

Borland C++ 3.0 & Turbo C++ 3.0
for Windows

程序员手册

李振格 编译

北京航空航天大学出版社

Borland C++ 3.0 & Turbo C++ 3.0 for Windows

程序员手册

李振格 编译

北京航空航天大学出版社

(京)新登字 166 号

内 容 简 介

本书介绍利用 Borland C++ 3.0 & Turbo C++ 3.0 for Windows 库函数、新增的丰富的类库进行传统的 DOS 和先进的 Microsoft Windows 应用程序设计,着重讲述利用 Turbo C++ for Windows 进行 Microsoft Windows 应用程序设计的技巧。最后介绍 Turbo C++ 3.0 for Windows 的 OBJECTWINDOWS 类库;介绍该类库成员的类型、基类、派生类、友元、拥有友元、成员函数和用法等信息。

Borland C++ 3.0 & Turbo C++ 3.0 for Windows
程序员手册

Borland C++ 3.0 & Turbo C++ 3.0 for Windows
Chéngxùyuán shǒucè

李振格 编译
责任编辑 陶金福

北京航空航天大学出版社出版
新华书店总店科技发行所发行 各地新华书店经销
朝阳科普印刷厂印刷
787×1092 1/16 印张:34.25 字数:876 千字
1992 年 2 月第一版 1992 年 2 月第一次印刷 印数:7000 册
ISBN 7-81012-338-6/TP·072 定价:26.50 元

前 言

随着计算机软件技术的不断发展,90年代迎来了面向对象范型(Object-Oriented Paradigm)的软件开发新时代。在C语言基础上发展起来的C++语言就是一种面向对象开发方法的程序设计语言。由于C++提供了把数据和在数据上的操作封装在一起的类、对象和方法的机制,并通过派生、继承、重载和多态性等特性,实现了人们追求已久的软件重用技术,使得软件,特别是大型复杂软件的构造和维护变得更加有效和容易,并使软件的开发能更自然地反映事物的本来面貌,从而大大提高软件开发的效率和质量。

1991年美国Borland公司在原Turbo C++基础上,推出了全新的Borland C++系列软件。Borland C++ 3.0除了实现了AT&T C++ 2.1版本的全部功能外,还支持ANSI C和Misrosoft Windows应用软件的开发,因此成为当今国际上最受欢迎的面向对象程序设计软件。

为了让读者对Borland的C++系列产品有一个完整的了解,下面介绍一下Borland公司和它的C++系列产品。

Borland公司到目前为止已开发出了众多的语言产品及其相应的支持产品。在国内用得比较多的语言产品有Borland C++、Turbo C++、Turbo C、Turbo Pascal、Turbo Prolog、Turbo Basic和Turbo Assembler。

Borland公司的语言产品除了Turbo Assembler之外,都具有可以归结为如下与众不同的特点:它们都是一个集编辑、编译、调试、运行和剖析优化等功能于一体的具有联机帮助和热键触发等特点的优秀的软件开发环境。

Borland公司与语言产品配套的实用程序也非常有名,Turbo Debugger和Turbo Profiler等程序与语言产品形成一个不可分割的整体。

到目前为止,Borland已经发行了许多种C++包装,其中有些只针对单个程序员,另外一些针对专业开发者。不同包装的编译器之间在功能上有差别。

☐ Turbo C++

Turbo C++编译器可以创建K&R、ANSI和C++程序,它是以AT&T C++ 2.0为模板实现的,软件包含了Borland类库,但不包含Turbo Vision和ObjectWindows。虽然它不作为一个非常好的个人编译器,但它不能产生运行于Windows下的应用程序,而且不包含独立的Turbo Debugger、Turbo Assembler和Turbo Profiler。

☐ Turbo C++ & Turbo Vision

该编译器不仅包含Turbo C++编译器所有成分,而且包含面向对象的Turbo Vision库。该库是编写以DOS为基础的面向文本模式用户界面的工具,利用该类库可以完成窗口、菜单、编辑控制和其它的界面元素设计。对于不关心Windows程序设计的个人来说,该软件包已经足够了。

☐ Borland C++ 2.0

该编译器保留了Turbo C++编译器的所有性能,并且还包含了独立的Turbo Debugger

Turbo Assembler 和 Turbo Profiler。虽然 Borland C++ 2.0 不提供 Turbo Vision 和 ObjectWindows,但它足以编写各种 Windows 应用程序。

Whitewater Resource Toolit (WRT)使 Borland C++ 2.0 能够创建 Windows 资源(图标、光标、位图、菜单、字体和字符串),它是一个能很好地与 Borland 产品和与 Windows 兼容的编译器一起工作的资源管理工具。

☐ Borland C++ 2.0 & Application Frameworks

该软件简称为 Borland C++ 2.0 & AF,虽然它仍然是运行于 DOS 的 IDE,但是它提供了专业的开发者用来创建 DOS 和 Windows 应用程序的任何工具和环境。

Borland C++ 2.0 & AF 包含了 Borland C++ 2.0 编译器、DOS 应用程序的类库 Turbo Vision 和 Windows 应用程序的类库 ObjectWindows,是 Borland C++ 2.0 平台上最完整的发行包装。

☐ Borland C++ 3.0

Borland C++ 3.0 提供了一个专业开发人员的集成环境,能生成高质量的 C 和 C++ 代码、DOS 和 Windows 代码。在 Borland C++ 3.0 中,C++ 支持提高到 AT&T C++ 2.1 版本的水平。这种包装既包含 Borland C++ 3.0 编译器,也包含 Turbo C++ for Windows 编译器,支持宿主 Windows 的用户界面,提供了所有的 Windows 和 DOS 开发工具、Turbo Assembler、Turbo Profiler 和 Turbo Debugger,但并不提供 Turbo Vision 和 ObjectWindows。Borland C++ 和独立的调试器仍使用 DOS 的文本接口,但其中的 Turbo C++ for Windows 是一个真正的 Windows 图形用户界面(GUI)的应用程序,并且新增了 Resource Workshop 资源管理程序。在 Windows 的 Program Manager 的 Borland C++ 3.0 程序组中,具有支持 Windows 的程序 Turbo C++ for Windows、Resource Workshop、Winsight、ImportLib、Turbo Debugger for Windows、WRemote、WRSetup、Turbo Profiler for Windows 和 FConvert。

☐ Borland C++ 3.0 & Application Frameworks

这种包装是读者最好的选择,它能运行于 Windows,并且提供了专业的开发者用来创建 DOS 和 Windows 应用程序的任何工具和环境,其中包含新的 Turbo Vision 和 ObjectWindows、类库和 C 库函数的源代码。

为了系统和全面地使用 Borland 在 C++ 3.0 平台上的系列产品的功能,在《Borland C++ & Turbo C++ 用户手册》、《Borland C++ & Turbo C++ 库函数参考手册》和《Borland C++ & Turbo C++ 程序员手册》(它们针对 Borland C++ 2.0 和 Turbo C++ 1.x)基础上,把 Borland 的 C++ 3.0 平台新增的内容整理成两本书,它们就是《Borland C++ 3.0 & Turbo C++ for Windows 3.0 用户手册》和《Borland C++ 3.0 & Turbo C++ for Windows 3.0 程序员手册》。

《Borland C++ 3.0 & Turbo C++ for Windows 3.0 用户手册》分成四篇,第一篇介绍了 Turbo C++ for Windows 集成环境的用法,讨论了预编译头文件功能和用法。第二篇介绍了 Borland C++ 3.0 集成环境和命令行编译器的用法。在 Borland C++ 3.0 的系列软件中用 Resource Workshop 资源管理程序代替了 Borland C++ 2.0 中的 Whitewater Resource Toolkit,第三篇全面系统地论述 Resource Workshop 的功能和用法,说明了如何创建和管理 Windows 的对话框(dialog box)、菜单(menu)、加速器(热键)(accelerator)、字符串表(string table)、位图(bitmap)、图标(icon)、光标(cursor)、字体(font)、用户自定义和 fcddata 资源。第四篇按字母顺序介绍了 Borland C++ 3.0 新增和修改的 C 库函数,罗列了它们的功能、用法、返回值和示例。

《Borland C++ 3.0 & Turbo C++ for Windows 3.0 程序员手册》分成二篇,第一篇从程序设计的角度说明了 Turbo C++ for Windows 3.0 中的C++的语法,论述了C++的要素和特性,讨论了预处理器的功能和用法,介绍了C++流的概念、作用、内涵和用法,讲述了创建 Windows 应用程序的过程;ObjectWindows 是 Borland C++ 3.0 中用来开发 Windows 应用软件 的类库,利用该类库,用户用很少的代码就可以设计出完美的 Windows 应用程序的用户界面。第二篇的前面部分从介绍开发一个从简单到复杂 Windows 应用程序的步骤入手,全面地介绍了 ObjectWindows 功能和用法;后面部分按字母顺序列出了关于类库的类和流式类、它们的成员和用法的描述;最后还给出了全局变量和辅助类的描述。

Borland C++ 系列软件包含许多实用程序,《Turbo Debugger 3.0 调试手册》所描述的是其中最重要的实用程序 Turbo Debugger。书中介绍了如何使用 Turbo Debugger 菜单,如何在运行时在源程序上检查、监视和修改变量的值,如何进行表达式的求值,如何设置条件和无条件断点,如何进行单步跟踪,如何进行大程序双机虚拟调试、远程调试,如何调试设备驱动程序和 TSR,如何用 TDW 调试用 Borland 系列语言编写 Windows 应用程序等,最后给出了错误信息。

在本套书的编译过程中,程时言、黄磊光、吕良双、占卫兵、汪文、李兵、章忆文给予了极大的支持。在此一并致谢。

由于编译者水平有限,加上时间仓促,书中难免有缺点和错误,欢迎广大读者给予批评和指正。

编译者

1992 年于北航计算中心

目 录

前 言

第一篇 Turbo C++ for Windows 程序员手册

概 述

0.1 内容简介	(3)
0.2 形式化定义的介绍	(3)
0.2.1 语法和术语	(4)

第一章 词法元素

1.1 空白	(5)
1.1.1 行分隔符	(5)
1.1.2 注释	(5)
1.1.2.1 C 注释	(6)
1.1.2.2 嵌套注释	(6)
1.1.2.3 C++ 注释	(6)
1.1.2.4 注释分隔符与空白	(6)
1.2 词法符号	(7)
1.2.1 关键字	(7)
1.2.2 标识符	(8)
1.2.2.1 命名与长度限制	(8)
1.2.2.2 标识符与字母大小写	(9)
1.2.2.3 唯一性和作用域	(9)
1.2.3 常量	(9)
1.2.3.1 整常量	(11)
1.2.3.2 字符常量	(12)
1.2.3.3 浮点常量	(13)
1.2.3.4 串文字量	(15)
1.2.3.5 常量与内部表示	(15)
1.2.3.6 常量表达式	(17)
1.2.4 标点符号(也称隔离符)	(17)
1.2.4.1 中括号	(17)
1.2.4.2 括号	(17)
1.2.4.3 大括号	(18)
1.2.4.4 逗号	(18)

1.2.4.5	分号	(18)
1.2.4.6	冒号	(18)
1.2.4.7	省略号	(19)
1.2.4.8	星号	(19)
1.2.4.9	等号(初始值)	(19)
1.2.4.10	#号(预处理指令)	(19)

第二章 语言结构

2.1	说明	(20)
2.1.1	对象(object)	(20)
2.1.2	左值	(20)
2.1.2.1	右值	(21)
2.1.3	类型和存储类	(21)
2.1.4	作用域	(21)
2.1.4.1	块作用域	(21)
2.1.4.2	函数作用域	(21)
2.1.4.3	函数原型作用域	(21)
2.1.4.4	文件作用域	(21)
2.1.4.5	类作用域(仅限于 C++)	(21)
2.1.4.6	作用域和名字空间	(21)
2.1.5	可见性	(22)
2.1.6	生存期	(22)
2.1.6.1	静态生存期	(22)
2.1.6.2	局部生存期	(23)
2.1.6.3	动态生存期	(23)
2.1.7	编译单元	(23)
2.1.8	连接	(23)
2.1.8.1	名字重构	(24)
2.2	说明的语法	(24)
2.2.1	暂时定义	(25)
2.2.2	可能的说明	(25)
2.2.3	外部说明和定义	(29)
2.2.4	类型指明符	(31)
2.2.5	类型分类	(31)
2.2.5.1	类型 void	(32)
2.2.6	基本类型	(32)
2.2.6.1	整型	(32)
2.2.6.2	浮点型	(33)
2.2.6.3	标准转换	(33)
2.2.6.4	特殊的 char、int 和 enum 转换	(33)
2.2.7	初始化	(34)

2.2.7.1	数组、结构和联合	(35)
2.2.8	简单说明	(35)
2.2.9	存储类指明符	(36)
2.2.9.1	存储类指明符 auto 的使用	(36)
2.2.9.2	存储类指明符 extern 的使用	(36)
2.2.9.3	存储类指明符 register 使用	(36)
2.2.9.4	存储类指明符 static 的使用	(36)
2.2.9.5	存储类指明符 typedef 的使用	(36)
2.2.10	修饰符	(37)
2.2.10.1	常量修饰符	(37)
2.2.10.2	中断函数修饰符	(38)
2.2.10.3	volatile 修饰符	(38)
2.2.10.4	cdecl 与 pascal 修饰符	(39)
2.2.10.5	指针修饰符	(40)
2.2.10.6	函数类型修饰符	(40)
2.2.11	复杂说明与说明符	(40)
2.3	指针	(41)
2.3.1	指向对象的指针	(41)
2.3.2	指向函数的指针	(42)
2.3.3	指针说明	(42)
2.3.4	指针与常量	(43)
2.3.5	指针算术运算	(44)
2.3.6	指针转换	(44)
2.3.7	C++ 引用说明	(44)
2.4	数组	(44)
2.5	函数	(45)
2.5.1	说明与定义	(45)
2.5.2	说明与原型	(45)
2.5.3	定义	(46)
2.5.4	形参说明	(47)
2.5.5	函数调用和参数转换	(47)
2.6	结构	(48)
2.6.1	无标记结构和 typedef	(48)
2.6.2	结构成员说明	(49)
2.6.3	结构与函数	(49)
2.6.4	结构成员存取	(49)
2.6.5	结构字对齐	(50)
2.6.6	结构名字空间	(51)
2.6.7	不完整说明	(51)
2.6.8	位域	(51)

2.7 联合.....	(52)
2.7.1 无名联合(仅 C++ 专用).....	(53)
2.7.2 联合说明.....	(53)
2.8 枚举.....	(53)
2.9 表达式.....	(55)
2.9.1 表达式与 C++	(59)
2.9.2 求值次序.....	(59)
2.9.3 求值和溢出.....	(59)
2.10 操作符语义	(60)
2.11 操作符描述	(60)
2.11.1 单目操作符	(61)
2.11.2 双目操作符	(61)
2.11.2.1 加法类操作符	(61)
2.11.2.2 乘法类操作符	(61)
2.11.2.3 移位操作符	(61)
2.11.2.4 按位操作符	(61)
2.11.2.5 逻辑操作符	(61)
2.11.2.6 赋值操作符	(61)
2.11.2.7 关系操作符	(62)
2.11.2.8 等操作符	(62)
2.11.2.9 成员选择符	(62)
2.11.2.10 类成员操作符.....	(62)
2.11.2.11 条件操作符.....	(62)
2.11.2.12 逗号操作符.....	(62)
2.11.3 后缀和前缀操作符	(62)
2.11.3.1 数组下标操作[]	(62)
2.11.3.2 函数调用操作符	(62)
2.11.3.3 结构/联合成员操作符.....	(63)
2.11.3.4 结构/联合操作符—>.....	(63)
2.11.3.5 后缀量操作符++	(63)
2.11.3.6 后减量操作符——	(63)
2.11.4 增量和减量操作符	(63)
2.11.4.1 前增量操作符++	(63)
2.11.4.2 前减量操作符——	(63)
2.11.5 单目操作符	(63)
2.11.5.1 取地址操作符&	(64)
2.11.5.2 间接引用操作符*	(64)
2.11.5.3 单目加操作+	(64)
2.11.5.4 单目减操作符—	(64)
2.11.5.5 单目补操作符~	(64)

2.11.5.6	逻辑非操作符!	(65)
2.11.6	sizeof 操作符	(65)
2.11.7	乘法类操作符	(65)
2.11.8	加法类操作符	(66)
2.11.8.1	加操作符+	(66)
2.11.8.2	减操作符-	(66)
2.11.9	按位移位操作符	(66)
2.11.9.1	按位移位操作符<<和>>	(66)
2.11.10	关系操作符	(67)
2.11.10.1	小于操作符<	(67)
2.11.10.2	大于操作符>	(67)
2.11.10.3	小于等于操作符<=	(67)
2.11.10.4	大于等于操作符>=	(67)
2.11.11	相等操作符	(67)
2.11.11.1	等于操作符==	(68)
2.11.11.2	不等于操作符!=	(68)
2.11.12	按位与操作符&	(68)
2.11.13	按位异或操作符^	(69)
2.11.14	按位或操作符	(69)
2.11.15	逻辑与操作符&&	(69)
2.11.16	逻辑或操作符	(69)
2.11.17	条件操作符?:	(69)
2.11.18	赋值操作符	(70)
2.11.18.1	简单赋值操作符	(70)
2.11.18.2	复合赋值操作符	(70)
2.11.19	逗号操作符	(70)
2.11.20	C++操作符	(71)
2.12	语句	(71)
2.12.1	块(Blocks)	(73)
2.12.2	标号语句	(73)
2.12.3	表达式语句	(73)
2.12.4	选择语句	(73)
2.12.4.1	if 语句	(73)
2.12.4.2	switch 语句	(74)
2.12.5	循环语句	(74)
2.12.5.1	while 语句	(74)
2.12.5.2	do while 语句	(75)
2.12.5.3	for 语句	(75)
2.12.6	跳转语句	(75)
2.12.6.1	break 语句	(76)

2.12.6.2	continue 语句	(76)
2.12.6.3	goto 语句	(76)
2.12.6.4	return 语句	(76)
第三章 C++ 专用部分		
3.1	引用	(77)
3.1.1	简单引用	(77)
3.1.2	引用参数	(77)
3.1.2.1	实现 1	(78)
3.1.2.2	实现 2	(78)
3.1.2.3	实现 3	(78)
3.2	作用域存取操作符	(79)
3.3	new 和 delete 操作符	(79)
3.3.1	错误处理	(80)
3.3.2	关于数组的 new 操作符	(80)
3.3.3	关于数组的 delete 操作符	(80)
3.3.4	:: operator new	(80)
3.3.5	带有 new 操作的初始值	(80)
3.4	类	(81)
3.4.1	类名	(81)
3.4.2	类类型	(81)
3.4.3	类名作用域	(81)
3.4.4	类对象	(82)
3.4.5	类成员表	(82)
3.4.6	成员函数	(82)
3.4.7	关键字 this	(82)
3.4.8	内部函数	(83)
3.4.9	静态成员	(83)
3.4.10	成员作用域	(84)
3.4.10.1	嵌套类型	(85)
3.4.10.2	成员存取控制	(86)
3.4.11	基类和派生类存取	(87)
3.5	虚基类	(88)
3.6	类的友元	(89)
3.7	构造函数(constructor)和析构函数(destructor)	(90)
3.8	构造函数(constructor)	(91)
3.8.1	缺省构造函数	(92)
3.8.2	拷贝构造函数	(92)
3.8.3	构造函数的重载	(92)
3.8.4	构造函数的调用次序	(93)
3.8.5	类初始化	(94)

3.9	析构函数(destructor)	(96)
3.9.1	析构函数的调用	(97)
3.9.2	atexit、#pragma exit 和析构函数	(97)
3.9.3	exit 和析构函数	(97)
3.9.4	abort 和析构函数	(97)
3.9.5	虚析构函数	(98)
3.10	重载操作符	(99)
3.11	操作符函数	(100)
3.11.1	重载操作符和继承	(100)
3.11.2	new 和 delete 的重载	(100)
3.11.3	单目操作符的重载	(101)
3.11.4	重载二目操作符	(102)
3.11.5	赋值操作符=的重载	(102)
3.11.6	重载调用操作符()	(102)
3.11.7	重载下标操作符[]	(102)
3.11.8	重载类成员存取操作符->	(102)
3.12	虚函数	(103)
3.13	抽象类	(104)
3.14	C++作用域	(105)
3.14.1	类作用域	(105)
3.14.2	隐藏	(105)
3.14.3	C++作用域规则概括	(105)
3.15	模板(templates)	(106)
3.15.1	函数模板(Function templates)	(106)
3.15.1.1	重设模板函数	(107)
3.15.1.2	显式和隐式的模板函数	(108)
3.15.2	类模板	(109)
3.15.2.1	参数	(110)
3.15.2.2	尖括号	(110)
3.15.2.3	类型模板表	(110)
3.15.2.4	消除指针	(111)

第四章 预处理程序

4.1	空指令#	(114)
4.2	#define 和 #undef 指令	(114)
4.2.1	简单#define 宏	(114)
4.2.2	#undef 指令	(115)
4.2.3	定义选择项	(115)
4.2.4	关键字和保护字	(115)
4.2.5	带参量的宏	(116)
4.3	文件包含指令#include	(118)

4.3.1	用<头文件名>的头文件搜索	(118)
4.3.2	用“头文件名”的头文件搜索	(118)
4.4	条件编译	(119)
4.4.1	#if, #elif, #else 和 #endif 条件指令	(119)
4.4.1.1	defined 运算符	(119)
4.4.2	#ifdef 和 #ifndef 条件指令	(120)
4.5	#line 行控制指令	(120)
4.6	#error 指令	(121)
4.7	#pragma 指令	(121)
4.7.1	#pragma argsused	(122)
4.7.2	#pragma exit 和 #pragma startup	(122)
4.7.3	#pragma hdrfile	(123)
4.7.4	#pragma hdrstop	(123)
4.7.5	#pragma saveregs	(123)
4.8	预定义的宏	(123)
4.8.1	__CDECL__	(123)
4.8.2	__cplusplus	(123)
4.8.3	__DATE__	(123)
4.8.4	__DLL__	(124)
4.8.5	__FILE__	(124)
4.8.6	__LINE__	(124)
4.8.7	__MSDOS__	(124)
4.8.8	__PASCAL__	(124)
4.8.9	__STDC__	(124)
4.8.10	__TCPLUSPLUS__	(124)
4.8.11	__TEMPLATES__	(124)
4.8.12	__TIME__	(124)
4.8.13	__TURBOC__	(124)
4.8.14	__Windows	(124)

第五章 使用C++流

5.1	什么是流	(125)
5.2	iostream 库	(125)
5.2.1	streambuf 类	(125)
5.2.2	.ios 类	(126)
5.3	输出	(127)
5.3.1	基本类型	(127)
5.3.2	输出格式	(127)
5.3.3	操纵符	(128)
5.3.4	填充或补空	(129)
5.4	输入	(130)

5.5	用户定义类型的 I/O	(130)
5.6	简单文件 I/O	(131)
5.7	串流处理	(132)
5.8	流类参考	(133)
5.8.1	filebuf <fstream.h>	(134)
5.8.1.1	成员函数	(134)
5.8.2	fstream <fstream.h>	(135)
5.8.2.1	成员函数	(135)
5.8.3	fstreambase <fstream.h>	(135)
5.8.3.1	成员函数	(135)
5.8.4	ifstream <fstream.h>	(136)
5.8.4.1	成员函数	(136)
5.8.5	ios <iostream.h>	(136)
5.8.5.1	数据成员	(136)
5.8.5.2	成员函数	(137)
5.8.6	iostream <iostream.h>	(139)
5.8.7	iostream _withassign <iostream.h>	(139)
5.8.7.1	成员函数	(139)
5.8.8	istream <iostream.h>	(139)
5.8.8.1	成员函数	(139)
5.8.9	istream _withassign <iostream.h>	(140)
5.8.9.1	成员函数	(140)
5.8.10	istream <istream.h>	(140)
5.8.11	ofstream <fstream.h>	(141)
5.8.11.1	成员函数	(141)
5.8.12	ostream <iostream.h>	(141)
5.8.12.1	成员函数	(141)
5.8.13	ostream _withassign <iostream.h>	(142)
5.8.13.1	成员函数	(142)
5.8.14	ostrstream <strstream.h>	(142)
5.8.14.1	成员函数	(142)
5.8.15	streambuf <iostream.h>	(142)
5.8.15.1	成员函数	(142)
5.8.16	strstreambase <strstream.h>	(145)
5.8.16.1	成员函数	(145)
5.8.17	strstreambuf <strstream.h>	(145)
5.8.17.1	成员函数	(145)
5.8.18	strstream <strstream.h>	(146)
5.8.18.1	成员函数	(146)

第六章 数学处理

6.1 浮点处理	(147)
6.1.1 仿真 80x87 芯片	(147)
6.1.2 使用 80x87 代码	(147)
6.1.3 无浮点代码	(147)
6.1.4 快速浮点选择	(147)
6.1.5 寄存器和 80x87	(147)
6.1.6 禁止浮点异常	(148)
6.2 使用复数	(148)
6.2.1 BCD 数学库的用法	(149)
6.2.1.1 转换 BCD 数	(150)
6.2.1.2 二进制位数	(150)

第七章 BASM 和内部汇编

7.1 内部汇编语言	(151)
7.1.1 BASM	(151)
7.1.2 内部语法	(151)
7.1.3 操作码	(152)
7.1.3.1 串指令	(153)
7.1.3.2 前缀	(154)
7.1.3.3 跳转移指令	(154)
7.1.4 汇编指令	(154)
7.1.5 内部汇编对数据和函数的引用	(154)
7.1.5.1 内部汇编和寄存器变量	(154)
7.1.5.2 内部汇编、偏移量和长度控制	(155)
7.1.6 使用 C 结构成员	(155)
7.1.7 使用转换指令和标号	(155)
7.2 中断函数	(156)
7.3 使用低级实例	(157)

第八章 建立 Windows 应用程序

8.1 在 IDE 下的编译和连接	(160)
8.1.1 理解资源文件	(160)
8.1.2 理解模块定义文件	(160)
8.1.3 编译和连接 WHELLO	(160)
8.1.3.1 使用项目管理程序	(161)
8.1.3.2 设置编译和连接选择	(161)
8.2 WinMain	(161)
8.3 入口代码和出口代码(Prolog 和 Epilog)	(162)
8.3.1 Windows 所有函数都是可输出的(exportable)	(162)
8.3.2 Windows 显式输出函数	(162)
8.3.3 Windows 灵敏回调	(162)

8.3.4	Windows DLL 所有函数可输出	(163)
8.3.5	Windows DLL 显式输出函数	(163)
8.3.6	关键字 <code>_export</code>	(163)
8.3.7	入口、出口代码和输出综述	(163)
8.4	内存模式	(164)
8.5	模块定义文件	(164)
8.5.1	一个例子	(165)
8.6	Windows 下的连接	(166)
8.6.1	IDE 中的连接	(166)
8.7	动态连接库(DLL)	(166)
8.7.1	在 IDE 里编译和连接一个 DLL	(166)
8.7.1.1	输入库	(166)
8.7.2	创建输入库	(167)
8.7.2.1	选择一个重要的库	(167)
8.7.2.2	建立输入库	(167)
8.7.3	创建 DLL	(167)
8.7.3.1	LibMain 和 WEP	(167)
8.7.3.2	指针和内存	(168)
8.7.3.3	C++ 类和指针	(169)
第九章 错误信息		
9.1	错误信息的类型	(171)
9.1.1	编译时间信息	(171)
9.1.2	Help 编译程序信息	(172)
9.1.2.1	标题号(topic numbers)	(172)
9.1.3	运行时间错误信息	(173)
9.1.4	库管理信息	(173)
9.1.5	连接信息	(173)
9.2	信息解释	(174)
附录 A HC: Windows Help 编译程序		
A.1	建立一个开发系统:开发周期	(216)
A.1.1	Help 怎样向用户提供	(216)
A.1.2	如何出现作者 Help	(217)
A.1.3	程序员如何得到 Help	(218)
A.2	规划 Help 系统	(218)
A.2.1	提出规划	(218)
A.2.1.1	定义对象	(218)
A.2.1.2	规划内容	(219)
A.2.1.3	规划结构	(219)
A.2.1.4	显示上下文有关标题	(220)
A.2.2	决定标题文件结构	(220)