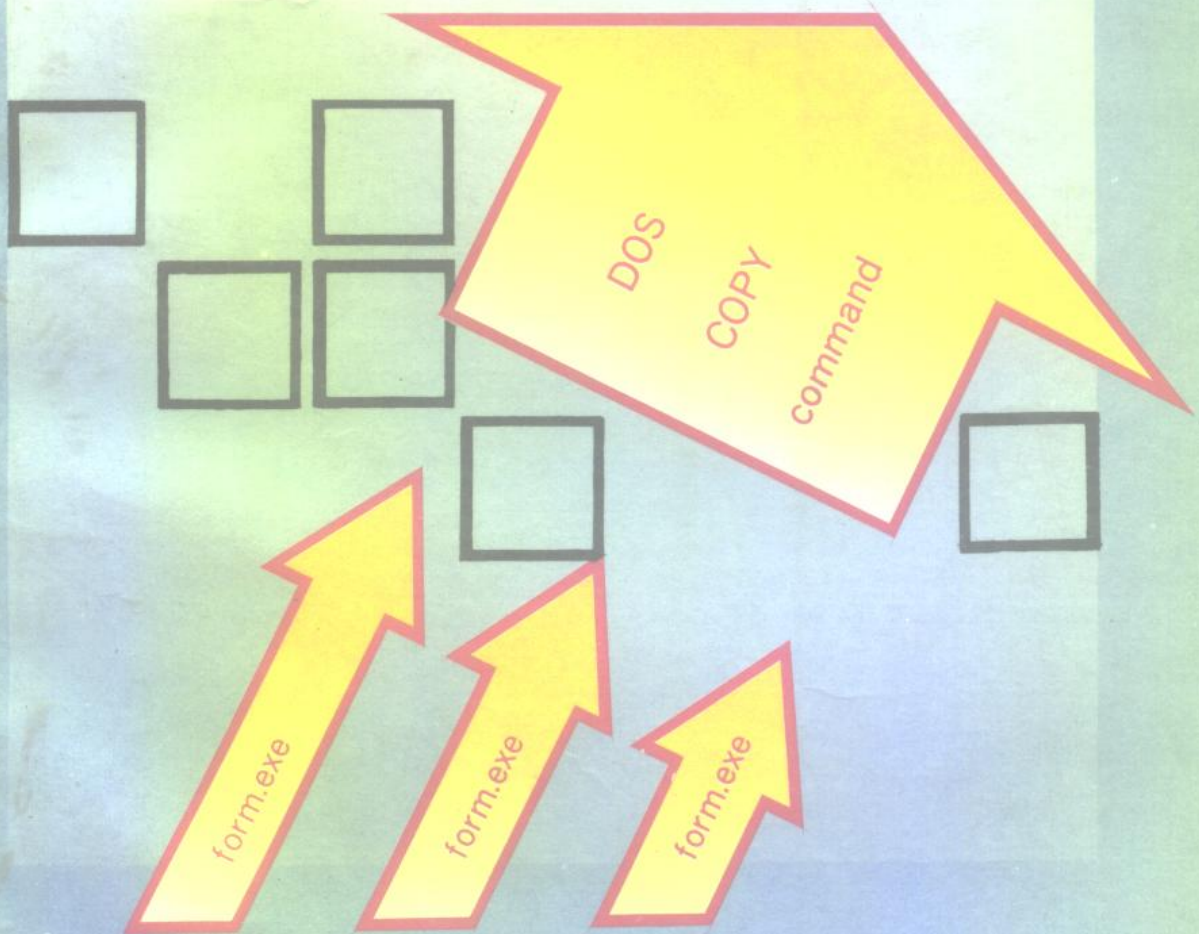
最新 Borland C++实用教程 4

Borland C++实用类、算法及类库教程

求 实 编著



科学出版社

4

最新 Borland C ++ 实用教程 4

Borland C ++实用类、算法及类库教程

求　实　编著

科学出版社

1994

（京）新登字 092 号

内 容 简 介

本书讲述具体算法、代码和优化程序的设计方法，给出了用 Borland C++设计的大量实用类程序和算法。这些实用的基础类包括：可扩充的动态字符串处理类，具有快速分类和统计功能的基础数组类，为 C++数组进行越界检查的子界与下标类及对各类错误产生输出报告的错误处理类。本书包含的各种算法覆盖了从一般字符串处理、数据处理、排序分类和统计、越界检查、文件处理到数据库设计的各个方面。

本书可供程序设计人员和高校计算机专业的师生阅读、参考。

最新 Borland C++实用教程 4

Borland C++实用类、算法及类库教程

求 实 编著

责任编辑 那莉莉

科学出版社出版
北京东黄城根北街 16 号
邮政编码：100717

北京市通县燕山印刷厂印刷
新华书店北京发行所发行 各地新华书店经售

*

1994 年 5 月第 一 版 开本：787×1092 1/16
1994 年 5 月第一次印刷 印张：31 3/4
印数：1 5 000 字数：732 000

ISBN 7-03-004041-4/TP・339

定价：28.50 元

前　言

本书是最新 Borland C++实用教程之四，系统地介绍了利用 Borland C++进行高级程序设计所需的类库、剖析工具及实用类库。

本书共分三个部分。

类库是高级程序设计中非常重要的工具，是衡量一个程序设计员对一种语言熟悉程度的重要标志，专业语言程序员考试中都有这方面的试题。因此准备参加高级程序员考试的读者不妨仔细阅读本书，并结合有关程序设计实例进行学习。类库最实际的作用在于它能大量减少程序代码，提高程序性能，扩充封装的威力。

本教程第一至第四章为第一部分，主要介绍 Borland C++的 Container 类库的两个版本，即基于 Object 的实现和基于模板的实现。

第五到第二十章为本教程的第二部分，重点介绍 Turbo Vision 具有什么功能，它怎样完成这些功能以及它为什么具有这些功能。一旦掌握了 Turbo Vision 的基本原理，读者就会觉得它是一个有益的、省时的、独具创造性的开发工具。用户能在比所想象的少得多的时间内完成复杂的、一致的、交互式的应用程序。Turbo Vision 为用户编写基于字符的高性能的、灵活的、一致的交互式应用程序用户接口提供了理想的工具。

第六章介绍 Turbo Vision 的基本概念和 Turbo Vision 应用程序的组成成分。第七章用实例介绍编写一个 Turbo Vision 应用程序的方法和步骤。

第八至第十五章主要针对 Turbo Vision 的使用技术进行了详细的讨论，其中第八章首先给读者讲解了类的层次结构，并介绍了大量的界面基本概念。第九章讨论了视图这个 Turbo Vision 应用程序的基本组成部分。第十章讨论事件驱动程序的设计方法，事件驱动是 Windows 的重要思想之一，这里讨论的事件包括鼠标事件、键盘事件、消息事件和“空”事件等。第十一章介绍设计事件驱动的交互式用户界面应用程序的安全方法。

要成为高级程序设计员，必须借助一些调试工具，分析程序的执行效率，找出影响程序效率的代码并进行修改，只有这样才能提高编程的技术，设计出高效代码。

为此，本教程第三部分专门介绍 Borland C++的高效剖析工具 Turbo Profiler。

这一部分包括第二十一至第二十八章：对 Turbo Profiler 进行了简要介绍，并给出一个简单的剖析实例；讲解 Turbo Profiler 环境中的每个菜单项和对话框选项；用类比法说明 Turbo Profiler 在用户程序运行时如何收集有关执行时间及次数的数据；还分别给出了 Turbo Profiler 的命令行选项，改变 Turbo Profiler 配置的方法，如何在扩展内存上运行 Turbo Profiler，并给出提示及错误信息。

作为 Borland C++实用教程之四的本书是为了进一步提高读者的程序设计能力而编写的，它能伴随读者进入更新的境界。

本实用教程分别针对 Borland C++入门知识、面向对象的 Windows 高级程序设计技术、面向对象的高级程序设计实例、类库及工具库等主题进行讨论，对希望进一步提高 C++程序设计技术的读者来说，本教程是一套不可多得的参考资料。

本书的前四章由谢宇、杜海燕、梁金枝、朱志勇、李雷、朱尽染、魏泱执笔；第五到第二十章由王常刚、刘秀英、何大庆、张勇、阎继忠、李波、何国辉、何晓珊、韩亮、魏宁、覃社庭、朱志坚、刘崇福、李宁、李宗昌、魏忠明、刘秀芳、车墩仁、蔡萌、欧阳琼、孟莎、张荣军执笔；第二十一至二十八章由魏忠才、程功、沈刚、石艳海执笔。祖宁对全书的结构作了合理的安排，诗颖、白琴为本书的编排付出了辛苦的劳动。在此谨对上述人员表示深深的感谢。

目　录

第一部分　Barland C++ Container 类库

第二部分 Turbo Vision 介绍

第三部分 Borland C++调试工具

第一部分　Borland C++ Container 类库

Borland C++ 3.x 版有两个 Container 类库：2.0 版本中提供的基于 Object 的库的增强版，以及新推出的基于模板的实现。我们假定读者熟悉 C++的语法和语义，以及面向对象程序设计（OOP）的基本概念。要理解基于模板的版本（称为 BIDS，即 Borland International Data Structure，“Borland 通用数据结构”的缩写），读者必须对 C++新采用的模板技术有所了解。

注意，要了解关于模板（template）的信息，请参见本教程之一的第七章。

第一部分为第一至第四章。第一章介绍如下内容：

（1）回顾 2.0 版类库与 3.x 版类库之间的区别。

（2）基于 Object 库与基于模板库的概况。

（3）Object Container 类的总览，介绍基本概念和术语。

第二章讨论 BIDS 库的概况。

第三章介绍两方面的内容：

（1）CLASSLIB 目录及其使用方法。

（2）新的调试工具。

第四章按字母顺序给出 Object Container 库参考指南，并列出每个类及其成员。

第一章　Container 类库

1.1　3.x 版对 2.0 版有什么改进？

2.0 版 Container 库是一个基于 Object 的实现。存储在其中的 Container 的对象与元素最终都是从类 Object 里派生出来的。而且，用于实现每个类的数据结构都是固定的，且（通常）对于程序员是透明的，它为大多数 Container 应用程序提供了简单有效的模型。在此基础上，3.x 版提供了先前基于 Object 的 Container 库的增强的、代码可移植的版本。我们称之为 Object Container 类库。此外，3.x 版同时提供了一个更灵活（但也更复杂）的基于模板的 Container 库，叫做 BIDS（Borland International Data Structure）。利用模板的威力，BIDS 允许用户改变 Container 的基本数据结构，并允许用户在 Container 里存储任意对象。如果提供合适的模板参数，BIDS 实际上可以模仿 Object Container 库。

在分析 Object 与 BIDS 之间的区别之前，我们先列出 2.0 版以来 Object Container 库的变化：

（1）新增加了 Btree 类和 PriorityQueue 类。

（2）新增加的 TShouldDelete 类为程序员提供了对 Container/元素所有权的控制，在对象从 Container 里分离出来，以及 Container 被去值（用新增加的 flush 方法）或者被消除时，用户可以控制这些对象。

（3）为了有效地分配（标记和释放）内存块和内存栈，新增加了内存管理类 MemBlocks 和 MemStack。

（4）为了加速应用软件的开发和调试，增加了提供复杂的 assert 技术的 PRECONDITION 和 CHECK 宏。

（5）新增加了 Timer 类，为用户计算程序运行（不在 Microsoft Windows 中）时间提供了计时器。

（6）新增加的 DLL 库及对 Windows 应用程序的支持。

现存的 Borland C++ 2.0 版 Container 类代码仍然可以在 3.x 版的库下运行。新 Object Container 类库在目录/CLASSLIB 里，由前缀 TC：TCLASSx. LIB，TCLASDBx. LIB，TCLASDLL. LIB 和 TCLASS. DLL 来区分，其中 x 代表存储模式，DB 指示特殊调试版本，为了避免累赘，我们将经常称这个 Container 实现为 Object 版或 TC 版。

注意，用户在 IDE 的 Link Libraries 对话框里选择 Container Class Library 后，基于 Object 的库将自动链接进来。

与新的基于模板的 Container 类相对应的库用前缀 BIDS 来区分：BIDSx. LIB，BIDSDBx. LIB，BIDSDLL. LIB 和 BIDS. DLL。下面我们分析提供两套 Container 库的理由。所有这些库的使用在第四章里讲述。

注意，要使用基于模板的库，用户必须显式地在工程或制作文件里加进适当的 BIDS

(DB) x. LIB 库。

1.2 为什么要有两套库?

在 Borland C++ 3.x 里,保留并增强了 Object Container 类,以提供与 2.0 版本的代码兼容性。学习它们比学习基于模板的 BIDS 库的难度小一些。Object Container 代码与模板版本相比较而言,编译要快一些,而执行则稍微慢一些,例子和演示程序的工程文件被设置为使用 Container 库的 Object 版。

BIDS 利用了 C++2.1 的令人兴奋的新模板功能。它为用户在对一个给定 Container 应用程序选择最好的基础数据结构方面提供了相当大的灵活性。在 Object 版本下,每个 Container 都用一个固定的数据结构来实现,用于满足大多数 Container 应用程序对空间/速度的要求。例如,Bag 对象用一个哈希表实现,Deque 对象用双向链表实现。使用 BIDS,用户可以通过改变 Container 实现方式,只改变最少的代码——通常只要有一个 typedef 就够了,来"微调"应用程序。用户可以简便地从 StackAsList 转换到 StackAsVector,并检验其结果。事实上,通过为通用类参数<T>设置适当的值,用户完全可以实现 Object 模型。有了 BIDS,用户甚至可以在 Object Container 模型的多态实现与非多态实现之间进行选择,这样在执行速度(非多态)和未来灵活性(多态)之间的选择可以不修改很多代码即进行测试。

Object 与 BIDS 版提供相同的函数接口。例如,对于所有 Stack 对象,push 和 pop 成员函数以同样方式起作用。这使新的基于模板的库与为 Object 库编写的代码有高度的兼容性。

注意,已存在的基于 Object Container 类的代码用新的 BIDS 类编译和运行都很正常,只要将它们与合适的库链接。

Object 库 Container 里存储的对象必须从 Object 类里派生出来,例如,要存储 int,用户必须从 Object 里派生一个 Integer 类(读者将在后面看到如何派生)有了 BIDS,用户可以完全自由地定义并直接控制 Container 中存储对象的类型。存储对象的类型只是一个作为模板参数传递的值。例如,BI_ListImp<int>给用户一个 int 链表,BI_IBtreeImp<Emp>给用户一个由指向 Emp(一个用户定义的 struct)指针组成的 B 树。

不管读者决定使用哪种 Container 类模型,都必须熟悉 Container 术语、Object 类层次结构以及每种 Container 类型提供的功能。尽管 BIDS 库中有不同的类名约定及专门的模板参数,但每个类成员的原型和功能都与 Object 库中的相同。

1.3 Container 的基本知识

我们先来描述在 Borland C++ 3.x 版里增强了的 Object Container 类层次结构。通过继承和多态性,这个层次结构提供了高度的模块性,用户可以按它们提交的形式来使用这些类,也可以扩充或者扩展它们,以产生一个符合用户需要的面向对象软件包。

注意,如果读者对 Borland C++ 2.0 版的 Container 库极其熟练,则在读到第二章之前,还必须先弄清楚 Object 库的增强性能。

在类层次结构的最上层是 Object 类(见图 1.1),这是一个抽象(abstract)类,不能够被实例化(instantiate),即不能声明这种类型的对象。抽象类可作为有关类的"伞"。它自身即