库现在成了标准 C++ 的一部分，其中包括以前称为 STL（标准模板库）的内容。新的 string 类型、一组顺序和关联容器类型（比如 vector、list、map 和 set），以及在这些类型上进行操作的一组可扩展的泛型算法（generic algorithm），都是这个新标准库的特性。本书不但包括了许多新的资料，而且还阐述了怎样在 C++ 中进行程序设计的新的思考方法。简而言之，实际上，不但 C++ 已经被重新创造，本书第三版也是如此。

在第二版中，不但对语言的处理方式发生了根本的变化，而且作者本身也发生了变化：首先，我们的人数已经加倍。而且，我们的写作过程也已经国际化了（尽管我们还牢牢扎根于北美大陆）：Stan Lippman 是美国人，Josée Lajoie 是加拿大人。最后，这种双作者关系也反映了 C++ 团体的两类主要活动：Stan 现在正在迪斯尼动画公司（Walt Disney Feature Animation）\* 致力于以 C++ 为基础的 3D 计算机图形和动画应用，而 Josée 正专心于 C++ 的定义与实现，同时她也是 C++ 标准的核心语言小组的主席\*\*，以及 IBM 加拿大实验室的 C++ 编译器组的成员。

Stan 是 Bell 实验室中与 Bjarne Stroustrup（C++ 的发明者）一起工作的早期成员之一，从 1984 年开始一直从事 C++ 方面的工作。Stan 曾经致力于原始 C++ 编译器 cfront 的各种实现，从 1986 年的版本 1.1 到版本 3.0，并领导了 2.1 和 3.0 版本的开发组。之后，他参与了 Stroustrup 领导的、Foundation Research Project 项目中关于程序设计环境的对象模型部分。

Josée 作为 IBM 加拿大实验室 C++ 编译器组的成员已经有八年时间了。从 1990 年开始她成为 C++ 标准委员会的成员。她曾经担任委员会的副主席三年，目前担任核心语言小组委员会的主席已经达四年之久。

本书第三版是一个大幅修订的版本，不仅反映了语言的变化和扩展，也反映了作者洞察力和经验的变化。

---

\* Stan Lippman 现已受雇于 Microsoft，成为 Visual C++ .NET 的架构设计师。

\*\* Josée Lajoie 现在在滑铁卢大学攻读博士学位，已不再担任该委员会的主席。现任主席为 Sun 公司的 Steve Clamage。

## 本书的结构

本书为 C++ 国际标准进行了全面的介绍。在此意义上，它是一个初级读本（primer），它提供了一种指导性的方法来描述 C++ 语言。（但是，它也为 C++ 语言提供了一种简单而温和的描述，从这个角度来看，它不是一本初级读物。）C++ 语言的程序设计要素，比如异常处理、容器类型、面向对象的程序设计等等，都在解决特定问题或程序设计任务的上下文环境中展示出来。C++ 语言的规则，比如重载函数调用的解析过程以及在面向对象程序设计下支持的类型转换，本书都有广泛的论述，这似乎超出了一本初级读本的范畴。我们相信，为了加强读者对于 C++ 语言的理解，覆盖这些内容是必要的。对于这些材料，读者应该不时地回头翻阅，而不是一次消化了事。如果开始的时候你发现这些内容比较难以接受或者过于枯燥，请把它们放到一边，以后再回头來看——我们为这样的章节加上了特殊的记号：※。

阅读本书不需要具备 C 语言的知识，但是，熟悉某些现代的结构化语言会使学习进展更快一些。本书的意图是作为学习 C++ 的第一本书；而不是学习程序设计的第一本书！为了确保这一点，我们会以一个公共的词汇表作为开始；然而，开始的章节涵盖了一些基本的概念，比如循环语句和变量等，有些读者可能会觉得这些概念太浅显了。不必担心：深层的内容很快就会看到。

C++ 的许多威力来自于它对程序设计新方法的支持，以及对程序设计问题的思考方式。因此，要想有效地学习使用 C++，不要只想简单地学会一组新的语法和语义。为了使这种学习更加容易，本书将围绕一系列可扩展的例子来组织内容。这些例子被用来介绍各种语言特性的细节，同时也说明了这些语言特性的动机所在。当我们在一个完整例子的上下文环境中学习语言特性时，对这些特性为什么会有用处也就变得很清楚了，它会使我们对于“何时以及怎样在实际的问题解决过程中使用这些特性”有一些感觉。另外，把焦点放在例子上，可使读者能够尽早地使用一些概念，随着读者的知识基础被建立起来之后，这些概念会进一步完整地解释清楚。本书前面的例子含有 C++ 基本概念的简单用法，读者可以先领略一下 C++ 中程序设计的概貌，而不要求完全理解 C++ 程序设计和实现的细节。

第 1 章和第 2 章形成了一个独立完整的 C++ 介绍和概述。第一篇的目的是使我们快速地了解 C++ 支持的概念和语言设施，以及编写和执行一个程序所需要的基础知识。读完这部分内容之后，你应该对 C++ 语言有了一些认识，但是还谈不上真正理解 C++。这就够了：那是本书余下部分的目的。

第 1 章向我们介绍了语言的基本元素：内置数据类型、变量、表达式、语句以及函数。它将介绍一个最小的、合法的 C++ 程序，简要讨论编译程序的过程，介绍所谓的预处理器（preprocessor），以及对输入和输出的支持。它给出了多个简单但却完整的 C++ 程序，鼓励读者亲自编译并执行这些程序。第 2 章介绍了 C++ 是如何通过类机制，为基于对象和面向对象的程序设计提供支持的，同时通过数组抽象的演化过程来说明这些设计思想。另外，它简要介绍了模板、名字空间、异常处理，以及标准库为一般容器类型和泛型程序设计提供的支持。这一章的进度比较快，有些读者可能会觉得难以接受。如果是这样，我们建议你跳过这一章，以后再回过头来看它。

C++ 的基础是各种设施，它们使用户能够通过定义新的数据类型来扩展语言本身，这些

新类型可以具有与内置类型一样的灵活性和简单性。掌握这些设施的第一步是理解基本语言本身。第3章到第6章（第二篇）在这个层次上介绍了C++语言。

第3章介绍了C++语言预定义的内置和复合数据类型，以及C++标准库提供的string、complex、vector类数据类型。这些类型构成了所有程序的基石。第4章详细讨论了C++语言支持的表达式，比如算术、关系、赋值表达式。语句是C++程序中最小的独立单元，它是第5章的主题。C++标准库提供的容器类型是第6章的焦点。我们不是简单地列出所有可用的操作，而是通过一个文本查询系统的实现，来说明这些容器类型的设计和用法。

第7章到第12章（第三篇）集中在C++为基于过程化的程序设计所提供的支持上。第7章介绍C++函数机制。函数封装了一组操作，它们通常形成一项单一的任务，如print()。（名字后面的括号表明它是一个函数。）关于程序域和变量生命期的概念、以及名字空间设施的讨论是第8章的主题。第9章扩展了第7章中引入的关于函数的讨论，介绍了函数的重载。函数重载允许多个函数实例（它们提供一个公共的操作）共享一个公共的名字（但是，要求不同的实现代码）。例如，我们可以定义一组print()函数来输出不同类型的数据。第10章介绍和说明函数模板的用法。函数模板为自动生成多个函数实例（可能是无限多个）提供了一种规范描述（prescription），这些函数实例的类型不同，但实现方式保持不变。

C++支持异常处理设施。异常表示的是一个没有预料到的程序行为，比如所有可用的程序内存耗尽。出现异常情况的程序部分会抛出一个异常——即程序的其他部分都可以访问到。程序中的某个函数必须捕获这个异常并做一些必要的动作。对于异常处理的讨论跨越了两章。第11章用一个简单的例子介绍了异常处理的基本语法和用法，该例子捕获和抛出一个类类型（class type）的异常。因为在我们的程序中，实际被处理的异常通常是一个面向对象类层次结构的类对象，所以，关于怎样抛出和处理异常的讨论一直继续到第19章，也就是在介绍面向对象程序设计之后。

第12章介绍标准库提供的泛型算法集合，看一看它们怎样和第6章的容器类型以及内置数组类型互相作用。这一章以一个使用泛型算法的程序设计作为开始。第6章介绍的iterator（迭代器）在第12章将进一步讨论，因为它们为泛型算法与实际容器的绑定提供了粘合剂。这一章也介绍并解释了函数对象的概念。函数对象使我们能够为泛型算法中用到的操作符（比如等于或小于操作符）提供另一种可替换的语义。关于泛型算法在附录中有详细说明，并带有用法的示例。

第13章到第16章（第四篇）的焦点集中在基于对象的程序设计上——即创建独立的抽象数据类型的那些类设施的定义和用法。通过创建新的类型来描述问题域，C++允许程序员在写应用程序时可以不用关心各种乏味的簿记工作。应用程序的基本类型可以只被实现一次，而多次被重用，这使程序员能够将注意力集中在问题本身，而不是实现细节上。这些封装数据的设施可以极大地简化应用程序的后续维护和改进工作。

第13章集中在一般的类机制上：怎样定义一个类，信息隐藏的概念（即，把类的公有接口同私有实现分离），以及怎样定义并封装一个类的对象实例。这一章还有关于类域、嵌套类、类作为名字空间成员的讨论。

第14章详细讨论C++为类对象的初始化、析构以及赋值而提供的特殊支持。为了支持这些特殊的行为，需要使用一些特殊的成员函数，分别是构造函数、析构函数和拷贝赋值操作符。这一章我们还将看一看按成员初始化和拷贝的主题（即指一个类对象被初始化为或者

赋值为该类的另一个对象)，以及为了有效地支持按成员初始化和拷贝而提出的命名返回值（named return value）扩展。

第 15 章将介绍类特有的操作符重载，首先给出一般的概念和设计考虑，然后介绍一些特殊的操作符，如赋值、下标、调用以及类特有的 `new` 和 `delete` 操作符。这一章还介绍了类的友元（它对一个类具有特殊的访问特权）及其必要性。然后讨论用户定义的转换，包括底层的概念和用法的扩展实例。这一章还详细讨论了函数重载解析的规则，并带有代码示例说明。

类模板是第 16 章的主题。类模板是用来创建类的规范描述，其中的类包含一个或多个参数化的类型或值。例如，一个 `vector` 类可以对内含的元素类型进行参数化。一个 `buffer` 类，可以对内含的元素类型以及缓冲区的大小进行参数化。更复杂的用法，比如在分布式计算中，IPC 接口、寻址接口、同步接口等，都可以被参数化。这一章讨论了怎样定义类模板，怎样创建一个类模板特定类型的实例，怎样定义类模板的成员（成员函数、静态成员和嵌套类型），以及怎样用类模板来组织我们的程序。最后以一个扩展的类模板的例子作为结束。

面向对象的程序设计和 C++ 的支持机制是第 17、18、19 和 20 章（第五篇）的主题。第 17 章介绍了 C++ 对于面向对象程序设计主要要素的支持：继承和动态绑定。在面向对象的程序设计中，用父/子关系（也称类型/子类型关系）来定义“有共同行为的各个类”。类不用重新实现共享特性，它可以继承了父类的数据和操作。子类或者子类型只针对它与父类不同的地方进行设计。例如，我们可以定义一个父类 `Employee`，以及两个子类型：`TemporaryEmpl` 和 `Manager`。这些子类型继承了 `Employee` 的全部行为。它们只实现自己特有的行为。

继承的第二个方面，称为多态性，是指父类型具有“引用由它派生的任何子类型”的能力。例如，一个 `Employee` 可以指向自己的类型，也可以指向 `TemporaryEmpl` 或者 `Manager`。动态绑定是指“在运行时刻根据多态对象的实际类型来确定应该执行哪个操作”的解析能力。在 C++ 中，这是通过虚拟函数机制来处理的。

第 17 章介绍了面向对象程序设计的基本特性。这一章说明了如何设计和实现一个 `Query` 类层次结构，用来支持第 6 章实现的文本查询系统。

第 18 章介绍更为复杂的继承层次结构，多继承和虚拟继承机制使得这样的层次结构成为可能。这一章利用多继承和虚拟继承，把第 16 章的模板类例子扩展成一个三层的类模板层次结构。

第 19 章介绍 RTTI（运行时刻类型识别）设施。使用 RTTI 我们的程序在执行过程中可以查询一个多态类对象的类型。例如，我们可以询问一个 `Employee` 对象，它是否实际指向一个 `Manager` 类型。另外，第 19 章回顾了异常处理机制，讨论了标准库的异常类层次机构，并说明了如何定义和处理我们自己的异常类层次结构。这一章也深入讨论了在继承机制下重载函数的解析过程。

第 20 章详细说明了如何使用 C++ 的 `iostream` 输入/输出库。它通过例子说明了一般的数数据输入和输出，说明了如何定义类特有的输入输出操作符实例、如何辨别和设置条件状态、如何对数据进行格式化。`iostream` 库是一个用虚拟继承和多继承实现的类层次结构。

本书以一个附录作为结束，附录给出了每个泛型算法的简短讨论和程序例子。这些算法按字母排序，以便参考。

最后，我们要说的是，无论谁写了一本书，他所省略掉的，往往与他所讲述的内容一样

重要。C++语言的某些方面，比如构造函数的工作细节、在什么条件下编译器会创建内部临时对象、或者对于效率的一般性考虑，虽然这些方面对于编写实际的应用程序非常重要，但是不适合于一本入门级的语言书籍。在开始写作本书第三版之前，Stan Lippman 写的《Inside the C++ Object Model》（参见本前言最后所附的参考文献中的 [LIPPMAN96a]）包含了许多这方面的内容。当读者希望获得更详细的说明（特别是讨论基于对象和面向对象的程序设计）时，本书常常会引用该书中的讨论。

本书故意省略了 C++标准库中的某些部分，比如对本地化和算术运算库的支持。C++标准库非常广泛，要想介绍它的所有方面，则远远超出了本书的范围。在后面所附的参考文献中，某些书更详细地讨论了该库（见[MUSSER96] 和 [STROUSTRUP97]）。我们相信，在这本书出版之后，一定还会有更多的关于 C++标准库各个方面的书面世。

## 第三版的变化

本书第三版的变化主要是以下四个方面：

1. 涵盖了语言所增加的新特性：异常处理、运行时刻类型识别、名字空间、内置 bool 类型、新风格的类型强制转换。

2. 涵盖了新的 C++标准库，包括 complex 和 string 类型、auto\_ptr 和 pair 类型、顺序容器和关联容器类型（主要是 list、vector、map、set 容器），以及泛型算法。

3. 对原来的文字作了调整，以反映出标准 C++对原有语言特性的精炼、变化以及扩展。语言精炼的一个例子是，现在能够前向声明一个嵌套类型，这在以前是不允许的。语言变化的一个例子是，一个虚拟函数的派生类实例能够返回一个“基类实例的返回类型”的派生类。这种变化支持一个被称为 clone 或 factory 的方法[关于 clone()虚拟函数，见 17.5.7 节说明]。对原有语言特性进行扩展的一个例子是，现在可以显式地指定一个函数模板的一个或多个模板实参。（实际上，模板已经被大大地扩展了，差不多已经成为一个新特性！）

4. 加强了对 C++高级特性的处理和组织方式，尤其是对于模板、类以及面向对象程序设计。这几年 Stan 从一个相对较小的 C++提供者团体转到了一般的 C++用户团体，这种影响使他相信，越是深入地了解问题，则程序员越是能够高明地使用 C++语言。因此，在第三版中，许多情况下，我们已经把焦点转移到如何更好地说明底层特性的概念，以及怎样最好地使用它们，并在适当的时候指出应该避免的潜在陷阱。

## C++的未来

在出版这本书的时候，ISO/ANSI C++标准委员会已经完成了 C++第一个国际标准的技术工作。该标准已于 1998 年的夏天由 ISO 公布。

C++标准公布之后，支持标准 C++的 C++编译器实现也将很快会推出。随着标准的公布，C++语言的进化将会稳定下来。这种稳定性使得以标准 C++编写的复杂的程序库，可以被用来解决工业界特有的问题。因此，在 C++世界中，我们将会看到越来越多的 C++程序库。

一旦标准被公布，标准委员会仍然会继续工作（当然步伐会慢下来），以解决 C++标准的用户所提出的解释请求。这会导致对 C++标准作一些细微的澄清和修正。如果需要，国际

标准将会每五年修订一次，以考虑技术的变化和工业界的需要。

C++标准公布五年之后将会发生什么事情我们还不知道。有可能是，一个被工业界广泛使用的新库组件将被加入到 C++标准库的组件集中。但是，到现在为止，面对 C++标准委员会已经完成的工作，C++的命运就全掌握在用户的手中了。

## 致谢

特别的感谢总是给予 Bjarne Stroustrup，感谢他给予我们如此美妙的编程语言，以及这些年他对我们的关心。特别感谢 C++标准委员会成员的奉献和艰苦工作（常常是自愿的），以及他们对 C++所作的贡献。

下面这些人为本书稿的各个草稿提供了许多有益的建议：Paul Abrahams、Michael Ball、Stephen Edwards、Cay Horstmann、Brian Kernighan、Tom Lyons、Robert Murray、Ed Scheibel、Roy Turner 和 Jon Wada。尤其要感谢 Michael Ball 的建议和鼓励。特别感谢 Clovis Tondo 和 Bruce Leung 为本书所作的深刻评论。

Stan想把特别的感谢给予 Shyh-Chyuan Huang 和 Jinko Gotoh，感谢他们对 Firebird 给与的帮助和支持，当然还有 Jon Wada 和 Josée。

Josée 要感谢 Gabby Silberman、Karen Bennet 以及 IBM 高级研究中心（the Centre for Advanced Studies）的其他小组成员，感谢他们为写作这本书所提供的支持。最大的感谢要给予 Stan，感谢他带着她经历了这项伟大的冒险活动。

最后，我们两个都要感谢编辑们的辛苦工作以及巨大的耐心：Debbie Lafferty，他从一开始，就为本书操劳；Mike Hendrickson 以及 John Fuller。Big Purple 公司在排版上做了精彩的工作。6.1 节的插图是 Elena Driskill 做的。非常感谢 Elena 使我们能够再版它。

## 第二版的致谢

这本书是无数看不见的手在帮助作者的结果。最由衷地感谢 Barbara Moo。她的鼓励、建议，以及对原手稿无数遍地仔细阅读已经无法评价。特别感谢 Bjarne Stroustrup 持续不断的帮助和鼓励，以及他给予我们如此美妙的编程语言；还有 Stephen Dewhurst，他在我第一次学习 C++ 时给了许多支持；以及 Nancy Wilkinson，另一位 cfront 编写者和 Gummi Bears 的提供者。

Dag Brück、Martin Carroll、William Hopkins、Brian Kernighan、Andrew Koenig、Alexis Layton 以及 Barbara Moo 提供了特别详细和敏锐的建议。他们的评论大大完善了这本书。Andy Baily、Phil Brown、James Coplien、Elizabeth Flanagan、David Jordan、Don Kretsch、Craig Rubin、Jonathan Shopiro、Judy Ward、Nancy Wilkinson 以及 Clay Wilson 检查了书稿的各种草稿，提出了许多有益的建议。David Prosser 澄清了 ANSI C 的许多问题。Jerry Schwarz，他实现了 iostream 包，提供了附录 A 所依据的原始文档（在第三版中变成了第 20 章）。非常感谢他对附录的细致检查。感谢 3.0 版本开发小组的其他成员：Laura Eaves、George Logothetis、Judy Ward、和 Nancy Wilkinson。

以下的人对 Addison-Wesley 的手稿作了评论：James Adcock、Steven Bellovin、Jon Forrest、



Maurice Herlihy、Norman Kerth、Darrell Long、Victor Milenkovic 以及 Justin Smith。

以下的人指出了第一版的各种印刷错误：David Beckedorff、Dag Brück、John Eldridge、Jim Humelsine、Dave Jordan、Ami Kleinman、Andrew Koenig、Tim O'Konski、Clovis Tondo 和 Steve Vinoski。

我非常感谢 Brian Kernighan 和 Andrew Koenig 提供了大量可用的排版工具。

## 参考文献

下面的文献，有的影响了本书的撰写，有的则是关于 C++ 的重要推荐资料。

[BOOCH94] Booch, Grady, *Object-Oriented Analysis and Design*, Benjamin/Cummings, Redwood City, CA (1994) ISBN 0-8053-5340-2.

[GAMMA95] Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns*, Addison Wesley Longman, Inc., Reading, MA (1995) ISBN 0-201-63361-2.

[GHEZZI97] Ghezzi, Carlo, and Mehdi Jazayeri, *Programming Language Concepts*, 3rd Edition, John Wiley and Sons, New York, NY (1997) ISBN 0-471-10426-4.

[HARBISON88] Samuel P. Harbison and Guy L. Steele, Jr, *C: A Reference Manual*, 3rd Edition, Prentice-Hall, Englewood Cliffs, NJ (1988) ISBN 0-13-110933-2.

[ISO-C++97] Draft Proposed International Standard for Information Systems — Programming Language C++ - Final Draft (FDIS) 14882.

[KERNIGHAN88] Kernighan, Brian W., and Dennis M. Ritchie, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, NJ (1988) ISBN 0-13-110362-8.

[KOENIG97] Koenig, Andrew, and Barbara Moo, *Ruminations on C++*, Addison Wesley Longman, Inc., Reading, MA (1997) ISBN 0-201-42339-1.

[LIPPMAN91] Lippman, Stanley, *C++ Primer*, 2nd Edition, Addison Wesley Longman, Inc., Reading, MA (1991) ISBN 0-201-54848-8.

[LIPPMAN96a] Lippman, Stanley, *Inside the C++ Object Model*, Addison Wesley Longman, Inc., Reading, MA (1996) ISBN 0-201-83454-5.

[LIPPMAN96b] Lippman, Stanley, Editor, *C++ Gems*, a SIGS Books imprint, Cambridge University Press, Cambridge, England (1996) ISBN 0-13570581-9.

[MEYERS98] Meyers, Scott, *Effective C++*, 2nd Edition, Addison Wesley Longman, Inc., Reading, MA (1998) ISBN 0-201-92488-9.

[MEYERS96] Meyers, Scott, *More Effective C++*, Addison Wesley Longman, Inc., Reading, MA (1996) ISBN 0-201-63371-X.

[MURRAY93] Murray, Robert B., *C++ Strategies and Tactics*, Addison Wesley Longman, Inc., Reading, MA (1993) ISBN 0-201-56382-7.

[MUSSER96] Musser, David R., and Atul Saini, *STL Tutorial and Reference Guide*, Addison Wesley Longman, Inc., Reading, MA (1996) ISBN 0-201-63398-1.



# 目 录

---

译 序  
前 言

## 第一篇 C++概述

第 1 章 开始 .....	3
1.1 问题的解决 .....	3
1.2 C++程序 .....	4
1.2.1 程序流程控制 .....	8
1.3 预处理器指示符 .....	10
1.4 注释 .....	13
1.5 输入/输出初步 .....	15
1.5.1 文件输入和输出 .....	16
第 2 章 C++浏览 .....	18
2.1 内置数组数据类型 .....	18
2.2 动态内存分配和指针 .....	21
2.3 基于对象的设计 .....	23
2.4 面向对象的设计 .....	32
2.5 泛型设计 .....	40
2.6 基于异常的设计 .....	46
2.7 用其他名字来命名数组 .....	49
2.8 标准数组——向量 .....	54

## 第二篇 基本语言

第 3 章 C++数据类型 .....	61
3.1 文字常量 .....	61
3.2 变量 .....	64
3.2.1 什么是变量? .....	66
3.2.2 变量名 .....	68
3.2.3 对象的定义 .....	69
3.3 指针类型 .....	71

3.4 字符串类型 .....	76
3.4.1 C 风格字符串 .....	76
3.4.2 字符串类型 .....	78
3.5 const 限定修饰符 .....	83
3.6 引用类型 .....	86
3.7 布尔类型 .....	90
3.8 枚举类型 .....	91
3.9 数组类型 .....	93
3.9.1 多维数组 .....	96
3.9.2 数组与指针类型的关系 .....	97
3.10 vector 容器类型 .....	99
3.11 复数类型 .....	103
3.12 typedef 名字 .....	103
3.13 volatile 限定修饰符 .....	104
3.14 pair 类型 .....	105
3.15 类类型 .....	106
 第 4 章 表达式 .....	116
4.1 什么是表达式? .....	116
4.2 算术操作符 .....	118
4.3 等于、关系和逻辑操作符 .....	120
4.4 赋值操作符 .....	123
4.5 递增和递减操作符 .....	126
4.6 复数操作 .....	128
4.7 条件操作符 .....	131
4.8 sizeof 操作符 .....	132
4.9 new 和 delete 表达式 .....	134
4.10 逗号操作符 .....	135
4.11 位操作符 .....	136
4.12 bitset 操作 .....	139
4.13 优先级 .....	142
4.14 类型转换 .....	146
4.14.1 隐式类型转换 .....	147
4.14.2 算术转换 .....	148
4.14.3 显式转换 .....	149
4.14.4 旧式强制类型转换 .....	153
4.15 栈类实例 .....	154

<b>第 5 章 语句</b> .....	159
5.1 简单语句和复合语句 .....	159
5.2 声明语句 .....	160
5.3 if 语句 .....	163
5.4 switch 语句 .....	170
5.5 for 循环语句 .....	176
5.6 while 语句 .....	180
5.7 do while 语句 .....	182
5.8 break 语句 .....	183
5.9 continue 语句 .....	184
5.10 goto 语句 .....	185
5.11 链表示例 .....	186
5.11.1 给出一个通用链表类 .....	203
<b>第 6 章 抽象容器类型</b> .....	209
6.1 我们的文本查询系统 .....	209
6.2 vector 还是 list? .....	213
6.3 vector 怎样自己增长 .....	214
6.4 定义一个顺序容器 .....	217
6.5 迭代器 .....	221
6.6 顺序容器操作 .....	224
6.6.1 删除操作 .....	226
6.6.2 赋值和对换 .....	227
6.6.3 泛型算法 .....	227
6.7 存储文本行 .....	228
6.8 找到一个子串 .....	231
6.9 处理标点符号 .....	236
6.10 任意其他格式的字符串 .....	239
6.11 其他 string 操作 .....	241
6.12 生成文本位置 map .....	247
6.12.1 定义并生成 map .....	247
6.12.2 查找并获取 map 中的元素 .....	251
6.12.3 对 map 进行迭代 .....	252
6.12.4 单词转换 map .....	253
6.12.5 从 map 中删除元素 .....	255
6.13 创建单词排除集 .....	256
6.13.1 定义 set 并放入元素 .....	256
6.13.2 搜索一个元素 .....	257
6.13.3 迭代一个 set 对象 .....	257

6.14	完整的程序.....	258
6.15	multimap 和 multiset.....	267
6.16	栈.....	269
6.17	队列和优先级队列.....	271
6.18	回顾 iStack 类.....	272

## 第三篇 基于过程的程序设计

第 7 章	函数.....	277
7.1	概述.....	277
7.2	函数原型.....	279
7.2.1	函数返回类型.....	279
7.2.2	函数参数表.....	280
7.2.3	参数类型检查.....	281
7.3	参数传递.....	382
7.3.1	引用参数.....	284
7.3.2	引用和指针参数的关系.....	286
7.3.3	数组参数.....	289
7.3.4	抽象容器类型参数.....	291
7.3.5	缺省实参.....	293
7.3.6	省略号(ellipse).....	295
7.4	返回一个值.....	297
7.4.1	参数和返回值与全局对象.....	300
7.5	递归.....	301
7.6	inline 函数.....	302
7.7	链接指示符: extern “C”.....	304
7.8	main(): 处理命令行选项.....	306
7.8.1	一个处理命令行的类.....	313
7.9	指向函数的指针※.....	315
7.9.1	指向函数的指针的类型.....	316
7.9.2	初始化和赋值.....	317
7.9.3	调用.....	317
7.9.4	函数指针的数组.....	318
7.9.5	参数和返回类型.....	319
7.9.6	指向 extern “C”函数的指针.....	322
第 8 章	域和生命期.....	325
8.1	域.....	325
8.1.1	局部域.....	327

8.2 全局对象和函数 .....	330
8.2.1 声明和定义 .....	330
8.2.2 不同文件之间声明的匹配 .....	331
8.2.3 谈谈头文件 .....	333
8.3 局部对象 .....	335
8.3.1 自动对象 .....	335
8.3.2 寄存器自动对象 .....	336
8.3.3 静态局部对象 .....	337
8.4 动态分配的对象 .....	338
8.4.1 单个对象的动态分配与释放 .....	339
8.4.2 <code>auto_ptr</code> ※ .....	341
8.4.3 数组的动态分配与释放 .....	345
8.4.4 常量对象的动态分配与释放 .....	346
8.4.5 定位 <code>new</code> 表达式 .....	347
8.5 名字空间定义 ※ .....	349
8.5.1 名字空间定义 .....	351
8.5.2 域操作符 ( <code>::</code> ) .....	353
8.5.3 嵌套名字空间 .....	354
8.5.4 名字空间成员定义 .....	356
8.5.5 <code>ODR</code> 和名字空间成员 .....	358
8.5.6 未命名的名字空间 .....	359
8.6 使用名字空间成员 ※ .....	361
8.6.1 名字空间别名 .....	361
8.6.2 <code>using</code> 声明 .....	362
8.6.3 <code>using</code> 指示符 .....	363
8.6.4 标准名字空间 <code>std</code> .....	366
<b>第9章 重载函数</b> .....	369
9.1 重载函数声明 .....	369
9.1.1 为什么要重载一个函数名 .....	369
9.1.2 怎样重载一个函数名 .....	370
9.1.3 何时不重载一个函数名 .....	372
9.1.4 重载与域 ※ .....	373
9.1.5 <code>extern "C"</code> 和重载函数 ※ .....	376
9.1.6 指向重载函数的指针 ※ .....	377
9.1.7 类型安全链接 ※ .....	378
9.2 重载解析的三个步骤 .....	379
9.3 参数类型转换 ※ .....	381
9.3.1 精确匹配的细节 .....	382

9.3.2	提升的细节 .....	386
9.3.3	标准转换的细节 .....	388
9.3.4	引用 .....	391
9.4	函数重载解析细节 ※ .....	393
9.4.1	候选函数 .....	394
9.4.2	可行函数 .....	397
9.4.3	最佳可行函数 .....	399
9.4.4	缺省实参 .....	403
第 10 章	函数模板 .....	405
10.1	函数模板定义 .....	405
10.2	函数模板实例化 .....	411
10.3	模板实参推演 ※ .....	414
10.4	显式模板实参 ※ .....	417
10.5	模板编译模式 ※ .....	420
10.5.1	包含编译模式 .....	421
10.5.2	分离编译模式 .....	421
10.5.3	显式实例化声明 .....	423
10.6	模板显式特化 ※ .....	424
10.7	重载函数模板 ※ .....	428
10.8	考虑模板函数实例的重载解析 ※ .....	431
10.9	模板定义中的名字解析 ※ .....	437
10.10	名字空间和函数模板 ※ .....	442
10.11	函数模板示例 .....	446
第 11 章	异常处理 .....	449
11.1	抛出异常 .....	449
11.2	try 块 .....	452
11.3	捕获异常 .....	455
11.3.1	异常对象 .....	456
11.3.2	栈展开 .....	459
11.3.3	重新抛出 .....	459
11.3.4	catch-all 处理代码 .....	461
11.4	异常规范 .....	463
11.4.1	异常规范与函数指针 .....	465
11.5	异常与设计事项 .....	466
第 12 章	泛型算法 .....	468
12.1	概述 .....	468

12.2 使用泛型算述 .....	471
12.3 函数对象 .....	481
12.3.1 预定义函数对象 .....	482
12.3.2 算术函数对象 .....	484
12.3.3 关系函数对象 .....	484
12.3.4 逻辑函数对象 .....	485
12.3.5 函数对象的函数适配器 .....	486
12.3.6 实现函数对象 .....	486
12.4 回顾 iterator .....	488
12.4.1 插入 iterator .....	488
12.4.2 反向 iterator .....	489
12.4.3 istream iterator .....	490
12.4.4 istream_iterator .....	491
12.4.5 ostream_iterator .....	492
12.4.6 五种 iterator .....	493
12.5 泛型算法 .....	494
12.5.1 查找算法 .....	495
12.5.2 排序和通用整序算法 .....	495
12.5.3 删除和替换算法 .....	496
12.5.4 排列组合算法 .....	496
12.5.5 算术算法 .....	496
12.5.6 生成和异变算法 .....	496
12.5.7 关系算法 .....	496
12.5.8 集合算法 .....	497
12.5.9 堆算法 .....	497
12.6 何时不用泛型算法 .....	497
12.6.1 list::merge() .....	498
12.6.2 list::remove() .....	498
12.6.3 list::remove_if() .....	498
12.6.4 list::reverse() .....	499
12.6.5 list::sort() .....	499
12.6.6 list::splice() .....	499
12.6.7 list::unique() .....	500

## 第四篇 基于对象的程序设计

第 13 章 类 .....	503
13.1 类定义 .....	503
13.1.1 数据成员 .....	504



13.1.2	成员函数.....	505
13.1.3	成员访问.....	506
13.1.4	友元.....	507
13.1.5	类声明和类定义.....	508
13.2	类对象.....	509
13.3	类成员函数.....	511
13.3.1	inline 和非 inline 成员函数.....	512
13.3.2	访问类成员.....	513
13.3.3	私有与公有成员函数.....	514
13.3.4	特殊的成员函数.....	517
13.3.5	const 和 volatile 成员函数.....	517
13.3.6	mutable 数据成员.....	520
13.4	隐含的 this 指针.....	521
13.4.1	何时使用 this 指针.....	523
13.5	静态类成员.....	525
13.5.1	静态成员函数.....	529
13.6	指向类成员的指针.....	532
13.6.1	类成员的类型.....	534
13.6.2	使用指向类成员的指针.....	536
13.6.3	静态类成员的指针.....	538
13.7	联合：一个节省空间的类.....	539
13.8	位域 (bit-field)：一种节省空间的成员.....	544
13.9	类域 ※.....	545
13.9.1	类域中的名字解析.....	548
13.10	嵌套类.....	551
13.10.1	在嵌套类域中的名字解析.....	557
13.11	作为名字空间成员类 ※.....	559
13.12	局部类 ※.....	562

第 14 章	类的初始化、赋值和析构.....	565
14.1	类的初始化.....	565
14.2	类的构造函数.....	567
14.2.1	缺省构造函数.....	572
14.2.2	限制对象创建.....	573
14.2.3	拷贝构造函数.....	574
14.3	类的析构函数.....	576
14.3.1	显式的析构调用.....	579
14.3.2	可能出现的程序代码膨胀.....	579
14.4	类对象数组和 vector.....	581

14.4.1 堆数组的初始化 ※ .....	583
14.4.2 类对象的 vector .....	585
14.5 成员初始化表 .....	587
14.6 按成员初始化 ※ .....	592
14.6.1 成员类对象的初始化 .....	595
14.7 按成员赋值 ※ .....	597
14.8 效率问题 ※ .....	600
<b>第 15 章 重载操作符和用户定义的转换</b> .....	<b>605</b>
15.1 操作符重载 .....	605
15.1.1 类成员与非成员 .....	608
15.1.2 重载操作符的名字 .....	611
15.1.3 重载操作符的设计 .....	612
15.2 友元 .....	614
15.3 操作符 = .....	616
15.4 操作符 [] .....	618
15.5 操作符 operator() .....	619
15.6 操作符 -> .....	620
15.7 操作符 ++和-- .....	622
15.8 操作符 new 和 delete .....	626
15.8.1 数组操作符 new[] 和 delete[] .....	629
15.8.2 定位操作符 new() 和 delete() .....	631
15.9 用户定义的转换 .....	633
15.9.1 转换函数 .....	636
15.9.2 用构造函数作为转换函数 .....	640
15.10 选择一个转换 ※ .....	642
15.10.1 函数重载解析——回顾 .....	644
15.10.2 候选函数 .....	645
15.10.3 类域中的函数所调用的候选函数 .....	647
15.10.4 对用户定义的转换序列划分等级 .....	648
15.11 重载解析和成员函数 ※ .....	652
15.11.1 重载成员函数的声明 .....	652
15.11.2 候选函数 .....	653
15.11.3 可行函数 .....	653
15.12 重载解析和操作符 ※ .....	656
15.12.1 候选的操作符函数 .....	657
15.12.2 可行函数 .....	660
15.12.3 二义性 .....	661

第 16 章 类模板 .....	664
16.1 类模板定义 .....	664
16.1.1 Queue 和 QueueItem 类模板的定义 .....	669
16.2 类模板实例化 .....	671
16.2.1 非类型参数的模板实参 .....	675
16.3 类模板的成员函数 .....	679
16.3.1 Queue 和 QueueItem 模板成员函数 .....	680
16.4 类模板中的友元声明 .....	682
16.4.1 Queue 和 QueueItem 的友元声明 .....	684
16.5 类模板的静态数据成员 .....	687
16.6 类模板的嵌套类型 .....	689
16.7 成员模板 ※ .....	691
16.8 类模板和编译模式 ※ .....	695
16.8.1 包含编译模式 .....	696
16.8.2 分离编译模式 .....	697
16.8.3 显式实例声明 .....	699
16.9 类模板特化 ※ .....	700
16.10 类模板部分特化 ※ .....	703
16.11 类模板中的名字解析 ※ .....	705
16.12 名字空间和类模板 ※ .....	707
16.13 模板数组类 .....	709

## 第五篇 面向对象的程序设计

第 17 章 类继承和子类型 .....	719
17.1 定义一个类层次结构 .....	721
17.1.1 面向对象的设计 .....	724
17.2 确定层次的成员 .....	728
17.2.1 定义基类 .....	728
17.2.2 定义派生类 .....	732
17.2.3 小结 .....	734
17.3 基类成员访问 .....	736
17.4 基类和派生类的构造 .....	743
17.4.1 基类构造函数 .....	744
17.4.2 派生类构造函数 .....	745
17.4.3 另外一个类层次结构 .....	746
17.4.4 迟缓型错误检测 (Lazy error detection) .....	749
17.4.5 析构函数 .....	749

17.5 基类和派生类虚拟函数 .....	751
17.5.1 虚拟的输入/输出 .....	753
17.5.2 纯虚拟函数 .....	757
17.5.3 虚拟函数的静态调用 .....	759
17.5.4 虚拟函数和缺省实参 .....	760
17.5.5 虚拟析构函数 .....	762
17.5.6 eval()虚拟函数 .....	764
17.5.7 虚拟 new 操作符 .....	768
17.5.8 虚拟函数、构造函数和析构函数 .....	770
17.6 按成员初始化和赋值 ※ .....	772
17.7 UserQuery 管理类 .....	776
17.7.1 定义 UserQuery 类 .....	780
17.8 组合起来 .....	784
 第 18 章 多继承和虚拟继承 .....	 790
18.1 准备阶段 .....	790
18.2 多继承 .....	794
18.3 public、private 和 protected 继承 .....	800
18.3.1 继承与组合 (composition) .....	802
18.3.2 免除 (exempting) 个别成员的私有继承影响 .....	803
18.3.3 protected 继承 .....	804
18.3.4 对象组合 .....	804
18.4 继承下的类域 .....	806
18.4.1 多继承下的类域 .....	809
18.5 虚拟继承 ※ .....	813
18.5.1 虚拟基类声明 .....	815
18.5.2 特殊的初始化语义 .....	816
18.5.3 构造函数与析构函数顺序 .....	819
18.5.4 虚拟基类成员的可视性 .....	820
18.6 多继承及虚拟继承实例 ※ .....	823
18.6.1 带有范围检查的 Array 派生类 .....	825
18.6.2 排序的 Array 派生类 .....	827
18.6.3 多重派生的 Array 类 .....	832
 第 19 章 C++ 中继承的用法 .....	 835
19.1 RTTI .....	835
19.1.1 dynamic_cast 操作符 .....	836
19.1.2 typeid 操作符 .....	840
19.1.3 type_info 类 .....	842

19.2 异常和继承 .....	845
19.2.1 定义为类层次结构的异常 .....	845
19.2.2 抛出类类型的异常 .....	846
19.2.3 处理类类型的异常 .....	847
19.2.4 异常对象和虚拟函数 .....	849
19.2.5 栈展开和析构函数调用 .....	851
19.2.6 异常规范 .....	852
19.2.7 构造函数和函数 try 块 .....	854
19.2.8 C++标准库的异常类层次结构 .....	855
19.3 重载解析过程和继承 ※ .....	859
19.3.1 候选函数 .....	859
19.3.2 可行函数和用户定义的转换序列 .....	862
19.3.3 最佳可行函数 .....	864
 第 20 章 iostream 库 .....	868
20.1 输出操作符 << .....	872
20.2 输入 .....	876
20.2.1 字符串输入 .....	880
20.3 其他输入/输出操作符 .....	886
20.4 重载输出操作符 << .....	891
20.5 重载输入操作符 >> .....	895
20.6 文件输入和输出 .....	897
20.7 条件状态 .....	906
20.8 string 流 .....	908
20.9 格式状态 .....	911
20.10 强类型库 .....	917
 附录 泛型算法 (按字母排序) .....	919
accumulate() .....	920
adjacent_difference() .....	921
adjacent_find() .....	922
binary_search() .....	923
copy() .....	924
copy_backward() .....	925
count() .....	926
count_if() .....	927
equal() .....	929
equal_range() .....	930
fill() .....	932

fill_n() .....	933
find() .....	934
find_if() .....	935
find_end() .....	936
find_first_of() .....	937
for_each() .....	939
generate() .....	939
generate_n() .....	940
includes() .....	941
inner_product() .....	942
inplace_merge() .....	943
iter_swap() .....	944
lexicographical_compare() .....	945
lower_bound() .....	947
max() .....	948
max_element() .....	948
min() .....	948
min_element() .....	949
merge() .....	950
mismatch() .....	951
next_permutation() .....	952
nth_element() .....	953
partial_sort() .....	954
partial_sort_copy() .....	955
partial_sum() .....	956
partition() .....	957
prev_permutation() .....	958
random_shuffle() .....	959
remove() .....	960
remove_copy() .....	960
remove_if() .....	961
remove_copy_if() .....	961
replace() .....	962
replace_copy() .....	963
replace_if() .....	964
replace_copy_if() .....	964
reverse() .....	965
reverse_copy() .....	965
rotate() .....	966

rotate_copy()	966
search()	967
search_n()	968
set_difference()	969
set_intersection()	970
set_symmetric_difference()	970
set_union()	971
sort()	972
stable_partition()	973
stable_sort()	974
swap()	975
swap_range()	975
transform()	977
unique()	978
unique_copy()	978
upper_bound()	980
堆算法	981
make_heap()	981
pop_heap()	981
push_heap()	982
sort_heap()	982

英汉对照索引	984
--------	-----