HOPE 北京希望电脑公司Borland C++3.0培训教程之一

C++3.0基础教程

马启文 冯矢勇 编

海洋出版社

C++ 3.0 基础教程

马启文　冯知勇　编

海洋出版社

1996 年·北京

内 容 提 要

C＋＋是面向对象程序设计语言(OOP)中最成熟和最宜于普及的一种。Borland C＋＋ 3.0(包括 2.0)是目前国内用户最广泛的版本。本教程的目的是帮助学员迅速理解C＋＋的精华，并掌握使用 Borland C＋＋ 3.0 软件的要点。本教程共分三册，第一册"C＋＋基础教程"的内容包括 OOP 的发展、Borland C＋＋ 3.0 简介、集成开发环境(IDE)和实用程序、C＋＋对 C 的扩充、C＋＋基本要素、高度技术初步等。

本教程可作为计算机软件应用及开发人员、研究生、大学生有关课程的教材，也是开发复杂大型计算机应用软件的工程技术人员的重要参考资料。

C＋＋ 3.0 基础教程

马启文　冯矢勇　编

*

海洋出版社出版发行(北京复兴门外大街 1 号)
常熟市教育印刷二厂印刷
开本：787×1092 毫米 1/16　印张：14　字数：335 千字
1993 年 3 月第一版　1996 年 5 月第二次印刷
印数：1—10000 册　定价：15.00 元
ISBN7-5027-3273-X/TP・166

前　言

“软件产业的革命!”,这是国内外计算机专家和软件专家对面向对象技术的看法。而C++语言是面向对象程序设计语言(OOP)中最成熟、最适宜于普及应用的工具,它是一颗正在上升的耀眼的高级语言新星。

首先,C++受到C语言程序员和软件开发人员的欢迎。今后将要受到各个领域(如管理、经济、工程技术、科学研究等)中越来越多专家的接受和喜爱。

C语言程序员之所以欢迎C++,是因为只要学习几天就可利用C++“小规模”的优点,在软件开发中收到立竿见影的效果。

软件开发人员之所以欢迎C++,是因为他们发现C++就是梦寐已求的开发大型复杂软件的工具。

C++程序设计的方法和传统的方法有根本不同:

OOP软件设计的关键在于上层设计,而不是它的实现和细节。所以这种特点会使越来越多的非软件领域的专家所欢迎和接受。从某种角度上说,他们对软件的上层设计能提出比程序员更合理的方案。

可以预见,C++不但将普及到程序员,并且将普及到所有对计算机技术有兴趣的人员。这样,C++可成为软件人员和其他领域工作者共同相通的语言,这个意义是很巨大的。

在国内,近年来也开始逐渐形成“C++热”,主要反映在应用程序的编程和图书出版物的迅速增加,单从图书和资料的出版来讲,品种数量已达半百之多,按其内容区分,大致可分为如下三类:

- C++软件包的手册译文;
- C++的应用技术;
- C++语言的教材。

我们编写这套培训教材的目的在于吸取前述各类图书的优点和长处,结合国内读者的口味,并由浅入深地使学员迅速进入C++,理解C++的精华,掌握Borland C++ 2.0及3.0软件在应用方面的关键,从而树立良好的程序设计风格。本教程是一座桥梁,它引导读者快速地进入C++殿堂。

本教程共分三册出版,第一册是C++的入门基础教程,第二册是C++应用方面的中级教程,第三册是C++专题应用技术的高级教程。

本教程第一册的一、三、四、六、七章由冯矢勇高工编写,二、五、八章等由马启文高工编写。该书在编写过程中,由卢琳琪及陈魏峰整理和校对了部分文稿和程序,在此向他们表示衷心的感谢。

编者

1993年2月28日于苏州

目　录

前言
第一章　高级语言的发展与OOP设计导论 …… (1)
1.1　C的成熟 …… (2)
1.2　新的期望 …… (2)
1.3　库的失败 …… (2)
1.4　OOP的兴起 …… (3)
1.5　C++应运而生 …… (3)
1.6　OOP的基本思想 …… (4)
第二章　Borland C++ 3.0简介 …… (5)
2.1　Borland C++ 3.0概述 …… (5)
2.2　Borland C++ 3.0的主要特点 …… (5)
2.3　Borland C++ 3.0的运行环境 …… (7)
2.4　Borland C++ 3.0软件包和资料 …… (7)
2.5　Borland C++ 3.0的安装 …… (8)
2.5.1　使用INSTALL安装Borland C++ …… (8)
2.5.2　README文件 …… (10)
2.5.3　HELPME!.DOC文件 …… (11)
2.5.4　示例程序 …… (11)
2.5.5　设置IDE …… (11)
第三章　C++对C的扩充(非类部分的特性) …… (12)
3.1　注释 …… (12)
3.2　标帜符长度 …… (13)
3.3　函数中void参数 …… (13)
3.4　函数原型 …… (14)
3.5　枚举名、结构名和类名 …… (14)
3.6　不允许多次定义全局变量 …… (14)
3.7　输入输出操作 …… (14)
3.8　寄存器变量 …… (16)
3.9　强类型机制 …… (16)
3.10　引用 …… (17)
3.11　块内的说明 …… (20)
3.12　内联函数 …… (20)
3.13　new和delete …… (21)
3.14　作用域访问运算符 …… (22)
3.15　# pragmas …… (22)
3.16　const的扩充作用 …… (23)

3.17 无名联合 …… (23)
3.18 函数重载 …… (24)
3.19 字符型常量不自动扩展成整数型 …… (25)
3.20 使用默认值的函数参数 …… (25)
3.21 参数个数不定的函数 …… (26)
第四章 C++入门 …… (28)
4.1 建立类的例子 …… (28)
4.2 成员函数 …… (29)
4.3 调用成员函数 …… (30)
4.4 成员访问控制 …… (30)
4.5 操作和数据的封装 …… (32)
4.6 构造函数和析构函数 …… (32)
4.7 继承 …… (35)
4.8 小结 …… (44)
第五章 集成开发环境(IDE) …… (45)
5.1 IDE 的启动和退出 …… (45)
5.1.1 保护模式下运行 C++的准备工作 …… (45)
5.1.2 Windows 和保护模式 …… (46)
5.1.3 IDE 启动命令及选参项 …… (46)
5.1.4 退出 Borland C++ IDE …… (47)
5.2 IDE 的组成 …… (48)
5.2.1 菜单结构 …… (48)
5.2.2 常用热键及功能 …… (49)
5.2.3 Borland C++窗口系统 …… (51)
5.2.4 状态行 …… (54)
5.2.5 会话框 …… (55)
5.2.6 编辑功能 …… (57)
5.3 Turbo C++ 3.0 for Windows 的 IDE …… (57)
5.3.1 Turbo C++ 3.0 for Windows IDE 的扩充特性 …… (58)
5.3.2 安装 Turbo C++ for Windows …… (58)
5.3.3 从 IDE 中启动 Turbo C++ for Windows …… (58)
5.3.4 使用加速棒(Speed Bar) …… (61)
5.3.5 配置文件和项目文件简介 …… (61)
5.4 对象浏览器简介 …… (63)
5.4.1 浏览类 …… (64)
5.4.2 检查函数 …… (65)
5.4.3 检查变量 …… (65)
5.4.4 检查源代码的符号 …… (65)
5.5 IDE 菜单及选项 …… (66)
5.5.1 ≡(系统)菜单 …… (66)

5.5.2 File 菜单 …… (66)
5.5.3 Edit 菜单 …… (69)
5.5.4 Search 菜单 …… (71)
5.5.5 Run 菜单 …… (73)
5.5.6 Compile 菜单 …… (75)
5.5.7 Debug 菜单 …… (76)
5.5.8 Project 菜单 …… (83)
5.5.9 Browse 菜单 …… (84)
5.5.10 Options 菜单 …… (86)
5.5.11 Window 菜单 …… (107)
5.5.12 Help 菜单 …… (109)
5.6 多文件项目的管理 …… (110)
5.6.1 使用项目管理器的例子 …… (111)
5.6.2 跟踪错误 …… (113)
5.6.3 使用不同的文件译码器 …… (115)
5.6.4 取代标准库 …… (116)
5.6.5 其他功能 …… (116)
第六章 C++基本要素 …… (119)
6.1 类的组织 …… (119)
6.2 类指针 …… (120)
6.3 关键字 this …… (121)
6.4 内联函数 …… (122)
6.5 友元函数 …… (124)
6.6 对象数组 …… (128)
6.7 类的静态成员 …… (131)
6.8 结构和联合 …… (132)
6.9 枚举 …… (134)
6.10 类的继承 …… (135)
6.11 函数的缺省实参数 …… (142)
6.12 传递对象给函数 …… (143)
6.13 运算符重载 …… (144)
6.14 多态性 …… (147)
第七章 调试手段和方法 …… (156)
7.1 跟踪程序的流程和跟踪高层流程 …… (156)
7.2 跟踪被调函数 …… (158)
7.3 设置断点 …… (159)
7.4 查看数据 …… (159)
7.5 求值和修改变量 …… (159)
7.6 设置监视 …… (160)
7.7 查找函数定义和调用关系 …… (160)

7.8 预防措施 …… (161)
7.9 调试技巧 …… (161)
7.10 常见错误 …… (162)
7.11 调试实例 …… (169)
第八章 Borland C++ 3.0 实用程序 …… (176)
8.1 IMPDEF …… (176)
8.1.1 使用 C++类的 DLL …… (177)
8.1.2 DLL 中的函数 …… (177)
8.2 IMPLIB …… (178)
8.3 IMPLIBW …… (179)
8.3.1 选择一个 IMPORT(引入)库 …… (179)
8.3.2 建立 IMPORT 库 …… (180)
8.4 MAKE …… (180)
8.4.1 MAKE 是怎样工作的 …… (180)
8.4.2 启动 MAKE …… (180)
8.4.3 MAKE 的简单运用 …… (183)
8.4.4 创建 makefile 文件 …… (184)
8.4.5 makefile 文件的组成 …… (184)
8.5 TLIB …… (185)
8.5.1 为什么使用目标模块库 …… (185)
8.5.2 TLIB 命令行 …… (185)
8.5.3 使用应答文件 …… (187)
8.5.4 建立一个扩充字典:/E 选项 …… (188)
8.5.5 设置页的大小:/P 选项 …… (188)
8.5.6 高级操作:/C 选项 …… (188)
8.5.7 例子 …… (189)
8.6 TLINK …… (189)
8.6.1 调用 TLINK …… (189)
8.6.2 TLINK 选项 …… (196)
8.6.3 模块定义文件 …… (201)
8.6.4 模块定义方法 …… (203)
8.7 WinSight …… (207)
8.7.1 启动 …… (207)
8.7.2 选择一个视口 …… (208)
8.7.3 使用窗口树 …… (208)
8.7.4 类的操作 …… (209)
8.7.5 取消时间 …… (210)
8.7.6 选择跟踪的信息 …… (210)
8.7.7 WinSight 窗口 …… (215)

第一章　高级语言的发展与OOP设计导论

本章阐明C++怎样会成为软件发展中的“里程碑”，及其简介。下面分六节叙述：

1. C的成熟
2. 新的期望
3. 库的失败
4. OOP的兴起
5. C++应运而生
6. OOP的基本思想

本教程的各种例子程序都用一致的风格，并作下列约定，以便阅读：

(1)变量名采用匈牙利命名法；

(2)结构名(struct、class、union)以大写开始，大小写间写：如classDemo，Box，unionExample；

(3)函数名以小写开始，后面必须有一个大写字母，以区别库函数(一般都是全部小写)和工具库中的函数(以大写开始)。这样做的目的，一方面便于识别，另一方面可避免函数名的冲突。例如：

fnExample，boxSet，setColor，Setx

(4)名字取用，尽可能便于阅读、记忆及理解。

匈牙利命名法(Hugarian Naming)是一种建立函数名和变量名的约定，它被Windows程序设计者广泛使用，它使代码变得易读和易维护。它采用这样的方案：

短前缀＋较长的具有描述性的名字

其中第一部分前缀用小写，而第二部分以大写开始。举例说明一些Windows下采用的短前缀约定：

a(array)数组
ch (character)字符
dw (double word)无符号长整型
cb (counter byte)字节计数
h (handle)16位句柄
hdc 设备连接器的句柄
i (index)索引(复合类型)
l (long int)长整型
lp (long pointer)长指针
n 整型
np (near pointer)近指针
pt (pointer)点

sz 以 '0' 为结尾的字串。

w (word)无符号整型

1.1 C 的成熟

目前,各种计算机语言仍在同时使用,但是,作为职业程序员,多数选中了 C 语言,并且 C 语言的应用大军在我国正日益庞大。C 语言的模块性,易读性,可移植性及可扩充性等特性是其优点,但与其他语言相比也并不是占有绝对优势;当前用户非常需要 C 语言来完成的(而其他高级语言几乎很难完成的)是可以访问和控制硬件特征和处理各种低级细节。C 语言既是高级语言,可以用结构化程序设计方法,又可以像汇编语言一样调用各种硬件资源。

ANSI C 标准的确定;用 C 语言开发很多世界上大型软件的成功经历;各种 C 语言编译器的优美版本的出现,丰富的资料和软件工具库的涌现,都标志着 C 语言已经处于成熟阶段。

C 语言既然具有双重性(结构化的控制语句,强化的类型检查,汇编语言式的位,字节,寻址等低级操作,也就有较少的"安全性"。其他的高级语言好象在规道上开车,编译通过时,是安全的。而 C 是在没有轨道的路上开车,开发者必须自己知道在做什么,编译系统不会为"安全"负责。因此,C 语言最适用于职业化的高级程序员。其他高级语言因适合不同人员的使用而共存。

1.2 新的期望

进入九十年代,计算机的应用呈几何级数上升的发展趋势。对计算机的硬件和软件的期望之高是空前的。

这些期望可以从三个角度(难度,大型,大量)反映出来:

难度急速上升:在软件开发前和开发过程中,系统本身及需求或者不甚明确或者一直在变化;随着计算机应用不断深入到不同的专业,一个成功的软件必定会涉及到很多领域的专业知识;往往在开发初、中期,用户和开发者对系统的认识和对系统的要求是模糊的,或者是不甚明了的。

软件大型化:开发者和用户都趋向于把系统的功能设置得完善,用户的界面要漂亮友好,软件的规模也越来越大,因而在开发软件系统时除了计算机硬件,软件人员外,还有不同专业、不同层次的人员参与(管理、技术、直接用户等),开发人员间的协调与管理也日益困难。

软件需求大量化:硬件系统的价格不断降低和计算机在很多领域中的成功应用,无疑刺激了计算机技术渗透到几乎"所有的"经济和生活领域中去。软件的"生产"已经面临着不是手工作坊式"单件"生产,而要逐步转向流水线式的批量生产。

以下三点都对软件的生产提出了很高而全新的期望。

1.3 库的失败

C 语言本身非常简洁,它把大量的功能性任务都放到函数库中,一方面可以允许用户自己重新定义,另一方面可以由用户再扩充库。本来期望这样的思想,能够完成要求越来越高的软件开发任务;但实际上,当程序的规模到了十万行时,用 C 语言开发,其困难程度几乎已到极

点。这种库失败的主要原因是函数(过程)和数据是分开的,相互独立的。在函数设计阶段,必须时时留心数据类型,不同的类型相同的处理,也要设计不同的函数。在函数使用阶段,有可能使用正确的数据而调用错误的函数。库的失败使C语言作为一种"万能的"开发工具的地位动摇了。人们迫切希望走出困境。

1.4 OOP的兴起

为了解决软件技术落后于硬件及六七十年代间发生的"软件危机",60年代中期出现了第一种面向对象的程序设计语言,这是有历史意义的一步。

我们处理现实问题,往往用某种计算机语言对计算机实施一些操作,以此去映射问题的解。计算机实体可称为解空间对象。面向对象的程序设计语言(OOPL)中的对象,是让程序员自己去定义解空间对象,首先描述对象及其行为,再用传统的方法去实现。

从存贮的机制看,对象是一个封装数据和操作的实体,内有数据及操作(C++中叫函数—相当于C中的函数)。其他对象不能操作这里的私有数据。

七八十年代相继从其他语言开发出OOPL来:

LIST--Flavors--Common Lisp Object System

Logo--Object Logo

Algol--Smalltalk 80--Object C

Pascal--Object Pascal

C--C++

C--Eiffel

面向对象的技术,不单单是应用在程序设计语言方面,几乎在计算机技术各个方面都在蓬勃发展。目前在面向对象的数据库技术,面向对象的系统分析与设计方法,面向对象的操作系统,面向对象的软件开发环境及硬件支持等方面都有很大进展。面向对象很有希望成为程序设计语言,数据库和人工智能等的汇合点。"面向对象"是当今一句十分时髦的口号。

面向对象程序设计包括对象的建立,所建立的对象将数据及对数据进行操作的代码结合在一起并封装。对象中有公共的和私有的成员,对其中的私有成员只允许本对象的成员函数可以访问,而公共成员则可由程序的任何部分访问。

使用OOPL的优点:对象是由各成员组成的一个简单逻辑实体,比传统的结构容易理解和控制;程序员可以从最高层到最底层建立对象的层次,每个对象继承上一层对象的特性。这样,OOPL最根本目的是使程序员能更好理解、更易管理大型复杂程序。

1.5 C++应运而生

C语言在80年代到达它的最辉煌的顶点,无论在大型的著名软件或者是各种企事业内部的事务处理,甚至在作为一些单片机汇编语言的工具等方面,C语言都成功地被人们使用着。同时随着对软件要求的提高C语言的局限性也越来越暴露出来。

面向对象概念的成熟,导致出好几种OOP语言。其中C++独树一帜,倍受欢迎。除了C++能满足主要OOP要求外,作者以为还有两点理由:第一是,C程序员是非常广大的一批用户,由于C++没有摒弃C的特征而是继承和扩充C,所以C++与其他OOP语言相比,更容

易为他们所接受。第二是，BORLAND 和 MICROSOFT 公司这两大系统软件霸主 在 C++商品软件上的竞争，导致 C++的商品软件性能优异。

当然，近年 BORLAND 公司通过代理正式在我国直接合法销售也起了良好的推动作用。

1.6 OOP 的基本思想

OOP 方法有三大特性：

(1)封装性(Encapsulation)

把数据结构和处理数据的方法(C++中称为函数)组合在一个完整的类中称为封装。类是一种新的结构机制。

(2)继承性(Inheritance)

可以从一个类(不一定是基类)建立一个派生类，则产生派生类的上级类称为此派生类的基类。派生类可以继承其基类的数据和行为；派生类也可以依次创立新的派生类，处于下层的派生类可以继承其"祖先"的数据和行为。

(3)多态性(Polymorphism)

在类的层次结构中共享一个函数的名字，而不同层次结构的同名函数各按自己的方式实现。

使用 OOP 的目的是使人们按照自然的思维习惯建立问题认知模型，更加自然地模拟客观世界。OOP 方法希望能使程序员从数据格式和函数的束缚中解脱出来，以集中精神于处理对象。又因为 OOP 中对象的内部实现细节与外界无关(数据抽象和信息隐藏)，对象类( )提供了最理想的模块化机制和可重用性方法。

用 OOP 来表示知识具有很强的表达能力，既能说明广泛领域内的知识，也能阐述复杂的问题；因此，OOP 方法也很适合于知识处理，专家系统等人工智能领域的应用。

第二章　Borland C++ 3.0 简介

2.1　Borland C++ 3.0 概述

C++语言是 AT&T 公司 BELL(贝尔)实验室的 B. Stroustrup 在 C 语言的基础上,扩充引进了面向对象的概念而发展成功的,并于 1983 年正式向外界公开推出。可以说,C++ 是 C 语言发展的一个新的里程碑,它既具有面向对象的语言的全部特点,又与 C 语言完全兼容,而且当时使用 C++ 语言编程时可以利用 UNIX 环境中的各种工具,预期这种语言将会得到越益广泛的应用,尤其在软件工程领域和人工智能领域方面。

美国 BORLAND 公司自 1990 年 5 月推出 Turbo C++ 1.0 之后,陆续又推出 Turbo C++ 2.0、Borland C++ 2.0、Turbo C++ 3.0 和 Borland C++ 3.0,到目前为止已经发展到最新的第三代 Borland C++ 3.1 版。

Borland C++ 3.0 可满足 C++ 和 C 程序员对于专业化和优化的编译器的要求,它同时又可服务于 AT&T 公司的 C++ 2.1 版和 ANSI C 的程序员。C++ 3.0 快速而有效,从而让用户可方便地创建包括 Microsoft Windows 应用程序在内的任何应用程序。

C++ 3.0 是一个面向对象(OOP)的程序设计语言,又是 C 语言的进一步发展,因此也是可移植的,所以可以很容易地将一个用 C++ 编写的应用程序从一个系统移植到另一个系统上,不管在任何地方,你都可以用 C++ 语言从事任何程序开发工作。

2.2　Borland C++ 3.0 的主要特点

Borland C++ 3.0 包含有许多用户所要求的最新特点。

(1)C 和 C++:Borland C ++ 3.0 提供 C 和 C++ 程序设计的全部功能,它完全实现 AT&T C++ V2.1 版规范指标,并 100%提供美国国家标准协会(ANSI)的最新标准 C 语言编译器,完全支持 Kernighan 和 Ritchie 定义。C++ 编译器和 C 编译器均包含了集成环境(IDE)版本和命令行版的版本。Borland C++ 3.0 同时还提供一个 C++ 的类库,首次完全实现了商用模板(Complate),在其中可使用参数化类型来创建高效的集合类。

(2)全局优化:C++ 3.0 的一整套优化选项使你能完全控制代码植成,所以用户可以使用最方便的方式来编程,从而可产生短小、高速和高效的程序代码。

(3)快速编译:Borland C++ 3.0 可使 C++ 的编译时间缩短一半,Borland 所独有的预编译头文件,可以大大缩短程序的编译时间,优化也很快,不必等待多久,即可获得高品质的代码。

(4)DPMI 编译器:当编译受内存限制的大程序,C++ 3.0 使用工业标准 DPMI 保护模式,它允许编译器(IDE、链接程序和其他程序也一样)在 DOS 或 Windows 386 增强模式的保护模式下运行。

(5)Microsoft Windows 编程:Borland C++ 3.0 可用来编制 Windows 应用程序,包括动态连接库(DDLs)和可执行文件(EXEs)。它增加了许多新的功能,包括资源编译器(Resource Compiler)、帮助编译器(Help Compiler)、资源工作库(Resource Workshop),另外还包括一些 C 和 C++ Windows 应用程序的实例,以便帮助用户更好地工作。

(6)EasyWin:这是一种自动 Windows 转换特性,它可将你的标准 DOS 应用程序(使用 printf、canf 和其他标准 I/O 函数)转换为 Windows 应用程序,而不改变代码行,仅仅设置编译器开关(或在 IDE 中选择"Windows Application"),你的 DOS 程序就可以在 Windows 中运行。

(7)程序员平台:Borland C++ 3.0 中出现了新的程序员平台,即 Borland 开放结构集成环境(IDE),它提供全范围内的编程工具和实用程序的访问,其中包括如下:

- 多重文件编辑,功能如同一个工业标准 Common User Acces (CUA)接口和一个 Alternate 接口,Alternate 接口与 Borland C++ 的以前版本兼容。
- 高级 Turbo 编辑器宏语言(Turbo Editor Macro Language,TEML)和编译器。
- 支持使用鼠标器的多重复盖窗口。
- 集成资源连接,使得在单个环境中开发 Windows 应用程序更容易。
- 运行在 DPMI(DOS 保护模式接口)上的全集成 debugger,用于调试大型应用程序。
- 内部汇编器支持内部汇编代码。
- 使用对大缓冲区的废除(undo)和重做(redo)功能等等。

(8)宿主 Windows 的 IDE:Turbo C++ for Windows 的 IDE 让用户在 Windows 环境下进行编辑、编译和运行程序。在建立 Windows 程序时,用户就没必要在 Windows 和 DOS 间进行来回切换,这样能大大提高在 Windows 环境下的编程效率,Turbo C++ for Windows 的 IDE 还包括如下功能:

- 内部 ObjectBrowser,它能让用户从视觉上检查类的层次、函数和变量,定位继承函数和数据成员,并能让你立即浏览所选择的元素的源码。
- 可见 SpeedBar,可对常用的菜单选项实行点按(point_and_click)访问。

(9)WinSight:Windows 信息跟踪实用程序,它使用户能看清自己的程序与 Windows 的相互作用。

(10)VROOMM:Borland C++ 面向对象的实时虚拟存贮管理功能,可使用户简单地复盖代码,选择复盖的代码段,VROOMM 会处理剩下的事情,使用户代码存入 640K 的空间中。

(11)Help 联机内容敏感的帮助:用复制和传输函数的示例来试验每一个函数。

(12)Streams:Borland C++ 3.0 全力支持 C++ 的 I/O 流,对流库的 Borland 扩充能使用户定位文本、设置屏幕属性和其他的在 Windows 环境下可完成的其它操作。

(13)Container classes:高级的 Container 类库提供的 sets、bags、lists、arrays、B_trees 和其他可重用数据结构,以最大的灵活性实现了模板和以对象为基础的 Container。

(14)Windows API(应用程序接口):在联机求助中增加有 Windows API 的文档。

(15)其它方面的特点:

- 超过 200 个新的库函数,这些库函数是灵活而兼容的。
- 包括复数的 BCD 数学运算,快速的复杂算术运算。
- 远程对象和大容量数组、内存管理和堆检测函数。
- 在 Windows 应用程序的 DLL 中的运行库。

- 新 BGI 字体和支持完整 ASCII 字符集的 BGI。
- 共享项目、配置、桌面文件，使得程序员不管是使用程序员平台，还是使用运行于 Windows 的 IDE，其工作环境都相同。
- 命令行编译器响应文件。
- 与 NMAKE 兼容，从而更容易从 Microsoft C 进行转换。

2.3 Borland C++ 3.0 的运行环境

Borland C++ 3.0 运行于 IBM 系列计算机和兼容机上，其中包括 AT（286 系列）、386、486 微机，PS/2 机等。

Borland C++ 3.0 需要 DOS 3.31 以上的版本，机器应配置一个硬盘驱动器，一个软盘驱动器，且至少需要 640K 内存，另外再加上 1MB 扩充内存，需要 80 列的单色或彩色监示器。Turbo C++ for Windows 的 IDE 需要保护模式 Windows 3.0 或更高版本，至少需要 2MB 的扩充内存以及 Windows 兼容的监示器。

Borland C++ 3.0 包含有浮点运算子程序，从而使你的程序能利用 80X87 协处理器。若没有协处理器，它可进行仿真运算，但运行的效率却大为降低。

Borland C++ 支持鼠标器，虽然它不是必需的，但 Resource Workshop（资源工作库）是需要鼠标器的。若配置有鼠标器，你就必须拥有下列实用程序之一，才能做到全兼容。

- Microsoft Mouse 6.1 版本或更高版本，或者任何与此兼容的 mouse 版本。
- Logitech Mouse 3.4 版或更高版本。
- Mouse Systems 的 PC Mouse 6.22 版或更高版本。
- IMSI Mouse 6.11 版或更高版本。

2.4 Borland C++ 3.0 软件包和资料

Borland C++ 3.0 软件包包括磁盘文件和 9 本手册。

Borland C++ 软件包全部实现了 AT&T C++ 2.1 版的功能，并实现了 ANSI 的 C 编译标准。另外，Borland C++ 3.0 包含有一定的扩展成分让用户能使用混合语言和混合模型编程来扩充 PC 机的能力。〈程序员指南〉第一章到第四章完整介绍了 Borland C++ 的语言标准。

Borland C++ 3.0 包含如下手册：

- Borland C++ 用户手册；
- Borland C++ 工具和实用程序手册；
- Borland C++ 程序员指南；
- Borland C++ 库函数参考手册；
- Resource Workshop 资源工作库（或称管理程序）用户手册；
- Turbo Debugger 用户手册；
- Turbo Profiler 用户手册；
- Turbo Assembler 用户手册；
- Turbo Assembler 快速参考。

此外，还提供快速查找卡。发行原盘包含所有程序、文件和库，利用它们可以建立、编译、连接和运行 Borland C++ 程序，还包括有几个样本程序，几个卓有成效的实用程序，紧缩的提示文件、集成调试器以及那些手册中没有包括进去的附加 C/C++ 文档。

2.5 Borland C++ 3.0 的安装

Borland C++ 软件包包括两种 Borland C++ 编译器的不同版本，即 IDE(集成开发环境)和 DOS 命令行编译器，还包括运行于 Windows 下的 Turbo C++ for Windows. 由于使用了文件压缩技术，用户必须使用安装程序 INSTALL 去安装 Borland C++ 3.0，而不能简单地将 Borland C++ 文件拷贝到用户的硬盘上，使用 INSTALL 可自动地把发行源盘上的文件拷贝到用户的硬盘上。安装盘上的 README 文件包含一个 Borland C++ 软件的所有文件的清单。

假定用户已熟悉 DOS 命令，则可用 DISKCOPY 命令去备份 Borland C++ 3.0 软件，然后将源盘放到安全处。

2.5.1 使用 INSTALL 安装 Borland C++

INSTALL 检查所使用的硬件，并且适当地配置 Borland C++ ，它也能随需要创建目录，并且从配置盘上把文件拷贝到硬盘上，它的操作是自解释性的。为了安装 Borland C++ ，操作步骤如下：

(1)插入安装盘(第一张盘)到驱动器 A，键入下述命令，然后按 Enter 键：

A:INSTALL 〈Enter〉

(2)在安装屏幕上，按〈Enter〉；

(3)顺序回答所有的提示；

(4)在安装过程完毕后，将以下行添加到 CONFIG. SYS 文件中：

FILES=20

将以下行添加到 AUTOEXEC. BAT 文件中

PATH=C:\BORLANDC\BIN

注意：当安装结束时，INSTALL 程序显示 README 文件的内容，让用户了解关于 Borland C++ 的最新信息，文件 HELPME! .DOC 回答一些经常遇到的技术支持问题。

在退出 README 文件之后，启动 Microsoft Windows，安装程序在 Windows 程序管理器中创建和安装了 Borland C++ 程序组，该程序组包含下列 Borland C++ 程序和实用程序的图标：

- Borland C++
- Turbo Profile
- Turbo Debugger for Windows
- Turbo C++ for windows
- Resource Workshop
- Winsight
- Import Librarian
- Fconvert utility

注意:INSTALL 假定 Microsoft Windows 已安装在用户指定的目录中,同时还假定当启动 Windows 时,程序管理器将作为 Windows 的 shell 自动启动。若平常使用不同于程序管理器的 shell 命令,应编辑 Windows 目录中的 SYSTEM.INI 文件,使之包含以下之行:

SHELL=PROGMAN.EXE

否则,当安装 Borland C++ 之后首次启动 Windows 时可能会接收到"Cannot Communicate with Program Manager"的信息,并且 Borland C++ 试图创建新的程序管理器组。一旦程序管理器组中安装了 Windows Turbo C++ ,可以检查其设置,若需要的话,可重新在 Alternate 命令 shell 中安装它们。

2.5.1.1 保护模式和内存

Borland C++ 利用 DPMI(DOS 保护模式接口)在保护模式下运行编译器,在不对换(swap)程序的情况下,使用系统所拥有的全部内存。保护模式接口对用户来说是透明的,除了很少数的例外,用户几乎很少需要考虑它的存在。

(1)**DPMIINST**

当用户首次运行 Borland C++ 时可能出现异常,Borland C++ 利用不同机器特点的内部数据库来决定如何打开机器的保护模式,并作相应的配置。若 Borland C++ 不认用户的计算机,则用户会收到一个出错信息。

Machine not in d atabase (RUN DPMIINST)

若接收到该信息,只需在 DOS 提示符下运行 DPMIINST,键入

DPMIINST 〈Enter〉

然后按程序指令行事。DPMIINST 通过一系列测试来决定启动保护模式的最佳方式,且自动设置 Borland C++ ,一旦用户运行过 DPMIINST,下次就不必再运行了。

(2)**DPMIMEM**

Borland C++ DPMI 接口缺省时可利用所有可用的扩充内存,若不想让 DPMI 内核独占内存资源,可设置使用内存的最大值,为了达到此目的,必须设置一个环境变量,来指定最大的内存使用量。变量可直接在 DOS 提示符下键入,也可以在 AUTOEXEC.BAT 文件中加入一行命令行,使用如下语法:

DPMIMEM=MAXMEM nnnn

其中,nnnn 是 K 字节数。

例如,若系统有 4M 内存,而只让 DPMI 内核使用 2M,剩余 2M 不用,可设置如下:

C:>set DPMIMEM=MAXMEM 2000

当正运行于 386 增强模式下的 Windows 3.0 时,不应设置 DPMIMEM 变量,而可用 Windows PIF 文件设置 Borland C++ 的内存使用量。

在 Windows 标准模式下,我们建议运行 DPMIRES.EXE,但 Borland C++ DPMI 内核应在运行 Windows 之前预先装入,当 DPMIRES.EXE 和 Windows 一起使用时,应将 DPMIMEM 变量设置成比最大内存变量小,以保证 Windows 有足够的物理空间供运行之用。

(3)**DPMIRES**

DPMIRES 是一个 Borland 实用程序,它能同 Borland C++ 3.0 一起使用,在某些条件下,可提高 Windows 语言工具的某些性能,特别是可提高以下工具的性能

- BCC
- TASMX