普通高等教育“十一五”国家级规划教材

C++程序设计系列教材

C++程序设计教程详解——过程化编程

钱能 著

清华大学出版社

普通高等教育“十一五”国家级规划教材

C++程序设计系列教材

C++程序设计教程详解——过程化编程

钱能 著

清华大学出版社
北京

内 容 简 介

本书的前身先是《C++程序设计教程》，曾获教育部全国高校优秀教材奖，后是《C++程序教程教程（第二版）》，获教育部普通高等教育“十一五”规划教材及普通高等教育精品教材。将第二版分为过程化编程与对象化编程两部分作进一步细述。过程化编程部分即为本书，对象化编程部分（即《C++程序设计教程详解——对象化编程》）有待完成。

本书共分9章三个部分，以C++问题分析求解展开，阐述程序设计的初级方法。

书中经常比照C语言，对C++饮水思源，介绍C++的改进缘由和特征，以使读者不致因C++的优越感而迷失。书中始终围绕内部特性与抽象编程两条主线，以使读者学到更深刻的问题求解技巧和程序组织要诀。内部特性在于体现C++的编程技巧性特征，抽象编程在于充分发挥C++灵活的编程方法和技术。

本书内容涉及初、高级编程诸问题，主要针对初学编程的学生和自学者，适宜作为初学编程的教材。该书对于执着彻求C++编程奥秘的读者将会受益尤甚，对教师和程序员也不乏参考作用。

图书在版编目（CIP）数据

C++程序设计教程详解：过程化编程 / 钱能著. —北京：清华大学出版社，2014
C++程序设计系列教材
ISBN 978-7-302-35865-7

Ⅰ. ①C…　Ⅱ. ①钱…　Ⅲ. ①C语言－程序设计－高等学校－教学参考资料　Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2014）第060946号

责任编辑：郑寅堃
封面设计：傅瑞学
责任校对：焦丽丽
责任印制：刘海龙

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954

印 刷 者：清华大学印刷厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：41.25　　**字　　数**：998千字
版　　次：2014年5月第1版　　**印　　次**：2014年5月第1次印刷
印　　数：1～2000
定　　价：79.00元

产品编号：019748-01

前言与导读

导读首先要解决学习选择问题，也就是要清楚自己以怎样的学习路线来学习。

其次要了解本书的内容体系，以了解各章节之间的关联以及重点和难点。

再次要解决学习目标问题。学习的关键是什么？心理上应做好什么准备？如何学习才能达到目标？初学者被告诫需要克服的编程操作的困难有哪些？

之后是获得学习要领以及从本书的编排特色来了解本书阅读的方法。

最后是成书缘由，介绍本书写作中所依赖的课程教学背景和实验环境以及成书的关键原因。

1. C/C++之选择

C 与 C++的关系非常密切，C 和 C++程序员既互相融合，又各领风骚，于是就有初学计算机语言，是先学 C 还是 C++的问题。这个问题是自 C++诞生以来就一直在争议的。

1）C++出于 C 而胜于 C

由于 C++从 C 中继承而来，所以 C 程序也基本是 C++程序，有时不需说明便可以通过 C++编译；反之，C++程序则不能通过 C 编译器编译。所以，C++作为 C 的超集表现出编程方式方法的更多灵活选择。

C 编译器的集成开发环境相对 C++比较简单，C++的计算机实验环境比之 C 的实验环境在功能和操作上更便捷一些，甚至许多 C 学习环境都是借助于 C++的编译和开发环境来进行实验操作的。

C++从设计理念上更新和颠覆 C 语言，带来了更好的语言表现；而 C 语言的发展也在循着现代程序设计的要求向 C++的设计特征靠拢。从这个意义上说，C++更反映了现代程序设计的发展趋势。

C++具有更丰富的编程资源与库支持，语句描述更简洁，加上语言的强类型性，更容易获得程序正确性和安全性验证，因而初学切入更快，更容易克服编程表达的难关。

2）内容因素

（1）共同的初学内容

C 与 C++的学习，与问题解决的规模有很大的关系。一般认为，C++适合各种大小规模的编程；而 C 则更适合小规模和高效的编程。但在初级学习中，两者体现了共同的初学内容：

① 通过相对简单的程序框架结构，来表现小规模编程简单和自然的风采；

② 通过语言中语句表达技巧的学习，来体现小规模编程简捷、精巧的优势；

③ 通过良好编程风格的培养，洞悉语言理解和程序构造方法，为学习大规模编程打下基础。

程序设计学习，都是从理解和实践小规模编程开始，即使采用 C++语言来切入程序设计课程的学习，所看到和学到的实例也不外乎是一些小规模程序。

（2）后继学习的差别

C 在语言上不支持对象化编程方法，所以，如果后继还要学习对象化编程（如学了 C 再学 C++），要注意一些语言特征上的区别。例如，函数重载，编译指令的表达，for 循环的初始描述特征以及强化的类型匹配等，这些特征都是 C 所不具有的。C 语言的学习者在过渡到 C++时，需要处处留心，避免犯错，C 学习者在进一步学习 C++时，经历了语言比较，但却会因语言混淆而困惑。

因此，若从 C++着手来初学编程，则一步到位，似乎更靠谱。

本书总是比照 C 程序设计代码，作为可选阅读，让读者无缝切换到 C 语言学习与运用，弥补了 C++学习者比之从 C 过渡的学习者在经历上的不足。

3）方法因素

（1）共同的方法切入

编程的学习目标，不仅仅是通过学一些语言语法，学会程序控制的描述技巧，更重要的是通过编程方法和程序结构与组织的学习，掌握问题分析和设计的方法和手段。

无论从 C 开始学习还是从 C++开始学习，都是从简单的过程控制着手，学会分析问题，用编程控制来描绘解决问题的框架，用计算表达技巧，去描述解决问题的细节。

高级编程方法，或者说对象化编程的学习，同样需要过程结构和过程控制描述技巧的学习。所以学习 C 与 C++，在方法上，其学习切入点是相同的。

（2）灵活的后继选择

因为 C++具有更丰富的编程资源，支持了 C++在方法上表现出的更具抽象化的编程特征。所以从抽象化编程入手：

① 同步学习内在特性，便可以直入汇编语言，面向嵌入式或系统编程；

② 若递进学习对象化编程，则 C++比之 C 具有更完善的方法体系，更容易以方法体系来搭建学习模式，自然衔接对象化编程和其他高级编程的学习。由此可以展开编程规模，趋向软件工程。

因此从编程方法特征看，学习 C++具有更合理的后继选择。

本书既在 C++中表现 C 的精巧和性能优势，又注重抽象编程，强调结构，从框架学习上着眼，启发更多的学习感悟。

2．内容体系

C++的内容大致分为过程化方法编程与对象化方法编程两部分。过程化方法是对象化方法的基础，对象化方法的程序结构离不开以过程（函数）作为驱动（调用）的框架。

过程化编程以基本语句控制和内部数据类型的学习为开端，以分析问题、解决问题的训练为主线，学习程序构造和函数设计。

对象化编程为程序员搭建语言运用的框架，包括搭建以类型为基本单元的程序设计方法和对应的程序结构，将 C++过程控制技巧升华到大规模程序框架的搭建，完善整个编程知识体系，指导更高级的编程。

1）过程化编程

（1）第一部分 初涉编程

初涉编程是从语句着手，学习循环运行控制和内部数据类型的表示和使用，学习针对

问题描述的C++编程求解。通过对C++各种语句和表达式的组合描述，了解C++的各种语言特性。

读者将在内部特性上把握编程技术，学到构造问题求解技巧和语句描述技巧。

读者将会学习函数体描述中的过程控制——大到各种循环控制结构，小到输入/输出控制和各种表达式计算控制。

书中自然地展开数据运用，详细介绍内部数据类型的构造和各种操作。

作者认为第一部分对于初学者至关重要，其内容涉及初学编程之实践中绝大部分的困惑。

（2）第二部分　拓展技巧

解决了编程一开始将会遇到的初级困惑，便会渴望了解C++编程技巧，这种现象就是激发了的学习兴趣之表现。于是，作为第二部分，介绍的是如何解决比初涉编程更“难”一点的问题。例如，数据很多，需要按一定顺序排列，需要有一定的数据结构来组织，需要有一定的计算表达技巧。

本部分首先介绍语言的原理，语言的表达，以便了解程序构成的逻辑。其次介绍表达式计算的描述技巧以及它们在编程中发挥简捷、精确、高效的作用，使得编程学习从跌跌撞撞的无序性，走向优雅规范的艺术性。

接着，循序渐进地介绍批量数据处理方法，将数组指定为处理批量数据的主要数据实体，辅之以结构，使得较为复杂的计算可以化为简单的计算模型而得到解决。

最后介绍数据指针，从中了解指针的访问形式、通过指针申请内存空间的方法、指针的安全性以及指针构造链表的方法，特别点出指针既是编程难点，但又表现出很大的编程灵活性。

（3）第三部分　组织程序

从过程化程序框架着手，介绍程序赖以扩张的函数机制。函数从函数说明三部曲，介绍函数定义，函数指针，函数传值参数，指针、引用参数，到函数的内部结构，函数的栈结构、函数递归、重载以及参数默认，通过从函数设计原理上的解读，可达到对函数的深度把握。

除了函数之外，程序文件的构成，数据组织与时效，程序框架，编译与链接也是编程实践中需要掌握的重要内容。

2）对象化编程

（1）第一部分　对象化基础

过程化编程主要是从程序控制方法的层面上展开C++的编程学习。对象化编程则主要是从程序设计方法的层面上展开C++的学习。

程序架构，涉及程序方法，所以通过对象化编程来介绍。对象化编程的基础是将自定义数据类型作为模块搭建，引入类型和类系。所以，本部分介绍类与对象创建、存储、操作诸方法。

对象化编程的本质是分离类型编程。而类型编程是将自定义类型融入程序一部分的编程过程。它从应用编程中分离，使得由对象融入的过程控制，可以自如地调整过程控制的抽象性框架。所以，数据类型的自定义机制，带来了对象化编程，从而带来了程序设计方法的变革。

（2）第二部分　面向对象

本部分介绍对象化基础工作之后的进一步编程，即类型编程独立出应用编程后，不但需要对数据类型的机制进行贯通，从定义实体和操作到创建和使用，展开诸多情节，而且还需要突出其数据封装、继承和多态技术，以满足应用编程的抽象性、安全性要求和可扩展性要求，同时又能高效地处理批量对象数据。

为了在应用编程中，尽量适应可扩展性编程，在一个类系中表现多态的同时，应用模板编程的方法，以表现任何类型的多态特征。

为了配合面向对象，介绍异常处理是必要的。大规模编程的抽象性程序结构和层层安全保障体系，使得擅长对错误进行归类和处理的异常机制必不可少。

对象化编程一方面对数据类型的定义和使用的一整套机制展开描述，另一方面，围绕程序设计方法，介绍过程化、对象化、面向对象以及模板化编程，将它们互相融合，以达到性能最优，或者最优雅、最简捷、最通用的目的。所以，读者将在程序组织和抽象思维的训练上获得长足的提高。

3．学习目标

1）学习准备

编程学习的知识前提，基本上只需要小学数学运算和逻辑思维能力。从理解的角度上说，计算机程序是人为安排的、按一定语言结构构造的计算机工作顺序。编程学习就是在学习操作中不断了解计算机语言，适应语言的运用，达到正确表达解决问题思路的目的。这个目的一旦达到，就可以让计算机代替人来解决实际问题了。

我国学习编程，特别是学习C++的人群多为大学生，传统的中小学义务教育之教学内容和教学要求都围绕升学考试而无暇顾及编程学习。许多已达就业年龄的人群，学习编程只能作为业余的兴趣爱好，并不能系统地学习，更谈不上实际应用。然而编程学习确实由其内在的兴趣性吸引着众多的爱好者。因而有必要也引领这些人群走上自学成才之路。

C++编程在高校是作为计算机专业的基础课程，在计算机科学体系中，编程既作为应用开发的手段，也是许多研究方法的基础。它往往是走进大学殿堂的第一门技能性课程。许多其他专业也开设C或C++编程课程，以作为本专业的研究手段与工具。

2）课程学习

无论作为随堂课程学习，还是自学C++，都希望通过学习来达到他自己预定的目标。开设“C++程序设计”课程，是为了更好地适应科学发展形势，适应计算机人才的培养要求，与计算机产业现状接轨。

将C++编程作为课程学习，有利于对C++体系化的学习和认识。本书配套的课程目标是通过竞赛型实验和考试以及课外编程竞赛活动，使综合心理素质得到锻炼，编程学习的兴趣得以提高，从而深入C++编程学习，将重点置于问题的解决和代码调试，获得真正意义上的实践动手能力。并以此为契机，自然拓展C++编程学习的知识面，对应用开发和计算机专业知识体系有所了解，达到培养自学能力的目的。因而对于自学者，可以有意识地进行一定强度的以问题解决为目的的自我训练，来考察自己的能力提高。

3）三个层次

学习C++编程，旨在掌握计算机编程各个步骤的操作。通过充分的上机实验，获得调

制一个完整程序的能力，即掌握代码编辑、编译、调试，并付诸不同环境下运行的能力。

学习 C++编程，旨在学习程序设计基本思想与方法。通过充分的问题分析与设计实践，养成一定的编程风格，了解高效代码的编制技巧，掌握数据类型的创建与使用，准确控制多重循环，学会使用常用的 C++标准库，从而掌握基本的编程方法和技巧。

学习 C++编程，旨在获得进一步学习的方法。通过掌握 C++平台的使用，就能很容易熟悉作为计算机开发工具的各种 C++开发平台的使用方法和计算机应用开发技术的学习，通过对 C++体系化的初步认知，便容易理解 C++的整个体系，获得深化 C++学习的方法，更容易理解计算机科学的体系。

这三个学习的层次，便是学习本门课程的基本目标、一般目标和高级目标。

4）升华学习理念

学好编程在于对编程感兴趣，之后，从什么样的高度去认识编程就很重要。编程学习本是一种技能的学习。在技能学习中，若从知识、能力、素质三个要素上去把握，便会得到更大的收益，能够提高自己的综合学习能力，开拓科技创新意识和思想。

知识以能力为先导，中国学生知识运用能力相对薄弱，没有操作技能和充分的编程实践，其知识只是建立在虚无缥缈的空中楼阁之上。知识对能力有方向性指导作用，在素质的激发下，能深层次地获得和提升解决问题的能力。

能力基于扎扎实实的实践活动，通过有意识的素质磨炼以及勤于对知识的感悟与理解，便能获得珍贵的实践能力，它是编程学习的中心目标。

素质是一种适应性能力，它是在一定强度的对抗训练中获得的。高素质前提下的技能培养，能够帮助找到一条适合自身特点的学习之路，激发出创新思想。

C++编程学习过程体现在感性认识带动理性认知。它需要从实践中培养 C++知识的理解力。许多对书面文字的错误理解，随着实践的深入而矫正。没有实践的引导，许多知识会理解不了或只能片面地理解。实践不仅仅是一种操作或实验，它是一个不断深化的对知识和方法的理解验证过程。因此，不要寄希望于看看书就会编程的学习模式，或者，只经历很少的实践活动，而通过大量的看书“理解”来学习，这并不能达到预期的学习效果。

编程学习不但是计算机工具的操作学习，也是问题分析和解决方法的学习和计算机科学体系基础知识的学习。培养编程能力，是以培养计算机高素质创新人才为目标，注重综合素质的提高，重视基础学习和实践能力的培养。

同时，完成本书的编程学习也是继续深入学习的起点，在具备了继续学习的知识、能力和方法之后，后面的学习将更能自我把握。

4. 学习要领

学习要领是学习本书时需要领会的方法，本书也在这个方面多倾注了笔墨。学习方法很重要，当你学完了本书，进一步学习其他内容时，你能明白本书所刻意强调的学习方法吗？

1）学语言与学编程

（1）学语言

学语言是要了解语言的特性，了解其表达程序和组织程序的方式，一般当做知识密集型课程来学习。

无论先学 C 还是 C++，学习中都要防止追求单纯的语言知识的全面性，而忽略分析问题、解决问题能力的激发和培养。编程方法是包括编程技巧在内的一种把握编程框架体系的分析问题与解决问题的综合能力。脱离实践的知识全面性，不是真正意义上的理论水平。而具备了初步的编程能力，就可以带动语言学习能力的提高，其知识全面性便立足于实践，具有更强的高级编程指导性。这便是从初级学习逐渐深入的学习过程。

在学会了一门程序设计语言的编程方法后，有了一定的分析问题、解决问题的基础，便可以学语言文法的方式去学第二门编程语言。此时往往会选择描述语言语法特征以及语法使用的《C++程序设计语言》类书来学习。

（2）学编程

学编程就是学习如何运用语言来描述程序，然后通过计算机来解决问题。

将程序看做符合计算机语言之语法的描述集合，那么学习编程，必然要对照语言的语法子集进行反复练习。

但是，并不是要等到全部语言内容都学遍，再下手编程的，而是边学编程，边了解语言，甚至许多编程中的用法暂时还不理解，先模仿实践，然后再寻求理解。这也是所有自然语言成功学习的模式。不注重语言运用环境，片面讲求英语语法及结构，而不注重口语表达和应用训练的中国学生学习英语的学习模式，千万不要成为时下 C++编程的学习模式。

本书是学习编程的书，注重如何下手编程，如何分析问题，如何解决问题。它以介绍解决问题的编程方法为己任。在涉及编程诸内容时，伴随语言语法介绍，有相应的分析解决问题之道。其叙写结构也是按照编程学习的顺序逐一展开的。

2）理解分析与设计

本书的编程学习以解决问题作为学习形式，最终仍以解决问题的能力作为衡量掌握编程的标志。要提高解决问题的能力：

首先是理解和分析问题的能力提高，这个过程是螺旋式上升的。随着编程实践的深入，才能认识到问题所含更多经常会忽视的文字信息。这就促进了问题理解和分析的全面性和深入性，进而深化编程实践，提高编程能力。

其次是要知道对于同一个问题，有许多方法，这也是融会贯通运用知识，提高编程能力的途径。通过本书的各种编程描述技巧，将问题的解决方案转化成合理的程序结构框架和对应的简捷高效代码。掌握这一技巧，便是编程能力提高的体现。所以，编程学习含有很多方法性、技能性、技巧性要素。

3）积累调试经验

程序付诸运行是为了见证程序的正确性和高效性。在程序运行中跟踪数据变化，有助于发现运行错误，改善运行性能。这便是调试的功效。

首先，要把握 C++开发环境的操作，了解其特征，了解整个编程步骤，克服初学 C++过程中阻碍程序正确运行的诸多操作因素。

其次，要多学习改正编译错误的经验。在将问题的解决方案用程序语言描述出来，成为程序代码之后，便进入编译和链接阶段，这个过程也需要许多经验，以进一步强化对语言正确描述的敏感度与熟练度。

第三，致力于提高调试能力。调试涉及的范围非常广，需要不断积累经验，充分借鉴别的程序员在编程中犯的错误，以提高自己。

调试涉及计算机运行环境和内存布局，涉及C++语言的编译器差别，涉及代码的逻辑和结构。调试的错综复杂性决定了C++编程实践必须循序渐进，先把握C++操作环境，再把握调试方法。从解决最简单的问题开始，逐步深化。

随着学习的深入，代码结构逐渐复杂，代码规模逐渐扩大，调试难度也在逐渐加大，调试经验因而就在编程经历的增加中自然积累，调试手段也会在实践中完善。

编程中的调试能力因素也决定了C++编程学习不可以局限于知识密集型的学习。

4）内部特性与抽象编程

（1）内部数据特性

数据的内部表示取决于数据类型的描述，程序运行的机制取决于内存布局、异常处理的策略以及数据传递的方式。这些内部特性对于提高编程质量有着举足轻重的作用。本书特别细致地描述了整型与浮点型这两大内部数据类型的类系，并且特别注重通过位操作来观察内在数据变化，并衍生种种编程技巧。

（2）抽象编程

抽象编程是编程中框架设计的技巧。通过将繁复的代码抽象化为过程和函数，来简化过程结构和设计，从而达到复杂问题的简单化实现。

C++可以有多种构造程序的方法，视其程序的规模和作用的不同而不同。C++特别擅长以各个层次的数据模块和代码模块来扩充程序的规模，并不失其可维护性。C++是通过抽象编程的理念将各种编程方法融合在一起，从而发挥出“小大由之”的巨大威力。

（3）并重的意义

内部特性作为底部架构和设施是实现抽象编程的基础，而抽象编程则突出和发挥了内部特性的效率优势。既重视内部特性，又关注抽象编程，有助于从更深的层面上理解C++编程。

5）进阶提高的方向

应该看清，编程能力的关键是由分析问题、解决问题（设计解决方案）、调试代码这三种能力构成的。对于有一定深度理解能力的程序员来说，分析问题可以很好地把握；对于有一定实践经历的程序员来说，具有了一定程度的调试能力，那么，重点就在于提高解决问题的能力了。初学编程，对于问题解决这一块，并没有充分展开，所有的问题解决都是围绕着编程技巧和编程方法的学习展开的，所以，书中有意识、有目的地罗列一些适度的可以综合一些编程技巧加以解决的问题。编程进一步的学习，会涉及数据结构、算法以及程序质量保证方面的系统学习，那是纯粹为了提高设计和解决问题能力的，对于应用领域的开发，那是功到自然成的事。

5. 编排特色

1）比照C代码

采用作者之前的畅销书《C++程序设计教程（修订版）——设计思想与实现》（2009年清华大学出版社出版）的做法，经常比照C程序实现。这一方面是因为C++本身包容C，即C程序在C++编译中合法；另一方面，C代码本身就是C++编程中的一种可选方法，对于处理简单的结构和数据问题时，C虽然失去了代码安全性和简捷性，但却具有一定的性能优势。

在学习过程中，随时比照C语言，可以饮水思源，了解C++的改进缘由和特征，也不致因为C++的优越感而迷失。

2）习题与例题

本书十分倚重可有效验证代码正确性和测试有效性的问题描述形式。这种形式源自ACM国际大学生程序设计竞赛。这种形式有利于分析问题和解决问题能力的培养。

问题描述形式是对问题的描述，包括题目、问题基本描述、输入数据描述、输出数据描述、样本输入描述和样本输出描述。这种描述能够无二义性地把握问题，准确定位数据处理要求，从而得到唯一的计算结果。

习题与例题如果在题目后面标上打了括号的数字，表示在浙江工业大学的程序设计学习平台 cpp.zjut.edu.cn 的 problems 选项页中能找到其对应的实验题目，以给读者提供练习的环境。

3）章首和章末

每章首均以精炼的文字描述本章所要讲述的内容及其意义。

每章结束时，亦以生动的文字，给出本章内容的概括性描述，指出每节内容相互之间的关联以及本章在全书中的重要程度，以指导学生进一步学习。

4）观点加注与评鉴

本着对教科书负责的态度，作者在此处声明，本书带有许多对C++的个人见解。较之作者以前出版的书，更多了些个人的观点，并且为了描述的通俗性，对术语作了多样性描述。虽然一些特别之处已经作了加注，但更多的观点或见解融合在了书中的字里行间，无法分离。请读者见谅。

作者认为，不同的观点表达了不同的教学和学习场景。本书内容只是作者自身的不间断C++教学经验和经历的凝炼，但书中的观点应该是更有利于读者的学习，至少是初始学习。

从导读开始，作者就在发表个人观点了。课程教学中理论与实践教学比例的争执在许多大学都存在，本书不予评论；课程内容讲什么，不讲什么，强调什么等等，都是教学经验问题；谈论课程教学，总是论及体系问题，但实际操作又有许多具体问题，何况对于C++技术与内涵，存在争议也很自然。欢迎广大教师和读者来函讨论。

5）代码风格

书中全部代码都是作者亲手在VC 8和BCB 6两个环境中分别调试的。

作者的代码风格是一如既往的，本书也不例外。

代码中的控制语句总是以锯齿形方式书写，以表达控制的结构。有些花括号对是可以省略的，视可读和优雅性而为之。

代码头上的预编译部分以斜体列出，并以注释虚线与下面的函数正文代码分割。每个函数之间也分割以注释虚线。语言中的关键字以黑体字标出。

书中可以运行的完整代码，总是以注释行引导，说明该代码的功能、名称、章节和序号。例如下列代码：

```
//=====================================
// X0201.cpp
```

```
//一个@字符矩形
//=====================================
#include<iostream>
using namespace std;
//-------------------------------------
int main()
{
 int n;
 cin>>n;                          // 获取循环次数
 for(int i=1; i<=n; i++)          // 循环n次
 {
  cout<<"@@@@@@@@@@\n";
 }
}//=====================================
```

代码名称 X0201.cpp 中的 02 表示第 2 章，01 表示该章中第一个可运行代码。

6）超前引用

书中经常会出现一些名称和实例，它们在后续章节才解释其概念。

所有这些名称都会有一个超前引用标志，指向后面的章节。例如，“字节（☞CH3.1.3 字节）”，表示后面第 3.1.3 小节中有关于“字节”概念的介绍。

一个概念要靠一组概念来描述，因而需要预先展开一组概念的描述。这个格局往往会嵌套追究下去，或者产生彼此依赖的死结。流行的描述方式是，先引用一个超前概念用来描述和理解当前的问题，然后在读者希望了解的任何时候，循着超前引用的指引，去展开理解超前概念。

本书虽然是在《C++程序设计教程（第二版）》（2005 年清华大学出版社出版）基础上进行的深化描述，但是有些描述还是与该书互相补充，要到该书上才能找到那些概念或实例。

6. 成书缘由

许多高校教师，将作者的《C++程序设计教程（第二版）》一书用作教材，来信讨论编程的教学思路，交流教学经验以及对教材使用的感受。作者也从自身的教学实践中，体会到《C++程序设计教程（第二版）》作为教材的某些内容需要改进，某些内容需要展开或细述。

许多学生的提问富有见解，特别是其中曾经获得 ACM 编程竞赛大奖，目前颇有成就的学生金冠卓、金天鹏、金启为、徐立、夏超伦、许恬菁、金顺敬、朱佳一、王跃、金波、陈益波、俞骋超等，给予了作者许多忠告并提供了自己积累的宝贵资料。

作者还通过认真回答网上读者专门请教编程诸问题的来信，形成了对许多问题的新思考、新视角。

上述种种原因促使了作者决定开始写这样的一本书，来细述《C++程序设计教程（第二版）》中没有展开的编程方法细节。

本书写完之后，一定还有些内容有待于更进一步完善，有些新知识又需要新的细述与展开。在《C++程序设计教程（第二版）》出版之后，总是不断有全国各地的读者来信给予

高度评价，作者感到欣慰之时也非常感谢读者的关心和支持，也欢迎读者来信探讨任何编程问题。书中叙述不到之处也请读者见谅。

本书也得益于作者所处的课程教学环境，它来自于学校和学院所营造的课外学习氛围。其中，每年举行的 C++程序设计学术竞赛，每月举行的程序设计竞赛，每年举行的 ACM 程序设计校赛、省赛、亚洲区域赛与国际赛，都在深深地激发着学生的学习兴趣。作者曾亲历指导 ACM 程序设计竞赛与培训，并因此而长期采纳 ACM 竞赛方式的考试作为课程教学的重要环节，激发学习兴趣，让学生边学习边评估，以从编程中获得最大程度的快乐。

本书还离不开课程实践环境 cpp.zjut.edu.cn 的支撑，因为课内外实验与考试都是在该平台上。很多学生从中得到了能力的训练与提高。作者一直致力于丰富题库的日常建设，本书的许多例子和每章后面的习题，都是从该环境的题库中提炼而来。周楚、杨玲、曹瑞杰这三位学生为题库建设做了许多有益的工作。并且，这个环境能够一直很好地稳定工作，得益于单位同事陈波和刘端阳两位计算机系统和网络专家的奉献。

作者尤其要感谢人生导师王国东先生的帮助和关怀，写书过程也是获得其智慧与灵感加持的过程。作者因而经常处于深度探究 C++的超然状态，每次写书，都能心绪宁静、思路敏捷与灵感迭现。书中诸多精彩无疑是在这样的写作状态下完成的。

作者电子邮件地址：qianneng@mail.hz.zj.cn。

钱 能

于杭州自在居

2013 年 12 月

目　录

第一部分　初涉编程

第1章　编程认识……3
1.1　编程语言……3
1.1.1　语言与编程……3
1.1.2　程序翻译……5
1.1.3　语言影响编程……7
1.2　操作与环境……9
1.2.1　文件种类……9
1.2.2　操作流程……10
1.2.3　控制台编程……11
1.2.4　C++环境……12
1.3　从最小程序切入……13
1.3.1　带输出的最小程序……13
1.3.2　程序解释……13
1.3.3　比照C程序……15
1.3.4　实现与实践……16
1.4　语句编排……16
1.4.1　程序构成要素……16
1.4.2　变量与处理……20
1.4.3　空格与注释……23
1.4.4　编排技巧……24
1.4.5　正确性问题……29
1.5　重复执行问题……31
1.5.1　重复与循环……31
1.5.2　简单循环控制……32
1.5.3　重复执行语句集合……36
1.6　规范问题描述……37
1.6.1　问题求解中的循环控制……37
1.6.2　问题求解模式……38
1.6.3　循环输出字符矩形……39
1.6.4　循环输出……41
1.7　目的归纳……41
1.7.1　编程语言……41

1.7.2 操作与环境 ····· 42
1.7.3 从最小程序切入 ····· 42
1.7.4 语句编排 ····· 42
1.7.5 重复输出问题 ····· 42
1.7.6 规范问题描述 ····· 43
1.8 练习 ····· 43
A. 一个@字符矩形（1163） ····· 43
B. 一个#字符正方形（1164） ····· 44
C. 一个字符三角形（1165） ····· 44
D. 正方形面积（1167） ····· 45
E. A–B（1166） ····· 45
第 2 章 过程控制 ····· 46
2.1 简单 for 循环 ····· 46
2.1.1 for 循环结构 ····· 46
2.1.2 次数控制方式 ····· 49
2.1.3 输入控制方式 ····· 51
2.2 分支语句 ····· 53
2.2.1 if 语句 ····· 53
2.2.2 switch 语句 ····· 57
2.2.3 if 与 switch ····· 61
2.3 循环语句 ····· 67
2.3.1 for 循环嵌套 ····· 67
2.3.2 while 循环 ····· 71
2.3.3 for 与 while ····· 77
2.3.4 do-while 循环 ····· 82
2.4 转移语句 ····· 90
2.4.1 break 语句 ····· 90
2.4.2 continue 语句 ····· 94
2.4.3 goto 语句 ····· 97
2.5 初涉函数 ····· 100
2.5.1 函数使用三部曲 ····· 100
2.5.2 分类与表达 ····· 104
2.6 输入/输出初步 ····· 106
2.6.1 标准输入/输出 ····· 106
2.6.2 输入流 ····· 108
2.6.3 输出流控制 ····· 108
2.6.4 printf 格式控制 ····· 113
2.6.5 scanf 格式控制 ····· 119
2.7 目的归纳 ····· 122

2.7.1　简单 for 循环 …… 122
2.7.2　分支语句 …… 122
2.7.3　循环语句 …… 122
2.7.4　转移语句 …… 123
2.7.5　初涉函数 …… 123
2.7.6　输入/输出 …… 123
2.7.7　归纳 …… 123
2.8　练习 …… 124
A. 字母三角形 （1138） …… 124
B. 字符菱形（1169） …… 125
C. 背靠背字符三角形（1170） …… 125
D. 交替字符倒三角形（1171） …… 126
E. 格式阵列一（1172） …… 127
F. 格式阵列二（1173） …… 127
第 3 章　数据类型 …… 129
3.1　整型 …… 129
3.1.1　内部表示原理 …… 130
3.1.2　整型长度与范围 …… 133
3.1.3　整型数 …… 136
3.1.4　整型操作 …… 138
3.1.5　整数问题 …… 140
3.2　整型子类 …… 145
3.2.1　字符型 …… 145
3.2.2　枚举型 …… 150
3.2.3　布尔型 …… 152
3.3　浮点型 …… 153
3.3.1　浮点型数 …… 153
3.3.2　十进制小数 …… 158
3.3.3　32 位浮点数 …… 160
3.3.4　三种浮点型 …… 167
3.3.5　浮点数问题 …… 170
3.4　C 字串 …… 174
3.4.1　C 串表示 …… 174
3.4.2　C 串操作 …… 178
3.5　string 字串 …… 186
3.5.1　string 串 …… 186
3.5.2　string 流 …… 194
3.6　输入方式 …… 199
3.6.1　基本输入 …… 199

3.6.2 文件输入 ····· 200
3.6.3 控制台输入 ····· 203
3.6.4 标准键盘输入 ····· 204
3.6.5 读入行 ····· 205
3.6.6 读入字符 ····· 207
3.7 目的归纳 ····· 208
3.7.1 整型数 ····· 209
3.7.2 浮点型 ····· 209
3.7.3 C 串 ····· 209
3.7.4 string ····· 210
3.7.5 输入方式 ····· 210
3.8 练习 ····· 210
A. 1!到 n!的和（1174） ····· 210
B. 等比数列（1175） ····· 211
C. 平均数（1179） ····· 211
D. 级数求和（1180） ····· 211
E. 整数内码（1183） ····· 212
F. 算菜价（1700） ····· 212
G. 去掉双斜杠注释（1216） ····· 213

第二部分 拓 展 技 巧

第 4 章 数组与结构 ····· 217
4.1 一维数组 ····· 217
4.1.1 定义与初始化 ····· 217
4.1.2 空间与元素 ····· 221
4.1.3 数组意义 ····· 223
4.2 一维数组应用 ····· 225
4.2.1 可用数组解决 ····· 225
4.2.2 用数组更高效 ····· 232
4.2.3 通常用数组解决 ····· 237
4.3 多维数组 ····· 243
4.3.1 定义与初始化 ····· 243
4.3.2 多维数组应用 ····· 246
4.4 结构 ····· 252
4.4.1 结构概念 ····· 252
4.4.2 结构定义 ····· 253
4.4.3 结构操作 ····· 254
4.5 结构数组 ····· 258
4.5.1 定义与使用 ····· 258