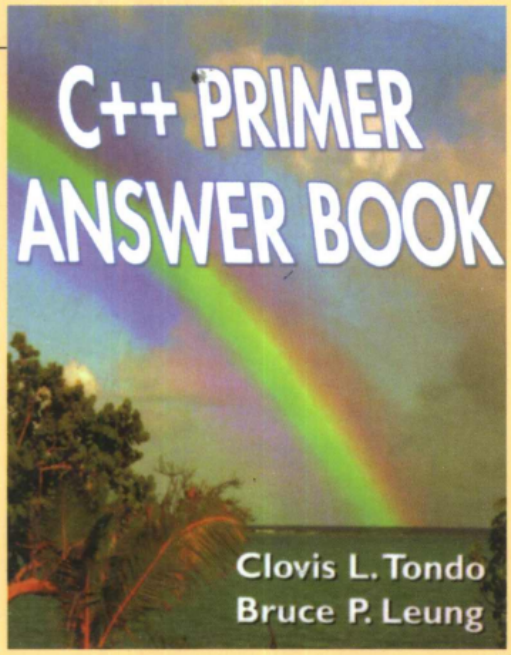
HUSTP

C++ Primer Answer Book

3rd edition

Clovis L. Tondo & Bruce P. Leung 著



题解

侯捷 译

华中科技大学出版社
http://press.hust.edu.cn

Pearson Education (培生教育出版集团)

同筑书山路　共造学海舟

C++Primer题解

很显然，学习C++ — 不只是学习其语言架构，还包括学习如何应用 — 的最佳方法，就是通过问题的解决和实际的操作来进行。也因此，Stanley Lippman和Josee Lajoie所著的畅销书《C++Primer 3/e》中出现了许多习题，用来协助C++程序员获得实际经验，并能更深刻地了解这一语言的复杂度。

但是当其中的某些习题阻挡了你的进步时，你将如何是好？现在，你可以打开这本《C++Primer题解》。这是《C++Primer》的伴随小册子，提供书中所有习题的解答，让你学习如何面对并解决程序设计上的挑战。拥有了这本习题解答，你将拥有技术上的解释、实用的技术，以及实际程序代码。它能终结你的挫败感，通过这些解答获得工作上的帮助。

本书内容完全跟得上时代，涵盖ANSI/ISO C++标准规格、扩充规格、标准程序库，以及STL。这些习题和其解答涵盖许多 C++主题，包括：

- 数据类型（data types）
- 抽象的容器类型（abstract container types）
- 泛型算法（generic algorithms）
- 类别范本（class templates）
- 多重继承和虚拟继承（multiple and virtual inheritance）
- iostream程序库

C++程序员如果以这本习题解答搭配《C++Primer》进行学习，将能因彻底掌握习题而更了解C++的概念，也会因重新审阅这些解答而有新的洞察。

作者Clovis L. Tondo，T&T Tech Works，Inc.的总裁。 这家公司提供C，C++，UNIX工具等技术训练，以其他公司为主要客户，并提供书稿给技术出版社出版。Tondo博士也是《The C Answer Book》的作者，以及其他8本书籍的协同作者，写作范围涵盖C、驱动程序、MAKE、数据结构。

作者Bruce P. Leung，Connected Components Corporation的软件工程师。他在那里参与开发一个先进的C++编译器。过去12年里他参与了许许多多研究用或工业用的编译器开发计划。他和Tondo博士合著有一本数据结构教科书。

译者 侯捷，电脑杂志专栏作家与自由撰稿人。曾任职于台湾工研院机械所、电通所，目前在元智大学授课。著有《多型与虚拟》和《深入浅出MFC》，译有《C++Primer》、《Effective C++》、《More Effective C++》、《Essential COM》等。

本书之英文支持网站：　http://www.awl.com/cseng/
本书之中文支持网站：　http://www.jjhou.com（繁体）或 http://jjhou.csdn.net（简体）

ISBN 7-5609-2704-1
9 787560 927046 >
定价：48.00元

使用者指引
入门　进阶　专家

C++ Primer 3/e

Answer Book

题 解

Clovis L. Tondo

Bruce P. Leung

侯 捷 译

华中科技大学出版社

C++ Primer 题解

C++ Primer Answer Book

Clovis L. Tondo & Bruce P. Leung

图书在版编目(CIP)数据

C++ Primer 题解/(美)Clovis L. Tondo & Bruce P. Leung 著；侯捷 译

武汉：华中科技大学出版社，2002 年 6 月

ISBN 7-5609-2704-1

Ⅰ. C…

Ⅱ. ①C… ②B… ③侯…

Ⅲ. C 语言-题解

Ⅳ. TP312

责任编辑：周 筠(http://yeka.xilubbs.com)

技术编辑：肖 翔

出版发行：华中科技大学出版社 （武昌喻家山 邮编：430074）

录 排：华中科技大学惠友科技文印中心

印 刷：湖北省新华印务有限公司

开 本：787×1092 1/16 印 张：27.5 字 数：400 000

版 次：2002 年 6 月第 1 版 印 次：2002 年 6 月第 1 次印刷

印 数：1—8 000 定 价：48.00 元

ISBN 7-5609-2704-1/TP・467

译序（侯捷）

作为一项重要的软件技术，OO（Object Oriented）已经盛行多年，日益成熟.

万丈高楼平地起，OO 的一大堆技术、思维、分析、设计… 最终要落实到编程（programming）来．OOP（Object Oriented Programming）成了踏入 OO 领域的根本.

全世界的 OO 程序设计语言中，目前以 C++ 的接受程度最广，使用的人口最多，近年来甚至被国内各大计算机科学系视为必修课程．一本好的 C++ 教科书，对于软件尖兵的技术培养有不小的贡献.

不论作为上课或自修，《*C++ Primer* 》都是一本很好的 C++ 教科书．虽然它“是否适合初学者”的正反意见始终不断，但这类讨论的症结在于每个人所谓的初学者意义都不相同，这类讨论其实没有交点，也没有意义.

我于 1999 年将《*C++ Primer*》*3e* 译成《*C++ Primer* 中文版》（繁体）．其中有许多较具突破性的翻译理念和制作理念（详见该书导读，或 `http://www.jjhou.com/cpp-primer-chap0.pdf`）．该书读者大致可分为两类，第一类把它当作工具书（查阅用），第二类把它当作学习书．前一类读者当然不需要书中的练习题，因为他们都已能熟用 C++ 语言．后一类读者可能很需要完成书中的练习题，以便验证自己的学习并提升自己的功力.

Tondo & Leung 所著的这本《*C++ Primer Answer Book*》，获得了《*C++ Primer*》作者 Lippman & Lajoie 的认同，可谓是一本“官方”解答．为了让上述第二类读者在练习时有一个凭借，我因而决定将此解答本一并中文化.

本书繁体版由黄向阳先生担任初译工作，并同挂译者之名；我负责技术和文字的全盘检阅与修润．一般所谓合译是“你译一半，我译一半，两不相干”，这是最为我所诟病的一种方式．本书由向阳完成全部初稿，再由我完成全部的技术检阅与文字修润．向阳在初始工作上帮了很大的忙，但由于未涉简体版的任何环节，所以未列名于简体版封面．我要在此感谢向阳的贡献.

我也要感谢担任繁简转译工作的肖翔先生。两岸 IT 术语分歧极大，我们需要优秀而仔细的术语转换。

一本书籍的英译中、繁转简，有很多很多专业技术的、行文遣字的、庶务性的工作在其中。我们都希望将书籍做到完美，但人世间没有完美。请上本书支持网站（载于封底）观看后续的讨论、勘误，以及程序源码下载。

侯捷 2002.03.18 于新竹

jjhou@jjhou.com
http://www.jjhou.com （繁体）
http://jjhou.csdn.net（简体）

p.s. 本书对照英文版，采页页对译方式，俾得以保留原文索引。

序

足足有一个年代（10 年）之久，*C++ Primer* 的读者通过各式各样的途径，用各式各样的方式，在研讨会大厅、电梯的一角、签名会现场，或是借着潮水般的电子邮件，追问 Stan 为什么不（以及何时要）出版习题解答。

唔…嗯… 眼睛盯着鞋带，“这是一个传统，”Stan 终于回答说：“我的意思是在贝尔实验室中，这些练习是没有解答的，至少，没有以文字形式呈现。”一阵沉默之中，他知道人们正这么想：“但是 Kernighan 和 Ritchie（*K&R*）有一本习题解答，Bjarne 也有一本。”“呃…喔…”（以及一阵胆怯的耸肩），是的，Stan 从不曾梦想过他也会有一本独立的 *Primer* 习题解答。

唔，那是过去式了。现在终于有了令大家心满意足的东西。我们要向你介绍 *C++ Primer Answer Book*!

Clovis Tondo 和 *C++ Primer*、*Inside the C++ Object Model* 两本书可以说是关系渊远。他于 1986 年初次检阅了 *Primer* 第一版的草稿，提出来的见解既有帮助又带鼓励，对于一个初尝写作的人而言，价值匪浅。接下来他又对第二版提出一份极佳的评论，并给予 *Inside the C++ Object Model* 许多有益的见解和鼓励——这对于一个写作老手而言，依然价值匪浅。最后，身为第三版的检阅者，他又提出一份非常深入的阅读心得。

当有人建议我们，Tondo 博士和 Bruce Leung 可以为第三版写一本习题解答时，我们觉得这真是绝美的搭配，因为他仿佛已为此准备了 10 年。当然，Tondo 已经为 *K&R* 教本写下了 *The C Answer Book*，本书只不过是他第二次显身手。无论如何，我们为此感到十分兴奋。

C++ Primer 书中的习题企图 (a) 增强书中资料的关键元素；(b) 提供设计工作与实际编程工作的一个具体组合，让读者得以练习他们新获得的专业知识；(c) 某些时候诱发读者思考 C++ 语言本身的设计，毕竟理论往往因实际考量而有所取舍；这类习题并不一定都有非常明确的答案。

C++ Primer 书中呈现的习题是绝对完美的设计吗？仔细阅读这本解答之后，你会发现有少数习题历经了不少修正．我们十分佩服作者的耐心．举个例子，我们居然要求 `OrQuery` 二元运算符派生自抽象的 `UnaryOperator` class（译注：练习 17.8）．真是抱歉哪！Stan 以为他写的是 `NotQuery`！那才是他原本的意思．

也因此，*C++ Primer Answer Book* 中的解答可被归为两大类．第一类用来解答习题所提出的每一个项目究竟正确或不正确．这些资料不但提供答案，还为每一节的重点提供了有用的总结；第二类解答涵盖程序设计与实现，这是我们特别喜爱的，也许是因为它们特别令人惊艳吧！毕竟我们对于第一类习题大都可以回答得很好，第二类解答则展现了两位作者的专业素养．6, 12, 17 章的程序尤其令人印象深刻．

我们相信你会满意 *C++ Primer Answer Book*，并从中获益．我们也衷心感谢两位作者所做的工作．

Stanley B. Lippman
Josée Lajoie

前言

C++ 已经成为广受欢迎的程序设计语言．渴望学习 C++ 语言的人士则对 *C++ Primer* 青睐有加．然而，学习一种程序设计语言，不只需要理解其语言架构，还必须动手，写下属于自己的程序代码，并从饶富经验的程序员所写的代码中学习．

为了这样的目的，Lippman 和 Lajoie（*L&L*）准备了许多习题，遍及 *C++ Primer* 全书，鼓励读者自行检验对书中内容的了解程度．本书则为那些练习题提供了一份解答．

C++ Primer Answer Book 企图与 *C++ Primer* 结合，成为不可分的伙伴．我们假设读者在做练习题之前，已经阅读过 *L&L*．我们提供解答，并带说明，但不重述出现于 *L&L* 的内容．只有 *L&L* 书中介绍过的观念和架构，才会被我们采用．

如果某个解答涉及整个程序，我们通常会含入完整的程序代码，使每个解答具备独立性．所有程序皆通过 Microsoft Visual C++ 5.0 的编译．少数一些编译器不符标准之处，我们会采取迂回的做法，并附上批注．

我们推荐你藉由 *L&L* 学习 C++，动手做书中习题，并仔细检阅本书所呈现的解答．我们希望 *C++ Primer Answer Book* 能够在你苦思解答而不可得的挫折过程中，帮助你了解 C++．

致谢

中译本略

Clovis L. Tondo

Bruce P. Leung

目 录

1

起步走

Getting Started

本章无任何习题

2

纵览 C++

练习 2.1

你认为为什么内建的数组不支持数组和数组之间的赋值操作(assignment)？如果要支持这个操作，需要哪些信息？

数组名，其实代表着一个常量指针，所以将一个数组赋值给另一个数组，就好像是将常量 5 赋值给常量 3 一样，虽然语法正确，但在语意层面上会产生错误。

C++ 语言并未支持数组的赋值操作。编译器必须在执行时期知道数组的长度，才能产生程序代码，支持数组对数组的赋值操作。

练习 2.2

第一级的数组应该支持什么样的操作（operations）？

虽然 C++ 对数组提供了内建支持，但是这份支持仅局限于对个别元素的读写而已。C++ 并未支持数组的抽象性（abstraction）；它也未支持我们可能希望对数组的所有操作，例如数组与数组之间的赋值操作（assignment），两个数组之间的比较操作（comparison），或是对于数组大小的询问操作（L&L, p.26；简体版 p.21）。换句话说，第一级的数组支持应该允许我们将数组视为一个单元，同时又允许我们处理个别的元素。

第一级数组应该支持的操作包括：

1. 数组初始化（initialization）
2. 数组赋值（assignment）
3. 数组间的比较（comparison）

4. 数组大小的查询
5. 索引范围的检验

练习 2.3

请解释以下 4 个对象的差异：

```
(a) int ival = 1024;
(b) int *pi = &ival;
(c) int *pi2 = new int( 1024 );
(d) int *pi3 = new int[ 1024 ];
```

```
(a) int ival = 1024;
```

对象 ival 的内存分配属性是 static 或 automatic。这个定义要求编译器分配足够的内存空间来放置任何类型为 int 的数值，并将该空间命名为 ival，然后将数值 1024 放入该空间中（L&L, p.27; 简体版 p.21）。

```
(b) int *pi = &ival;
```

对象 pi 是个指针，可以保存一个 int 的内存地址。取地址操作符（&）返回 int 对象 ival 的地址。所以，pi 是一个指向 int 的指针，它被初始化为 int 对象 ival 的地址。

```
(c) int *pi2 = new int( 1024 );
```

对象 pi2 是个指针，可以保存一个 int 的内存地址。new 操作符分配了一个类型为 int 的无具名对象，并将它初始化为 1024，然后返回其内存地址。于是 pi2 以该地址作为初值。

```
(d) int *pi3 = new int[ 1024 ];
```

对象 pi3 是个指针，可以保存一个 int 的内存地址。new 操作符分配了一个拥有 1024 个整数元素的数组（但每一个元素都未被初始化），并返回第一个元素的内存地址。于是 pi3 以该地址作为初值。

对象 ival 的类型是 int，可以保存“能够被涵盖于底层机器所支持之 int 内”的任何正值或负值。所以 (a) 保存的是一个数值，而 (b), (c), (d) 则是一个指针，保存的是地址。

练习 2.4

以下程序片段做了些什么事？它有什么重大错误？注意，面对 pia 使用 subscript（下标）操作符是可以的。之所以可以这么做，3.9.2 节有说明其理由。

```
int *pi = new int( 10 );
int *pia = new int[ 10 ];
```

```
while ( *pi < 10 ) {
        pia[ *pi ] = *pi;
        *pi = *pi + 1;
}
delete pi;
delete [] pia;
```

这一段程序代码企图将 pia 所指的一个由 ints 组成的数组加以初始化。pi 指向一个无具名的 int 对象，初值为 10。while 循环所使用的测试将 pi 所指的对象拿来和 10 比较，只要小于 10，就执行循环体。问题是第一次测试就失败了，于是循环体一次也没有执行起来。一个可能的做法是：

```
int *pi = new int( 10 );
int *pia = new int[ 10 ];
int i = 0;

while ( i < 10 ) {
      pia[ i ] = *pi;
      i++;
}
```

其余程序代码则是正确地释放了 pi 所指的无具名 int 对象，以及 pia 所指的 ints 数组。

练习 2.5

C++ class 的一个关键性质是："接口"与"实现"分离。所谓接口是一组操作行为，用户可以将它施行于此 class 所派生的任何对象身上。它由三个部分构成：操作名称、返回值、参数列表。一般而言 class 的用户只需知道这些就够了。至于私底下的实现内容，由算法（algorithms）和数据构成。观念上，虽然 class 的接口可以成长，却不可以改头换面变得与过去版本不兼容。至于私底下的实现内容，则可以自由演化。现在，请选择以下一个抽象性事物，写出其 public 接口：

```
(a) Matrix    (c) Person    (e) Pointer
(b) Boolean   (d) Date      (f) Point
```

(a) Matrix

```
class Matrix {
public:
```

```
                                    // default constructor
  Matrix( int =defaultRowSize,
          int =defaultColumnSize );
  Matrix( const Matrix & );         // copy constructor
  ~Matrix();                        // destructor
                                    // copy assignment operator
  Matrix &operator=(const Matrix &);
                                    // equality operator
  bool operator==(const Matrix &) const;
                                    // inequality operator
  bool operator!=(const Matrix &) const;
                                    // indexing operators
  int &operator()( int,int );
  int operator()( int,int ) const;
  int getRowSize() const;           // number of rows
  int getColumnSize() const;        // number of columns
                                    // iostream output operator
  friend ostream &operator<<(ostream &,const Matrix &);
                                    // iostream input operator
  friend istream &operator>>(istream &,Matrix &);

private:
  int **im;                         // implementation...
  int rows;                         // how many rows
  int cols;                         // how many columns
  static const int defaultRowSize = 5;
  static const int defaultColumnSize = 5;
};
```

针对 class Matrix，我提供了两个 constructors：

- default constructor `Matrix()`（L&L, p.698；简体版 p.572）
- copy constructor，接受另一个 `Matrix` 引用作为其参数（L&L, p.700；简体版 p.574）

Destructors 和 constructors 不一样，它不可以接受任何参数。所以我们只能为 `Matrix` 产生唯一一个 destructor。

我另外还提供了一个 copy assignment 操作符：

```
Matrix &operator=(const Matrix &);
```

以及 equality 操作符和 inequality 操作符（L&L, p.738；简体版 p.657）：

```
bool operator==(const Matrix &) const;
bool operator!=(const Matrix &) const;
```

我也提供了类似 Fortran 风格的 index 操作符：