第一章 引 言

1.1 C++语言的发展过程

六、七十年代计算机技术领域所出现的"软件危机"迫使一批计算机科学家和软件人员去寻求最有效的克服危机和提高效率的手段。面向对象的程序设计语言(OOPL)由此应运而生，从70年代开始，相继推出许多早期的OOPL，包括有下列几种语言。

- CLOS (Common Lisp Object System)；
- Smalltalk 80；
- Modula－3；
- Commonloops；
- Eiffel；
- Object Logo、Object C、Object Pascal。

上述这些早期的OOPL对于现代的OOPL主流语言C++曾产生相当大的推动作用。

70年代末，一位英国剑桥大学的博士生B. Stroustrup在其博士学位论文"分布式计算机系统中的通信和控制"中采用当时条件下的Simula仿真语言作为模拟程序的编写工具语言。在编写和运行该模块程序的实践中，B. Stroustrup体会到Simula中的类(class)的概念非常有助于系统描述和程序组织，并且能方便地实现并发性。但由于当时Simula的实现仅适用于小规模程序，并且运行效率很低，以致几乎无法得到所需的模拟结果数据。因此，他不得不改用BCPL语言(C语言前身)重写模拟程序，虽然该模拟程序运行性能很好，并获得全部结果，但由于BCPL没有任何类型检查，也无运行库支持，所以其编码和调试过程相当困难。

1979年，B. Stroustrup进入AT&T公司Bell实验室计算机科学研究中心工作，他在分析UNIX内核及其在计算机局域网络的分布程度的项目时，需要拥有一种表达复杂系统的模块结构及模块间通信模式的手段。为此，Stroustrup开始自己动手编制其所需的软件工具，其编制标准是如下几点：

- 具有BCPL那样的运行效率和易于组合个别编译模块的能力，能通过简单的约定联结其他语言编制的模块；
- 具有simula那样的类及类层次的概念，并支持一定形式的并发控制、类型检查和程序设计；
- 具有较高的可移植能力，适应不同的计算机平台和硬件平台。

根据上述标准编制的在C语言中扩充类(class)的予处理工具程序称为Cpre，并于79年10月投入运行。随后，Stroustrup开始将目标从"工具"转向"语言"。1980年3月，他推出了"C with Classes"(带有类的C语言)，并在1983年出版的"Software Practice and Experience"杂志上发表了该语言的详细说明。

C with Classes在C语言的基础上扩充了下列特点：

(1) 类(class)和派生类(derived classes)；

(2) 友类(friend class)；

(3) 公有/私有(public/private)访问控制；

(4) 构造函数(constructor)和析构函数(destructor)；

(5) 调用和返回函数；

(6) 插入替换函数(inline function)；

(7) 赋值算符的重载；

(8) 函数参数的类型检查和转换；

(9) 默认(缺省)的参数。

B. Stroustrup 最终选择 C 语言作为其编制 OOPL 语言的基础，其主要原因在于 C 的灵活性、高效性、可用性(任何机器上均配有 C 编译程序)和可移植性。虽然从语言学的角度讲，C 不是最好的语言，但确实能解决问题。

Stroustrup 又进一步对 C with Classes 做了设计和实现方面的扩充和改进，改进后的版本称为 C84。不久，Stroustrup 采纳了 Rick Masciti 的建议，在他所发表的论文“Data Abstraction in C”中首次使用了 C++的名称。

1984 年 1 月，第一本 C++参考手册正式发表，名称为“The C++ Reference Manual”。1985 年 10 月，发布了 C++编译程序的前端处理程序 Cfront 1.0，接着 1986 年 6 月和 1987 年 2 月发布了 Cfront 1.1 和 Cfront1.2。

Cfront 1.0 相对 C with Classes 增加了下列特点：

- 虚拟函数(virtual function)；
- 函数名和算符的重载；
- 引用(reference)；
- 常量(const)；
- 用户控制的自由内存；
- 改进了类型检查。

1989 年 6 月发表了 Cfront 2.0 版，它与 1.x 版的主要差别在于如下。

- 增加了多重继承(Multiple inheritance)机制；
- 提供类型安全的联结；
- 改进了对重载函数的处理；
- 赋值和初始化的递归定义；
- 用户定义的内存管理特点；
- 静态成员函数；
- 常量成员函数；
- 受保护的成员(最早出现在 1.2 版中)；
- －＞算符的重载；

指向成员的指针(最早出现在 1.2 版中)。

1990 年 4 月发表了 Cfront 2.1 版，排列了 2.0 版中的一些错误。1991 年 9 月又推出了 Cfront 3.0 版，其中加进了抽象类 Template。1993 年发布 Cfront 4.0，其中含有异常处理(exception handling)的特点，并支持多字节字符集。

自 1985 年以来，C++语言逐年增长吸引更多的人参与研究工作，1986 年出版了 Strous-

trup 的第一本著作“The C++ Programming Language”，其中给出了关于 1.x 版 C++语言的详细定义。

Stroustrup 的 C++设计目标是如下几点：

(1) 提供构造程序的一般机制而不是面向特定的应用领域。

(2) 对于传统由 C 语言解决的“计算问题”，C++提供更好的程序组织结构，而不用增加运行时的额外开销，即在运行时间、代码和数据的紧凑性方面接近于 C 语言。

(3) 可应用于任何适合使用 C 的场合。

由于动态类型检查必然增加运行的开销，因此 C++采用静态类型系统而不是动态类型系统。

基于运行效率和移植性的考虑，C++中不提供内存无用单元收集(garbage collection)，但可以在必要时任选地增加无用单元收集功能。

由于不同的应用对并发控制的需求存有许多差异，难以高效地实现一种通用的并发控制机制。C++中不明显地提供并发控制的语法成份，而采用库程序实现并发控制。

由于应用需要，C++中保留了 C 的低级操作(例如位操作等)和显式不检查类型(不安全)的转换。

1987 年 11 月，UNIX 用户协会 USENIX 召开了第一次有关 C++语言的专门会议(有 214 人参加)。随后，AT&T 公司进一步完善 C++之外，其他厂商也开始提供 C++编译器(程序)产品，主要有 GNC、TauMetric、Sun、HP、Certerline、DEC、IBM、Zortech、Microsoft 和 Borland 等。其中以 Zortech、Microsoft 及 Borland 公司的 C++编译器最为著名，尤其是 Borland 公司的 C++，非常适用于广泛的应用问题，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、Borland C++ 3.0 及 Borland C++ 3.1，到目前为止已发展到最新的第四代产品 Borland C++ 4.0 版。本系列教材就是以 C++ 4.0 版为基础而编写的。

值得指出，在 1988 年 6 月推出的所有 PC 机上的 C++编译程序都是从 Cfront 移植的。

1988 年 6 月，Zortech 发布自行开发的 C++编译器，当前最新版本为 3.1 版；

1992 年 3 月，Microsoft 推出自己的 C++ (C/C++ 7.0)。

1992 年 2 月，DEC 公司宣布自行开发的 DEC C++ 1.0。同年 5 月，IBM 公司也宣布了自行开发的 C++产品。

随着 C++产品不断开发，C++用户也迅速增长，据保守的统计，1992 年的 C++用户已达到至少 150 万，相当于 1985 年的 3000 倍。

各类有关 C++的刊物也大量涌现，例如从 1989 年 1 月起，SIGS 开始出版不定期刊物“The C++ Report”，从 1991 年改为季刊，并改名为“The C++ Journal”；现在许多刊物，如 Computer Language、The Journal of Object-Oriented Programming、Dr. Dobbs Journal、The C Journal 都包含有 C++的专栏。而在电子信息网络中也包含有 C++的专用公告牌，例如 usenet 的 comm. lang. C++，BIX 的 C. plus. plus 等。

在我们国内，近年来也开始逐渐形成 C++热，主要反映在应用领域的扩展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++的应用教材和 C++的培训教材。

1.2 C++语言的标准化工作

随着C++语言及其应用的发展，其标准化问题必然提到议事日程上。1989年由HP公司发起，联合AT&T、DEC和IBM公司共同向ANSI提议开展标准化工作。HP公司的Dmirty Lenkov起草一份ANSI立项建议书，其中列举开展C++标准化工作的理由：

- C++正以比其它语言更快的速度为用户们所接受；
- 延迟标准化将导致出现更多的方言；
- 需要对每个语言特征给出详细的定义，并提供完整的语义；
- C++尚缺乏许多重要的特征，诸如异常处理、多重继承、支持参数多态性的特征以及标准库等。

建议书同时还强调了C++与ANSI C的兼容性问题。1990年AT&T出版纯描述C++语言的文本，即著名的"The Annotated C++ Reference Manual"(简称ARM)。

1989年，ANSI X3J16 C++标准化委员会成立，同年11月在华盛顿召开了第一次委员会。1991年6月，以IEC/JTC1/SC22/WG21 C++工作组正式成立，并在瑞典的Lund召开了第一次工作组会议。两个组织决定联合工作，用一个标准文本同时作为ANSI和ISO的标准。

C++标准化工作组划分成以下几个小组：

- 核心组——专门研究C++语言的精确定义；
- 扩充组——专门研究对C++语言所必须的扩充；
- 库组——专门研究C++标准库的组成；
- C/C++兼容性驻——专门研究C++与C的兼容性问题；
- 环境组——专门研究C++对支撑环境的要求和接口。

到1993年6月底为止，ANSI X3J16的成员总数已达到148个单位，总共227人。ISO/IEC/JTC1/SC22/WG21的成员包括澳大利亚、加拿大、中国、丹麦、法国、德国、意大利、日本、荷兰、新西兰、瑞典、英国和美国的技术专家，共有35人，其它还有联络员4人，以及奥地利、波兰、俄罗斯和英国的个人参加者4人。

从1990年以来，两个组织频繁举行会议。1991年6月起都是联席会议，至今已召开过12次全体工作组会议，收到各类提案500多件，以ARM为基础的标准草案工作文本已是第六稿。

在1992年年底确定的标准化工作进程计划列于如下：

- 1994年9月，完成工作草案；
- 1994年10月，将工作草案作为SC22的委员会草案登记；
- 1995年3月，作为委员会草案正式发布；
- 1995年5月，委员会草案表决期结束；
- 1995年8月，JTC1作为第一个DIS(标准草案)发布；
- 1996年3月，开始DIS的表决；
- 1996年6月，DIS表决结束，成为正式标准。

像其他语言一样，C++的标准化工作也遇到许多困难。最大的问题就是扩充，因为各个应用领域的有关人员都希望语言中具有该领域最适用的机制。标准化小组在这方面十分谨慎，除抽象类和异常处理之外，基本能确定的扩充就是名字空间的处理和对强制(cast)的动态类

型检查。另一个重要问题就是C++与ANSI C(即ISO 9899)的兼容性问题。虽然ANSI C的标准采纳了C++的许多定义,两者之间的差距比传统C与C++的差距缩小了许多,但C++还是不可能做到百分之百的兼容ANSI C,只能尽可能接近ANSI C。标准草案工作文本的附录中详细列出了C++与传统C和ANSI C的差别,以及这些差别可能造成的影响。

此外,ISO 10646字符编码集的发布,也对C++提出了国际化的新问题。C++标准对这个问题的处理方法是直接采用即将完成的ISO 9899补篇中对多字节字符支撑的扩充,并在C++中增加了多字节字符串的类库。

总之,随着C++标准化工作的进展,C++语言也将更趋完善和更显魅力,新的编译器版本将会及时实现已被确认的修改和扩充,各种C++的类库也将继续扩充,应用C++的开发工具和环境将会逐步流行,有关C++语言和程序设计的教育培训也会更加广泛。可以予料,几年以后,C++语言将会成为最流行最热门的程序设计语言之一。

下面将言归正传,详细介绍Borland C++ 4.0的主要内容和开发工具等。

1.3 Borland C++ 4.0的新特点

Borland C++是用于建立和维护由C语言和C++语言编写的DOS、Windows、Win 32s和Windows NT应用程序的专业性工具。Borland C++ 4.0相对于先前版本增加了很多新的特点,现列出如下:

- 32位编译器和工具为Win 32s和Windows NT生成32位目标;
- 你可以从Windows IDE生成DOS程序;
- IDE(集成开发环境)拥有一种图形集成调试器(程序)。用于调试16位Windows应用程序;
- IDE具有一个增强的编辑器(程序),它让你记录击键宏命令,工作在一个编辑器窗口中的多个方格,并搜索使用规定表达式的文本。你可配置使用Brief或Epsilon击键的编辑器,或者你可创建自己的击键;
- 右边鼠标器按钮带出SpeedMenus,它列出你单击(click)对象所特有的命令。例如,某些通用的编辑命令就位于所有编辑器窗口的SpeedMenu上。(为了访问右边鼠标按钮的老功能,按压Ctrl+click右边鼠标按钮)。
- IDE拥有一种新型的多目标项目管理器(程序),它清楚地显示出文件相关性,并让你管理多个程序。
- IDE具有一种新型的多窗口ObjectBrowser,它显示出类的关系。
- 利用AppExpert,你可以快速地生成ObjectWondows 2.0 Windows程序。ClassExpert帮助你修改和组织你的AppExpert应用程序。

本书可作为用户指南教程,它教会你如何安装和使用Borland C++ for Windows,同时也教你如何使用集成开发环境(IDE)中的各种成份,包括集成调试器(debugger),浏览器(browser)、AppExpert、ClassExpert和项目管理器(程序)。本书也提供命令行工具,包括编译器、链接器、库和MAKE。

总之,Borland C++ 4.0能做到如下:

(1) 集成开发DOS、Windows、Win32s和Windows NT应用程序。你可以从单一项目文件中构造多种应用类型。

(2) 使用 AppExpert 快速而方便地建立 ObjectWindows 应用程序。随后，ClassExpert 通过跟踪类和事件辅助你维护该应用程序，并通过 Resource Workshop(资源工具库)来管理应用程序中所创建和使用的资源。

(3) 帮助你调试和浏览你的应用程序，不需要使用一个独立的调试器。

(4) 包含一个可定制的编辑器。你可使用由 Borland C++提供的键盘简捷键(shortcut)，或者你定制自己的编辑器。

本书也介绍两个可帮助调试应用程序的 Windows 程序：WinSight 和 WinSpector。

第二章　Borland C++ 4.0 的安装与使用

Borland C++ 4.0 是一种开发软件包，它包含有 Windows 工具、命令行工具和库，这些工具帮助你开发 DOS、Windows、Win32s 和 Windows NT 的应用程序。本章给出 Borland C++ 4.0 产品的工作描述，这些产品包括 IDE、项目管理器(程序)、AppExpert、工具和实用程序。

通过本章可学习到下列内容：

- 安装和配置 Borland C++
- 使用编辑器(程序)
- 使用代码的语法重点
- 使用 SpeedBar(加速棒)
- 使用 Message(信息)窗口
- 浏览你的代码
- 启动和使用 IDE 的其它工具

2.1　Borland C++的安装

Borland C++包括开发 DOS 和 Windows 两部分应用程序。在安装 Borland C++之前，要保证计算机的最小硬件和软件要求。

2.1.1　硬件和软件要求

为了使用 Borland C++ 4.0，计算机要达到如下配置：

- DOS 4.01 以上版本；
- 运行在 386 增强方式下的 Windows 3.1 以上版本；
- 对于基本安装，具备可用磁盘空间 40MB 的硬盘(对于完全安装，需要 80MB)；
- 一个 1.44MB 软盘驱动器或 CD ROM(用于安装)；
- 至少 4MB 扩充内存；
- 一个 Windows 兼容的鼠标器。

若想增加计算机的性能，最好配置如下：

- 8MB RAM；
- 一个 80x87 数学协处理机(若你正在编制使用浮点数值的程序)。若没有协处理机，Borland C++就仿真一个数学芯片。

2.1.2　安装步骤

Borland C++安装程序可安装 Borland C++产品(诸如 IDE、命令行工具、ObjectWin-

dows 和 Turbo Debuggers)以及安装 Win32s(允许你在 16 位 Windows 下运行 32 位程序)。安装程序可工作在 Windows、Win32s 和 Windows NT 环境下;然而,并非所有程序都可运行在 Windows NT 下面。

安装之前,首先检查计算机是否满足硬件和软件要求。若你想知道关于安装 Borland C++的更多信息,请阅读位于 Disk 1(磁盘 1)上的在线文件 INSTALL. TXT(这个文件未被压缩,所以可使用任何文本编辑程序来查看它)。

对于软盘和 CD 的安装指令是基本相同的,但你应该阅读 INSTALL. TXT 文件,或者当你从 CD ROM 安装时要阅读 CD 要点说明。

现在介绍从软盘安置 Borland C++ 4.0。

1. 将 Disk 1 插入你的软盘驱动器(通常是 A 或 B)。

2. 启动 Windows,并从 Program Manager 中选择 File|Run。

3. 键入 a:\install <Enter> (或是 b:\install <Enter>)。出现安装对话框。在这个对话框的底部,你将看到为实现完全安装所需的硬盘空间量(目标要求 Target Requirements)。你也将看到你的机器上可用的磁盘空间量。在继续下去之前,要保证有足够多的空间可用于安装。若你的计算机使用磁盘压缩实用程序,请读 INSTALL. TXT 文件;你可能需要比列出的可用空间更多的空间。

4. 若你想仅选择特定文件用于安装,请单击(click) Customize BC 4.0 Installation 按钮。出现另一个对话框,并带有按钮和产品方面的说明。对于希望定制的一个内容单击按钮。相应于那个内容的对话框就出现了,在此你无需检验不想安装的文件(缺省安装将全部文件安装到你的机器上)。对任何想定制的内容单击 OK 及重复这个过程。单击 OK 返回到第一个安装对话框。

5. 安装程序列出将要安装文件的缺省目录,若你要 Borland C++安装到一个不同的目录,只要键入另一个路径。

- **Borland C++目标目录**是主目录。所有其它的文件均被安装到该目录下(缺省情况下,该目录是 C:\BC4)。
- **Borland C++ Working(工作)目录**是安装程序存放 Borland C++应用程序的目录(通常是 C:\BC4\BIN)。

6. 在缺省(默认)情况下,安装程序创建一个 Windows 组(group),该组放置所有的 Borland C++图标。若你不希望创建一个组,就不检验 Create Borland C++ Group。

7. Win32s 也在缺省下安装。若你不想要 Win32s,不用检验这选项。当运行 32 位应用时需要 Win32s。

8. 仅当你在带有 LAN Windows 的机器上安装时,才要检验 LAN Windows Configuration。

9. 单击 Install 开始将文件拷贝到你的机器上。当完成安装时,你可读 README. TXT 文件。这个文件包含有产品、资料和在线 Help 方面的最新变化。对于安装建立的所有图标的描述,请查看 Disk 1 和 C:\BC4 上的 INSTALL. TXT。

安装之后,要保证你的 CONFIG. SYS 文件中的 FILES 和 BUFFERS 等于 40 以上(关于 CONFIG. SYS 文件的信息,请参看 DOS 资料)。

这样的安装对机器中的现有文件实现了如下的改变:

- AUTOEXEC. BAT 现在包含通向 Borland C++的路径(缺省时,C:\BC4\BIN);

- WIN.INI 包含一个段[BCW4.0 INSTALL],它被 TASM 安装程序所使用,用来对 Borland C++安装到机器上的位置定位。此外,在[EXTENSIONS]段,扩展 IDE 关联到 IDE(BCW.EXE)。
- SYSTEM.INI 包含两个设备行:

  device = C:\bc4\bin\tddebug.386

  device = C:\bc4\bin\windpmi.386
- 若你运行在 Windows NT 下,则 CONFIG.NT 中要加上 NTCMDPROMPT。

FILELIST.TXT 列出随 Borland C++一起提供的每一个文件。若你需要释放磁盘空间,在你删除任何 Borland C++文件之前查阅这个文件。

2.1.3 启动 Borland C++

为了启动 IDE,要双击(double-click) Windows 中的 Borland C++图标。IDE 允许你编写、编辑、编译、链接、调试和管理编程项目。IDE 由下列组成(图 2.1 所示)

- 一个编辑器和浏览器
- 一个项目管理器
- 一个调试器

(注:这里的器就是指程序。)

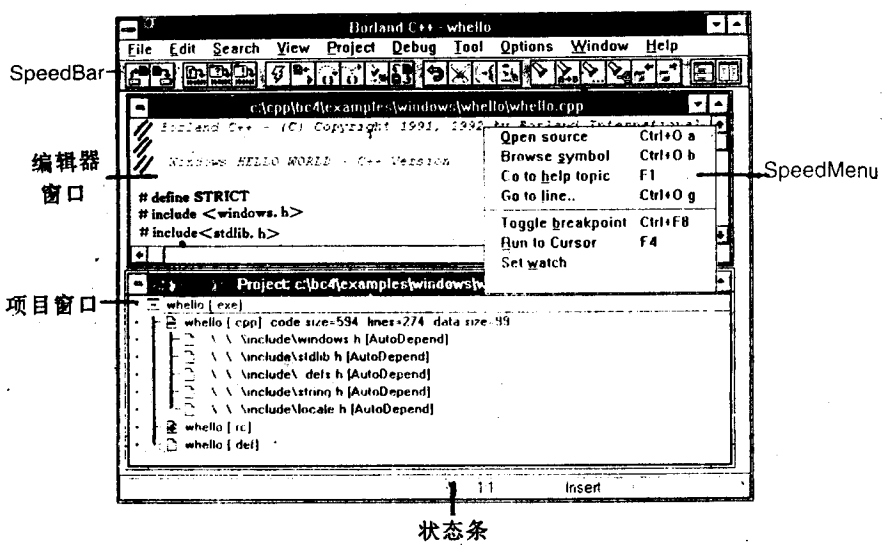

图 2.1 IDE 的组成

IDE 含有内容敏感的 SpeedMenus(加速菜单),它让你很快修改对象。为了观察一个 SpeedMenu,在一个窗口中或者窗口中的一个条目上进行右击(right-click),或者按压 Alt+F10(SpeedMenu 根据所选的内容发生变化)。例如,为了跳转到编辑器窗口中的一行,在编辑器窗口右击,选择 Go to line,然后键入你想选择的行数。仅当选择一个编辑器窗口时,才会出现菜单条目"Go to line"。若你在一个项目窗口中打开 SpeedMenu,你将看到一个完全不同的菜单条目组。

SpeedBar(加速棒)也根据你所选的窗口而改变。对于编辑器、浏览器、调试器、项目管理器、信息窗口(message wondow)、桌面(desktop)和 ClassExpert 都存在一个可配置的 Speed-

Bar。(为了配置一个 SpeedBar,参看下一节)。当你在 SpeedBar 的一个按钮上设置鼠标器指针时,在 IDE 底部的状态行就出现描述此按钮的帮助(help)行。

在 SpeedBar 上的某些按钮有时是不清楚的。这表明描述该按钮的命令在当前含义中对你是不可用的。例如,若打开一个编辑窗口,来自 Clipboard(剪辑板)按钮的 Paste Text 是不清楚的(若在 Clipboard 中不存在文本)。

2.1.4 获得帮助

Borland c++ Help 系统让你在线访问关于 Borland c++的详细信息。你可找到在手册和在线 Help 两部分中的大多数产品信息。然而下面列出的题目仅含在在线 Help 中,

- IDE 菜单命令
- 编辑器 KEYMAPPER
- 运行时库例子代码
- Windows API

为了获取在线 Help(帮助),做下列操作:

- 在 IDE 中,从菜单选择 Help 或者按压 F1。
- 在对话框中,单击 Help 按钮或者按压 F1。
- 对于菜单命令,选择菜单命令然后按压 F1。

2.2 IDE 的配置

你可配置 IDE 自动地实现任务(诸如在编辑器窗口中保存文件的备份)或处理事件。本节描述可配置的内容。

Options | Environment 对话框允许你配置编辑器、浏览器、调试器、项目窗口和其他的 IDE 组成(这些选项被保存在一个称为 BCCONFIG. BCW 的文件中)。

为了打开 Environment Options(环境选项)对话框,选择 Options | Environment。则出现对话框,在其左边带有题目表。某些题目在其下含有子题目。例如,Editor 题目拥有称为 Options、File 和 Display 的子题目。当一个题目具有未被显示的子题目时,该题目在名字之后含有一个"+"。当你单击一个题目的"+"符号时,它的子题目出现于其下面,而"+"转变成"-"(你此时可单击该"-"以清除此表)。不包含子题目的题目会出现,在其名字之后带有小点。当你单击一个题目时,它的特征出现在对话框内的右方。

这里不准备讨论所有的 Options | Environment 题目,详情可参考在线 Help(单击 Help 按钮)。某些同任务或 IDE 组成相关的题目将在本书的相应处予以讨论。

2.2.1 SpeedBar(加速棒)

IDE 拥有针对编辑器、浏览器、调试器、项目、信息、桌面和 ClassExpert 窗口的 SpeedBars。当你选择这些窗口类型之一时,就会出现相应的 SpeedBar。你可定制每一种 SpeedBar,使它们仅包含所需的按钮。

为了从任何 SpeedBar 中增加或删除按钮,进行如下操作:

1. 从主菜单选择 Options | Environment。
2. 选择左方的 SpeedBar 题目。对话框的右方显示所有 SpeedBar 的总选项。

这里的选项让你选择 SpeedBar 的出现方式(IDE 的顶部或底部),以及 SpeedBar 的行为(当你传送鼠标器指针到一个按钮上时,检查 Use flyby help 来察看状态行上的 help 提示)。

3. 选择 SpeedBar 下面的题目 Customize。在右方的选项显示出关于 SpeedBar 的信息。
4. 从窗口下拉菜单中选择你想修改的 SpeedBar 类型(编辑器、浏览器、调试器、项目、信息、桌面、或 ClassExpert)。

   在左方的列(Available Buttons)显示所有可用的(未被使用的)按钮,在按钮功能说明后面带有名字。在右方的列(Active Buttons)仅显示被选 SpeedBar 的按钮。
5. 为了增加一个按钮,在 Available Buttons 表中双击按钮图标,或者选择它并单击右指向箭头。该按钮移动到 Active Buttons 表。
6. 为了从 SpeedBar 中取消一个按钮,双击 Available Buttons 表中的按钮图标,或选择它并单击左指向箭头。该按钮移动到 Available Buttons 表。

为了重新按排 SpeedBar 的按钮位置顺序,使用向上(up)和向下(down)箭头。在 Active Buttons 表中的被选按钮向上移动或向下移动该表(顶部按钮出现在 SpeedBar 的远左方;在表中的最后按钮出现在远右方)。

你也可通过选择 Window 表中的 SpeedBar 来制作同样的所有 SpeedBars,此时按压 Copy Layout 按钮。出现一个对话框,你在其中检查希望做得与被选 SpeedBar 相同的所有 SpeedBars。例如,若你首先选择 Editor SpeedBar,然后单击 Copy Layout,出现带有朦胧 Editor 的对话框。若你此时检查 Project(项目)和 Message(信息),这些 SpeedBar 将完全相同于 Editor(编辑器)SpeedBar。

你可通过在 Window 表中选择 SpeedBar,并单击 Restore Layout 按钮而将任意 SpeedBar 恢复到原先的缺省状态。

分隔符(separator)在两个按钮之间放置空间。你可通过从 Active Buttons 表中选择一个按钮,并单击 Separator 按钮而将分隔符加到任意 SpeedBar 中。要在被选按钮之前增加分隔符。

2.2.2 设置 IDE 优先权和保存 IDE 设置

优先权(preferences)让你定制想自动保存的内容以及某些窗口的工作方式。

为了设置优先权,进行如下操作:

1. 选择 Options|Environment|Preferences。
2. 检验及不检验所要的选项。对于多个选项的说明参看在线 Help(按压 Help 按钮)。
3. 选择 OK。

当你退出 IDE、构造或实现一个项目、使用一个传递工具、运行集成调试器、或者打开(或者关闭)或者保存一个项目时,IDE 自动保存信息。可通过如下手段来控制自动保存过程,即从 Environment Options 对话框(从主菜单选择 Options|Environment)选择 Preferences,并为自动保存设置选项。

为了人工保存你的设置,进行如下操作:

1. 选择 Options|Save。
2. 从 Environment Options 对话框的 Editor(编辑器)、Syntax Highlighting(语法重点)、SpeedBar(加速棒)、Browser(浏览器)和 Preferences(优先权)段,检验 Environment 来

保存设置，这些设置被保存在一个名为 BCCONFIG.BCW 的文件中。

3. 检验 Desktop 来保存有关打开窗口及其位置的信息。这个信息被保存在称为<prj-name>.DSW 的文件中。若你并不具有一个打开的项目，该信息被保存到称为 BCWDEF.DSW 的文件中。
4. 检验 Project 将变化保存到你的 project(.IDE)文件内，包括构造选项和节点属性。

2.3 编辑器(Editor)的使用

编辑器窗口是建立和编辑程序代码的场所。当你编辑一个文件时，IDE 状态条显示光标的行号和字符位置。例如，若光标位于一个编辑器窗口的第一行和第一字符处，将在状态条看到 1：1；若光标位于行 43 和字符 13，将看到 43：13。IDE 状态条也指出，光标是否将修改或插入字符(按压 Insert 来触发这个选项)，并当你已经在被选编辑器中对文件作出任何改变时显示字 Modified。

编辑器通过选择 Edit|Undo 或按压 Alt+Backspace 让你废除多个编辑。例如，若你删除一个文本行，然后贴补某个文本，可废除这些编辑：首先废除最近的贴补，然后删除之。可通过选择 Options|Environment|Editor|Options 和设置 Undo Limit 来设定允许的废除作用数。

2.3.1 配置 IDE 编辑器

你可以对编辑器进行配置，使它具有类似于其它编辑器(如 Brief 和 Epsilon)的行为。IDE 编辑器使用键盘映像文件(.CKB)，它为编辑器设置键盘“简捷”键(shortouts)(这些文件也改变了其他窗口的击键)。

你可使用四种缺省 .CKB 文件之一，其操作方法是选择 Options|Environment|Editor 并单击一种 SpeedSetting(default Keymapping、IDE classic、BRIEF emulation 或 Epsilon)。为了学会如何编辑或建立你自己的 .CKB 文件，请参看在线 Help(搜索“Keymapper”)。

2.3.2 语法重点(Syntax Highlighting)

Syntax Highlighting 让你为某些代码成分定义色彩和字体属性(像粗体等)。例如，你可用兰色来显示注释，而用红色显示字符串。Syntax Highlighting 的缺省值是接通(on)。为了断开它，则进行如下操作：

1. 选择 Options|Environment|Syntax Highlighting。
2. 不检验 Use Syntax Highlighting。

Syntax Highlighting 工作于这样的文件上，其扩展名列在 Syntax Extensions 表中(缺省时为 .CPP、.C、.H、和 .HPP)。你可从这表中增加或删除任何扩展名，但必须用分号来分隔扩展名。

Syntax Highlighting 段显示缺省色彩方案以及你可使用的四种予定的色彩设置(按钮)。为了使用一种予定的色彩方案，进行如下操作。

1. 选择 Options|Environment|Syntax Highlighting。
2. 单击其按钮选择四种色彩方案之一；示例代码发生变化以便使用选择的色彩方案。你可使用一种色彩方案作为定制 Syntax Highlighting 的出发点。

为了人工选择 Syntax Highlighting 色彩，进行如下操作：

1. 选择 Options|Environment|Syntax Highlighting|Customize。在 Environment Options 对话框的右上方出现组成和示例代码。
2. 从组成表中选择你想修改的一个组成(例如注释),或者单击示例代码中的组成(这选择组成表中的名字)。你可滚动示例代码来查看更多的组成。示例代码使用 Environment Options 对话框的 Editor|Display 段中所选的字体。
3. 对组成选择一种色彩。在示例代码中的组成色彩反映了你的选择。使用左边鼠标按钮来选择组成的前景色彩(FG 出现在这种色彩中)。使用右边鼠标按钮来选择一种背景色彩(BG 出现在这种色彩中)。若 FB 出现在色彩中,此色彩就被用作背景和前景两者的色彩。
4. 若你需要,选择一种属性(Attribute),诸如粗体。
5. 你可检验缺省 FG(前景)或 BG(背景)以便为一个组成使用 Windows 缺省色彩。
6. 对于想修改的组成重复步骤 2～4。

2.4 Message(信息)窗口的使用

当编译程序时,Message 窗口显示出错和警告。当你在 Message 窗口选择信息时,编辑器将光标放置在你的代码发生出错或警告之点。若含有错误的文件在编辑器窗口未被装入,按压 Spacebar 来装入它(你也能按压 Alt+F10 和从 SpeedMenu 选择 View source(观察源))。该信息窗口仍然是所选的,所以你能从信息到信息进行传送。

为了查看与错误或警告相关的代码,可做两种操作之一,或者在信息窗口中选择信息,按 Enter,并双击该信息,或者按压 Alt+F10,并从 SpeedMenu 中选择 Edit(编辑)源。光标出现在你的源代码中最可能发生错误的行与列处(信息窗口移到背景)。使用 Alt+F7 移向下一个出错信息,而 Alt+F8 转到前一个出错信息。

你也能让光标通过信息窗口中的整个信息。当选择一个信息时,在编辑器窗口的光标移动到错误发生之处(这被称为自动错误跟踪)。仅当含有错误的文件被显示在一个编辑器窗口中时自动错误跟踪才工作。若选择的下一个信息参照另一个源文件(不是在当前编辑器窗口中的文件),那么你继续自动错误跟踪之前,必须选择显示出关联于该信息的源文件的编辑器窗口。

你可通过从信息窗口 SpeedMenu 中选择 Remove all message 来清除该信息(右击或按压 Alt+F10 来查看 SpeedMenu)。

2.5 浏览器(程序)

对象浏览器(Object Browser)是一个充分利用 Windows 图形环境的程序设计工具,浏览器让你查看应用程序所使用的对象层次、类、函数、变量、类型和常量。浏览器也让你做到如下。

- 从图形上查看应用程序中的层次,然后选取你的选择对象,并查看它包含以及继承的符号和函数。
- 列出应用程序所定义的变量,然后选择一个变量并查看它的声明,列出应用程序中对于它的所有引用,或者在源代码中编辑它的声明。

在使用浏览器之前,必须在 Project Options 对话框中设置这些选项(选择 Options|Project)和编辑你的应用程序:

• 选择 Compiler|Debugging 和在 OBJs 中检查 Debug information(调试信息)；
• 选择 Compiler|Debugging 和在 OBJs 中检查 Browser reference information(浏览器引用信息)；
• 选择 Linker|General 和检查 Include debug information(包含调试信息)。

为了启动浏览器，选择 Search|Browse Symbol、View|Classes 或者 View|Globals。你也能将光标放置在代码中的一个符号上，并选择 Search|Browse 符号带出浏览器。若该程序在当前编辑器窗口中还未被编译，你必须在使用浏览器之前编译和链接带有调试信息的程序。若你试图浏览一个类定义(或者没有符号调试信息的任何符号)，将得到一个出错信息。

你可利用 Environment Options 对话框来设置几个浏览器选项。选择 Options|Environment，单击 Browser 题目并选择想使用的选项。单一窗口表明你一次仅能拉出一个浏览器窗口；当每次执行一个浏览作用(诸如从主菜单中选择 View|Globals)，多个窗口就打开一个新的浏览器窗口。

2.5.1 浏览对象(类观察)

选择 View|Classes 查看"big picture"(大图案)，即应用程序中的对象层次以及小细节。当你选择 View|Classes 时，浏览器引出你的对象和用水平树显示它们的祖－孙(ancestor－descendant)关系。在层次中的红线有助于更清楚地查看当前被选对象的中间祖－孙关系。图 2.2 是显 WHELLO 应用的结构。

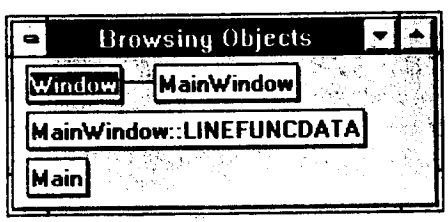

图 2.2 查看一个应用中的类

为了看到特定对象的更多信息，双击它。若你并不使用鼠标器，通过使用箭头键和按 Enter 而选择该对象。

1. 过滤器

当你浏览一个特定符号时，识别该符号的相同字母出现在浏览器窗口底部的过滤器(Filters)矩阵中。你可使用过滤器来选择想看到的符号型式(参见表 2.1)。

过渡器矩阵对每个字母拥有一列。单击顶部或底部行来移动该字母(顶部行的一个字母表示，浏览器显示带有那种识别的符号；在底部的一个字母表示浏览器排斥带有那种识别的符号)。

为了限制特定类型符号的查看，单击字母列的底部单元，如左方所示。例如，为了清除在当前选择对象中显示的所有变量，单击 V 列中底部单元。

在某些情况下，多个字母出现在一符号之后。第二个字母正好出现在识别符号类型的字母之后，并进一步描述该符号。参看表 2.1(过滤器识别符号清单)。

表 2.1　浏览器中的字母符号

字母	符号
F	Functions(函数)
T	Types(类型)
V	Variables(变量)
C	Integral constants(整常量)
?	Debuggable(可调试的)
I	Inherited form an ancestor(从祖先继承)
V	Virtual method(虚拟方法)

2. 查看所列符号的声明

使用下列方法之一来查看表中显示的特定符号的声明：

- 双击该符号。
- 选择该符号并按 Enter。
- 选择该符号，按压 Alt+F10 查看 SpeedMenu，然后选择 Browse Symbol。

该符号声明在现在一个窗口中，如图 2.3 所示。

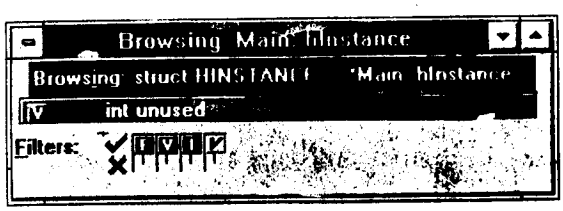

图 2.3　符号声明窗口

2.5.2　浏览全局符号

选择 View|Globals 打开一个窗口，它按字母顺序列出你的应用程序中的每一个全局符号。该浏览器列出对象中所用的符号(函数、变量等等)。图 2.4 显示 WHELLO 程序的全局符号。

一个或多个字母出现在对象中每个符号的左方。这些字母说明它是哪一类符号。你可利用浏览器窗口底部的过滤器表来滤出符号。(可参看前一节“过滤器”)。

为了获取有关一个特定符号的更多信息，或者单击该符号，或者使用光标键来选择它。在窗口底部的一个 Search(搜索)输入框让你快速地搜索全局符号表，只要键入该符号名字的前几个字母，你也能键入正规表达进行搜索(例如，可使用?、* 和+)。当键入时，在表框中的加亮条移向匹配该键入字符的一个符号。你可通过选择符号和按 Enter 来查看符号声明。(参看前一节“查看所列符号的声明”)。

1. 在浏览器中使用正规表达

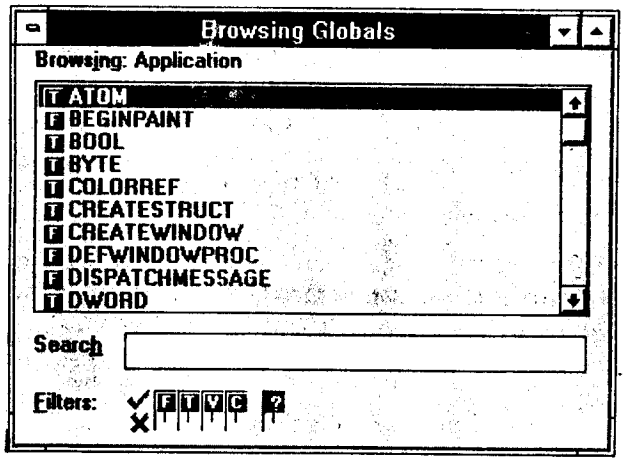

图 2.4 查看全局符号

你可在某些浏览器窗口的搜索框中使用表达。参见表 2.2(允许符号的清单)。

表 2.2 浏览器搜索表达

字符	功能
.	匹配任何一个字符
*	匹配零或多个先前字符。例如 * 是一个错误,因为不存在先前字符。 f0* 匹配由一个"f"开始的任何字符 f0*x 匹配"fx"、"f0x"、"f000x"
+	匹配一个或多个先前字符。例如, +是一个错误 f0+匹配由"f0"开始的任何字符 f0+x 匹配"f0x"、"f000x"
?	匹配零或一个先前字符。例如, ? 是一个错误 f0? 匹配由"f"起始的任何字符 f0? x 仅匹配"fx"或"f0x"

2.5.3 浏览代码中符号

你也可浏览代码中的任何符号,而不需要首先查看对象层次或符号表。从下列方法中选择:

- 加亮你的代码中的符号,并选择 Search|Browse Symbol。
- 当选择一个编辑器窗口来显示 SpeedMenu 时,单击右边鼠标按钮或按压 Alt+F10,然

后选择 Browse Symbol。

2.6 命令行工具

Borland C++含有几种命令行工具，它们让你做 IDE 中能做的相同的任务。Borland C++包含一个命令行编译器、一个链接器、一个资源编译器、一个库管理程序、一个项目构造器(称为 MAKE)和其它工具。这些工具的大多数都在本书中给以介绍。某些内容在联机文件中进行说明。所有的工具都在在线 Help 中给以介绍。

你既可使用 IDE，也可使用命令行工具，因为它们产生相同的结果，但当你的程序使用诸如 Brief 那样的 DOS 编辑器时，最好选用命令行工具。下面给出命令行工具清单：

(1)BCC. EXE 和 BCC32. EXE 是 16 位和 32 位编译器。在第三章中介绍。

(2)TLINK. EXE 和 TLLINK32. EXE 链接 .OBJ 文件和 .LIB 文件以形成 .EXE 和 .DLL 文件。在第五章中介绍。

(3)IMPLIB. EXE 和 TLIB. EXE 帮助你同库一起工作，并建立库。在第六章中介绍。

(4)HC31. EXE 编译在线 Help 文件，并创建大多数 Windows 应用程序可使用的 .HLP 文件。在线 Help 中进行介绍。

(5)BRCC. EXE、BRCC32. EXE、BRC. EXE、BRC32. EXE 和 RLINK. EXE 是资源工具，它们编译应用程序的资源。在第六章中介绍。

(6)MAKE. EXE 和 MAKER. EXE 通过构造已发生改变的文件来帮助管理项目。在第七章中进行介绍。

2.6.1 DPMI 和命令行工具

命令行编译器使用 DPMI(DOS 保护方式接口)以保护方式运行在 286、386、i486 或 Pentium 机器上，它至少带有 640K 常规 RAM 和至少 1MB 扩充内存。

虽然 Borland C++运行在保护方式下，但它仍然生成运行在实址方式下的应用程序。在保护方式下使用 Borland C++的优点在于，编译器要比运行在实址方式下拥有更多的运行空间，所以它可较快地编译较大的项目而不需要扩展的磁盘交换。

2.6.2 内存和 MAKESWAP. EXE

当你运行 32 位命令行工具时，若看到 DOS 出错"Out of Memroy"(没有从 Windows 运行 DOS)，就用 MAKESWAP 实用程序建立一个交换文件(swap file)。MAKESWAP 以 K 字节来表示建立文件的大小，例如：

```
MAKESWAP 12000
```

在当前目录中建立一个 12MB 的交换文件，称为 EDPMI. SWP，当它们需要附加内存时命令行工具使用此文件。为了建立一个交换文件，在 DOS 提示符使用 DPMIMEM 环境变量，或者将此行加到你的 AUTOEXEC. BAT 文件中：

```
set DPMIMEN = SWAPFILM<location of swap file>\EDPMI. SWP
```

在你运行 Borland C++ 3.1 命令行工具或其它的 16 位基于 DPMI 的可执行软件(诸如 Paradox)之前，你必须清除这个环境变量。为了清除此变量，在 DOS 提示符下键入：

```
set DPMIMEN =
```