

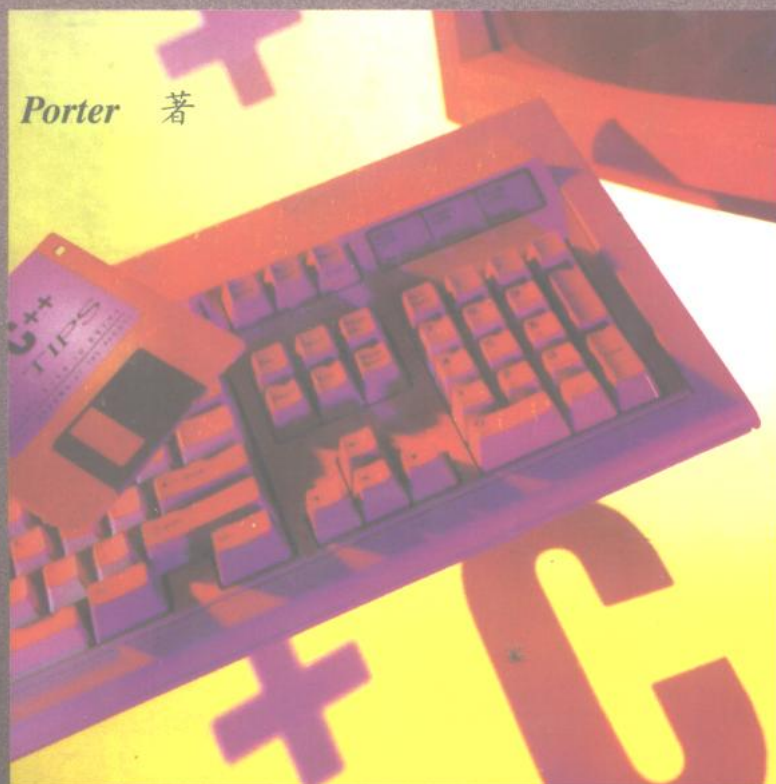
Osborne



最优 C/C++ 编程秘诀

The Best C/C++ Tips Ever

〔美〕Anthony Porter 著



雷君玲南
冬惠 向
秦孙殷陶

译



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

The Best C/C++ Tips Ever

最优 C/C++ 编程秘诀

[美]Anthony porter

秦冬雷 孙惠君 殷玲 陶向南 译

电子工业出版社

Publishing House of Electronics Industry

内 容 简 介

本书为 C/C++ 语言程序设计指南,分三个部分共十五章,系统全面地介绍了 C/C++ 程序设计的一些较常见的问题及解决的方法。其中第一部分为 C 和 C++ 语言,第二部分为标准函数,第三部分为 MS-DOS 特别说明,共阐述了 405 种常见的问题及编程秘诀。本书对提高 C/C++ 编程能力及技巧有较大的帮助。

本书适合于大专院校师生、培训班师生、研究生、程序员和其它从事计算机软件开发和应用的人员使用。

Copyright © 1993 by McGraw-Hill. Chinese Edition Copyright © 1995 by publishing house of Electronics industry, All right reserved.

本书英文版由 McGraw-Hill, Inc. 所有,中文版经 McGraw-Hill 于 94 年 10 月授予于电子工业出版社独家出版。未经出版者书面许可,不得以任何形式或任何手段复制或抄袭本书内容。

The Best C/C++ Tips Ever

最优 C/C++ 编程秘诀

[美]Anthony Porter 著

秦冬雷 孙惠君 殷玲 陶向南 译

责任编辑:崔围荣

*

电子工业出版社出版

北京市海淀区万寿路 173 信箱(100036)

电子工业出版社发行 各地新华书店经销

河北省大厂县胶印厂印刷

*

开本:787×1092 毫米 1/16 印张:21.5 字数:547 千字

1995 年 12 月第一版 1995 年 12 月北京第一次印刷

印数:5000 册 定价 36.00 元

ISBN7-5053-3133-7/TP·1115

著作权合同登记号图字:01-1995-157

前 言

本书并不是一本入门书——它假定你已至少了解 C 或 C++ 的基本语法和语义。该书希望能引导你通过语言的陷阱并告诫你如何提高编程水平。如果你曾经用了一二天时间查找一个程序故障，仅当某人提示：“嘿，一个通常的错误——我也曾犯过”，然后你就会认识到预先告诫的益处。

一旦你已经编写了一定数量的程序，就能知道如何去避免最通常的错误，并且更关心程序的优美性、有效性和易维护性。本书中的指南是针对解决通常的编程问题的。

本书覆盖了 C 和 C++，许多关于 C 的指南也可用于 C++，多数程序员仍使用 C 语言，尽管许多人现在已转向 C++，如果你仍使用 C，我想给你一个忠告，转向 C++。你可以使用 C++ 作为结构化程序设计，并且可利用 C++ 的类型检查以及许多对 C 的改进及扩充的功能，不要听信某些人告诉你：C++ 和面向对象程序设计是无效的——这对于以前的 C++ 翻译器可能是对的，但现代 C++ 编译器和 C 编译器一样有效。

该书支持 ANSI C，有不少程序员仍在使用 Pre-ANSI C，特别是在 UNIX 世界 Pre-ANSI C 包容在 UNIX 系统中。如果你仍在使用 Pre-ANSI C 编译器，则若不使用 C++ 那么绝对应该转向 ANSI C 编译器，Pre-ANSI C 的程序员对于 ANSI C 或 C++ 编译器能指出的运行错误可能要花相当长的时间去查找。

关于本书

本书分为三部分，第一部分为 C 和 C++ 语言，第二部分为标准库，第三部分为 MS-DOS 特别说明。这可能对 UNIX 程序员来说是不公平的，但多数程序员使用 MS-DOS 且甚至有一些 UNIX 程序员也为 MS-DOS 编写程序。

当然这并不是一本必须一页页研读的书，它是一本你编程时可以得到指导的书，甚至你会发现经常从一个指南导致另一个指南，并且可能某一个指南建立在一个或多个先前指南的基础上，如果你发觉不能理解一个指南时，可以返回来读前面的一些指南。

我已经在二个 C++ 编译器上测试了所有的例子，但并不能保证所有的例子可以在所有的 C++ 编译器上正确编译。某些编译器并不支持语言的所有特性，所以你可能会发现一些编译错误。

本书的结构

第一部分：C 和 C++ 语言

前三章主要针对 C 编程，但也是 C++ 程序员所感兴趣的。第一章列举了 10 种最通常

的 C 语言编程错误，这对 C 语言的初级编程者特别有用，第二、三章的指南覆盖了 C 语言的不同侧面，我希望这可以提高对 C 语言的理解。

四到八章覆盖 C++。根据你的经验，随着章节的深入，叙述将变得更加的有兴趣或更复杂。

第九章针对 C 预处理。其中一部分指南对 C++ 程序也是有用的。C++ 提供了为许多通常的程序任务替代预处理宏的特性。

第十章针对程序设计风格。你自己的编程风格可能并不和我的相同，另外，可能你的雇主要求以一定的或标准的风格编程。第十章中的指南并不是强加的，并且提供了你可以改进你自己的或雇主的编程风格的方法。这些指南也常常提供一些替代风格。只要前后一致，选择多少风格并没有多大关系。

第二部分：库函数

第十一章覆盖 ANSI C 函数，十二章覆盖 C++ 流库。十一章中的许多指南对 C++ 程序员也是有用的。目前对流库还没有一个标准，但十二章中的指南趋向 defacto 标准。你会发现这些指南对任何一种你用的编译器都是有用的，但也许会发现你的编译器的流库使用了与本书代码例子不同的函数或标识符。有些流程序包也提供了十二章中没有涉及的附加特性。

某些编译程序包括通用类的附加库。这些库常常包括一些诸如字符串，列表和数组等通用存储类。不幸的是，对这些类还没有一个标准，并且每一个编译器的编写者都以不同的方法来实现它们。既然我避免了编译特别说明指南，因此同样也避免了使用这些通用类的特别指南。

第三部分：MS-DOS 说明

十三章涉及了 INTEL 分段结构。这一章对那些编写针对 DOS 或 WINDOWS 的 16 位应用程序是有用的，因为它帮助你避免了许多通常的分段程序设计问题。甚至如果为 WINDOWS 或 OS/2 编写应用程序，这一章也会帮助你理解 INTEL 的体系结构。

MS-DOS 文件和目录服务可能是应用程序设计者仍需面对的 MS-DOS 中一个最复杂的部分，这在十四章中阐述。PC 的硬件或通讯端口的接口也是相当复杂的，但是很少有程序员直接处理。大部分程序员使用设备驱动程序或例程库去控制硬件和通讯口。

最后的十五章涉及一些针对 INTEL 硬件体系的调试程序技术。如果你至少有一些计算机如何执行编译代码的概念，那么即使你并不是一个装配程序员，也会发现查找 C 程序的故障是较容易的。

目 录

第一部分 C 和 C++ 语言

第一章 十种避免常见 C 语言编程错误的方法	3
1、在使用所有数据之前须先将其初始化	4
2、不能将堆栈里的对象作为指针返回值	4
3、当给字符串分配空间时，不要忘记空字符“\0”	5
4、请注意数组的最后一个元素的序号值比该数组的大小少一	5
5、使用函数原型	6
6、在表达式中使用 char 类型时要小心	7
7、不要混淆逻辑运算符与位运算符	8
8、不要依赖表达式的运算次序	9
9、函数形参的运算次序没有定义	10
10、不要混淆“=”和“==”运算符	10
第二章 标识符与数据	11
11、不要用 C++ 的关键词作为 C 语言程序的标识符	12
12、避免标识符含有双下划线或以下划线作为标识符的起始符	12
13、避免临时定义	12
14、如何编排长字符串	13
15、文件路径要用双反斜杠来表示	13
16、在一字符串中加入十六进制常数时要谨慎	13
17、转义符必须小写	14
18、何时需在问号(?)前加转义符	14
19、何时使用三元符	14
20、如果编辑器和程序使用同一扩展字符集则可以在字符串中使用扩展字符	15
21、对于小整数使用 char 型定义，效果并不好	15
22、枚举类型常量比 #define 语句更方便	15
23、当说明一个 enum 型的变量时没有必要使用 enum 关键字	17
24、数组在函数调用时是作为指针传递的	17
25、只有针对函数的形式参数而言，数组与数组的指针是等价的	18
26、如果形式参数是多维数组，须定义其大小	18
27、对于多维数组形式参数，第一维的大小是随意的	19

28、不恰当的对大型多维数组的操作会导致系统颠簸	19
29、允许超出数组一个单位的元素，但只能是超出一个	20
30、下标运算符位置可以交换	20
31、C 语言没有定义一种决定数组边界的方式	21
32、如果变量或函数在一模块中是私有的，将其定义为 static 型	22
33、一个函数内部的 static 型数据项在多个函数调用期间将被保存	22
34、对于内部初始化的数组，static 型数组很有用	23
35、如果不允许函数修改一个数组参数，那么将其说明为 const 型	24
36、const 修饰符单独使用同 const int 等价	24
37、怎样说明常量指针	25
38、使用现代编译器后没有必要再用 register 说明	25
39、何时使用 volatile 修饰符	26
40、一个变量可同时说明为 const 和 volatile	26
41、在一条语句中定义几个指针时，要谨慎	26

第三章 函数与运算符 **29**

42、如果函数没有形式参数，应将其原型说明为 void	30
43、传统函数定义将 float 类型视为 double 型	30
44、main 函数应返回一个结果	31
45、读程序命令行参数	31
46、尽可能不要使用递归函数	32
47、如果函数没有返回，说明其为 void	33
48、对定点数应避免使用浮点运算	34
49、避免对浮点数进行相等比较	34
50、谨慎整数表达式中会出现暂时溢出	35
51、避免在除法运算中混合使用符号数与无符号数	35
52、如果一个除法式中两个操作之一是负数，那么其结果的取整值取决机器的运行	36
53、避免进行符号与无符号数的比较	36
54、逻辑运算是从左向右进行求值直至获得表达式的值	36
55、逻辑非运算符！在测试是否为零时非常有效	37
56、赋值号可以连在一起	37
57、了解前缀“++、--”与后缀“++、--”的不同之处	38
58、位运算符对存储标志很有用处	38
59、给字中的某些位清零，使用取反运算符	39
60、要反置一位，可用异或运算符	39
61、怎样使用位运算符压缩小数位	40
62、算术移位与逻辑移位的区别	41
63、认清赋值运算符	41
64、条件表达式作为函数参数特别有效	42
65、条件表达式的优先级较低	43
66、位运算的优先级低于关系运算符	43
67、在 C 语言中，分程序块标识符对函数内变量的局部化十分有效	43

68、在一个嵌套的 if 语句中, else 语句与其前的 if 语句相匹配	44
69、switch 语句比嵌套 if—else 语句更有效	45
70、break 语句放在 switch 语句中最后一个 case 语句后	46
71、可在 while 循环体内初始化和测试变量	46
72、逗号运算符可用于 for 循环内多于一个循环变量的处理情况	47
73、不要混淆 break 与 continue 语句	47
74、通过检验循环变量可以判断循环语句是否执行了 break 语句	48
75、for 循环中的初始表达式是可任意的	48
76、自加符“++”只增加指针的值, 而不增加指针所指向内容的值	49
77、当查询一数组时, 使用数组指针要比使用数组下标更迅速	49
78、当有嵌套时, 数组指针比多维数组更有效	50
79、使用函数指针作为回调	51
80、用 typedef 来定义函数指针	52
81、理解复杂的说明	52
82、使用 typedef 来命名一个结构	53
83、ANSI C 允许结构赋值	54
84、当想传递一指针时, 不要无意中传递结构参数	54
85、如果要给数组赋值, 请将其置于结构中	55
86、当存储标志时, 位字段提供另一种屏蔽方法	55
87、可用位字段存储小的数值	56
88、位字段是运行依赖的	57
89、谨慎未初始化的指针	57
90、C 语言不允许做指针加法, 但可以做减法	58
91、弄清当指针加 1 时, 将会发生什么	58
92、空指针	59
93、null 指针不必每位都是零	60
94、在 ANSI C 对 void 型指针使用类型转换	60
95、何时使用多级指针	61
第四章 C++ 说明语句	63
96、在 C++ 中, 当定义变量时, 没有理由不将其初始化	64
97、怎样使用缺省形式参数	64
98、用缺省形参来扩展一个已非常广泛使用的函数	64
99、可在 for 和 while 条件语句中说明循环变量	65
100、如果在 for 和 while 条件语句中说明变量, 那么它在该语句后的作用域内仍有效	65
101、一个变量若是定义在条件语句内, 则不能在该条件语句作用域外被访问	66
102、如果在一 case 语句中初始化变量, 那么须将此 case 语句封闭在一个块中	66
103、在 C++ 语言中, 可取出一个寄存器的变量地址	67
104、:: 运算符可存取隐含在当前作用域中的数据项	67
105、条件运算符可写在表达式的左边	68
106、可以从普通 C 的 structure 中导出 class (类)	69
107、使用引用 (reference) 作为函数的形参	70

108、使用引用作为返回类型	71
109、处理出现在返回 reference 型函数中的错误条件值	71
110、何时使用内联函数	72
111、定义在类说明语句中的函数自动设置为内联函数	73
112、使用内联函数来访问一些私有成员	73
113、隐含类数据可方便地改变程序执行	73
114、引用型成员函数可读写成员	75
115、如何在类库中使用已被保护的成员	76
116、何时使用虚拟 (Virtual) 函数	76
117、一般情况下, 不要说明函数为 Virtual 型	78
118、对于导出类的 Virtual 函数没有必要使用 Virtual 关键词	78
119、在使用 Virtual 函数时一个常见的错误就是仅在某一类中改变形参类型	79
120、使用带有纯 Virtual 函数的抽象类	79
121、纯虚拟函数可以被定义	80
122、必须在导出类中重新定义所有纯 Virtual 函数, 否则导出类将也是抽象的	81
123、如果纯 Virtual 函数非正确导出, 编译器将会指出错误	81
124、当不使用 Virtual 函数全部形参时, 如何避免警告	82
125、基本类中应含有虚拟析构函数 (Virtual Destructor)	83
126、当没有一个导出类含有 Destructor 时, 基本类可省略 Virtual Destructor	83
127、弄清成员指针	83
128、如何使用指向成员函数的指针	84
129、对非静态函数运用 Callbacks	85
130、使用 Template 来实现成员函数指针	87
131、说明静态类成员	89
132、不要在头文件中给出静态数据定义	89
133、静态成员函数仅能访问静态数据	89
134、使用静态指针来指定活动的对象	90
135、使用全静态类来组合函数	90
136、Const 可取代 #define 或 enum 语句	91
137、如何在类中说明数组的大小	92
138、C++ 的程序怎样调用 C 函数	92
139、如何使头文件在 C 与 C++ 中可移植	93

第五章 创建和撤消 C++ 对象 95

140、怎样使用 New 和 Delete 会导致内存泄漏	96
141、C++ 允许删除空指针	96
142、当在一个类中使用指针时, 应在构造函数中首先将它们置成空指针, 然后在析构函数中将它们删除	97
143、当一个类包含指针成员时, 在类中说明一个拷贝构造函数和一个赋值运算符	98
144、书写赋值运算符时, 牢记 X=X	100
145、拷贝构造函数可以调用赋值运算符	100
146、当类无须拷贝时, 说明拷贝运算符为私有, 而无须对它们定义	101

147、在重定义赋值运算符时，必须拷贝所有的成员	101
148、赋值运算符不能被继承	103
149、除了在构造函数中，其它情况下给指针赋值前，都必须删除原先的内容	104
150、如果在析构函数以外释放指针，应紧随其后将该指针置为空指针	104
151、可为类提供一个初始化函数，而不必使用构造函数	105
152、试图使 New 和 Delete 运算符局限于一个单独类中	106
153、如何将 New 和 Delete 重新定义为全局运算符	108
154、一个类可以有自己的 New 和 Delete 运算符	109
155、一个重载的 Delete 运算符能够接受一个任意大小的参数	110
156、如何为 New 运算符指定布局参数	111
157、当类重新定义使用布局参数的 New 运算符时，它也应提供一个 不带参数的 New 运算符	112
158、如何在一个固定分配缓冲区分配类对象	113
159、在重新定义 New 运算符时，也需重新定义 Delete 运算符	114
160、当 New 运算符被重新定义时，如何重新定义 Delete 运算符	114
161、全局 New 常用于创建对象数组，即使类重新定义了 New 运算符	116
162、用 Delete [] 运算符撤消数组	116
163、一个重定义的 New 和 Delete 运算符如何处理导出类	117
164、如何重定义 New 去查找内存泄漏	118
165、为特定类优化 New	119
166、对于虚拟存储系统，New 函数怎样会导致系统颠簸	121
167、重载 new 而不使用 set-new-handler	121
168、若一个类不含显式构造函数，编译程序将提供缺省的构造函数	122
169、说明带参数的构造函数，将屏蔽缺省的构造函数	122
170、一个带有缺省参数的构造函数等同于缺省的构造函数	123
171、当类已屏蔽缺省构造函数时，导出类必须提供构造函数	124
172、即使基本类型也有构造函数	124
173、如何构造临时对象	125
174、避免在函数调用时，创建临时对象	125
175、不要从同一类的另一构造函数中调用缺省构造函数	126
176、理解拷贝构造函数	127
177、拷贝构造函数必须带常量参数	127
178、拷贝构造函数可以带有另外的缺省参数	128
179、虚拟构造函数	128
180、一个虚拟构造函数的虚拟模拟函数	129
181、为什么一个析构函数未被调用	130
182、如何在一个构造函数的初始化表中，对成员项初始化	131
183、不能使用初始化表初始化导出类成员	134
184、如何显式调用析构函数	134
185、自动类型的对象仅当超出它们所在的块时被撤消	135
186、当临时对象不再被调用时，它们将被释放	136
187、尽早使用一个程序块去撤消自动型对象	136
188、确保全局对象在使用前被初始化	137

第六章 类型转换和重载 139

189、转换成类	140
190、一个带有缺省参数的构造函数也能用作类型转换	140
191、用类型赋值运算符来避免创建临时对象所产生的额外开销	141
192、将类转换成另一类型	141
193、如有可能，仅为基本类型和库函数提供转换运算符	142
194、如在两类间既说明了一个转换运算符又说明了一个转换构造函数， 将可能产生二义性	143
195、记住转换运算符也可以是虚拟的	143
196、当说明类型转换及重载表达式时，请注意二义性问题	143
197、警惕过多使用类型转换	144
198、什么时候重载函数	144
199、当调用重载函数时，使用常量后缀	146
200、当调用重载函数时避免二义性	146
201、Const 特性对于区分重载函数的参数已足够了	147
202、对象的 Const 特性已足够区分重载的成员函数	147
203、切记，函数重载不可通过函数指针完成	148
204、用重载代替缺省参数	149
205、用重载函数设置或返回类中成员值	150
206、运算符表	151
207、什么时候重载运算符	151
208、注意运算符的运算优先级	151
209、将二元运算符定义为友元函数	152
210、=、()、[] 和->运算符必须是成员函数	153
211、将运算符参数说明为常量引用	154
212、前缀和后缀自加与自减运算符的差异	154
213、前缀运算符对自加或自减的对象更有效	154
214、重载运算符的返回类型未被定义	155
215、理解函数调用运算符	156
216、使用带有非整数参数的下标运算符	157

第七章 继承和成员访问控制 159

217、在 C++ 中，Struct 是公有类	160
218、在说明结构时，无需使用 typedef	160
219、指定存取说明符时，不必保持成员次序	161
220、限制成员存取并不影响其可视性	161
221、当私有派生时，如何选择存取权限	162
222、当基本类被公有继承时，不可取消基本类中特定成员的存取权限	163
223、避免使成员数据公有	163
224、当一个类从 C 的结构继承时，使其私有派生	163

225、何时公有派生	164
226、何时私有派生	164
227、当编写 Template 类时，从私有基本类派生	165
228、在 Template 基本类中，使用保护成员	167
229、谨慎使用多重继承	168
230、解决多重继承的二义性	168
231、使用虚拟基本类	169
232、不要将所有类定义为虚拟类	171
233、解决多重继承中二义性的虚拟函数调用	171
234、如何避免过多的虚拟基本类函数调用	173
235、虚拟基本类必须被任何导出类初始化	175
236、用多重继承转换一个类可以改变指针值	177
237、当从无关联基本类导出时，多重继承是易实现的	178
238、用隐含类代替多重继承	178
239、为多重继承设计类	180
第八章 C++ 技术	183
240、当两个类彼此引用时	184
241、在执行文件中包含被引用的类	185
242、将类分层（级）来屏蔽执行细节	186
243、为指针保留引用计数值	188
第九章 预处理	193
244、使用宏实现条件编译	194
245、用宏作为标志	195
246、用 #if 0 语句来实现代码的注释	196
247、在宏和函数之间提供一个选择	196
248、#include “文件名”和 #include <文件名> 之间的差别	197
249、include 语句中的文件名可以是一个宏	197
250、include 中的文件可以嵌套	198
251、怎样避免多次嵌入头文件	198
252、头文件中不应该包含有变量数据	200
253、用 #define 语句说明一个常量	200
254、使用函数宏	200
255、使用括号定义函数宏	201
256、扩充一个宏成多行	202
257、使用字符化运算符	202
258、用字符化运算符来扩展宏	203
259、使用连接运算符	203
260、用连接运算符定义类型函数	204
261、使用预定义宏	205

262、使用 #error 指令	206
263、pragma 指令的作用	206
264、使用 #line 指令	207
265、标识 #endif 指令	207
266、怎样写一个多语句的宏	208
267、怎样按照整型的长度有条件地编译	209
268、用预处理仿真 C++ 样板	210

第 10 章 程序设计风格问题 213

269、在一个数据项的名字中指出它的类型	214
270、使用较长的描述性名字	214
271、两个命名惯例	215
272、命名循环计数器	216
273、命名宏	216
274、类型定义 (typedefs)	217
275、C++ 类的命名	217
276、类对象的命名	218
277、函数的命名	218
278、命名作为数据包容使用的函数成员	219
279、使用花括弧的三种风格	221
280、表达式的写法	223
281、划分 if 和 while 语句	223
282、怎样划分 for 语句	225
283、长的函数调用怎样划分	225
284、把 else if 语句当作一个语句对待	226
285、尽量避免使 for 和 while 语句的语句体为空	226
286、将 do 语句中的 while 放到紧接 do 语句的右括号之后的同一行上	227
287、switch 语句的格式	227
288、函数定义的写法	228
289、定义类的两种方法	229
290、编排内联函数	230
291、分隔内联函数	230
292、预处理语句不一定必须从第一列开始	231
293、让编译器来优化表达式	232

第二部分 库函数

第十一章 标准 C 库函数 237

294、使用支持库函数的头文件	238
-----------------------	-----

295、使用 C 的库函数后应该检查错误信息	238
296、一个函数调用成功时, errno 中也可能包含有一个代码	238
297、main 之后的退出函数	238
298、memcpy 函数不处理重叠区	239
299、用 swab 函数交换字节的顺序	240
300、在长整型上交换字节顺序	240
301、用 getenv 读取环境变量	241
302、使用字符分类函数	241
303、创建唯一的文件名	242
304、获取有关文件信息	243
305、用 seek 函数找出文件的长度	244
306、文件 I/O 字符例程使用整型	244
307、选择流或低级文件 I/O	244
308、用“W”方式打开一个已存在的文件会使文件被重写	245
309、在修改方式进行读写	245
310、用 freopen 重定向标准流	245
311、重定向标准流回到原来状态	246
312、何时使用上锁函数	247
313、calloc 函数初始化数据为 0	247
314、calloc 或 memset 不能用于初始化指针和浮点数	248
315、用 Time 作为随机数种子	248
316、用 qsort 函数排序数组	248
317、使用带 qsort 函数的样板	250
318、在 qsort 函数中使用指针数组	251
319、在排好序的数组中查找	252
320、使用 assert 函数	252
321、在维护失败时设置断点	253
322、用 Strf time 格式化时间	254
323、strftime 与流结合起来格式化日期和时间	255
324、使用 strrchr 函数提取一个文件路径中的文件名	256
325、用 strcat 函数合并字符串时, 要给它分配存储空间	257
326、sprintf 函数也可以连接字符串	258
327、使用 longjmp 和 setjmp 函数进行出错恢复	259
328、复制字符串	260
329、大小写转换函数不能用于所有的欧洲式字符上	261
330、应小心使用大写的欧洲文本	261
331、用 strchr 函数来标记“,”分隔符	262
332、用 strtol 和 strtod 函数分析列表中的字段	263
333、删除一个字符串中最后一个字符	265
334、scanf 中的参数必须是地址	266
335、用 scanf 查找字符集	266
336、scanf 抑制赋值运算符	267
337、恢复 scanf 中的错误	267

338、避免用 scanf 函数直接从键盘读入	268
339、用 printf 函数实现动态精度	268
340、用 printf 函数实现动态对齐	269
341、存取可变参数表	270
342、传递可变参数给其它的函数	271
343、传递可变参数给 printf	272

第十二章 C++ I/O 流库..... 275

344、尽量使用输入输出流，少用 printf 和 scanf	276
345、理解流	276
346、读输入之前先刷新输出	277
347、使用流操作函数	278
348、为十六进制数设置大写字符	278
349、指定一个字段的宽度	279
350、居左、右、中间对齐	280
351、显示十六进制或八进制的基	281
352、使点 (.) 前导	281
353、调节十六进制数以适合字长	282
354、设置浮点数精度的格式	282
355、使用 ios 类公共成员函数	283
356、写入存储缓冲器	284
357、使用带内部缓冲的 ostrstream	284
358、用 ostrstream 写图形用户接口	285
359、写自己的操纵算子	285
360、写一个带整型或长整型的操纵算子	286
361、写带非整型参数的操纵算子	287
362、定义一个复杂类型的操纵算子	288
363、写一个带多个参数的操纵算子	288
364、重载 << 运算符去流一个类	290
365、提取值时处理分隔符	291
366、在分析时使用 seekg 和 tellg 函数	291
367、二进制文件上的流	294
368、流类对象	294
369、查看档案类	297
370、用 get 函数读取定界符	297
371、重定义 cin 和 cout	298

第三部分 MS-DOS 说明

第十三章 分段式体系结构	301
--------------------	-----

372、了解 Intel 的分段式体系结构	302
373、何时使用 32 位的编译器	302
374、选择 16 位的存储器模式	303
375、对 Near, Far 和 Huge 地址变换器使用宏	304
376、用经过优化的 Smaller 模式来建立应用程序	304
377、当使用 Medium 和 Large 模式时, 说明局部函数为静态 Near	304
378、定义一些数据项为 Far, 从而使得可以保留在一种存储器模式中	305
379、使用 Huge 指针	306
380、把 V 表移入代码段, 以便程序保留在 Medium 模式中	306
381、匹配存储器模式和函数库	306
382、建立适合任何模式的库	307
383、连接标准库和混合模式应用程序	307
384、如果需要指定 Far 指针, 则对引用也须如此	308
385、把同类函数组合在一个段内以优化存储管理	308
第十四章 目 录	312
386、显示 DOS 目录中的文件	312
387、列目录时将日期和时间同时列出	312
388、找出目录中的特殊文件和子目录	314
389、将目录排序	315
390、chdir 函数不改变 DOS 驱动器	316
391、搜索环境路径字符串	317
392、目录名可以有后缀	318
393、文件说明缓冲器最长为 120 个字符	318
394、如果你要保存文件路径指定, 使用从堆里分配的内存	319
第十五章 调 试	321
395、检查未赋值的返回值	322
396、检查复杂的函数参数	322
397、使用调试程序测试各个子程序	323
398、Windows 函数的参数是自左向右入栈的	324
399、尽可能在一个保护模式的操作系统下开发应用程序	324
400、不学习 Windows 程序设计而在 Windows 下进行开发	324
401、中断 Windows 程序	325
402、中断后不要在系统程序中单步执行	325
403、在库函数出错时找到调用点	325
404、如果重复删除一个对象将会出现什么	326
405、不要攻击编译程序作者	326



PART
1

第一部分 C 和 C++ 语言