

北京宝兰—英博思信息技术有限公司 推荐用书

BORLAND
INPRISE
核心技术丛书

Borland C++ Builder: The Complete Reference
Borland C++ Builder: The Complete Reference
Borland C++ Builder: The Complete Reference
Borland C++ Builder: The Complete Reference
Borland C++ Builder: The Complete Reference
Borland C++ Builder: The Complete Reference

Borland C++ Builder: The Complete Reference

(美) Herbert Schildt Greg Guntle 著 周海斌 王安鹏 牛韬 黄晓东 等译

C++ Builder

技术大全



机械工业出版社
China Machine Press



Education

Borland/Inprise核心技术丛书

C++Builder技术大全

(美) Herbert Schildt 著
Greg Guntle

周海斌 王安鹏 牛 韬 黄晓东 等译
前导工作室 审校



机械工业出版社
China Machine Press

本书涵盖了Borland C++的基本编程知识和技巧。本书内容共分为四个部分,第一部分介绍了C++的基础知识,包括控制语句、运算符、预处理器指示符和数据类型;第二部分主要讲述C++ Builder函数库;第三部分主要讲述类和对象、构造函数、析构函数、例外处理、模板等面向对象编程方面的知识和技巧;第四部分详细说明了C++ Builder集成开发环境(IDE),并解释了如何创建、编译和运行应用程序。

本书编排独特、阅读方便、覆盖面广,众多的示例贯穿全书,使读者能学以致用。

Herbert Schildt and Greg Guntle: Borland C++Builder: The Complete Reference (ISBN 0-07-212778-3).

Copyright © 2001 by the McGraw-Hill Companies, Inc.

Authorized translation from the English language edition published by McGraw-Hill, Inc.

All rights reserved. For sale in the People's Republic of China.

本书中文简体字版由机械工业出版社和美国麦格劳-希尔国际公司合作出版。未经出版者书面许可,不得以任何方式复制或抄袭本书内容。

版权所有,侵权必究。

本书版权登记号:图字:01-2001-3930

图书在版编目(CIP)数据

C++Builder技术大全/(美)斯奇得(Schildt, H.), (美)金托(Guntle, G.)著;周海斌等译. -北京:机械工业出版社, 2002. 2

(Borland/Inprise核心技术丛书)

书名原文: Borland C++Builder: The Complete Reference

ISBN 7-111-09691-6

I. C… II. ①斯… ②金… ③周… III. C语言 - 程序设计 IV. TP312

中国版本图书馆CIP数据核字(2001)第096648号

机械工业出版社(北京市西城区百万庄大街22号 邮政编码 100037)

责任编辑:曾朝阳 张鸿斌

北京市密云县印刷厂印刷·新华书店北京发行所发行

2002年2月第1版第1次印刷

787mm × 1092mm 1/16 · 44.25印张

印数:0 001 - 4 000册

定价:68.00元

凡购本书,如有倒页、脱页、缺页,由本社发行部调换

译者序

C和C++语言通常被认为是一种“中级计算机语言”，这种称谓说明C语言是一种功能强大的语言；同时还说明，一方面它并不像某些高级语言那样难以使用，另一方面也不像汇编语言那样容易出现错误。它包含了高级语言的元素和汇编语言的内容。因此，作为程序员的一种基本语言，C语言的应用特别广泛。

这本书是有关Borland C++Builder的最全面的指南。它的主旨就是要帮助你简单快速地建立功能强大的应用程序。

本书是一本极好的参考手册，它按照主题进行组织，共分为四个部分。第一部分覆盖了C++的基础知识，包括控制语句、运算符、预处理器指示符和数据类型；第二部分细化了扩展的C++Builder函数库；第三部分带你探索面向对象编程（OOP），学习类和对象、构造函数、析构函数、例外处理、模板等等，其中还包括标准模板库（STL）的细节问题，这是C++最强大、最先进的特性之一；第四部分介绍C++Builder集成开发环境（IDE），并解释了如何创建、编译和运行应用程序，以及如何使用调试器修补bug。众多的示例贯穿全书，这是本书的特色之一，它能够给编程带来很大的帮助。

本书对于C++程序员来说，是一本不得不看的参考书。作为本书的翻译者，拿到这本书后爱不释手，并以最大的努力完成了本书的翻译工作。希望我们的努力能够给你带来一些帮助。

本书由中国互动出版网（<http://www.china-pub.com>）组织翻译，参与翻译的人员有周海斌、王安鹏、牛韬、黄晓东、姚继锋、安智平、刘伟娜、祝汉寿、王镭、黄建文、周永奎、林艳喜、徐继伟、王顺、谢君英、李挺文、陆绍飞、周波、邢飞、陈贵敏、李平等。最后由前导工作室统一审校。由于时间仓促，译者较多，译文中难免出现不当或欠妥之处，恳请广大读者批评和指正。

2001年10月

前 言

本书讲述Borland公司的C++Builder。从80年代起，Borland一直在制作精彩绝伦的编译器。C++Builder是他们最强大，也是最独到的编译器。它以其编译的速度和生成代码的高效率而闻名。事实上，C++Builder是一个二合一的编译器：首先它是C编译器（C是在C++开发出来以前的语言）；其次它是C++编译器。C++Builder可以生成Windows 95/98/NT /2000程序，也可以生成Windows下的DOS窗口应用程序。无论从哪个方面讲，它都是能得到的最好的程序开发环境之一。本书的目的是帮助你最大程度地掌握它。

关于本书

到现在，我编写关于Borland编译器的书已经有多年了。这本书是我写的Borland C++的最新版本。以前的版本叫《Borland C++: The Complete Reference》。尽管如此，本书还是独一无二的，因为它第一次涵盖了新的“Builder”环境。

除了新的“Builder”环境，还有一个非常重要的事件使得本书与以前的版本有很大的不同，即C++ ANSI/ISO标准的采用。这个标准包含了很多新的特性、函数和类，极大地拓展了C++语言的能力。正像人们所期待的，C++Builder完全支持该标准。因此，本书经过了完整的修订，以囊括ANSI/ISO标准C++的全部新特性。比如，现在进一步涵盖了STL、新的I/O类以及string类。关于模板的讲述也扩展了。坦白地说，C++标准化带来的变化在几个章节中是非常明显的。

本书的内容

本书完整地描述了C++Builder开发环境。同时也非常详尽地讨论了C和C++语言以及它们的库。还说明了如何使用IDE和C++Builder提供的应用程序开发工具。其中包含了许多例子程序来帮助说明C++和Builder环境的要素。

本书适用于各种技术水平的程序员。如果你刚刚开始学习编程，这本参考书是任何入门指南的极好伴侣，为你的细节问题提供了答案。如果你是一位熟练的C/C++程序员，这是一本必备的案头参考书。

本书是如何组织的

C++ Builder是一个非常大的主题，因此本书的厚度也很可观。为了把如此庞大的信息安排得井井有条，本书分成四个部分：

第一部分 C++基础：C子集

第二部分 C++Builder函数库

第三部分 C++的详细特性

第四部分 C++Builder集成开发环境

这样组织使得C程序员可以迅速找到与C相关的资料，而同时C++程序员又能及时找到适于C++的资料。另外，如果你是正准备转向C++的C程序员，这种结构可以使你不必再阅读已经了解的大量信息。你可以仅仅把注意力放在书中的C++部分。

Web上的源代码

本书中的所有程序源代码都可以从Osborne的网站www.osborne.com免费获取。

特别感谢

非常感谢Greg Guntle在本书的准备过程中所给予的帮助。由于我的写作日程非常紧凑而且需要非常大的变动，我自己要单枪匹马地在出版商规定的时间内准备这一版是不可能的。为此，我不得不求助于Greg Guntle。他是一位编程专家，在过去曾经作为我的几本书的技术审核给了我帮助。他非常细致地雕琢各个章节，进行必要的修订和更正。他还起草了第28、29和30章。他做了非常好的工作，本书的大部分荣誉应归于Greg。在此，我再一次表示衷心的感谢。

Herbert Schildt

目 录

译者序

前言

第一部分 C++基础：C子集

第1章 C概述2

1.1 C语言的起源2

1.2 一种中级语言2

1.3 一种结构化语言3

1.4 一种程序员的语言4

1.5 解释器与编译器5

1.6 C程序的结构6

1.6.1 库与链接7

1.6.2 单独编译7

1.6.3 C程序的内存映像8

1.7 术语回顾8

第2章 变量、常量、运算符和表达式10

2.1 标识符名10

2.2 数据类型10

2.2.1 类型限定符11

2.2.2 访问限定符12

2.3 变量的声明12

2.3.1 局部变量13

2.3.2 形式参数14

2.3.3 全局变量15

2.4 存储类说明符16

2.4.1 extern16

2.4.2 static变量18

2.4.3 static局部变量18

2.4.4 static全局变量19

2.4.5 register变量20

2.5 赋值语句21

2.5.1 多重赋值21

2.5.2 赋值中的类型转换21

2.5.3 变量的初始化22

2.6 常量23

2.7 运算符24

2.7.1 算术运算符24

2.7.2 增量和减量运算符25

2.7.3 关系运算符与逻辑运算符26

2.7.4 位运算符27

2.7.5 ?运算符30

2.7.6 &与*指针运算符30

2.7.7 编译时运算符sizeof32

2.7.8 逗号运算符32

2.7.9 点号(.)与箭头(->)运算符33

2.7.10 []与()运算符33

2.7.11 优先级说明33

2.8 表达式34

2.8.1 表达式中的类型转换34

2.8.2 强制类型转换35

2.8.3 空白符和圆括号36

2.8.4 C语言的简写36

第3章 程序控制语句37

3.1 True与False37

3.2 选择语句37

3.3 if语句37

3.3.1 嵌套if39

3.3.2 if-else-if阶梯39

3.3.3 ? 选择符40

3.4 switch42

3.5 循环语句45

3.6 for循环45

3.6.1 for循环变量46

3.6.2 无限循环48

3.6.3 循环体为空的for语句	48	5.5 多维数组	88
3.7 while循环	49	5.6 索引指针	88
3.8 do-while循环	50	5.7 数组的分配	90
3.9 跳转语句	51	5.8 数组初始化	91
3.10 break	51	5.9 一个例子: Tic-Tac-Toe	94
3.11 exit()	53	第6章 指针	97
3.12 continue	54	6.1 指针就是地址	97
3.13 标号与goto语句	55	6.2 指针变量	97
3.14 表达式语句	55	6.3 指针运算符	98
3.15 块语句	56	6.4 指针表达式	99
第4章 函数	57	6.4.1 指针赋值	99
4.1 函数的一般形式	57	6.4.2 指针算术运算	100
4.2 return语句	57	6.4.3 指针比较	101
4.2.1 从函数中返回	57	6.5 动态内存分配与指针	102
4.2.2 返回值	58	6.6 理解const指针	103
4.2.3 main()函数返回什么	59	6.7 指针与数组	104
4.3 理解函数的作用域	60	6.7.1 字符数组指针	105
4.4 函数参数	60	6.7.2 指针数组	107
4.4.1 值调用与引用调用	60	6.8 指针的指针: 多重间接访问	108
4.4.2 建立引用调用	61	6.9 初始化指针	109
4.4.3 使用数组调用函数	62	6.10 函数指针	110
4.5 argc与argv——main()函数的参数	66	6.11 指针带来的问题	112
4.6 函数原型	70	第7章 结构、联合与用户自定义类型	114
4.7 旧式与新式的参数声明	72	7.1 结构	114
4.8 “隐含整型”规则	73	7.1.1 访问结构成员	116
4.9 声明可变长度参数列表	74	7.1.2 结构赋值	116
4.10 返回指针	74	7.2 结构数组	117
4.11 递归	75	7.3 传递结构给函数	122
4.12 函数指针	76	7.3.1 传递结构成员给函数	123
4.13 实现的问题	78	7.3.2 传递整个结构给函数	123
4.13.1 参数与通用函数	79	7.4 结构指针	124
4.13.2 效率	79	7.4.1 声明结构指针	124
第5章 数组	80	7.4.2 使用结构指针	124
5.1 一维数组	80	7.5 结构内的数组与结构	127
5.2 生成数组指针	81	7.6 位域	128
5.3 向函数传递一维数组	81	7.7 联合	129
5.4 二维数组	84	7.8 枚举	131

7.9 C与C++之间的一个重要区别	133
7.10 使用sizeof以保证可移植性	133
7.11 typedef	134
第8章 输入、输出、流与文件	136
8.1 C与C++ I/O	136
8.2 流与文件	136
8.2.1 流	137
8.2.2 文件	137
8.3 控制台I/O	138
8.3.1 读写字符	138
8.3.2 读写字符串: gets()与puts()	140
8.4 格式化控制台I/O	141
8.4.1 printf()	141
8.4.2 scanf()	147
8.5 C文件系统	152
8.5.1 文件指针	153
8.5.2 打开一个文件	153
8.5.3 写入一个字符	154
8.5.4 读出一个字符	155
8.5.5 关闭一个文件	155
8.5.6 使用fopen()、getc()、putc()和fclose()	155
8.5.7 使用feof()	157
8.5.8 对字符串操作: fgets()与fputs()	158
8.5.9 fread()与fwrite()	158
8.5.10 fseek()与随机访问I/O	160
8.5.11 fprintf()与fscanf()	163
8.5.12 清除文件	163
8.5.13 ferror()与rewind()	163
8.6 与控制台的连接	164
第9章 预处理器与注释	166
9.1 #define	166
9.2 #error	169
9.3 #include	169
9.4 条件编译指示符	170
9.4.1 #if、#else、#elif与#endif	170
9.4.2 #ifdef与#ifndef	172
9.5 #undef	173

9.6 使用defined	173
9.7 #line	174
9.8 #pragma	174
9.9 #	177
9.10 #import	177
9.11 #与## 预处理运算符	177
9.12 预定义宏名	178
9.13 注释	180

第二部分 C++Builder函数库

第10章 链接、库和头文件	181
10.1 链接器	181
10.2 库文件和目标文件	182
10.3 标准库和C++ Builder 的扩展	182
10.4 头文件	183
第11章 I/O函数	185
11.1 int access (const char *filename, int mode)	185
11.2 int chmod (const char *filename, int mode)	186
11.3 int chsize (int handle, long size)	186
11.4 void clearerr FILE *stream	187
11.5 int close (int fd) int rtl_close (int fd)	188
11.6 int_creat (const char *filename, int pmode) int_rtl_creat (const char *filename, int attrib) int creat_new (const char *filename, int attrib) int creattemp (char *filename, int attrib)	189
11.7 int dup (int handle) int dup2 (int old_handle, int new_handle)	190
11.8 int eof (int fd)	191
11.9 int fclose (FILE *stream) int _fcloseall (void)	192
11.10 FILE *fdopen (int handle, char *mode)	192
11.11 int feof (FILE *stream)	193

11.12	int ferror (FILE *stream)	193	11.36	int getch (void)	
11.13	int fflush (FILE *stream)	194		int getche (void)	209
11.14	int fgetc (FILE *stream)	194	11.37	int getchar (void)	209
11.15	int fgetchar (void)	195	11.38	char *gets (char *str)	210
11.16	int *fgetpos (FILE *stream, fpos_t *pos)	195	11.39	int getw (FILE *stream)	211
11.17	char *fgets (char *str, int num, FILE *stream)	196	11.40	int isatty (int handle)	212
11.18	long filelength (int handle)	197	11.41	int lock (int handle, long offset, long length)	212
11.19	int fileno (FILE *stream)	197	11.42	int locking (int handle, int mode, long length)	212
11.20	int _flushall(void)	198	11.43	long lseek (int handle, long offset, int origin)	213
11.21	FILE *fopen (const char *fname, const char *mode)	198	11.44	int open (const char *filename, int accesss, unsigned mode) int _rtl_open (const char *filename, int access)	215
11.22	int fprintf (FILE *stream, const char *format, arg-list)	200	11.45	void perror (const char *str)	216
11.23	int fputc (int ch, FILE *stream)	200	11.46	int printf (const char *format, arg-list)	217
11.24	int fputchar (int ch)	201	11.47	int putc (int ch, FILE *stream)	219
11.25	int fputs (const char *str, FILE *stream)	201	11.48	int putch (int ch)	219
11.26	size_t fread (void *buf, size_t size, size_t count, FILE *stream)	202	11.49	int putchar (int ch)	219
11.27	FILE *freopen (const char *fname, const char *mode, FILE *stream)	202	11.50	int puts (const char *str)	220
11.28	int fscanf (FILE *stream, const char *format, arg-list)	203	11.51	int putw (int i, FILE *stream)	220
11.29	int fseek (FILE *stream, long offset, int origin)	204	11.52	int read (int fd, void *buf, unsigned count) int _rtl_read (int fd, void *buf, unsigned count)	220
11.30	int fsetpos (FILE *stream, const fpos_t *pos)	205	11.53	int remove (const char *fname)	221
11.31	FILE *_fsopen (const char *fname, const char *mode, int shflg)	206	11.54	int rename (const char *oldfname, const char *newfname)	222
11.32	int fstat (int handle, struct stat *statbuf)	206	11.55	void rewind (FILE *stream)	223
11.33	long ftell (FILE *stream)	207	11.56	int _rtl_chmod (const char *filename, int get_set, int attrib)	223
11.34	size_t fwrite (const void *buf, size_t size, size_t count, FILE *stream)	207	11.57	int scanf (const char *format, arg-list)	224
11.35	int getc (FILE *stream)	208	11.58	void setbuf (FILE *stream, char	

*buf)	226	12.2 int isalpha(int ch)	239
11.59 int setmode (int handle, int mode)	227	12.3 int isascii(int ch)	239
11.60 int setvbuf (FILE *stream, char *buf, int mode, size_t size)	227	12.4 int iscntrl(int ch)	240
11.61 int sopen (const char *filename, int access, int shflag, int mode)	228	12.5 int isdigit(int ch)	240
11.62 int sprintf (char *buf, const char *format, arg-list)	229	12.6 int isgraph(int ch)	241
11.63 int sscanf (char *buf, const char *format, arg-list)	229	12.7 int islower(int ch)	242
11.64 int stat (char *filename, struct stat *statbuf)	230	12.8 int isprint(int ch)	242
11.65 long tell (int fd)	231	12.9 int ispunct(int ch)	243
11.66 FILE *tmpfile (void)	231	12.10 int isspace(int ch)	243
11.67 char *tmpnam (char *name)	232	12.11 int isupper(ch)	244
11.68 int ungetc (int ch, FILE *stream)	232	12.12 int isxdigit(int ch)	244
11.69 int ungetch (int ch)	233	12.13 void *memcpy(void *dest, const void *source, int ch, size_t count)	245
11.70 int unlink (const char *fname)	234	12.14 void *memchr(const void *buffer, int ch, size_t count)	245
11.71 int unlock (int handle, long offset, long length)	234	12.15 int memcmp(const void *buf1, const void *buf2, size_t count) int memicmp(const void *buf1, const void *buf2, size_t count)	246
11.72 int vprintf (const char *format, va_list arg_ptr) int vfprintf (FILE *stream, const char *format, va_list arg_ptr) int vsprintf (char *buf, const char *format, va_list arg_ptr)	235	12.16 void *memcpy(void *dest, const void *source, size_t count)	247
11.73 int vscanf (const char *format, va_list arg_ptr) int vscanf (FILE *stream, const char *format, va_list arg_ptr) int vsscanf (const char *buf, const char *format, va_list arg_ptr)	235	12.17 void *memmove(void *dest, const void *source, size_t count)	248
11.74 int write (int handle, void *buf, int count) int _rtl_write (int handle, void *buf, int count)	236	12.18 void *memset(void *buf, int ch, size_t count)	248
第12章 字符串、内存和字符函数	238	12.19 void movmem(const void *source, void *dest, unsigned count)	248
12.1 int isalnum(int ch)	238	12.20 void setmem(void *buf, unsigned count, char ch)	249
		12.21 char *strcpy(char *str1, const char *str2)	249
		12.22 char *strcat(char *str1, const char *str2)	249
		12.23 char *strchr(const char *str, int ch)	250
		12.24 int strcmp(const char *str1, const char *str2)	251

12.25	int strcoll(const char *str1, const char *str2)	251	*str2)	260
12.26	char *strcpy(char *str1, const char *str2)	251	12.45	char *strupr(char *str)
12.27	size_t strcspn(const char *str1, const char *str2)	252	12.46	size_t strxfrm(char *dest, const char *source, size_t count)
12.28	char *strdup(const char *str)	252	12.47	int tolower(int ch) int _tolower(int ch)
12.29	char *_strerror(const char *str)	253	12.48	int toupper(int ch) int _toupper (int ch)
12.30	char *strerror(int num)	253	第13章	数学函数
12.31	int stricmp(const char *str1, const char *str2) int strcmpi(const char *str1, const char *str2)	253	13.1	double acos(double arg) long double acosl(long double arg)
12.32	size_t strlen(const char *str)	254	13.2	double asin(double arg) long double asinl(long double arg)
12.33	char *strlwr(char *str)	254	13.3	double atan(double arg) long double atanl(long double arg)
12.34	char *strncat(char *str1, const char *str2, size_t count)	255	13.4	double atan2(double y, double x) long double atan2l(long double y, long double x)
12.35	int strncmp(const char *str1, const char *str2, size_t count) int strnicmp(const char *str1, const char *str2, size_t count) int strncmpi(const char *str1, const char *str2, size_t count)	256	13.5	double cabs(struct complex znum) long double cabs1(struct _complex1 znum)
12.36	char *strncpy(char *dest, const char *source, size_t count)	257	13.6	double ceil(double num) long double ceil1(long double num)
12.37	char *strnset(char *str, int ch, size_t count)	257	13.7	double cos(double arg) long double cos1 (long double arg)
12.38	char *strpbrk(const char *str1, const char *str2)	257	13.8	double cosh(double arg), long double cosh1(long double arg)
12.39	char *strrchr(const char *str, int ch)	258	13.9	double exp(double arg), long double expl(long double arg)
12.40	char *strrev(char *str)	259	13.10	double fabs(double num), long double fabsl(long double num)
12.41	char *strset(char *str, int ch)	259	13.11	double floor(double num), long double floor1(long double num)
12.42	size_t strspn(const char *str1, const char *str2)	259	13.12	double fmod(double x, double y), long double fmod1(long double x, long double y)
12.43	char *strstr(const char *str1, const char *str2)	260	13.13	double frexp(double num, int *exp), long double frexpl(long double num,
12.44	char *strtok(char *str1, const char			

int *exp)	270	14.3 char *ctime(const time_t *time)	281
13.14 double hypot(double x, double y) long double hypotl(long double x, long double y)	271	14.4 double difftime(time_t time2, time_t time1)	282
13.15 double ldexp(double num, int exp) long double ldexpl(long double num, int exp)	271	14.5 void disable(void) void _disable(void)	282
13.16 double log(double num) long double logl(long double num)	272	14.6 unsigned _dos_close(int fd)	283
13.17 double log10(double num), long double log10l(long double num)	272	14.7 unsigned _dos_creat(const char *fname, unsigned attr, int *fd) unsigned _dos_creatnew(const char *fname, unsigned attr, int *fd)	283
13.18 int _matherr(struct exception *err), int _matherrl(struct _exceptionl *err)	273	14.8 void _dos_getdate(struct dosdate_t *d), void _dos_gettime(struct dostime_t *t)	284
13.19 double modf(double num, double *i), long double modfl(long double num, long double *i)	274	14.9 unsigned _dos_getdiskfree(unsigned char drive, struct diskfree_t *dfptr)	285
13.20 double poly(double x, int n, double c[]) long double polyl(long double x, int n, long double c[])	274	14.10 void _dos_getdrive(unsigned *drive)	286
13.21 double pow(double base, double exp) long double powl(long double base, long double exp)	275	14.11 unsigned _dos_getfileattr(const char *fname, unsigned *attrib)	286
13.22 double pow10(int n) long double pow10l(int n)	275	14.12 unsigned _dos_getftime(int fd, unsigned *fdate, unsigned *ftime)	287
13.23 double sin(double arg) long double sinl(long double arg)	276	14.13 unsigned _dos_open(const char *fname, unsigned mode, int *fd)	288
13.24 double sinh(double arg) long double sinhl(long double arg)	276	14.14 unsigned _dos_read(int fd, void *buf, unsigned count, unsigned *numread)	289
13.25 double sqrt(double num) long double sqrtl(long double num)	277	14.15 unsigned _dos_setdate(struct dosdate_t *d) unsigned _dos_settime(struct dostime_t *t)	289
13.26 double tan(double arg) long double tanl(long double arg)	277	14.16 void _dos_setdrive(unsigned drive, unsigned *num)	290
13.27 double tanh(double arg) long double tanh1(long double arg)	278	14.17 unsigned _dos_setfileattr(const char *fname, unsigned attrib)	291
第14章 时间、日期以及系统相关函数	279	14.18 unsigned _dos_setftime(int fd, unsigned fdate, unsigned ftime)	291
14.1 char *asctime(const struct tm *ptr)	280	14.19 long dostounix(struct date *d, struct time *t)	293
14.2 clock_t clock(void)	280		

14.20	unsigned _dos_write(int fd, void *buf, unsigned count, unsigned *numwritten)	293	15.3	void free(void *ptr)	307
14.21	void enable(void), void _enable (void)	294	15.4	int heapcheck(void)	307
14.22	void ftime(struct timeb *time)	294	15.5	int heapcheckfree(unsigned fill)	308
14.23	void geninterrupt(int intr)	295	15.6	int heapchecknode(void *ptr)	309
14.24	void getdate(struct date *d), void gettime(struct time *t)	295	15.7	int _heapchk(void)	310
14.25	void getdfree(unsigned char drive, struct dfree *dfptr)	295	15.8	int heapfillfree(unsigned fill)	310
14.26	int getftime(int handle, struct ftime *ftptr)	296	15.9	int _heapmin(void)	310
14.27	struct tm *gmtime(const time_t *time)	297	15.10	int _heapset(unsigned fill)	311
14.28	int kbhit(void)	298	15.11	int heapwalk(struct heapinfo *hinfo), int _rtl_heapwalk (_HEAPINFO *hinfo)	311
14.29	struct tm *localtime(const time_t *time)	298	15.12	void *malloc(size_t size)	313
14.30	time_t mktime(struct tm *p)	299	15.13	void *realloc(void *ptr, size_t newsize)	313
14.31	void setdate(struct date *d), void settime(struct time *t)	299	第16章	目录函数	315
14.32	int setftime(int handle, struct ftime *t)	300	16.1	int chdir(const char *path)	315
14.33	void sleep(unsigned time)	301	16.2	int _chdrive(int drivenum)	315
14.34	int stime(time_t *t)	301	16.3	void closedir(DIR *ptr) DIR *opendir(char *dirname) struct dirent *readdir(DIR *ptr) void rewinddir(DIR *ptr)	315
14.35	char *_strdate(char *buf), char *_strtime(char *buf)	301	16.4	unsigned _dos_findfirst(const char *fname, int attr, struct find_t *ptr) unsigned _dos_findnext (struct find_t *ptr)	316
14.36	size_t strftime(char *str, size_t maxsize, char const *fmt, const struct tm *time)	302	16.5	int findfirst(const char *fname, struct fblk *ptr, int attrib) int findnext(struct fblk *ptr)	317
14.37	time_t time(time_t *time)	302	16.6	void fnmerge(char *path, const char *drive, const char *dir, const char *fname, const char *ext) int fnsplit(const char *path, char *drive, char *dir, char *fname, char *ext)	319
14.38	void tzset(void)	304	16.7	char *_fullpath(char *fpath, const char *rpath, int len)	320
14.39	void unixtodos(long utime, struct date *d, struct time *t)	304	16.8	int getcurdir(int drive, char *dir)	320
第15章	动态分配	305	16.9	char *getcwd(char *dir, int len)	321
15.1	void *alloca(size_t size)	305	16.10	char *_getcwd(int drive, char *path, int len)	322
15.2	void *calloc(size_t num, size_t size)	306			

16.11	int getdisk(void)	322	int execl(char *fname, char *arg0, ...,	
16.12	int _getdrive(void)	323	char *argN, NULL) int execlp(char *fname, char *arg0, ...,	
16.13	void _makepath(char *pname, const char *drive, const char *directory, const char *fname, const char *extension)	323	char *argN, NULL) int execlpe(char *fname, char *arg0, ..., char *argN, NULL, char *envp[])	
16.14	int mkdir(const char *path)	324	int execvp(char *fname, char *arg[])	
16.15	char *mktemp(char *fname)	324	int execv(char *fname, char *arg[], char *envp[])	
16.16	int rmdir(const char *path)	325	int execvp (char *fname, char *arg[])	
16.17	char *searchpath(const char *fname)	325	int execvpe(char *fname, char *arg[], char *envp[])	332
16.18	int setdisk(int drive)	326	17.7 void exit(int status) void _exit (int status)	333
16.19	void _splitpath(const char *fpath, char *drive, char *directory char *fname, char *extension)	326	17.8 int getpid(void)	334
第17章	进程控制函数	328	17.9 int spawnl(int mode, char *fname, char *arg0, ..., char *argN, NULL)	
17.1	void abort(void)	328	int spawnle(int mode, char *fname, char *arg0, ..., char *argN, NULL, char *envp[])	
17.2	int atexit(void (*func)(void))	328	int spawnlp(int mode, char *fname, char *arg0, ..., char *argN, NULL)	
17.3	unsigned long _beginthread(void (*func) (void *), unsigned stksize, void *arglist) unsigned long _beginthreadex(void *secattr, unsigned stksize, unsigned (*start) (void *), void *arglist, unsigned create flags, unsigned *threadID) unsigned long _beginthreadNT(void (*func) (void *), unsigned stksize, void *arglist, void *secattr, unsigned create flags, unsigned *threadID);	329	int spawnlpe(int mode, char *fname, char *arg0, ..., char *argN, NULL, char *envp[])	
17.4	void _c_exit(void) void _cexit(void)	331	int spawnnv(int mode, char *fname, char *arg[])	
17.5	void _endthread(void) void _endthreadex(unsigned threadvalue)	331	int spawnve(int mode, char *fname, char *arg[], char *envp[])	
17.6	int execl(char *fname, char *arg0, ..., char *argN, NULL) int execlp(char *fname, char *arg0, ..., char *argN, NULL) int execlpe(char *fname, char *arg0, ..., char *argN, NULL, char *envp[])		int spawnvp(int mode, char *fname, char *arg[])	
			int spawnvpe(int mode, char *fname, char *arg[], char *envp[])	335
			17.10 int wait(int *status)	337
			第18章 基于屏幕的文本函数	338
			18.1 char *cgets(char *inpstr)	338
			18.2 void clrerr(void) void clrscr(void)	339
			18.3 int cprintf(const char *fmt, ...)	340
			18.4 int cputs(const char *str)	340
			18.5 int cscanf(char *fmt, ...)	341
			18.6 void delline(void)	342
			18.7 int gettext(int left, int top, int right, int bottom, void *buf)	342
			18.8 void gettextinfo(struct text_info *info)	343
			18.9 void gotoxy(int x, int y)	343

18.10	void highvideo(void)	344	char *buf)	357	
18.11	void inline(void)	344	19.15	char *getenv(const char *name)	357
18.12	void lowvideo(void)	345	19.16	char *getpass(const char *str)	358
18.13	int movetext(int left, int top, int right, int bottom, int newleft, int newtop)	345	19.17	unsigned getpid(void)	358
18.14	void normvideo(void)	346	19.18	char *itoa(int num, char *str, int radix)	358
18.15	int puttext(int left, int top, int right, int bottom, void *buf)	346	19.19	long labs(long num)	359
18.16	void textattr(int attr)	346	19.20	ldiv_t ldiv(long numerator, long denominator)	360
18.17	void textbackground(int color)	347	19.21	void *lfind(const void *key, const void *base, size_t *num, size_t size, int (*co- mpare) (const void *, const void *)) void *lsearch(const void *key, void *base, size_t *num, size_t size, int (*compare) (const void *, const void *))	360
18.18	void textcolor(int color)	347	19.22	struct lconv *localeconv(void)	362
18.19	void textmode(int mode)	348	19.23	void longjmp(jmp_buf envbuf, int val)	362
18.20	int wherex(void) int wherey(void)	349	19.24	char *ltoa(long num, char *str, int radix) char *ultoa(unsigned long num, char *str, int radix)	363
18.21	void window(int left, int top, int right, int bottom)	349	19.25	unsigned long _lrotl(unsigned long l, int i) unsigned long _lrotr(unsigned long l, int i)	364
第19章	杂项函数	350	19.26	max(x,y) min(x,y)	365
19.1	int abs(int num)	350	19.27	int mblen(const char *str, size_t size)	365
19.2	void asct(int exp)	351	19.28	size_t mbstowcs(wchar_t *out, const char *in, size_t size)	366
19.3	double atof(const char *str) long double _atold(const char *str)	351	19.29	int mbtowc(wchar_t *out, const char *in, size_t size)	366
19.4	int atoi(const char *str)	352	19.30	int putenv(const char *evar)	366
19.5	long atol(const char *str)	352	19.31	void qsort(void *base, size_t num, size_t size, int (*compare) (const void *, const void *))	367
19.6	void *bsearch(const void *key, const void *base, size_t num, size_t size, int (*compare) (const void *, const void *))	353	19.32	int raise(int signal)	368
19.7	unsigned int _clear87(void)	354	19.33	int rand(void)	369
19.8	unsigned int _control87(unsigned fpword, unsigned fpmask)	355			
19.9	div_t div(int numerator, int denominator)	355			
19.10	char *ecvt(double value, int ndigit, int *dec, int *sign)	356			
19.11	void __emit__(unsigned char arg, ...)	356			
19.12	char *fcvt(double value, int ndigit, int *dec, int *sign)	356			
19.13	void _fpreset(void)	357			
19.14	char *gcvt(double value, int ndigit, char *buf)	357			

19.34	int random(int num)	
	void randomize(void)	369
19.35	unsigned short _rotl(unsigned short val, int num)	
	unsigned short _rotr(unsigned short val, int num)	370
19.36	void _setcursortype(int type)	371
19.37	int setjmp(jmp_buf envbuf)	371
19.38	void _searchenv(const char *fname, const char *ename, char *fpath)	372
19.39	char *setlocale(int type, const char *locale)	372
19.40	void (*set_new_handler(void (*newhand)()))()	373
19.41	void (*signal (int signal, void (*sigfunc) (int func)))(int)	373
19.42	void srand(unsigned seed)	374
19.43	unsigned int _status87(void)	375
19.44	double strtod(const char *start, char **end) long double _strtold(const char *start, char **end)	375
19.45	long strtol(const char *start, char **end, int radix)	
	unsigned long strtoul(const char *start, char **end, int radix)	376
19.46	void swab(char *source, char *dest, int num)	377
19.47	int system(const char *str)	377
19.48	to_ascii (int ch)	378
19.49	unsigned umask(unsigned access)	378
19.50	int utime(char *fname, struct utimbuf *t)	378
19.51	void va_start(va_list argptr, last_parm) void va_end(va_list argptr) type va_arg(va_list argptr, type)	379
19.52	size_t wcstombs(char *out, const wchar_t *in, size_t size)	381

19.53	int wctomb(char *out, wchar_t in)	381
-------	-----------------------------------	-----

第三部分 C++的详细特性

第20章	C++概述	383
20.1	C++的起源	383
20.2	什么是面向对象编程	384
20.2.1	封装	384
20.2.2	多态	385
20.2.3	继承	385
20.3	C++基础	385
20.4	进一步考察头与名空间	388
20.4.1	新式的头	388
20.4.2	名空间	389
20.5	C++类简介	389
20.6	函数重载	392
20.7	运算符重载	395
20.8	继承	395
20.9	构造函数与析构函数	398
20.10	C++关键字	401
20.11	两种新的数据类型	402
第21章	进一步考察类与对象	403
21.1	参数化构造函数	403
21.2	友元函数	407
21.3	缺省函数参数	411
21.4	类与结构是相关的	414
21.5	联合与类是相关的	416
21.6	内联函数	417
21.7	把对象传递给函数	420
21.8	返回对象	421
21.9	对象赋值	422
21.10	对象数组	423
21.10.1	初始化对象数组	424
21.10.2	创建时初始化与未初始化的数组	426
21.11	对象指针	426
第22章	函数与运算符重载	429
22.1	重载构造函数	429
22.2	局部化变量	430