

科海培训中心系列教材

面向对象的程序设计系统
TURBO C++应用教程
(中)

林亨利 陈维兴 成渝 编译

清华大学出版社

面向对象的程序设计系统 Turbo C++应用教程

(中 册)

林亨利 陈维兴 成 渝 编译

清 华 大 学 出 版 社

内 容 提 要

本书为“面向对象程序设计系统 Turbo C++应用教程”的中册。C++语言是为适应近代软件工程发展而开发的 C 语言的超集，它增加了面向对象的抽象数据类型——类以及围绕类增加的必要的语言机制。本书包括如下内容：一个完整的 C 和 C++语言的参考；C++的类、重载和流；六种内存模式、浮点处理和先进的覆盖技术；与汇编语言的接口和混合语言（包括 Pascal 语言）编程技术，视频函数介绍和运行库函数的分类功能。在附录 B 和 C 部分列出错误信息和 ANSI 特殊实现标准。

本书可供计算机软件应用和开发人员参考，也可作为有关专业的研究生、本科生的教学参考书。

(京) 新登字 158 号

面向对象的程序设计系统 Turbo C++应用教程 (中 册)

林亨利 陈维兴 成 渝 编译

☆

清华大学出版社出版

北京 清华园

门头沟胶印厂印刷

新华书店总店科技发行所发行

☆

开本: 787×1092 1/16 印张: 183/8 字数: 470 千字

1992 年 1 月第 1 版 1992 年 1 月第一次印刷

印数: 0001~8000

ISBN7-302-01025-0/TP·373

定价: 12.50 元

前 言

近年来面向对象的程序设计 (OOP) 技术越来越受到人们的重视, 无论在软件工程领域还是在人工智能领域都得到日益广泛的应用。

美国 BORLAND 公司于 1990 年推出了最新的面向对象程序设计系统——TURBO C++ 1.0 版。该系统提供了两个编译器, 既可以编译运行按照最新 ANSI C 标准书写的 C 程序, 又可使用 C++ 编程的全部功能 (实现了 AT&T 2.0 版的 C++)。

我们在长期从事有关学科的教学和科研的基础上, 根据最近一段时期使用 TURBO C++ 系统的经验, 参照 BORLAND 公司提供的资料编写了本教程。本书不仅包括了 TURBO C++ 手册中全部有用的信息, 而且对手册中提到的命令操作和示例程序作了试用和测试, 在使用中发现和改正了原手册中的一些错误。此外, 补充写入了我们在使用该系统的过程中取得的经验, 以及我们学习、研究面向对象程序设计技术的体会。本书较原手册内容更系统、更集中、更丰富, 条理更清晰也更容易阅读, 不仅可以作为使用 TURBO C++ 系统的参考资料, 更是学习面向对象程序设计技术和 TURBO C++ 系统的好教材。

本书共四篇, 分为三册。上册包括第一、第二篇, 第一篇介绍面向对象程序设计的基本思想和基本概念, 面向对象方法学的要点, 面向对象程序设计步骤等, 并提供几个有用的实例帮助读者理解 TURBO C++ 语言的基本特性和使用方法。第二篇全面系统地介绍 TURBO C++ 系统所提供的程序设计环境。本册为中册包括第三篇。下册包括第四篇, 是运行库, 按功能分类介绍该系统所提供的全部库函数及全程变量。

本册系统介绍 TURBO C++ 语言标准, 包括一般 C 语言的标准和 TURBO C++ 语言提供的特殊功能。阅读本册应注意 C++ 语言不同于一般 C 语言的主要特点:

- 增加了 C 语言中没有的类 (class)。面向对象程序设计是一种试图模仿人类对事物分类和抽象的方法。C++ 提供的类具有的封装性、继承性和多态性的特点, 完全可以使面向对象的程序设计思想付诸实施。

- C++ 允许对已有的操作符或函数给予重载 (Overload) 即重定义, 使之以一种适当的用户定义的方式对自己的类对象进行操作。C++ 中的一些特殊操作符, 比如作用域分辨符:: 使得程序中可随时引用不在当前作用域的对象。

- C++ 的流库比 C 的流更有效和更灵活。比如用重载操作符 << 和 >> 表示“输出到”和“输入”, 使得程序员可在一条语句中输入一个对象序列。C++ 允许用户定义新类型, 对它们进行输入输出操作能象内部类型一样有效、方便。C++ 的流输入输出格式控制方式比 C 的流更简单直观——使用控制符直接控制格式。

- TURBO C++ 中的覆盖技术比一般编译程序的覆盖技术更先进, 它能够确定什么时候覆盖和怎样覆盖, 用户不必担心要调用的块被覆盖。而且比其它的覆盖方案更快和更有效。

本册章节编排上册的续号, 从第十一章至第三十二章, 并有附录 B 和 C。第十一章到第十九章主要介绍的是一般 C 语言标准, 对其中 C++ 特有的部分加有标志。第二十章至第二十四章为 C++ 的类、重载操作符、流。第二十五章为 TURBO C++ 预处理指令, 第二十

六章介绍头文件和运行库主要功能。第二十七章介绍屏幕与图形函数。第二十八章介绍内存模式和混合模式编程。第二十九章介绍 TURBO C++ 中的浮点数、复数和 BCD 数。第三十章介绍了从 TURBO C++ 调用汇编程序以及从汇编语言调用 TURBO C++。第三十一章介绍嵌入式汇编、伪变量和中断函数。第三十二章介绍 TURBO C++ 的覆盖技术。

第十一章至第二十六章由林亨利编写,第二十七章至第三十二章由陈维兴编写,附录部份由林亨利、成渝编写。

编 者

1991 年 12 月

目 录

第 三 篇 语言参考

前言

第十一章 Turbo C++的文法形式、程序结构和书写格式 (1)

11.1 C++的文法形式 (1)

11.2 C++程序的基本结构 (2)

11.3 注释、空格及反斜杠连接 (3)

11.3.1 注释 (3)

11.3.2 空白 (4)

11.3.3 连接两行的符号——反斜杠“\” (4)

第十二章 C++的单词 (5)

12.1 关键字 (5)

12.2 标识符 (6)

12.3 常量 (7)

12.3.1 整型常量 (8)

12.3.2 字符型常量 (9)

12.3.3 浮点常量 (11)

12.3.4 枚举型常量 (12)

12.3.5 字符串常量 (13)

12.3.6 常量的取值范围和内部表示 (13)

12.3.7 常量表达式 (14)

12.4 C++的操作符 (15)

12.4.1 一元操作符 (15)

12.4.2 二元操作符 (16)

12.5 分隔符 (17)

第十三章 说明 (20)

13.1 对象 (20)

13.2 作用域和可访问性 (20)

13.2.1 作用域 (20)

13.2.2 可访问性 (21)

13.3 类型、存储类和生存期 (22)

13.3.1 类型 (22)

13.3.2 存储类和生存期 (22)

13.4 连接属性 (24)

13.4.1 翻译单元.....	(24)
13.4.2 连接属性.....	(24)
13.5 左值.....	(25)
第十四章 说明的文法	(26)
14.1 说明的方式.....	(26)
14.2 基本类型.....	(27)
14.2.1 整型.....	(27)
14.2.2 浮点类型.....	(28)
14.3 无值 Void	(28)
14.4 初始化.....	(29)
14.5 简单说明.....	(30)
14.6 存储类说明符.....	(30)
14.7 修饰说明符.....	(31)
14.7.1 常量修饰符 const	(32)
14.7.2 易变修饰符 Volatile	(33)
14.7.3 中断函数修饰符 interrupt	(33)
14.7.4 cdecl 和 pascal 修饰符	(33)
14.7.5 指针修饰符.....	(34)
14.7.6 函数修饰符.....	(35)
14.8 说明的文法形式.....	(35)
14.8.1 简单说明和复合说明.....	(35)
14.8.2 说明的文法.....	(36)
第十五章 表达式和操作符语义	(41)
15.1 表达式.....	(41)
15.1.1 操作符的优先级.....	(44)
15.1.2 标准转换规则.....	(45)
15.1.3 求值顺序.....	(46)
15.1.4 出错和溢出.....	(46)
15.2 操作符语义.....	(46)
15.2.1 前缀和后缀操作符.....	(46)
15.2.2 增量++操作符和减量--操作符	(47)
15.2.3 一元操作符.....	(48)
15.2.4 求空间大小操作符 sizeof	(49)
15.2.5 算术运算符.....	(50)
15.2.6 移位操作符.....	(51)
15.2.7 关系运算符.....	(51)
15.2.8 相等类运算符.....	(52)
15.2.9 按位与运算符&	(53)
15.2.10 按位异或运算符^	(53)

15.2.11 按位或运算符	(53)
15.2.12 逻辑与运算符&&	(53)
15.2.13 逻辑或运算符	(54)
15.2.14 条件运算符?:	(54)
15.2.15 赋值运算符	(54)
15.2.16 逗号运算符	(55)
第十六章 枚举和指针	(56)
16.1 枚举	(56)
16.2 指针	(57)
16.2.1 指针的说明	(57)
16.2.2 指向对象的指针和指向函数的指针	(58)
16.2.3 指针修饰符	(58)
16.2.4 指针的操作	(59)
16.2.5 指针的类型转换	(61)
16.2.6 C++的引用	(61)
第十七章 数组、结构和联合	(62)
17.1 数组	(62)
17.2 结构	(63)
17.2.1 结构的说明	(63)
17.2.2 结构成员的访问	(65)
17.2.3 结构成员在内存中的安排	(66)
17.2.4 位域	(66)
17.3 联合	(67)
第十八章 函数	(69)
18.1 函数的说明和定义	(69)
18.2 说明与原型	(69)
18.3 函数的定义	(71)
18.4 形式参数的说明	(71)
18.5 函数的调用及参数的转换	(72)
18.5.1 参数的传递	(72)
18.5.2 参数的转换	(73)
18.5.3 函数返回值	(73)
18.6 指向函数的指针	(73)
第十九章 语句	(75)
19.1 块	(76)
19.2 标号语句	(76)
19.3 表达式语句	(76)
19.4 选择语句	(77)
19.4.1 if 语句	(77)

19.4.2	switch 语句	(78)
19.5	循环语句	(78)
19.5.1	while 循环语句	(78)
19.5.2	do-while 循环语句	(79)
19.5.3	for 语句	(79)
19.6	转移语句	(80)
19.6.1	break 语句	(80)
19.6.2	continue 语句	(80)
19.6.3	goto 语句	(80)
19.6.4	return 语句	(80)
第二十章	C++特殊的操作符	(82)
20.1	引用 (reference)	(82)
20.1.1	简单引用和引用的初始化	(82)
20.1.2	引用参数	(83)
20.2	作用域分辨符::	(84)
20.3	new 和 delete 操作符	(84)
20.3.1	对数组使用 new 操作符	(85)
20.3.2	:: operator new 操作符	(85)
20.3.3	用 new 操作符初始化	(85)
第二十一章	类	(86)
21.1	类的说明	(86)
21.1.1	类说明的语法形式	(87)
21.1.2	类名	(88)
21.1.3	类名作用域	(88)
21.1.4	类对象	(89)
21.1.5	类成员表	(89)
21.1.6	内部函数	(89)
21.2	调用成员函数和自引用关键字 this	(90)
21.3	静态成员	(91)
21.4	成员的作用域	(92)
21.5	成员的访问控制和三种类	(93)
21.6	类的友元	(94)
21.7	派生类	(95)
21.7.1	派生类的说明	(95)
21.7.2	虚基类	(97)
21.8	虚函数	(97)
21.9	抽象类	(99)
21.10	C++的作用域	(100)
第二十二章	构造函数和释放函数	(102)

22.1	构造函数	(103)
22.1.1	缺省构造函数	(103)
22.1.2	复制构造函数	(104)
22.1.3	重载构造函数	(104)
22.1.4	构造函数的调用顺序	(104)
22.1.5	类的初始化	(105)
22.2	释放函数	(107)
22.2.1	何时调用释放函数	(108)
22.2.2	atexit, #pragma, exit 和释放函数	(108)
22.2.3	exit 和释放函数	(108)
22.2.4	abort 和释放函数	(108)
22.2.5	虚释放函数	(109)
第二十三章	重载操作符	(110)
23.1	操作符函数	(111)
23.2	重载操作符和继承	(111)
23.3	new 和 delete 重载	(111)
23.4	重载一元操作符	(112)
23.5	重载二元操作符	(112)
23.6	重载赋值操作符 =	(112)
23.7	重载函数调用操作符 ()	(113)
23.8	重载下标操作符 []	(113)
23.9	重载成员访问操作符 →	(113)
第二十四章	C++ 的流	(114)
24.1	什么是流、如何使用流 I/O	(114)
24.1.1	流的概念	(114)
24.1.2	如何使用文件和流	(114)
24.2	iostream 库	(116)
24.2.1	streambuf 类	(116)
24.2.1	ios 类	(116)
24.3	四个标准流	(117)
24.4	输出	(118)
24.4.1	固有的输出类型	(118)
24.4.2	put 和 write 函数	(119)
24.4.3	输出格式化	(119)
24.4.4	转换基数	(120)
24.4.5	宽度	(120)
24.4.6	控制符	(120)
24.4.7	填充字符和补空	(122)
24.4.8	用户定义的输出符	(122)

24.5 输入	(123)
24.5.1 输入符的连用	(123)
24.5.2 固有的输入类型	(123)
24.5.3 putback 函数	(125)
24.5.4 用户定义的输入类型	(125)
24.6 初始化流	(125)
24.7 简单的文件 I/O	(126)
24.8 I/O 流错误状态	(128)
24.9 使用旧式流	(129)
24.10 升级到 2.0 版流的指南	(129)
第二十五章 Turbo C++ 预处理指令	(131)
25.1 预处理指令的语法形式	(131)
25.2 #define 和 undef 指令	(133)
25.2.1 简单#define 宏	(133)
25.2.2 #undef 指令	(134)
25.2.3 __D 和 __U 选择项	(135)
25.2.4 关键字和保护字	(135)
25.2.5 带参数的宏定义	(136)
25.3 用#include 包含文件	(138)
25.4 条件编译指令	(138)
25.4.1 #if, #elif, #else 和 #endif 条件指令	(139)
25.4.2 #ifdef 和 #ifndef 指令	(139)
25.5 行控制指令 #line	(140)
25.6 #error 指令	(141)
25.7 #pragma 指令	(141)
25.7.1 #pragma argsused	(142)
25.7.2 #pragma exit 和 #pragma startup	(142)
25.7.3 #pragma inline	(142)
25.7.4 #pragma option	(143)
25.7.5 #pragma save regs	(144)
25.7.6 #pragma warn	(144)
25.8 预定义的宏	(145)
25.8.1 __CDECL__	(145)
25.8.2 __cplusplus	(145)
25.8.3 __DATE__	(145)
25.8.4 __FILE__	(146)
25.8.5 __LINE__	(146)
25.8.6 __MSDOS__	(146)
25.8.7 __OVERLAY__	(146)

25.8.8	____PASCAL____	(146)
25.8.9	____STDC____	(146)
25.8.10	____TIME____	(146)
25.8.11	____TURBOC____	(146)
第二十六章	Turbo C++的 main 函数、头文件和运行库功能介绍	(147)
26.1	main 函数	(147)
26.1.1	main 函数的参数	(147)
26.1.2	使用 pascal 调用约定__P	(149)
26.1.3	main 函数返回值	(149)
26.2	Turbo C++头文件	(149)
26.3	运行库功能介绍	(151)
26.4	运行库函数源代码	(152)
第二十七章	屏幕与图形函数	(153)
27.1	图形适配器及视屏模式	(153)
27.2	文本模式下的程序设计	(153)
27.2.1	窗口	(153)
27.2.2	文本模式的种类	(154)
27.2.3	文本模式函数	(155)
27.3	图形模式下的程序设计	(160)
27.3.1	视口	(160)
27.3.2	象素与调色板	(160)
27.3.3	图形函数	(160)
第二十八章	内存模式和混合模式编程	(174)
28.1	内存模式	(174)
28.1.1	8086 寄存器	(174)
28.1.2	内存的段结构	(176)
28.1.3	近指针和远指针	(177)
28.1.4	六种内存模式	(177)
28.1.5	内存模式的选择	(181)
28.1.6	内存模式的编译选择项	(181)
28.2	混合模式编程	(181)
28.2.1	far (远程)	(182)
28.2.2	near (近程)	(183)
28.2.3	huge (巨形)	(183)
28.2.4	跨越内存模式的说明	(184)
28.2.5	Turbo C++的段修饰符	(185)
28.2.6	库文件的连接	(186)
28.2.7	混合模式的连接	(186)
28.2.8	简例	(187)

28.3 Turbo C++ RAM 的用法	(189)
第二十九章 浮点数、复数和 BCD 数	(190)
29.1 浮点数	(190)
29.1.1 仿真 80x87 芯片	(190)
29.1.2 使用 80x87 协处理器	(191)
29.1.3 无浮点代码	(192)
29.1.4 浮点运算的速度优化	(192)
29.1.5 87 环境变量	(192)
29.1.6 寄存器和 80x87	(193)
29.1.7 屏蔽浮点异常	(193)
29.2 复数	(194)
29.3 BCD 数	(196)
第三十章 Turbo C++与汇编语言的接口	(198)
30.1 参数传递方法	(198)
30.1.1 C 参数传递顺序	(198)
30.1.2 PASCAL 参数传递顺序	(199)
30.2 从 Turbo C++调用汇编程序	(200)
30.2.1 Turbo C++与汇编语言的接口机制	(201)
30.2.2 汇编语言与 Turbo C++的交互性	(208)
30.2.3 建立汇编代码函数	(213)
30.2.4 从 Turbo C++中调用汇编语言函数	(215)
30.3 从汇编语言调用 Turbo C++	(216)
30.3.1 调用说明和参数传递	(216)
30.3.2 汇编语言调用 Turbo C++函数	(219)
第三十一章 嵌入式汇编、伪变量和中断函数	(221)
31.1 嵌入式汇编	(221)
31.1.1 嵌入式汇编的工作过程	(222)
31.1.2 嵌入式汇编语句的格式	(226)
31.1.3 Turbo C++中使用的操作码	(227)
31.1.4 嵌入式汇编引用数据和函数	(228)
31.1.5 访问 C 结构成员	(229)
31.1.6 转移指令和标号的使用	(230)
31.2 伪变量	(230)
31.3 中断函数	(232)
31.4 实例	(233)
第三十二章 面向对象的实时虚拟存储管理覆盖技术	(235)
32.1 VROOMM 的工作原理	(235)
32.2 覆盖的使用	(236)
32.3 覆盖程序的设计	(237)

32.3.1	缓冲区的大小	(237)
32.3.2	驻留模块和覆盖模块	(238)
32.3.3	far 调用	(238)
32.3.4	覆盖中的外部过程	(238)
32.3.5	调试覆盖	(239)
32.4	交换	(239)
32.4.1	扩充内存	(239)
32.4.2	扩展内存	(239)
附录 B	错误信息	(241)
附录 C	ANSI 特殊实现标准	(271)

第 三 篇

第十一章 Turbo C++的文法形式、 程序结构和书写格式

Turbo C++实现了 X3J11 技术委员会在 1983 年 6 月至 1988 年 12 月间制定的 ANSI C 标准，并作了多处扩充。Turbo C++完全实现了 AT&T 公司的 C++2.0 版，此版本是贝尔实验室的 Bjarne Stroustrup 开发的面向对象 C 语言的超集。C++相对 C 来说除了具有许多新的特性和提供许多新的功能外还或多或少对 C 作了一些改动。在本书的介绍中我们将标出这些不同之处。

用户可以设置编译选择项来指出有无扩充成分，也可以让编译程序将 Turbo C++的扩充关键字当作普通标识符处理（参阅第二篇 8.2 节）。

为了处理非标准的和实现有关的特性，可以用 ANSI C 提供的标准编译指令 `#pragma` 提供“一致性”扩充。

11.1 C++的文法形式

本篇详细介绍 Turbo C++语言，并给出相应的单词文法和短语结构文法。被一种语言识别的类似于字的单元称作单词，单词文法涉及的是单词的不同种类。短语文法描述单词组成表达式、语句和其它有意义的单元的合法方式。本篇在各有关部分将给出相应的文法。文法形式为定义的语法范畴名后加冒号，第二行开始列出它的定义，多种选择的定义列在不同行中。或者在冒号后同一行上写“下列之一”，然后在下一行开始列出多种选择的定义。例如：

外部定义：

函数定义

说明

八进制数字：下列之一

0 1 2 3 4 5 6 7

定义中的可选项用尖括号括起来，例如：

整型后缀：

无符号后缀<长后缀>

11.2 C++程序的基本结构

Turbo C++程序是由 ASCII 字符序列组成的源代码。在 Turbo C++中, 基本的程序单元是文件, 通常是对应于 RAM 或磁盘上的 DOS 命名文件, 并具有扩展名.C 或.CPP。

所有 C++程序都包含一个或多个函数。函数是语句和数据项的集合。而且每个可执行的 C++程序都必须有一个而且只有一个 main() 函数, 它可以放在程序的任何位置, 通常把它放在程序的首部或尾部。

以下为一个命名为 SALESTAG.C 的程序:

```
/* SALESTAG.C...Example. it calculates a sales slip */
#include <stdio.h>          .....包含文件
#define RATE 0.065         .....宏定义
float tax(float amount);    .....函数原型
float purchase, tax__amt, total; .....全局变量说明
int main()                 .....主函数
{                           .....函数以{开始
    char inbuf (130);       .....此处一般应缩进一至二格
    printf("Amount of purchase: ");
    gets(inbuf);
    sscanf(inbuf, "%f", &purchase);
    tax__amt = tax(purchase);
    total = purchase + tax__amt;
    printf("\npurchase is: %f", purchase);
    printf("\n      Tax: %f", tax__amt);
    printf("\n      Total: %f", total);
    return 0;
}                           .....函数最后以}结束
float tax(float amount)
{
    return(amount * RATE);
}
```

这个程序的第一行由符号 /* 引起, /* 代表注释行开始, */ 表示注释结束。编译程序将略去注释符之间的所有字符。

由数字或#开始的行不是 C++的语句, 而是 Turbo C++的指令, 称为编译指令或预处理指令, 这些指令用来指导编译器的操作。指令#include<stdio.h>告诉 Turbo C++应把stdio.h (包含文件) 读入并编译。stdio.h 是一个头文件。头文件包含库函数的描述 (如printf, gets, sscanf 函数的原型, 数据说明等), 程序中用到的库函数必须用#include 指令把它所在的头文件包含在内, 这样编译程序才能利用这些描述把程序和库函数一起编译 (头文件的有关信息参阅 26.2 节)。

程序的下一行#define RATE 0.065, 定义了一个宏替换, 它告诉编译程序, 在源程序中遇到 RATE 时就用 0.065 代替。

接着是用户定义的函数原型说明, 函数定义在后面。然后是全局变量说明。main 函数和 float tax 函数都用一对花括号{}括起定义。程序的缩进排列将大大有助于提高程序的可读性和可维护性。

11.3 注释、空格及反斜杠连接

11.3.1 注释

注释用来注解一个程序的正文。好的程序应对数据和程序的功能等作适当注释。注释仅为程序员所用，编译程序在对源程序进行语法分析前先将注释剔除掉。

Turbo C++程序中可用两种注释方式：传统的 C 语言方法和 Turbo C++方法。程序中混合使用这两种方法，也可以选择其中一种，还可以用 Turbo C++扩充的嵌套注释。

C 语言注释法：

传统的 C 语言可将注释用 `/*` 为开始，用 `*/` 为结束。整个注释包括这四个注释标志符在内，在宏扩展后用一个空格代替。注意，有的 C 编译实现只是把注释去掉而不用空格代替。

在 Turbo C++中不能用不可移植的单词拼接方法，即用 `/**/` 把两个标识符拼在一起。但 Turbo C++中可以用 ANSI 特殊的双井号 `##` 来粘接两个标识符，例如：

```
#define VAR (i, j) (i/* */j)    (不合法)
#define VAR (i, j) (i##j)      (在Turbo C++中合法)
#define VAR (i, j) (i##j)      (也是合法的)
```

在 Turbo C++中，如下描述：

```
int/* declaration */i/* counter */;
```

将被分析成三个单词：`int i ;`

嵌套注释

ANSI C 标准不允许使用嵌套注释，如将上一行注释写成：

```
/* int/* declaration */i/* counter */; */
```

将产生语法错误，因为第一个 `*/` 与遇到的第一个 `/*` 配套，将它们去掉后，分析得到 `i; */`。

同样，Turbo C++也不允许使用嵌套注释，但可以通过编译命令选择忽略这种限制，可用 `-C` 选择子（命令行编译器）或集成环境中通过 `OlC|Source` 选择菜单选择使用嵌套注释。

C++语言注释

C++语言可以用以 `“//”`（双斜杠）作为一单行注释的开头，并用换行符作为结束。这种注释可以写在一行的任意位置。例如：

```
class x { // this is a comment
...};
```

注释中的分隔符和空格

在个别情况下，在 `/*` 和 `//` 之前和 `*/` 之后的空格虽然在语法上不是必须的，但却可以避免出现问题，例如下面的 C++代码：

```
int i=j/* */ * divide by k * / k;
+ m;
```