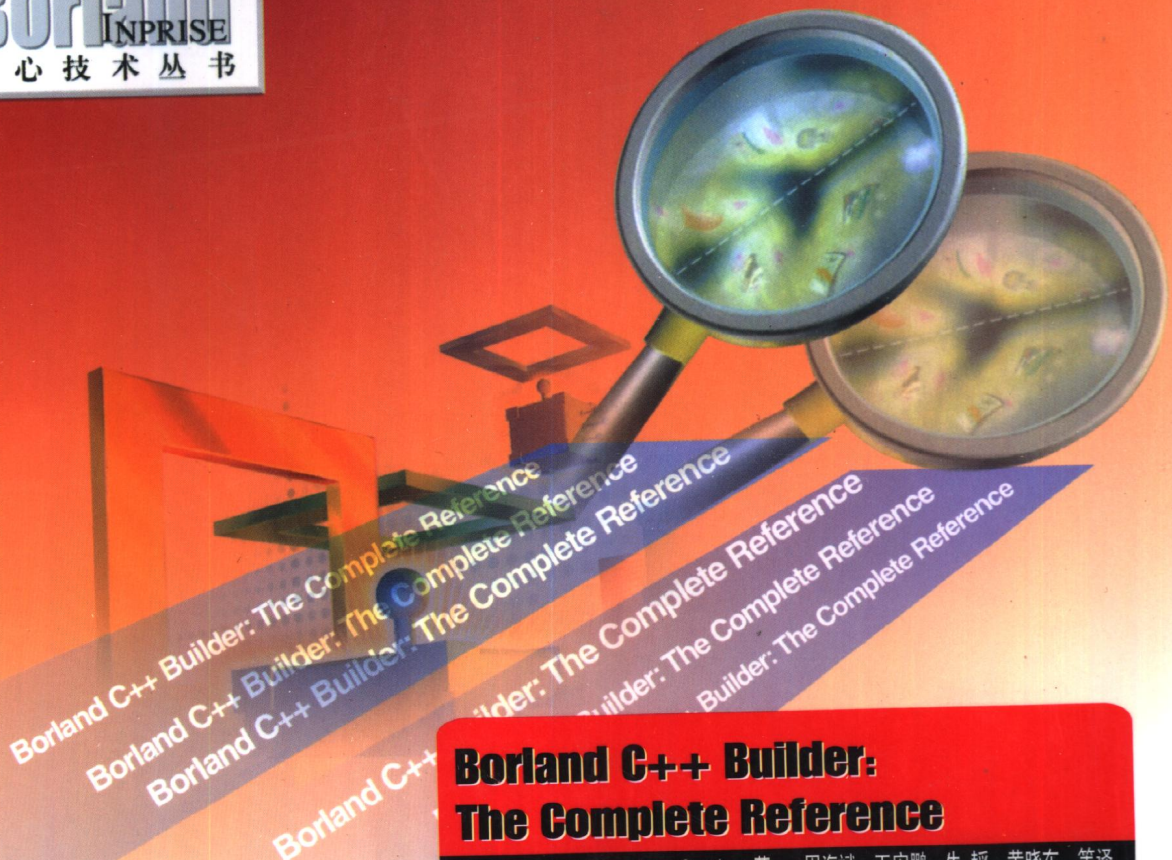




北京宝兰—英博思信息技术有限公司 推荐用书

**Borland**  
INPRISE  
核心技术丛书



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

 McGraw Hill Education

412

TP312C  
581f

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

# 目 录

译者序

前言

## 第一部分 C++基础：C子集

第1章 C概述 .....	2	2.5.2 赋值中的类型转换 .....	21
1.1 C语言的起源 .....	2	2.5.3 变量的初始化 .....	22
1.2 一种中级语言 .....	2	2.6 常量 .....	23
1.3 一种结构化语言 .....	3	2.7 运算符 .....	24
1.4 一种程序员的语言 .....	4	2.7.1 算术运算符 .....	24
1.5 解释器与编译器 .....	5	2.7.2 增量和减量运算符 .....	25
1.6 C程序的结构 .....	6	2.7.3 关系运算符与逻辑运算符 .....	26
1.6.1 库与链接 .....	7	2.7.4 位运算符 .....	27
1.6.2 单独编译 .....	7	2.7.5 ?运算符 .....	30
1.6.3 C程序的内存映像 .....	8	2.7.6 &与*指针运算符 .....	30
1.7 术语回顾 .....	8	2.7.7 编译时运算符sizeof .....	32
第2章 变量、常量、运算符和表达式 .....	10	2.7.8 逗号运算符 .....	32
2.1 标识符名 .....	10	2.7.9 点号(.)与箭头(->)运算符 .....	33
2.2 数据类型 .....	10	2.7.10 []与()运算符 .....	33
2.2.1 类型限定符 .....	11	2.7.11 优先级说明 .....	33
2.2.2 访问限定符 .....	12	2.8 表达式 .....	34
2.3 变量的声明 .....	12	2.8.1 表达式中的类型转换 .....	34
2.3.1 局部变量 .....	13	2.8.2 强制类型转换 .....	35
2.3.2 形式参数 .....	14	2.8.3 空白符和圆括号 .....	36
2.3.3 全局变量 .....	15	2.8.4 C语言的简写 .....	36
2.4 存储类说明符 .....	16	第3章 程序控制语句 .....	37
2.4.1 extern .....	16	3.1 True与False .....	37
2.4.2 static变量 .....	18	3.2 选择语句 .....	37
2.4.3 static局部变量 .....	18	3.3 if语句 .....	37
2.4.4 static全局变量 .....	19	3.3.1 嵌套if .....	39
2.4.5 register变量 .....	20	3.3.2 if-else-if阶梯 .....	39
2.5 赋值语句 .....	21	3.3.3 ?选择符 .....	40
2.5.1 多重赋值 .....	21	3.4 switch .....	42
		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	9.6 使用defined .....	173
7.10 使用sizeof以保证可移植性 .....	133	9.7 #line .....	174
7.11 typedef .....	134	9.8 #pragma .....	174
第8章 输入、输出、流与文件 .....	136	9.9 # .....	177
8.1 C与C++ I/O .....	136	9.10 #import .....	177
8.2 流与文件 .....	136	9.11 #与##预处理运算符 .....	177
8.2.1 流 .....	137	9.12 预定义宏名 .....	178
8.2.2 文件 .....	137	9.13 注释 .....	180
8.3 控制台I/O .....	138		
8.3.1 读写字符 .....	138	<b>第二部分 C++Builder函数库</b>	
8.3.2 读写字符串: gets()与puts() .....	140	第10章 链接、库和头文件 .....	181
8.4 格式化控制台I/O .....	141	10.1 链接器 .....	181
8.4.1 printf() .....	141	10.2 库文件和目标文件 .....	182
8.4.2 scanf() .....	147	10.3 标准库和C++ Builder的扩展 .....	182
8.5 C文件系统 .....	152	10.4 头文件 .....	183
8.5.1 文件指针 .....	153	第11章 I/O函数 .....	185
8.5.2 打开一个文件 .....	153	11.1 int access (const char *filename, int mode) .....	185
8.5.3 写入一个字符 .....	154	11.2 int chmod (const char *filename, int mode) .....	186
8.5.4 读出一个字符 .....	155	11.3 int chsize (int handle, long size) .....	186
8.5.5 关闭一个文件 .....	155	11.4 void clearerr FILE *stream .....	187
8.5.6 使用fopen()、getc()、putc()和fclose() .....	155	11.5 int close (int fd) int rtl close (int fd) .....	188
8.5.7 使用feof() .....	157	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
8.5.8 对字符串操作: fgets()与fputs() .....	158	11.7 int dup (int handle) int dup2 (int old_handle, int new_handle) .....	190
8.5.9 fread()与fwrite() .....	158	11.8 int eof (int fd) .....	191
8.5.10 fseek()与随机访问I/O .....	160	11.9 int fclose (FILE *stream) int _fcloseall (void) .....	192
8.5.11 fprintf()与fscanf() .....	163	11.10 FILE *fdopen (int handle, char *mode) .....	192
8.5.12 清除文件 .....	163	11.11 int feof (FILE *stream) .....	193
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		

- 11.12 int ferror (FILE \*stream) .....193
- 11.13 int fflush (FILE \*stream) .....194
- 11.14 int fgetc (FILE \*stream) .....194
- 11.15 int fgetchar (void) ..... 195
- 11.16 int \*fgetpos (FILE \*stream,  
fpos\_t \*pos) .....195
- 11.17 char \*fgets (char \*str, int num,  
FILE \*stream) .....196
- 11.18 long filelength (int handle) .....197
- 11.19 int fileno (FILE \*stream) .....197
- 11.20 int \_flushall(void) ..... 198
- 11.21 FILE \*fopen (const char \*fname,  
const char \*mode) ..... 198
- 11.22 int fprintf (FILE \*stream, const char  
\*format, arg-list) ..... 200
- 11.23 int fputc (int ch, FILE \*stream) ..... 200
- 11.24 int fputchar (int ch) ..... 201
- 11.25 int fputs (const char \*str,  
FILE \*stream) ..... 201
- 11.26 size\_t fread (void \*buf, size\_t size,  
size\_t count, FILE \*stream) ..... 202
- 11.27 FILE \*freopen (const char \*fname, const  
char \*mode, FILE \*stream) ..... 202
- 11.28 int fscanf (FILE \*stream, const  
char \*format, arg-list) ..... 203
- 11.29 int fseek (FILE \*stream, long offset,  
int origin) ..... 204
- 11.30 int fsetpos (FILE \*stream, const  
fpos\_t \*pos) ..... 205
- 11.31 FILE \*\_fsopen (const char \*fname,  
const char \*mode, int shflg) ..... 206
- 11.32 int fstat (int handle, struct stat  
\*statbuf) ..... 206
- 11.33 long ftell (FILE \*stream) ..... 207
- 11.34 size\_t fwrite (const void \*buf, size\_t  
size, size\_t count, FILE \*stream) ..... 207
- 11.35 int getc (FILE \*stream) ..... 208
- 11.36 int getch (void)  
int getche (void) ..... 209
- 11.37 int getchar (void) ..... 209
- 11.38 char \*gets (char \*str) ..... 210
- 11.39 int getw (FILE \*stream) ..... 211
- 11.40 int isatty (int handle) ..... 212
- 11.41 int lock (int handle, long offset,  
long length) ..... 212
- 11.42 int locking (int handle, int mode,  
long length) ..... 212
- 11.43 long lseek (int handle, long offset,  
int origin) ..... 213
- 11.44 int open (const char \*filename, int  
accesss, unsigned mode)  
int \_rtl\_open (const char \*filename,  
int access) ..... 215
- 11.45 void perror (const char \*str) ..... 216
- 11.46 int printf (const char \*format,  
arg-list) ..... 217
- 11.47 int putc (int ch, FILE \*stream) ..... 219
- 11.48 int putchar (int ch) ..... 219
- 11.49 int putchar (int ch) ..... 219
- 11.50 int puts (const char \*str) ..... 220
- 11.51 int putw (int i, FILE \*stream) ..... 220
- 11.52 int read (int fd, void \*buf,  
unsigned count)  
int \_rtl\_read (int fd, void \*buf, unsigned  
count) ..... 220
- 11.53 int remove (const char \*fname) ..... 221
- 11.54 int rename (const char \*oldfname,  
const char \*newfname) ..... 222
- 11.55 void rewind (FILE \*stream) ..... 223
- 11.56 int \_rtl\_chmod (const char \*filename,  
int get\_set, int attrib) ..... 223
- 11.57 int scanf (const char \*format,  
arg-list) ..... 224
- 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,<br>int mode, size_t size) .....  | 227 | 12.4  | int iscntrl(int ch) .....   | 240 |
| 11.61 | int sopen (const char *filename, int<br>access, int shflag, int mode) .....  | 228 | 12.5  | int isdigit(int ch) .....   | 240 |
| 11.62 | int sprintf (char *buf, const char<br>*format, arg-list) .....   | 229 | 12.6  | int isgraph(int ch) .....   | 241 |
| 11.63 | int sscanf (char *buf, const char<br>*format, arg-list) .....  | 229 | 12.7  | int islower(int ch) .....   | 242 |
| 11.64 | int stat (char *filename, struct stat<br>*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<br>*source, int ch, size_t count) .....   | 245 |
| 11.70 | int unlink (const char *fname) .....   | 234 | 12.14 | void *memchr(const void *buffer, int ch,<br>size_t count) .....   | 245 |
| 11.71 | int unlock (int handle, long<br>offset, long length) .....   | 234 | 12.15 | int memcmp(const void *buf1, const<br>void *buf2, size_t count)<br>int memicmp(const void *buf1, const<br>void *buf2, size_t count) ..... | 246 |
| 11.72 | int vprintf (const char *format,<br>va_list arg_ptr)<br>int vfprintf (FILE *stream, const char<br>*format, va_list arg_ptr)<br>int vsprintf (char *buf, const char *format,<br>va_list arg_ptr) .....    | 235 | 12.16 | void *memcpy(void *dest, const void<br>*source, size_t count) .....   | 247 |
| 11.73 | int vscanf (const char *format, va_list<br>arg_ptr)<br>int vfscanf (FILE *stream, const char<br>*format, va_list arg_ptr)<br>int vsscanf (const char *buf, const char<br>*format, va_list arg_ptr) ..... | 235 | 12.17 | void *memmove(void *dest, const void<br>*source, size_t count) .....  | 248 |
| 11.74 | int write (int handle, void *buf, int count)<br>int _rtl_write (int handle, void *buf, int<br>count) .....   | 236 | 12.18 | void *memset(void *buf, int ch, size_t<br>count) .....  | 248 |
| 第12章  | 字符串、内存和字符函数 .....  | 238 | 12.19 | void movmem(const void *source, void<br>*dest, unsigned count) .....  | 248 |
| 12.1  | int isalnum(int ch) .....  | 238 | 12.20 | void setmem(void *buf, unsigned count,<br>char ch) .....  | 249 |
|       |  |     | 12.21 | char *strcpy(char *str1,<br>const char *str2) .....   | 249 |
|       |  |     | 12.22 | char *strcat(char *str1,<br>const char *str2) .....   | 249 |
|       |  |     | 12.23 | char *strchr(const char *str,<br>int ch) .....  | 250 |
|       |  |     | 12.24 | int strcmp(const char *str1,<br>const char *str2) .....   | 251 |



- 12.25 int strcoll(const char \*str1,  
const char \*str2) .....251
- 12.26 char \*strcpy(char \*str1, const  
char \*str2) .....251
- 12.27 size\_t strcspn(const char \*str1,  
const char \*str2) .....252
- 12.28 char \*strdup(const char \*str) .....252
- 12.29 char \*\_strerror(const char \*str) .....253
- 12.30 char \*strerror(int num) .....253
- 12.31 int stricmp(const char \*str1, const  
char \*str2)  
int strcmpi(const char \*str1, const  
char \*str2) .....253
- 12.32 size\_t strlen(const char \*str) .....254
- 12.33 char \*strlwr(char \*str) .....254
- 12.34 char \*strncat(char \*str1, const char  
\*str2, size\_t count) .....255
- 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
- 12.36 char \*strncpy(char \*dest, const char  
\*source, size\_t count) .....257
- 12.37 char \*strnset(char \*str, int ch, size\_t  
count) .....257
- 12.38 char \*strpbrk(const char \*str1, const  
char \*str2) .....257
- 12.39 char \*strchr(const char \*str, int ch) .....258
- 12.40 char \*strrev(char \*str) .....259
- 12.41 char \*strset(char \*str, int ch) .....259
- 12.42 size\_t strspn(const char \*str1, const  
char \*str2) .....259
- 12.43 char \*strstr(const char \*str1, const  
char \*str2) .....260
- 12.44 char \*strtok(char \*str1, const char  
\*str2) .....260
- 12.45 char \*strupr(char \*str) .....261
- 12.46 size\_t strxfrm(char \*dest, const char  
\*source, size\_t count) .....262
- 12.47 int tolower(int ch)  
int \_tolower(int ch) .....262
- 12.48 int toupper(int ch) int \_toupper  
(int ch) .....262
- 第13章 数学函数 .....264
- 13.1 double acos(double arg) long double  
acosl(long double arg) .....264
- 13.2 double asin(double arg) long double  
asinx(long double arg) .....265
- 13.3 double atan(double arg) long double  
atanl(long double arg) .....265
- 13.4 double atan2(double y, double x)  
long double atan2l(long double y,  
long double x) .....266
- 13.5 double cabs(struct complex znum) long  
double cabsl(struct \_complexl znum) .....267
- 13.6 double ceil(double num) long double  
ceil(long double num) .....267
- 13.7 double cos(double arg) long double cosl  
(long double arg) .....268
- 13.8 double cosh(double arg), long double  
coshl(long double arg) .....268
- 13.9 double exp(double arg), long double  
expl(long double arg) .....269
- 13.10 double fabs(double num), long double  
fabsl(long double num) .....269
- 13.11 double floor(double num), long double  
floorl(long double num) .....270
- 13.12 double fmod(double x, double y),  
long double fmodl(long double x,  
long double y) .....270
- 13.13 double frexp(double num, int \*exp),  
long double frexpl(long double num,

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

- 16.11 int getdisk(void) .....322
- 16.12 int \_getdrive(void) .....323
- 16.13 void \_makepath(char \*pname, const char \*drive, const char \*directory, const char \*fname, const char \*extension) .....323
- 16.14 int mkdir(const char \*path) .....324
- 16.15 char \*mktemp(char \*fname) .....324
- 16.16 int rmdir(const char \*path) .....325
- 16.17 char \*searchpath(const char \*fname) .....325
- 16.18 int setdisk(int drive) .....326
- 16.19 void \_splitpath(const char \*fpath, char \*drive, char \*directory, char \*fname, char \*extension) .....326
- 第17章 进程控制函数 .....328
- 17.1 void abort(void) .....328
- 17.2 int atexit(void (\*func)(void)) .....328
- 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 createflags, unsigned \*threadID); .....329
- 17.4 void \_c\_exit(void) void \_cexit(void) .....331
- 17.5 void \_endthread(void) void \_endthreadex(unsigned threadvalue) .....331
- 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 execlp(char \*fname, char \*arg0, ..., char \*argN, NULL) int execlpe(char \*fname, char \*arg0, ..., char \*argN, NULL, char \*envp[ ]) int execv(char \*fname, char \*arg[ ]) int execve(char \*fname, char \*arg[ ], char \*envp[ ]) int execvp(char \*fname, char \*arg[ ]) int execvpe(char \*fname, char \*arg[ ], char \*envp[ ]) .....332
- 17.7 void exit(int status) void \_exit(int status) .....333
- 17.8 int getpid(void) .....334
- 17.9 int spawnl(int mode, char \*fname, char \*arg0, ..., char \*argN, NULL) int spawnle(int mode, char \*fname, char \*arg0, ..., char \*argN, NULL, char \*envp[ ]) int spawnlp(int mode, char \*fname, char \*arg0, ..., char \*argN, NULL) int spawnlpe(int mode, char \*fname, char \*arg0, ..., char \*argN, NULL, char \*envp[ ]) int spawnv(int mode, char \*fname, char \*arg[ ]) int spawnve(int mode, char \*fname, char \*arg[ ], char \*envp[ ]) int spawnvp(int mode, char \*fname, char \*arg[ ], char \*envp[ ]) 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
- 18.11 void inline(void) .....344
- 18.12 void lowvideo(void) .....345
- 18.13 int movetext(int left, int top, int right, int  
bottom, int newleft, int newtop) .....345
- 18.14 void normvideo(void) .....346
- 18.15 int puttext(int left, int top, int right, int  
bottom, void \*buf) .....346
- 18.16 void textattr(int attr) .....346
- 18.17 void textbackground(int color) .....347
- 18.18 void textcolor(int color) .....347
- 18.19 void textmode(int mode) .....348
- 18.20 int wherex(void) int wherey(void) .....349
- 18.21 void window(int left, int top, int right,  
int bottom) .....349
- 第19章 杂项函数** .....350
- 19.1 int abs(int num) .....350
- 19.2 void asset(int exp) .....351
- 19.3 double atof(const char \*str)  
long double \_atold(const char \*str) .....351
- 19.4 int atoi(const char \*str) .....352
- 19.5 long atol(const char \*str) .....352
- 19.6 void \*bsearch(const void \*key, const void  
\*base, size\_t num, size\_t size, int (\*compare)  
(const void \*, const void \*)) .....353
- 19.7 unsigned int \_clear87(void) .....354
- 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
- 19.15 char \*getenv(const char \*name) .....357
- 19.16 char \*getpass(const char \*str) .....358
- 19.17 unsigned getpid(void) .....358
- 19.18 char \*itoa(int num, char \*str,  
int radix) .....358
- 19.19 long labs(long num) .....359
- 19.20 ldiv\_t ldiv(long numerator, long  
denominator) .....360
- 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
- 19.22 struct lconv \*localeconv(void) .....362
- 19.23 void longjmp(jmp\_buf envbuf, int val) .....362
- 19.24 char \*ltoa(long num, char \*str, int radix)  
char \*ultoa(unsigned long num, char  
\*str, int radix) .....363
- 19.25 unsigned long \_lrotl(unsigned long l, int i)  
unsigned long \_lrotr(unsigned long l,  
int i) .....364
- 19.26 max(x,y) min(x,y) .....365
- 19.27 int mblen(const char \*str,  
size\_t size) .....365
- 19.28 size\_t mbstowcs(wchar\_t \*out,  
const char \*in, size\_t size) .....366
- 19.29 int mbtowc(wchar\_t \*out, const char \*in,  
size\_t size) .....366
- 19.30 int putenv(const char \*evar) .....366
- 19.31 void qsort(void \*base, size\_t num, size\_t  
size, int (\*compare) (const void \*, const  
void \* )) .....367
- 19.32 int raise(int signal) .....368
- 19.33 int rand(void) .....369

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

22.3 函数重载与多义性 .....	433	24.8.1 打开与关闭文件 .....	499
22.4 查找重载函数的地址 .....	435	24.8.2 读写文本文件 .....	501
22.5 this指针 .....	436	24.9 无格式和二进制I/O .....	502
22.6 运算符重载 .....	437	24.9.1 使用get( )和put( ) .....	503
22.7 引用 .....	446	24.9.2 使用read( )和write( ) .....	504
22.7.1 引用参数 .....	446	24.9.3 检测EOF .....	505
22.7.2 对对象传递引用 .....	448	24.9.4 随机存取 .....	506
22.7.3 返回引用 .....	449	第25章 模板、例外与RTTI .....	509
22.7.4 独立引用 .....	450	25.1 普通函数 .....	509
22.8 使用引用重载一元运算符 .....	451	25.1.1 一个带有两种普通类型的函数 .....	510
22.9 重载[] .....	454	25.1.2 显式重载一个普通函数 .....	511
22.10 应用运算符重载 .....	457	25.1.3 重载函数模板 .....	513
第23章 继承、虚函数与多态 .....	462	25.1.4 普通函数的限制 .....	513
23.1 继承与访问指示符 .....	462	25.2 普通类 .....	514
23.1.1 理解访问指示符 .....	462	25.3 例外处理 .....	518
23.1.2 基类访问控制 .....	464	25.4 例外处理基础 .....	518
23.2 派生类中的构造函数与析构函数 .....	466	25.4.1 捕获类的类型 .....	522
23.3 多重继承 .....	469	25.4.2 使用多重catch语句 .....	523
23.4 给基类传递参数 .....	471	25.4.3 处理派生类的例外 .....	524
23.5 指向派生类型的指针和引用 .....	472	25.5 例外处理选项 .....	525
23.6 虚函数 .....	474	25.5.1 捕获所有的例外 .....	525
23.7 为何要用虚函数 .....	479	25.5.2 限制例外 .....	526
23.8 纯虚函数和抽象类型 .....	482	25.5.3 重新引发例外 .....	528
23.9 前置绑定与滞后绑定 .....	484	25.5.4 理解terminate( )与unexpected( ) .....	529
第24章 C++ I/O类库 .....	486	25.5.5 设定Terminate和Unexpected处理器 .....	529
24.1 C++为什么有自己的I/O系统 .....	486	25.6 uncaught_exception( )函数 .....	530
24.2 旧式与新式的C++ I/O .....	486	25.7 例外处理的应用 .....	530
24.3 C++ 流 .....	486	25.8 运行时类型标识(RTTI) .....	531
24.4 C++ 流类 .....	487	25.9 强制转换运算符 .....	533
24.5 创建自己的插入符与提取符 .....	488	第26章 C++的其他主题 .....	537
24.5.1 生成插入符 .....	488	26.1 使用new和delete动态分配内存 .....	537
24.5.2 重载提取符 .....	490	26.1.1 为对象分配空间 .....	539
24.6 格式化 I/O .....	492	26.1.2 监视内存分配失败的另一种方法 .....	542
24.6.1 使用ios成员函数进行格式化 .....	492	26.1.3 重载new与delete .....	542
24.6.2 使用操纵符 .....	495	26.1.4 为数组重载new与delete .....	547
24.7 创建自己的操纵符函数 .....	497	26.2 static类成员 .....	549
24.8 文件I/O .....	499	26.2.1 static数据成员 .....	549

26.2.2 static成员函数 .....	550	27.6 map容器 .....	600
26.3 虚拟基类 .....	553	27.7 算法 (algorithm) .....	605
26.4 const成员函数与mutable成员函数 .....	556	27.7.1 计数 .....	607
26.5 可变的成员函数 .....	558	27.7.2 删除与替换元素 .....	608
26.6 使用asm关键字 .....	558	27.7.3 反转序列 .....	610
26.7 链接规范 .....	558	27.7.4 转换序列 .....	611
26.8 .*与->*运算符 .....	559	27.8 使用函数对象 .....	612
26.9 构造转换函数 .....	561	27.8.1 一元与二元函数对象 .....	612
26.10 拷贝构造函数 .....	563	27.8.2 使用内建的函数对象 .....	612
26.11 授权访问 .....	565	27.8.3 构造函数对象 .....	615
26.12 名空间 .....	567	27.8.4 使用绑定器 .....	617
26.12.1 名空间基础 .....	568	27.9 string类 .....	618
26.12.2 using .....	570	27.9.1 string的一些成员函数 .....	622
26.12.3 无名的名空间 .....	572	27.9.2 string是容器 .....	626
26.12.4 一些名空间选项 .....	573	27.9.3 把string放入其他的容器 .....	627
26.12.5 std名空间 .....	575	27.10 总结STL .....	627
26.13 显式的构造函数 .....	576	<b>第四部分 C++Builder集成开发环境</b>	
26.14 typename与export .....	577	第28章 集成开发环境 .....	629
26.15 C与C++之间的区别 .....	577	28.1 IDE的四个窗口 .....	629
第27章 标准模板库与字符串类 .....	579	28.2 Menu窗口 .....	630
27.1 STL概述 .....	579	28.2.1 File .....	630
27.1.1 容器 .....	579	28.2.2 Edit .....	632
27.1.2 算法 .....	580	28.2.3 Search .....	633
27.1.3 迭代器 .....	580	28.2.4 View .....	634
27.1.4 其他STL元素 .....	580	28.2.5 Project .....	635
27.2 容器类 .....	581	28.2.6 Run .....	637
27.3 操作的一般理论 .....	582	28.2.7 Component .....	637
27.4 vector容器 .....	583	28.2.8 Tools .....	639
27.4.1 通过迭代器访问vector容器 .....	586	28.2.9 Help .....	641
27.4.2 插入和删除vector容器中的元素 .....	587	28.2.10 工具栏 .....	641
27.4.3 在vector容器中保存类对象 .....	589	28.3 Object Inspector窗口 .....	642
27.5 list容器 .....	591	28.4 Form窗口 .....	642
27.5.1 理解end()函数 .....	594	28.5 Code窗口 .....	642
27.5.2 push_front()与push_back() .....	595	28.6 使用快捷菜单 .....	642
27.5.3 list排序 .....	596	28.7 使用上下文相关帮助 .....	642
27.5.4 合并list .....	597	第29章 使用IDE开发应用程序 .....	644
27.5.5 在list中保存类对象 .....	598		



29.1 应用程序的类型 .....	644	29.4.3 建立GUI窗体 .....	659
29.1.1 New .....	645	29.4.4 添加Label和Edit组件 .....	660
29.1.2 Project1 .....	646	29.4.5 使用ActionList和ImageList组件 .....	662
29.1.3 Forms .....	646	29.4.6 创建基本菜单 .....	667
29.1.4 Dialogs .....	646	29.4.7 创建工具栏 .....	668
29.1.5 Projects .....	646	29.4.8 建立命令按钮 .....	670
29.2 组件模板 .....	647	29.4.9 添加Help   About 对话框 .....	670
29.2.1 标准组件 .....	647	29.4.10 添加代码完成程序 .....	672
29.2.2 Additional组件 .....	648	第30章 使用C++Builder集成调试环境 .....	674
29.2.3 Win32组件 .....	649	30.1 准备要调试的程序 .....	674
29.2.4 System组件 .....	649	30.2 什么是源代码级的调试器 .....	674
29.2.5 Dialogs组件 .....	650	30.3 调试器基础 .....	674
29.2.6 Win3.1组件 .....	650	30.4 断点 .....	677
29.2.7 Samples组件 .....	651	30.4.1 设置无条件源代码断点 .....	677
29.2.8 ActiveX组件 .....	651	30.4.2 设置条件源代码断点 .....	678
29.2.9 Internet组件 .....	652	30.5 观测变量 .....	679
29.2.10 Servers组件 .....	652	30.5.1 观测表达式的格式 .....	680
29.3 创建控制台程序 .....	652	30.5.2 使变量名有效 .....	682
29.3.1 使用IDE创建控制台程序 .....	653	30.6 观测堆栈 .....	683
29.3.2 编译本书中的例子程序 .....	656	30.7 计算表达式 .....	684
29.3.3 使用命令行编译器 .....	656	30.8 暂停程序 .....	684
29.4 创建简单的Windows应用程序 .....	657	30.9 使用CPU窗口 .....	684
29.4.1 预备步骤 .....	657	30.10 调试技巧 .....	685
29.4.2 创建应用程序 .....	658		