

你必须知道的222个 C++语言问题

范立锋 李世欣 编著

222个编程新手最常遇到的C++语言问题

菜鸟想问不敢开口？

扫除入门者的障碍，开辟成长捷径



人民邮电出版社
POSTS & TELECOM PRESS

你必须知道的222个 C++语言问题

范立锋 李世欣 编著

人民邮电出版社
北京

图书在版编目 (CIP) 数据

你必须知道的222个C++语言问题 / 范立锋, 李世欣
编著. — 北京: 人民邮电出版社, 2010.6
ISBN 978-7-115-22496-5

I. ①你… II. ①范… ②李… III. ①C语言—程序设计—问答 IV. ①TP312-44

中国版本图书馆CIP数据核字(2010)第043584号

内 容 提 要

本书精选了 222 个在 C++ 程序设计中经常遇到的问题和典型功能, 覆盖了实际开发中的各种需求, 目的是帮助读者解决在 C++ 学习和开发中经常遇到的实际问题, 同时提高学习和开发的效率。本书涵盖了 C++ 与 C 语言的区别和联系、面向对象的设计思想、C++ 中的类和对象、继承、多态、指针与字符串、运算符重载、用户自定义数据类型、结构和枚举、类型转换与 RTTI、异常处理、标准模板库、通用函数及模板、C++ 中的输入与输出、内存管理、进程及线程、C++ 与 C 语言的综合应用和关于软件性能的思考等内容。本书所列出的问题均是作者在经过充分调研的基础上, 从实际开发项目中总结出来的典型问题, 提供的解决方法注重实用性。书中浓缩了作者多年从事项目开发的心得体会和实践经验教训, 并提供了程序设计的示例代码, 为初学者提供重要的参考价值。

本书适合于已经初步掌握 C++ 编程概念、方法的读者阅读, 可以帮助读者迅速解决实际开发中的疑难问题。

你必须知道的 222 个 C++ 语言问题

-
- ◆ 编 著 范立锋 李世欣
责任编辑 蒋 佳
 - ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
中国铁道出版社印刷厂印刷
 - ◆ 开本: 800×1000 1/16
印张: 24.5
字数: 548 千字 2010 年 6 月第 1 版
印数: 1—4 000 册 2010 年 6 月北京第 1 次印刷

ISBN 978-7-115-22496-5

定价: 45.00 元

读者服务热线: (010)67132692 印装质量热线: (010)67129223
反盗版热线: (010)67171154

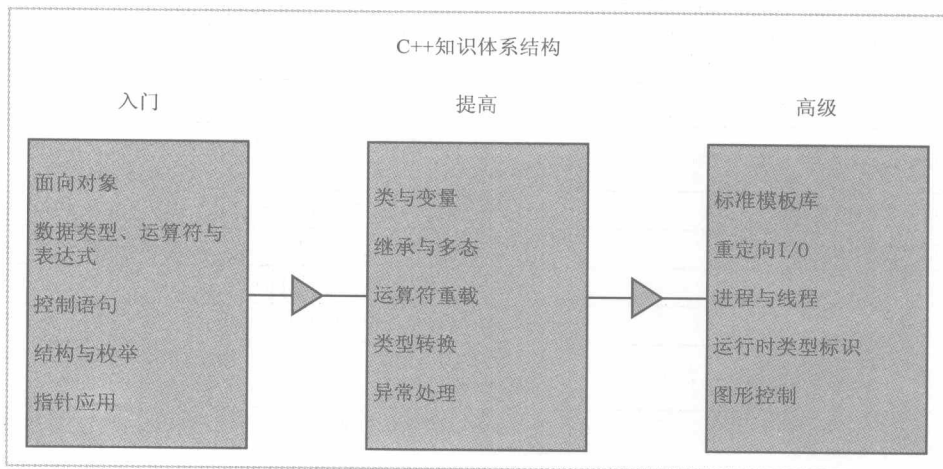
前言

随着计算机技术的迅速发展，C 语言（简称 C）在原有基础上扩充了面向对象机制而形成一种新的面向对象程序设计语言，该语言被称为 C++ 语言（简称 C++）。它除了继承 C 语言的全部优点和功能外，还支持面向对象程序设计。学习 C++ 不仅可以深刻理解和领会面向对象程序设计的特点和风格，掌握其方法和要领，而且可以使读者掌握一种十分流行并广泛应用的程序设计语言。

FAQ 是英文 Frequently Asked Questions 的缩写，其意思就是“经常问到的问题”或“常见问题解答”。本书主要以 C++ 语言环境为背景，解决学习与使用 C++ 语言时经常遇到的各种疑难问题，并给出专家在实际开发中对这些问题的建议。

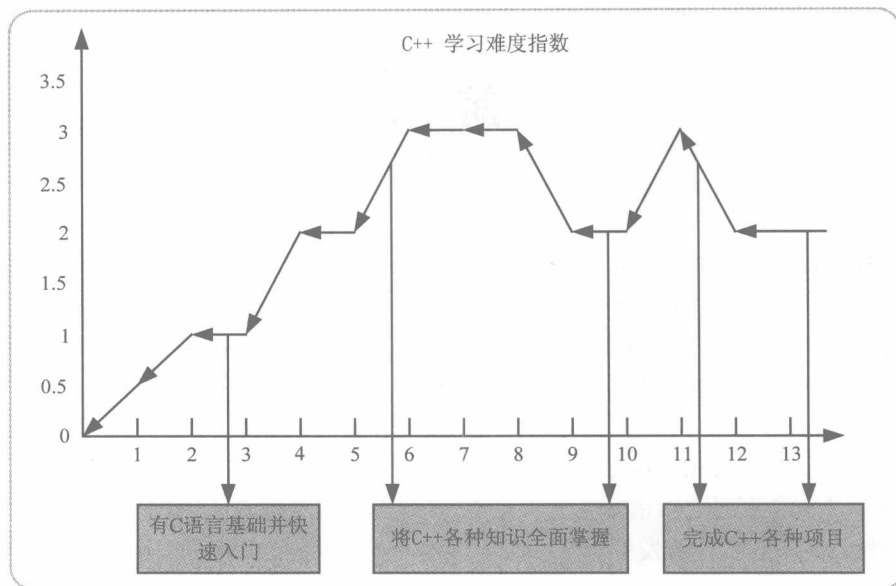
知识体系结构

C++ 本身是由 C 衍生而来的，它与其他高级语言相同，也具有丰富的知识体系结构。C++ 知识体系结构如下图所示。



学习 C++ 的难度分布图

为了让读者能够合理地学习 C++，笔者特意绘制了一个难度分布图，读者可以根据图中的标识进行学习。



本书结构

读者在学习与开发过程中,可能会遇到各种各样的问题,本书将根据这些问题按照不同的知识点分为各个章节,并且将问题进行深入,用实例代码的方式解决问题,使读者在学习和使用 C++ 时更加得心应手,做到学以致用。全书共分为 17 章,主要包含内容如下表所示。

章 节 名	章 节 内 容
第 1 章 C++与 C 语言的区别和联系	针对 C++在 C 语言基础上的扩充、C++与 C 语言的不同所产生的疑惑解答
第 2 章 面向对象的设计思想	针对 C++中面向对象编程技术所产生的疑惑解答
第 3 章 C++中的类和对象	针对 C++中类的定义、类的应用及对象的定义所产生的疑惑解答
第 4 章 继承	针对 C++中面向对象程序设计的重要特性继承所产生的疑惑解答
第 5 章 多态	针对 C++中继承和虚拟函数机制生成的动态多态所产生的疑惑解答
第 6 章 指针与字符串	针对 C++中指针和字符串的使用、指针和字符串的联系与区别所产生的疑惑解答
第 7 章 运算符重载	针对 C++中运算符重载的使用、规则及特殊运算符的重载所产生的疑惑解答
第 8 章 用户自定义数据类型、结构和枚举	针对 C++中结构类型的定义和使用、枚举类型及结构数组的使用所产生的疑惑解答
第 9 章 类型转换与 RTTI	针对 C++中数据类型的转换、静态类型检查及动态类型检查所产生的疑惑解答
第 10 章 异常处理	针对 C++中异常处理的原理、TRY_CATCH 语句的使用及 terminate() 函数和 unexpected() 函数所产生的疑惑解答

续表

章 节 名	章 节 内 容
第 11 章 标准模板库	针对 C++标准模板库中容器、算法所产生的疑惑解答
第 12 章 通用函数及模板	针对 C++中模板的作用、通用函数的使用及创建通用数组类所产生的疑惑解答
第 13 章 C++中的输入与输出	针对 C++中输入和输出操作、文件内容的读写所产生的疑惑解答
第 14 章 内存管理	针对 C++中内存分配的方式、内存常见错误所产生的疑惑解答
第 15 章 进程及线程	针对进程和线程的概念、操作所产生的疑惑解答
第 16 章 C++与 C 语言的结合应用	针对 C++中使用 C 语言的命令及操作所产生的疑惑解答
第 17 章 关于软件性能的思考	针对 C++中能够提供软件性能的方法所产生的疑惑解答

本书特点

目前市场上 C++相关图书很多,但以问答形式介绍 C++的特点与关键技术的书籍确实很少。作者在设计每个问答或需求时,都根据技术难度的不同加以标识,并给出在实际开发中专家的处理意见。

每个问答包括以下栏目。

- ❑ 问题简述——用简洁的语句将标题中的问题和需求描述清楚。
- ❑ 问题详述——将问题和需求所涉及背景、情况、需求及状况等信息进行详细的描述。
- ❑ 核心解答——给出问题的解决办法和满足需求的解决方案,并做适当延伸,使读者获得更多的知识。
- ❑ 疑难点评——根据解答出的内容,将问题的特点进行详细说明,并给出处理此问题的注意事项。
- ❑ 良好的编程习惯——是专家根据多年的软件研发经验总结出来的,有助于提高开发效率的小窍门及开发时需要注意的事项。
- ❑ 知识链接——指明了当前 FAQ 与其他 FAQ 相关知识点及原理的链接。

每个问答的特点如下。

- ❑ 常见问题一网打尽。本书中的问题是在实际开发与学习中提炼出来的,可以使读者快速找到问题的根源,短时间内解决问题。
- ❑ 示例详解,深入浅出。对于常见的问题注重于细节的讲解,尤其注重开发中每一个细小的环节,并将问题进行扩展,将每个问题进行全方位的介绍,使读者尽快找到问题的根源并及时解决。
- ❑ 与实际开发紧密结合。所有问题都以解决开发者在编程中遇到的实际问题和开发中应该掌握的技术为中心,每个问题都可以独立解决某个方面的疑题,例如,解决工作中的难题,同时提高工作效率并能提升工作价值。

学习路线

- 本书可以作为在学习使用 Visual C++开发应用程序过程中的工具书,读者可以带着在开发过程中遇到的疑惑按照本书的各章节名称进行检索,查询解开疑惑的答案,达到速查速用的目的。
- 对于编码性的技术问题,本书已将实例的关键代码全部给出,读者可以通过书中的代码了解并掌握解决问题的关键技术,并最终得到书中所涉及的实例的运行效果。

读者对象

- 有 C++编程基础的人群
- C++编程爱好者
- 大、中专院校的老师 and 学生
- 学习 C++存在问题的人群

技术支持

本书由范立锋、李世欣组织编写,参加编写的有范立锋、李世欣、于琦、唐瑶、张明、胡晶、吴新伟、吕正超、程峰、黄立东、霍晶馨、曹可欣、陈艳、王毅等。在本书编写过程中我们力求精益求精,虽然尽了最大的努力,但由于能力有限,难免存在一些错误及不足之处,敬请读者批评指正。感谢您购买本书,希望本书能够成为您的良师益友。

范立锋

2010年3月

目 录

第 1 章 C++与 C 语言的区别和联系	1
FAQ1.01 C++与 C 语言比较有哪些特点?	1
FAQ1.02 C++与 C 语言的代码注释风格有什么不同?	2
FAQ1.03 C++与 C 语言的程序结构有何不同?	3
FAQ1.04 C++与 C 语言是如何处理输入与输出的?	5
FAQ1.05 C++中局部变量的声明方式是否与 C 语言相同?	7
FAQ1.06 在函数原型的使用方面, C++与 C 语言是否一致?	8
FAQ1.07 在常量的定义方面, C++与 C 语言有何不同?	10
FAQ1.08 C 语言允许函数重载么? C++对于函数重载做出了哪些规定?	11
FAQ1.09 处理动态存储分配问题时 C++与 C 语言有何不同?	13
第 2 章 面向对象的设计思想	15
FAQ2.01 为什么采用面向对象编程模式?	15
FAQ2.02 面向对象技术包含哪些基本概念?	16
FAQ2.03 面向对象技术的基本特征包括哪些?	18
FAQ2.04 为什么封装对面向对象来说很重要?	20
FAQ2.05 接口与实现的分离有什么好处?	21
FAQ2.06 抽象是什么?	22
FAQ2.07 封装与抽象有何联系?	23
FAQ2.08 继承是否会削弱封装机制?	24
FAQ2.09 组合是怎样定义的, 它有什么作用?	25
FAQ2.10 组合的类型都包括哪些?	26
FAQ2.11 如何理解动态特性?	27
第 3 章 C++中的类和对象	29
FAQ3.01 如何理解类和对象?	29
FAQ3.02 如何理解对象的初始化?	32
FAQ3.03 如何理解对象的生存周期?	35

FAQ3.04	如何向函数传递对象?	37
FAQ3.05	编写 C++类时需要注意哪些问题?	40
FAQ3.06	如何理解构造函数?	42
FAQ3.07	默认构造函数是什么, 它有什么特点?	44
FAQ3.08	何时调用拷贝构造函数?	46
FAQ3.09	深拷贝与浅拷贝的区别是什么?	49
FAQ3.10	如何理解析构函数?	51
FAQ3.11	程序如何处理静态成员变量及静态成员函数?	54
FAQ3.12	为什么使用友元? 如何使用?	56
FAQ3.13	使用友元有什么优点?	58
第 4 章	继承	60
FAQ4.01	如何理解 C++中的继承?	60
FAQ4.02	C++程序的继承结构是怎样的?	61
FAQ4.03	当派生类与基类成员名称冲突时应如何解决?	64
FAQ4.04	为什么要使用虚基类?	65
FAQ4.05	继承体系中构造函数的调用顺序是怎样的?	67
FAQ4.06	在继承中如何使用 public、private 以及 protected 关键字?	69
FAQ4.07	类的默认访问权限是什么? 为什么使用它作为默认权限?	71
FAQ4.08	为什么要使用 protected 关键字? 如何使用?	72
FAQ4.09	为什么派生类不能访问其基类的 private 成员?	73
FAQ4.10	struct 与 class 之间有什么不同?	74
FAQ4.11	如何为基类构造函数传递参数?	75
第 5 章	多态	78
FAQ5.01	如何理解 C++中的捆绑?	78
FAQ5.02	如何理解和使用虚函数?	79
FAQ5.03	如何理解和使用纯虚函数?	81
FAQ5.04	如何理解和使用抽象类?	82
FAQ5.05	多态是如何实现的?	84
FAQ5.06	如何理解静态多态与动态多态?	86
FAQ5.07	如何理解虚函数和构造函数?	86
FAQ5.08	如何理解虚函数和析构函数?	87

第 6 章 指针与字符串	89
FAQ6.01 如何理解 sizeof 操作符?	89
FAQ6.02 指针是什么?	90
FAQ6.03 如何理解地址和指针的关系?	91
FAQ6.04 如何使用取地址操作符&?	92
FAQ6.05 指针与取地址操作符&怎样结合使用?	93
FAQ6.06 如何将指针与间接寻址操作符*结合使用?	94
FAQ6.07 指针的运算有哪些?	95
FAQ6.08 指针变量与引用有什么区别?	96
FAQ6.09 指针变量与变量指针有什么区别?	97
FAQ6.10 指针的比较指的是什么?	97
FAQ6.11 如何理解多级指针?	98
FAQ6.12 如何使用函数指针?	100
FAQ6.13 如何理解指针函数?	101
FAQ6.14 在指针中如何使用 const 限定符?	102
FAQ6.15 指针和数组之间有什么联系?	104
FAQ6.16 如何理解指针数组?	106
FAQ6.17 使用指针时有哪些常见的错误?	107
FAQ6.18 字符和字符串有什么区别?	109
FAQ6.19 常用的字符串操作函数有哪些?	110
FAQ6.20 如何理解字符数组和字符指针?	113
第 7 章 运算符重载	116
FAQ7.01 为什么使用运算符重载?	116
FAQ7.02 使用运算符重载应遵循哪些规则?	119
FAQ7.03 哪些运算符不能重载? 哪些可以重载?	121
FAQ7.04 为什么要使用友元函数重载运算符?	122
FAQ7.05 使用友元函数重载“++”、“—”运算符时可能会出现什么问题?	125
FAQ7.06 如何实现 NEW 和 DELETE 运算符的重载?	127
FAQ7.07 如何重载数组下标运算符?	129
FAQ7.08 如何将运算符函数作为成员函数使用?	131
FAQ7.09 成员运算符函数与友元运算符函数有什么区别?	133

第 8 章 用户自定义数据类型、结构和枚举	135
FAQ8.01 如何定义结构类型及结构声明?	135
FAQ8.02 对于无名结构是如何使用的?	137
FAQ8.03 如何通过点操作符访问结构?	138
FAQ8.04 结构数组是如何定义及使用的?	140
FAQ8.05 如何定义及使用结构中的结构?	142
FAQ8.06 如何使用多重结构嵌套?	144
FAQ8.07 如何将结构地址传递给函数?	146
FAQ8.08 什么是枚举类型? 如何使用枚举类型?	148
第 9 章 类型转换与 RTTI	151
FAQ9.01 C++预定义的类型转换有哪些方式?	151
FAQ9.02 如何实现类这种数据类型与其他数据类型的转换?	152
FAQ9.03 为什么需要转换函数? 如何创建转换函数?	156
FAQ9.04 C++新定义了哪几个强制转换运算符? 作用分别是什么?	158
FAQ9.05 如何区分静态类型检查和动态类型检查?	161
FAQ9.06 为什么要避免使用动态类型检查?	162
FAQ9.07 什么是运行时类型标识?	163
FAQ9.08 什么是 downcast?	166
FAQ9.09 为什么向下的类型转换是危险的?	168
FAQ9.10 dynamic_cast<T>()函数的作用是什么?	168
FAQ9.11 static_cast<T>()函数的作用是什么?	170
FAQ9.12 typeid()函数的作用是什么?	172
第 10 章 异常处理	175
FAQ10.01 C++异常处理的原理是什么?	175
FAQ10.02 异常处理是如何实现的?	176
FAQ10.03 使用异常时应该注意哪些方面?	182
FAQ10.04 抛出的异常和捕获的异常是否必须匹配?	185
FAQ10.05 如何处理 TRY 语句中函数抛出的异常?	186
FAQ10.06 程序在何时执行 CATCH 语句?	188
FAQ10.07 一个 TRY 语句是否可以使用多个 CATCH 语句? 如何使用?	189
FAQ10.08 对异常使用省略符有什么作用?	191
FAQ10.09 THROW 语句具体有什么作用?	193

FAQ10.10	如何实现重新抛出异常?	194
FAQ10.11	构造和析构对象时产生的异常应该如何处理?	196
FAQ10.12	如何使用默认函数参数避免异常和错误的发生?	197
FAQ10.13	处理异常时 <code>terminate()</code> 函数和 <code>unexpected()</code> 函数分别有什么作用?	198
第 11 章 标准模板库		201
FAQ11.01	什么是标准模板库? 为什么要使用标准模板库?	201
FAQ11.02	标准模板库包含哪些头文件?	202
FAQ11.03	如何理解容器?	204
FAQ11.04	标准模板库中容器的存储方式和访问方式	206
FAQ11.05	标准模板库中的容器是如何实现的?	207
FAQ11.06	标准模板库支持哪几个序列容器?	208
FAQ11.07	关联容器是如何工作的?	209
FAQ11.08	迭代器在标准模板库设计中有什么重要作用?	211
FAQ11.09	如何理解输入输出迭代器?	213
FAQ11.10	STL 包含哪些算法?	215
FAQ11.11	向量是什么? 如何使用向量?	218
FAQ11.12	如何构造一个 LIST 类型的对象?	221
FAQ11.13	如何为链表插入对象、移除对象以及为对象赋值?	223
FAQ11.14	如何遍历链表对象?	224
FAQ11.15	SLIST 和 LIST 有何区别?	225
FAQ11.16	什么是 DEQUE 容器? 如何使用这种容器?	226
FAQ11.17	如何对 DEQUE 容器使用 REVERSE 迭代器?	228
FAQ11.18	如何管理 DEQUE 队形的大小?	230
FAQ11.19	如何理解 MAP 对象?	231
FAQ11.20	如何使用成员函数管理 MAP?	232
FAQ11.21	如何控制 MAP 对象的大小及内容?	238
FAQ11.22	如何理解 SET?	239
第 12 章 通用函数及模板		242
FAQ12.01	如何理解模板?	242
FAQ12.02	如何显式重载通用函数?	244
FAQ12.03	什么情况下不能使用通用函数代替重载函数?	246
FAQ12.04	多个文件之间是否可以编译相同的函数模板定义?	247
FAQ12.05	类模板和模板类之间有什么关系?	248

FAQ12.06	当函数模板与同名非函数模板函数重载时如何进行调用?	249
FAQ12.07	如何使用模板定义通用类?	250
FAQ12.08	是否可以创建含有多个通用数据类型的通用类?	252
FAQ12.09	创建含有参数的操作符需要注意什么?	253
FAQ12.10	在函数模板中如何使用数组作为参数?	256
第 13 章 C++中的输入与输出		261
FAQ13.01	为什么 C++建立自己的输入输出系统?	261
FAQ13.02	如何设置并清除 ios 标志?	262
FAQ13.03	如何检测当前格式标志?	264
FAQ13.04	如何重载 setf()函数?	265
FAQ13.05	如何使用操作符格式化 I/O?	266
FAQ13.06	如何重载 "<<" ?	267
FAQ13.07	如何创建自定义的插入符?	269
FAQ13.08	如何打开文件流?	272
FAQ13.09	如何读写文本文件?	273
FAQ13.10	如何读写二进制文件?	275
FAQ13.11	如何实现每次读取一行文件?	277
FAQ13.12	如何检测文件是否结束?	278
FAQ13.13	如何控制输入输出格式?	280
FAQ13.14	如何创建自定义的提取运算符?	284
FAQ13.15	如何创建无参操作符?	286
FAQ13.16	如何创建带参数的操作符?	287
FAQ13.17	如何实现随机访问?	287
FAQ13.18	如何理解 C++的 I/O 状态?	289
第 14 章 内存管理		291
FAQ14.01	内存分配方式包括哪几种?	291
FAQ14.02	常见的内存错误有哪些? 如何解决?	293
FAQ14.03	指针参数如何传递内存?	295
FAQ14.04	free()和 delete()函数如何操作指针?	296
FAQ14.05	对同一指针执行两次删除操作会发生什么?	297
FAQ14.06	动态内存是否会被自动释放?	298
FAQ14.07	当 p 指向数组时能否执行 delete p 的操作?	299
FAQ14.08	malloc()/free()函数的使用要点有哪些?	300

FAQ14.09	new()/delete()函数的使用要点有哪些?	301
FAQ14.10	new()/delete()函数与 malloc()/free()函数有什么不同?	303
FAQ14.11	new 有哪些使用方式?	305
FAQ14.12	如何在预定内存位置建立对象? 如何释放该对象占用的空间?	308
FAQ14.13	如何应对内存耗尽问题?	309
FAQ14.14	定义了 p=new M(), 如果 M 构造函数抛出异常是否会出现内存 泄漏?	310
FAQ14.15	在使用 “delete this;” 操作时必须注意哪些问题?	311
FAQ14.16	如何释放一个没有明确析构函数对象所占用的空间?	312
FAQ14.17	最简单的防止内存泄漏的方法是什么?	313
第 15 章 进程及线程		315
FAQ15.01	如何更好地理解进程?	315
FAQ15.02	如何创建进程?	316
FAQ15.03	如何结束进程?	320
FAQ15.04	为什么要使用子进程? 如何使用子进程?	321
FAQ15.05	如何更好地理解线程?	322
FAQ15.06	何时需要创建线程? 何时不需要创建线程?	324
FAQ15.07	如何使用 CreateThread()函数创建简单线程?	325
FAQ15.08	操作系统创建线程分哪几步?	326
FAQ15.09	如何确定线程堆栈的大小?	327
FAQ15.10	如何获得当前线程或进程的句柄?	328
FAQ15.11	如何使用 GetThreadTimes()函数获得线程的执行时间?	330
FAQ15.12	GetProcessTimes()和 GetThreadTimes()函数有何不同?	331
FAQ15.13	如何更好地理解 GetQueueStatus()函数?	335
FAQ15.14	如何处理无句柄的异常?	336
FAQ15.15	如何结束线程?	337
FAQ15.16	如何确定线程或进程的 ID?	339
FAQ15.17	操作系统如何安排线程?	340
FAQ15.18	如何理解 Windows 系统的优先级类?	341
FAQ15.19	如何改变进程的优先级类?	342
FAQ15.20	如何设置线程的相对优先级?	344
FAQ15.21	如何获取线程的当前优先级?	345
FAQ15.22	如何获取线程的上下文?	347
FAQ15.23	线程的暂停及继续是如何实现的?	347

FAQ15.24	什么是线程同步?	348
FAQ15.25	Windows 系统支持的最常见的同步对象包括什么?	350
FAQ15.26	如何创建并使用简单的临界区?	351
FAQ15.27	WaitForSingleObject()和 WaitForMultipleObject()函数有何区别?	353
FAQ15.28	如何创建并使用互斥体?	357
FAQ15.29	如何理解信号量?	360
第 16 章 C++与 C 语言的结合应用		361
FAQ16.01	C++程序如何调用 C 程序?	361
FAQ16.02	C 程序如何调用 C++程序?	363
FAQ16.03	当 C++程序代码中包含 C 程序代码应该注意什么?	364
FAQ16.04	如何将 C++类的对象传递给 C 语言的函数?	365
FAQ16.05	C++的“iostream”和 C 语言的“stdio.h”是否可以一起使用?	367
FAQ16.06	C++的“iostream”和 C 语言的“stdio.h”比较有什么优势?	368
第 17 章 关于软件性能的思考		370
FAQ17.01	为什么编码方式会影响软件性能?	370
FAQ17.02	在不需要变量 <i>i</i> 原数值的情况下, <i>i++</i> 和 <i>++i</i> 哪个更好?	371
FAQ17.03	为什么使用最终类和最终成员函数能提高软件性能?	372
FAQ17.04	为什么成员对象能提高软件性能?	373
FAQ17.05	C++程序中如何从指针、引用的角度提高软件性能?	374
FAQ17.06	想要提高软件性能需要从哪几方面入手?	376

第 1 章

C++与 C 语言的区别和联系

C 语言（简称 C）是一种面向过程的结构化语言，适用于小型软件的开发设计，在开发一些大规模软件的时候，C 语言就显得有些力不从心了。C++语言（简称 C++）是在 C 语言基础之上发展而来的，其不仅保留了 C 语言的优点，更弥补了 C 语言的缺陷。面向对象编程机制是 C++有别于 C 语言最显著的标志。本章主要从编程语言的基础方面对 C++和 C 语言进行比较，使读者对 C++有一个整体认识，为后面章节的深入学习打下基础。

FAQ1.01 C++与 C 语言比较有哪些特点？

📖 难度系数：★

📖 问题几率：90%

问题详述

C 语言虽然是一门优秀的编程语言，但是在很多方面还是存在不足之处。C++是在 C 的基础上进行扩展的，提供了一些 C 语言不具备的功能。那么 C++与 C 语言比较到底有哪些特点呢？

核心解答

由于 C 语言提供了灵活的数据类型和强大的硬件操作能力，因此被广泛应用于操作系统软件以及硬件设备驱动的开发设计。但是在应用过程中，C 语言也暴露出一些语言本身设计上的局限性，其局限性有以下几点。

(1) C 语言几乎没有提供任何支持代码重用的语言机制，导致使用 C 语言所开发的程序中包含了大量重复的代码。

(2) C 语言的类型检测机制较弱，使得很多 C 程序中的错误不能在编译阶段被发现。

(3) C 语言的结构化特性使得其只适合开发中小型程序的开发设计，随着程序规模的不断扩大，开发人员很难控制程序的复杂性。

C++正是为了解决 C 语言出现的问题而设计的，它继承了 C 语言的优点并弥补了 C 语言的

不足。与C语言比较，C++的几个主要特点如下。

- (1) 全面兼容C语言，许多C语言编写的代码不用修改即可直接在C++程序中应用。
- (2) 支持面向对象，这是C++有别于C语言的最显著标志，也是C++最突出的特点。通过C++提供的面向对象机制，可以在程序中方便地构造出虚拟现实问题的属性和状态。
- (3) C++程序的可重用性、可以扩充性以及可维护性都十分良好，能够简化大中型项目的开发流程并提高开发效率。
- (4) C++程序结构清晰，运行效率高。

疑难点评

虽然从命名上来看C++与C语言相差不大，但这两种语言在编程思想方面存在着本质上的差别。对于那些拥有C语言基础的读者来说，学习C++时会轻松一些。如果没有任何语言基础也不要紧，只要掌握了正确的学习方法，同样能够学好并掌握C++。

良好的编程习惯

学习任何一门编程语言，都要掌握一定的技巧和方法，这样才能使学习做到事半功倍。正确的学习过程应该遵循以下步骤。

- (1) 首先必须掌握所学习语言的基本特征并了解其用途。
- (2) 准备好一款基础的开发工具。对于某些语言的学习，建议还是使用文本编辑器来编写程序，这是因为一些优秀的开发工具提供了很多附加功能实现，不利于初学者的学习。
- (3) 系统地学习并掌握该语言的设计基础知识，包括变量的定义及使用、流程控制的实现、运算符的应用和函数的工作原理等。
- (4) 严格依照该语言的编程规范进行开发设计。
- (5) 真正掌握了基础知识以后，可以尝试一些语言的高级技术实现，并学习在各种平台以及编程环境中使用该语言进行开发设计。

FAQ1.02 C++与C语言的代码注释风格有什么不同？

📖 难度系数：★

📖 问题几率：80%

问题详述

虽然代码注释不会影响应用程序的运行结果，但实际上它已经成为了程序设计过程中不可或缺的一个重要组成部分。C++和C语言分别对代码注释的风格做出了不同的规定，那么二者之间有什么区别呢？