

高等学校计算机科学与技术教材

# C++同构化对象程序设计原理

(向下兼容C语言结构程序设计)

周启海 著



清华大学出版社  
<http://www.tup.tsinghua.edu.cn>



北方交通大学出版社  
<http://press.njtu.edu.cn>



高等学校计算机科学与技术教材

全国教育科学规划教育部重点项目标志成果

本书得到西南财经大学学术专著出版基金资助

# C++同构化对象程序设计原理

(向下兼容 C 语言结构程序设计)

周启海 著

清华大学出版社

北方交通大学出版社

• 北京 •

## 内 容 简 介

本书是一本反映当今中国和世界教育创新历史潮流、时代要求与学科特点的原创性的专著，是全国教育科学规划教育部重点标志成果。它创立了“算法设计同构化，程序编码对象化（旧称面向对象）”新理论，突出了“算法设计为主导，C++编程为主体”新思想，创造了“既授人以鱼，更授人以渔”新方法，推出了“理论联系实际，算法制导程序，上机验证编程”新模式，独创了“思想新颖，功能完备，技术先进，使用简便”的新工具——算法周码。

全书共 13 章，内容包括：引论；顺序结构对象化设计；选择结构对象化设计；循环结构对象化设计；子算法结构对象化设计；用户定义类型与 C++/C 语言预处理；数组结构对象化设计；记录结构对象化设计；指针结构对象化设计；文件结构对象化设计；继承结构对象化设计；多态结构对象化设计；容错处理与异常处理。各章均向下兼容 C 语言结构化程序设计。

本书可供广大计算机初学者、使用者、爱好者、应用者、提高者、研究者阅读；可作为大专院校各专业 C++ 程序设计、面向对象程序设计等有关课程的教材或教学参考书；也可作为全国各地计算机等级考试、各类培训班的学习用书。

版权所有，翻印必究。

本书封面贴有清华大学出版社激光防伪标签，无标签者不得销售。

## 图书在版编目（CIP）数据

C++同构化对象程序设计原理 / 周启海著. —北京：清华大学出版社；北方交通大学出版社，2004.1

（高等学校计算机科学与技术教材）

ISBN 7-81082-199-7

I. C… II. 周… III. C 语言 — 程序设计 — 高等学校 — 教材 IV. TP312

中国版本图书馆 CIP 数据核字（2003）第 085580 号

责任编辑：谭文芳

出 版 者：清华大学出版社 邮编：100084 电话：010-62776969

北方交通大学出版社 邮编：100044 电话：010-51686045, 62237564

印 刷 者：北京鑫海金澳胶印有限公司

发 行 者：新华书店总店北京发行所

开 本：787×1092 1/16 印张：31 字数：793 千字

版 次：2004 年 1 月第 1 版 2004 年 1 月第 1 次印刷

印 数：1~5 000 册 定价：39.00 元

# 前言

## 1. 重要价值

本书是反映当今中国和世界教育创新历史潮流、时代要求与学科特点,具有开拓性、前瞻性、原创性的专著;是全国教育科学规划教育部重点项目“21 世纪高校程序设计教学改革与教育创新研究”(批准号 DIA010319)的标志成果;具有重要学术理论价值与重大社会应用价值,并具备“跨出国门,走向世界”的优秀潜质。

它填补了国内外程序设计教育研究空白,有利于提高我国高校程序设计课程的教学质量、水平,提升我国高等教育的国际地位、学术影响,促进高校学生创新意识、创新能力的培养,进一步丰富和深化现代课程论、教学论的研究、发展和创新。

## 2. 背景简介

计算机教育学,是事关创新人才“以计算机程序设计为基础的计算机应用能力”基本素质培养的重要新兴交叉学科。但迄今为止,这门全球计算机科学家几乎无暇顾及而各国教育家又很少给予关注的学科,远远落后于其母源学科——计算机科学与教育科学的发展进步,在不少方面还是空白。这就使得国内外高等院校的程序设计课程,一直沿用“发展迟滞、默守陈规、缺乏创新”的落后教学模式(包括其教育思想、教学内容、教材建设、教学方法、教学方式、教学手段等)。在教学上长期存在着三大根本弱点。

一是“移花不接木”——只着眼于把程序设计技术成果之花,直接生搬硬套地简单移用为程序设计课程教学内容之木,而未能在教学中实现“从计算机科学的技术成果到计算机教育学的课程内容”的整合、转化和创新。因而,不能顺应高校学生的认知规律和满足师生的教学需求。

二是“授鱼不授渔”——仅局限于用某种当时流行的计算机语言(如 BASIC, Pascal, FORTRAN, C 或 C++)展开程序设计教学,而未能在教学中揭示各种计算机语言程序设计的共性本质。因而,不能使高校学生掌握可指导各种计算机语言程序设计的共同原理,做到“举一反三”。

三是“叶落不知秋”——总拘泥于把面向过程的结构程序设计教学与面向对象的对象程序设计(注意:从“非结构化→结构化→对象化”的程序设计发展史来看,面向对象程序设计宜称对象化程序设计,简称对象程序设计)教学硬性割裂开来,分成互不相干的两部分(如国内外现行做法不是分别作为各自独立的两门课程、两种教材,就是在同一门课程、一本教材中作为各自独立的两大内容),而未能在教学中体现本应使结构程序设计与对象程序设计有机统一的客观规律。因而,不能使高校学生从对象化的科学视角来认识、学习和掌握结构程序设计与对象程序设计的辩证统一,浪费了高校教学资源,降低了国家教育效益。

## 3. 锐意创新

有哲理,才有理念;有理念,才有思想;有思想,才有分析;有分析,才能比较;有比较,

才能鉴别；有鉴别，才能发展；有发展，才有未来。作者历来认为：科学的创新理念哲学思考是追求科学创新的哲学理念前提。

1986年，作者提出“凡同类事物，客观上总存在着使其结构本质与构造实质趋于或达到一致的自然规律”的同构化基本原理。2000年，提出“凡异类事物，客观上总存在着使其结构本质与构造实质趋于或达到不同的自然规律”的异构化基本原理。2001年，提出“凡事物，客观上既存在着使其结构本质与构造实质趋于或达到一致的同构化自然规律，也存在着使其结构本质与构造实质趋于或达到不同的异构化自然规律”的特构化基本原理。

依据同构化、异构化、特构化三大原理，从容直面国内外计算机教育的历史和现状，锐意进行计算机教育学理论与实践相结合的反思、研究、探索和创新。

针对“移花不接木”弱点，作者坚持“科技上合理的事物未必在教育上也同样合理”的唯物辩证观，深刻认识“计算机教育学=计算机科学+教育科学”的交叉学科特色，正确运用认知论、课程论、教学论相结合的研究方法，认真研究程序设计技术与程序设计教学的根本区别、内在联系、交叉融合与学科转化，积极探索从程序设计技术历史成就向程序设计教学现实需要的交融创造和转化创新。

针对“授鱼不授渔”弱点，作者坚持“特殊与一般、共性与个性”的辩证统一观，探索和揭示寓于各种计算机语言程序设计中的、可指导各种计算机语言程序设计的共性本质和同构原理，进而创著出能适应当代科技进步与终身教育所要求的“举一反三”的专著型新一代程序设计创新教材。

针对“叶落不知秋”弱点，作者坚持“今天的程序设计教学内容、教材架构，绝不是程序设计进步历史及其轨迹的简单克隆、剪裁拼盘、直接翻版，而应是更深化的再认识与再创造”的历史统一观，研究和探求可使结构程序设计自然、合理、科学地融入对象程序设计的创新方法和认知捷径，使高校学生能直接从对象化的统一视角来深刻认识、快捷掌握结构程序设计与对象程序设计，以提高高校师生教学效能与学生创新能力。

## 4. 主要特色

本书具有如下主要原创性特色。

(1) 基础坚实——创立了“算法设计同构化，程序编码对象化”新理论。

从根本上讲，所有计算机语言都同构于同一种同构化算法设计语言，并且这一同构化算法设计语言宛如各种计算机语言的“世界语”，或者恰似我国的“普通话”。同构化算法设计语言是所有计算机语言的公共原象，而且这一公共原象是从它的各种映象——所有计算机语言中抽象、简化、浓缩、提炼、升华而成的。同医学中人体标本与人的身体的构造关系、数学中 $n$ 维欧氏空间与任何一个 $n$ 维空间的数学结构关系相类似，用同构化算法设计语言（例如作者先后创造的NS周图、周图、算法周码）表述的同构化算法与它所对应的任何一种具体计算机语言编写的程序之间的精神内核关系是同构的，即它们的结构本质与构造实质是完全一致的。同样，所有计算机语言下的程序设计，显然同构于用同构化算法设计语言表征的同构化程序设计。所以，从使“结构程序设计对象化与对象程序设计同构化”有机结合、和谐统一起来的同构化程序设计入手学习和把握程序设计，无疑是计算机程序设计普及、推广、应用的新捷径。（顺便指出，多年来，接受作者首创的这种先进教学法教育的学生们，其共同的亲身体会是：从算法到编码，其简单容易犹如将具体数据代入数学公式一样！）

(2) 思想科学——突出了“算法设计为主导，语言编程为主体”新思想。

在计算机科学中，“计算机程序设计=算法设计+计算机语言”，而“算法设计=算法+数据结构+程序设计方法学”，且其核心和基础是算法。这表明，“算法乃程序之母，程序为算法之子”。算法是“源”，是程序行为的主导思想；而程序是“流”，是算法描述的主体形式（因为程序只不过是用计算机语言描述的算法，即算法的最终表现形式而已）。因此，抓住了算法设计这个核心，就等于抓住了程序设计的灵魂！所以，在程序设计教学中，作者倡导突出“算法设计为主导，语言编程为主体”的新思想。

(3) 方法独到——创造了“既授人以鱼，更授人以渔”新方法。

古人云：“授人以鱼，不如授人以渔”。同理，让人们学懂某种计算机语言的程序设计，远不如教会人们掌握可指导各种计算机语言程序设计和程序移植的算法设计的基本原理和一般方法。所以，应当实行“既授人以鱼，更授人以渔”的新方法，以培养和形成学生的程序设计综合能力。为此，必须采用先“注重算法构造，突出算法设计；强化算法注释，点拨算法疑难”，再“结合某种语言（例如 C++），针对具体算法（例如秦九韶算法），实施程序编码（例如编写秦九韶 C++ 程序），上机调通程序”的新方法。

(4) 模式新颖——推出了“理论联系实际，算法制导程序，上机验证编程”新模式。

长期实践表明，这种新模式可切实做到授人以开启计算机程序设计宝库的金钥匙——进行计算机程序设计和程序移植的基本原理、一般方法与基础技术，使学生从根本上具备当今信息化社会所必需的计算机文化基本素质——“举一反三，触类旁通；各种语言，无师自通”的同构化对象程序设计应变能力和适应能力。

(5) 工具先进——独创了“思想新颖，功能完备，技术先进，使用简便”的新工具。

作者继 1986 年、1999 年先后创造出描述算法的 NS 周图、周图后，2001 年进一步新创出同构化算法设计描述新工具——算法周码。从根本上极大方便和提高了算法（及其程序）的设计效率。

大量事实证明：以同构化算法设计为指南来指导所有具体计算机语言下的程序设计，不但在思想上是先进的，在理论上是可靠的，在方法上是可行的，在模式上是创新的，在工具上是先进的，而且在实践上是成功的。毋庸置疑，这一新思路和新捷径，必将有利于消除我国同发达国家在计算机普及、推广、应用上的差距。

总之，本书是学生良师、教师益友、读者知音，它以出类拔萃的新面目、新体系、新方式、新效果，为广大读者开辟出一条易学好教、省力省时学习与应用对象化程序设计的新捷径！

## 5. 本书内容

本书创意新颖、体系独特，深入浅出、易学好用，注重实用、示例丰富（例题共 200 道，习题约 300 道），轻重缓急、安排得当，言简意赅、启迪性强。首创“同构化指导对象化，对象化涵盖结构化，结构化服务对象化”的科学方法，独创“最新 VC++ 作为上机实习平台，主流 C++ 作为编程主体语言，也向下兼容 C 语言基本内容”的特殊优势。全书共 13 章，内容包括：引论；顺序结构对象化设计；选择结构对象化设计；循环结构对象化设计；子算法结构对象化设计；用户定义类型与 C++/C 语言预处理；数组结构对象化设计；记录结构对象化设计；指针结构对象化设计；文件结构对象化设计；继承结构对象化设计；多态结构的对象化设计；容错处理与异常处理。各章均向下兼容 C 语言结构化程序设计。

## 6. 读者对象

无论是高等院校师生、工程技术人员、社会各界读者，也无论是编程新手、老手、高手，本书都能使之“雅俗共赏，各取所需”。

(1) 对毫无编程知识的新手，本书至少有 4 种可供选择的“零”入门方式（不含上机时间）。

- ① 选学本书第 1~5 章，需 35~45 小时，可升跃到“初级”对象化编程水平。
- ② 选学本书第 1~8 章，需 45~55 小时，可升跃到“中级”对象化编程水平。
- ③ 选学本书第 1~10 章，需 55~65 小时，可升跃到“准高级”对象化编程水平。
- ④ 全学本书第 1~13 章，需 90~100 小时，可升跃到“高级”对象化编程水平。

(2) 对已有编程经验的老手，本书起码可使其升级到“完美”对象化编程天地，沿着程序设计质量提高的升华轨迹——对→好→巧→妙→绝，去努力探索和寻求解决同一问题的算法的最优算法。

(3) 对颇有编程造诣的高手，本书完全可助其升华到“完备”对象化编程境界，即探索和实现“异构编程语言平台的同构化规范编程技术与方法，基于同构化的异构编程语言平台的对象程序设计自动化编程与自动化移植……”。

因此，本书可供广大计算机初学者、使用者、爱好者、应用者、提高者、研究者阅读，可作为大专院校各专业 C++ 程序设计、面向对象程序设计等课程的教材或教学参考书，也可作为全国各地计算机等级考试、各类培训班等的学习用书。

常言道：人无完人，金无足赤。故本书倘有疏误，则恳请读者不吝赐教指正，以便再版时予以更正。来信请寄：(610074) 成都市西南财经大学经济信息工程学院周启海教授；电话联系：028-87354172；E-mail: [zhouqh@swufe.edu.cn](mailto:zhouqh@swufe.edu.cn) 或 [zqhhqz@126.com](mailto:zqhhqz@126.com)。

谨以此书，献给我的母亲、家人、母校、祖国！

作 者

2004 年 1 月于西南财经大学

# 目 录

|                                   |      |
|-----------------------------------|------|
| 第 1 章 引论                          | (1)  |
| 1.1 计算机及其进步概要                     | (1)  |
| 1.1.1 计算机及其应用概要                   | (1)  |
| 1.1.2 计算机语言、算法、程序、程序设计概要          | (1)  |
| 1.1.3 程序设计进步简评                    | (2)  |
| 1.2 类、对象及其初始化、善后化行为简介             | (3)  |
| 1.2.1 类                           | (3)  |
| 1.2.2 对象                          | (4)  |
| 1.2.3 对象的最基本行为属性——初始化与善后化         | (4)  |
| 1.3 C++及其进入、工作、退出简述               | (5)  |
| 1.3.1 C++简述                       | (5)  |
| 1.3.2 Visual C++ 6.0 的进入、工作、退出    | (5)  |
| 1.4 算法与 C++程序的总体结构、注释结构概述         | (14) |
| 1.5 C++的字符集与标识符                   | (16) |
| 1.5.1 C++字符集                      | (16) |
| 1.5.2 C++/C 语言标识符                 | (17) |
| 1.6 C++的数据类型与基本操作概要               | (19) |
| 1.6.1 计算机的数据类型及数据形态总述             | (19) |
| 1.6.2 C++数据的数据类型及数据形态综述           | (19) |
| 1.6.3 C++/C 语言数据的最基本操作——输入与输出     | (21) |
| 1.6.4 C++/C 语言变量的最基础操作——赋值        | (33) |
| 1.7 C++数据形态与运算概要                  | (35) |
| 1.7.1 整数类型数据的形态与运算                | (35) |
| 1.7.2 实数类型数据的形态与运算                | (50) |
| 1.7.3 字符型和字符串型数据的形态与运算            | (52) |
| 1.7.4 (准)逻辑型数据的形态与运算              | (60) |
| 1.7.5 特殊运算——取地址(&)与取指物(*)         | (65) |
| 1.8 C++/C 语言指针与 C++别名(引用)的定义与使用概述 | (66) |
| 1.8.1 C++/C 语言指针的定义方法与使用特点        | (67) |
| 1.8.2 C++/C 语言指针的基本运算             | (70) |
| 1.8.3 C++别名(即引用)的定义与使用            | (71) |
| *1.9 从对象化 C++退化到结构化 C++/C 语言编程示例  | (72) |
| 习题 1                              | (73) |

|                                        |         |
|----------------------------------------|---------|
| <b>第 2 章 顺序结构对象化设计</b>                 | ( 77 )  |
| 2.1 算法及其程序的控制结构与分类                     | ( 77 )  |
| 2.1.1 传统控制结构                           | ( 77 )  |
| 2.1.2 对象控制结构                           | ( 77 )  |
| 2.1.3 程序控制结构                           | ( 77 )  |
| 2.2 类及其对象的简单定义、创建、拆除与使用                | ( 77 )  |
| 2.2.1 类的行为属性及其 C++ 函数成员的定义方式简介         | ( 78 )  |
| 2.2.2 对象的创建与初始化行为 (构造函数) 简介            | ( 83 )  |
| 2.2.3 对象的拆除与善后化行为 (析构函数) 简介            | ( 88 )  |
| 2.2.4 对象类与嵌套类用法简介                      | ( 89 )  |
| 2.2.5 当前对象指针 (this) 与当前对象 (*this) 用法简介 | ( 91 )  |
| 2.3 顺序结构设计                             | ( 92 )  |
| 2.3.1 顺序结构的特点                          | ( 92 )  |
| 2.3.2 顺序结构的描述                          | ( 93 )  |
| 2.3.3 始终结构的特点与描述                       | ( 94 )  |
| *2.4 从对象化 C++ 退化到结构化 C++/C 语言编程示例      | ( 99 )  |
| 习题 2                                   | ( 100 ) |
| <b>第 3 章 选择结构对象化设计</b>                 | ( 103 ) |
| 3.1 构成块——设计非顺序控制结构的关键                  | ( 103 ) |
| 3.1.1 构成块的特点                           | ( 103 ) |
| 3.1.2 构成块的描述                           | ( 103 ) |
| 3.2 选择结构概要                             | ( 104 ) |
| 3.3 双分支选择结构设计                          | ( 105 ) |
| 3.3.1 如果条件型双分支选择结构设计                   | ( 105 ) |
| 3.3.2 情况条件型双分支选择结构设计                   | ( 113 ) |
| 3.4 多分支选择结构设计                          | ( 116 ) |
| 3.4.1 如果条件型多分支选择结构设计 (if 嵌套)           | ( 117 ) |
| 3.4.2 情况条件型多分支选择结构设计 (switch)          | ( 125 ) |
| 3.4.3 混合条件型多分支选择结构设计 (if, switch 嵌套)   | ( 131 ) |
| *3.5 从对象化 C++ 退化到结构化 C++/C 语言编程示例      | ( 132 ) |
| 习题 3                                   | ( 134 ) |
| <b>第 4 章 循环结构对象化设计</b>                 | ( 139 ) |
| 4.1 循环结构概要                             | ( 139 ) |
| 4.2 单重循环结构设计                           | ( 139 ) |
| 4.2.1 当型单重循环结构设计                       | ( 139 ) |
| 4.2.2 直到型单重循环结构设计 (do-while)           | ( 152 ) |
| 4.2.3 步长型单重循环结构设计 (for)                | ( 161 ) |

|                                                   |                |
|---------------------------------------------------|----------------|
| 4.2.4 强制缩短与中止循环过程设计 (break, continue, goto) ..... | ( 172 )        |
| 4.2.5 三种循环结构的比较 .....                             | ( 176 )        |
| 4.3 双重循环结构设计 .....                                | ( 177 )        |
| 4.4 多重循环结构设计 .....                                | ( 182 )        |
| *4.5 从对象化 C++退化到结构化 C++/C 语言编程示例 .....            | ( 184 )        |
| 习题 4 .....                                        | ( 186 )        |
| <b>第 5 章 子算法结构对象化设计 .....</b>                     | <b>( 193 )</b> |
| 5.1 模块化与子算法简介 .....                               | ( 193 )        |
| 5.1.1 模块化结构简介 .....                               | ( 193 )        |
| 5.1.2 子算法结构简介 .....                               | ( 193 )        |
| 5.2 过程子算法结构设计 .....                               | ( 197 )        |
| 5.2.1 过程子算法的定义与调用 .....                           | ( 197 )        |
| 5.2.2 无参过程子算法结构设计 .....                           | ( 199 )        |
| 5.2.3 有参过程子算法结构设计 .....                           | ( 206 )        |
| 5.2.4 子算法调用关系与定义位置 .....                          | ( 219 )        |
| 5.2.5 标识符的作用域、生存期与误使用 .....                       | ( 220 )        |
| 5.2.6 静态变量与静态构成属性 .....                           | ( 223 )        |
| 5.2.7 单对象类与限数对象类 .....                            | ( 227 )        |
| 5.2.8 局部对象与全局对象 .....                             | ( 228 )        |
| 5.2.9 类的友元与初始化列表 .....                            | ( 231 )        |
| 5.3 函数子算法结构设计 .....                               | ( 235 )        |
| 5.4 嵌套、递推与递归 .....                                | ( 240 )        |
| 5.4.1 嵌套 .....                                    | ( 240 )        |
| 5.4.2 递推 .....                                    | ( 241 )        |
| 5.4.3 递归 .....                                    | ( 243 )        |
| 5.5 过程参数与函数参数 .....                               | ( 254 )        |
| 5.6 中、大型程序开发方法 .....                              | ( 257 )        |
| 5.6.1 中型程序开发方法——分立文件式设计 .....                     | ( 257 )        |
| 5.6.2 大型程序开发方法——文件集成式设计 .....                     | ( 260 )        |
| 5.7 命令行参数与操作系统可执行文件处理 .....                       | ( 260 )        |
| *5.8 从对象化 C++退化到结构化 C++/C 语言编程示例 .....            | ( 261 )        |
| 习题 5 .....                                        | ( 263 )        |
| <b>第 6 章 用户定义类型与 C++/C 语言预处理 .....</b>            | <b>( 267 )</b> |
| 6.1 用户定义类型及其变量定义的一般模式 .....                       | ( 267 )        |
| 6.2 枚举结构对象化设计 .....                               | ( 268 )        |
| 6.2.1 枚举型的类型定义与变量定义 .....                         | ( 268 )        |
| 6.2.2 枚举型量的运算与使用 .....                            | ( 269 )        |

|                                  |       |
|----------------------------------|-------|
| 6.3 C++/C 语言预处理的基本特点与使用用法        | (274) |
| 6.3.1 预处理、核心语言、标准库的主要作用与基本特点     | (275) |
| 6.3.2 文件包含命令                     | (275) |
| 6.3.3 宏处理——宏定义和宏替换               | (276) |
| 6.3.4 条件编译(选择编译)                 | (278) |
| *6.4 从对象化 C++退化到结构化 C++/C 语言编程示例 | (279) |
| 习题 6                             | (282) |

|                                  |       |
|----------------------------------|-------|
| <b>第 7 章 数组结构对象化设计</b>           | (284) |
| 7.1 数组及其下标变量的定义与使用               | (284) |
| 7.1.1 数组及其下标变量的有序化与宜用原则          | (284) |
| 7.1.2 数组型与数组的定义                  | (284) |
| 7.1.3 下标变量和下标的定义                 | (287) |
| 7.1.4 指针与数组、下标变量的联系              | (287) |
| 7.1.5 数组和下标变量的运算与使用              | (287) |
| 7.2 一维数组与单下标变量的应用                | (293) |
| 7.2.1 顺序存储与顺序标记                  | (294) |
| 7.2.2 下标变量型计数器、累加器与累乘器           | (297) |
| 7.2.3 排序分类                       | (301) |
| 7.2.4 名次排列                       | (309) |
| 7.2.5 查找检索                       | (310) |
| 7.2.6 有序插入                       | (314) |
| 7.2.7 有序删除                       | (316) |
| 7.3 二维数组与双下标变量的应用                | (317) |
| 7.4 多维数组与多下标变量的应用                | (323) |
| 7.5 一维字符型数组的字符串特殊用法示例            | (324) |
| 7.6 数组指针与指针数组的应用方法               | (326) |
| 7.6.1 数组指针应用示例                   | (327) |
| 7.6.2 指针数组应用示例                   | (328) |
| 7.7 数组对象与对象数组的应用方法               | (329) |
| 7.7.1 数组对象应用示例                   | (329) |
| 7.7.2 对象数组应用示例                   | (329) |
| 7.8 动态数组(new 和 delete)的定义与使用     | (331) |
| *7.9 从对象化 C++退化到结构化 C++/C 语言编程示例 | (333) |
| 习题 7                             | (335) |

|                        |       |
|------------------------|-------|
| <b>第 8 章 记录结构对象化设计</b> | (342) |
| 8.1 事物、实体与记录           | (342) |
| 8.2 记录的定义与使用           | (343) |

|               |                           |                |
|---------------|---------------------------|----------------|
| 8.2.1         | 记录及其字段的秩序化与宜用原则           | ( 343 )        |
| 8.2.2         | 记录型与记录的定义                 | ( 343 )        |
| 8.2.3         | 记录和字段的运算与使用               | ( 346 )        |
| 8.3           | 一般记录与位段记录的应用方法            | ( 347 )        |
| 8.3.1         | 一般记录应用示例                  | ( 347 )        |
| 8.3.2         | 位段记录应用示例                  | ( 350 )        |
| 8.4           | 数组记录与记录数组的应用方法            | ( 351 )        |
| 8.4.1         | 数组记录应用示例                  | ( 351 )        |
| 8.4.2         | 记录数组应用示例                  | ( 353 )        |
| *8.5          | 从对象化 C++退化到结构化 C++/C 编程示例 | ( 355 )        |
| 习题 8          |                           | ( 358 )        |
| <b>第 9 章</b>  | <b>指针结构对象化设计</b>          | <b>( 360 )</b> |
| 9.1           | 连通事物与指针结构                 | ( 360 )        |
| 9.1.1         | 连通事物                      | ( 360 )        |
| 9.1.2         | 指针结构=递归指针+递归指物            | ( 361 )        |
| 9.2           | 递归指针及其递归指物的定义与使用          | ( 362 )        |
| 9.2.1         | 递归指针型及其递归指针的定义            | ( 362 )        |
| 9.2.2         | 递归指针值和递归指物的定义与使用          | ( 363 )        |
| 9.3           | 指针结构在半动态、动态数据结构中的应用       | ( 368 )        |
| 9.3.1         | 栈的基本特性与类型定义               | ( 368 )        |
| 9.3.2         | 栈类的定义                     | ( 369 )        |
| 9.3.3         | 栈类的应用示例                   | ( 373 )        |
| *9.4          | 从对象化 C++退化到结构化 C++/C 编程示例 | ( 378 )        |
| 习题 9          |                           | ( 381 )        |
| <b>第 10 章</b> | <b>文件结构对象化设计</b>          | <b>( 384 )</b> |
| 10.1          | 原象数据、实体数据、内存数据与外存数据       | ( 384 )        |
| 10.2          | 文件及其数据的定义与使用              | ( 384 )        |
| 10.2.1        | 文件(类)型与文件变量(指针)的定义        | ( 384 )        |
| 10.2.2        | 文件的开启与关闭                  | ( 385 )        |
| 10.2.3        | 文件的创建与读写                  | ( 387 )        |
| 10.3          | 文件的应用                     | ( 395 )        |
| 10.3.1        | 文件的查看                     | ( 395 )        |
| 10.3.2        | 文件的复制                     | ( 395 )        |
| 10.3.3        | 文件的修改                     | ( 395 )        |
| 10.3.4        | 文件的增删                     | ( 395 )        |
| 10.3.5        | 文件的合并                     | ( 395 )        |
| 10.3.6        | 文件的排序                     | ( 396 )        |

|                                         |                |
|-----------------------------------------|----------------|
| 10.4 文本文件 .....                         | ( 396 )        |
| *10.5 从对象化 C++退化到结构化 C++/C 语言编程示例 ..... | ( 398 )        |
| 习题 10 .....                             | ( 401 )        |
| <b>第 11 章 继承结构对象化设计 .....</b>           | <b>( 404 )</b> |
| 11.1 演化衍生现象与类的继承派生 .....                | ( 404 )        |
| 11.2 继承性视角的类划分 .....                    | ( 404 )        |
| 11.3 类的继承关系与继承模式 .....                  | ( 405 )        |
| 11.3.1 类的继承关系 .....                     | ( 406 )        |
| 11.3.2 类的继承模式 .....                     | ( 409 )        |
| 11.3.3 类继承性的扩散效应、应用原则与简化描述 .....        | ( 414 )        |
| 11.3.4 类的继承表现域=继承关系域×继承模式域 .....        | ( 415 )        |
| 11.4 类及其对象的继承性特殊机制 .....                | ( 415 )        |
| 11.4.1 继承下的类初始化行为与善后化行为 .....           | ( 416 )        |
| 11.4.2 继承属性的子类中调整 .....                 | ( 419 )        |
| 11.4.3 多继承下的重名歧义性及其化解 .....             | ( 421 )        |
| *11.5 从对象化 C++退化到结构化 C++/C 语言编程示例 ..... | ( 425 )        |
| 习题 11 .....                             | ( 426 )        |
| <b>第 12 章 多态结构对象化设计 .....</b>           | <b>( 429 )</b> |
| 12.1 多样化现象与类的多态性概述 .....                | ( 429 )        |
| 12.2 子算法重载 (C++函数重载) .....              | ( 430 )        |
| 12.2.1 类外游离子算法的重载 (C++游离函数重载) .....     | ( 430 )        |
| 12.2.2 类中一般行为子算法的重载 (C++函数成员重载) .....   | ( 431 )        |
| 12.2.3 类中初始化行为子算法 (C++构造函数) 的重载 .....   | ( 432 )        |
| 12.2.4 类外友元子算法 (C++友元函数、友元类) 的重载 .....  | ( 438 )        |
| 12.3 运算符重载 .....                        | ( 438 )        |
| 12.3.1 运算符重载概述 .....                    | ( 438 )        |
| 12.3.2 类中行为方式的运算符重载 .....               | ( 439 )        |
| 12.3.3 类的友元方式的运算符重载 .....               | ( 443 )        |
| 12.3.4 类外游离运算符的重载 .....                 | ( 445 )        |
| 12.3.5 对象内部数据的信息泄漏与信息安全 .....           | ( 446 )        |
| 12.4 绑定多态性 .....                        | ( 446 )        |
| 12.4.1 子算法的绑定、绑定性与绑定多态性 .....           | ( 446 )        |
| 12.4.2 静态绑定的局限 .....                    | ( 447 )        |
| 12.4.3 动态绑定的实现——一般虚行为 (C++虚函数) .....    | ( 448 )        |
| 12.4.4 纯虚行为 (C++纯虚函数) 与抽象类 .....        | ( 449 )        |
| 12.5 类型多态性——模板及其模板类、模板子算法 .....         | ( 451 )        |
| 12.5.1 “常规多态性→类型多态性”的技术进步综述 .....       | ( 451 )        |

|               |                             |         |
|---------------|-----------------------------|---------|
| 12.5.2        | 模板                          | ( 452 ) |
| 12.5.3        | 模板类多态性实现轨迹                  | ( 452 ) |
| 12.5.4        | 模板子算法多态性实现轨迹                | ( 456 ) |
| *12.6         | 从对象化 C++退化到结构化 C++/C 语言编程示例 | ( 459 ) |
| 习题 12         |                             | ( 462 ) |
| <b>第 13 章</b> | <b>容错处理与异常处理</b>            | ( 465 ) |
| 13.1          | 容错处理 = 假想错处 + 排除错误          | ( 465 ) |
| 13.1.1        | 数据超限与运算违规的容错处理              | ( 465 ) |
| 13.1.2        | 数据溢出与精度损伤的容错处理              | ( 466 ) |
| 13.1.3        | 下标越界与文件误用的容错处理              | ( 466 ) |
| 13.2          | 异常处理 = 发现异常 + 处理异常          | ( 467 ) |
| 13.2.1        | 一般异常处理概述                    | ( 467 ) |
| 13.2.2        | C++异常处理概述                   | ( 468 ) |
| 13.2.3        | C++显式类型异常的捕获与处理             | ( 471 ) |
| 13.2.4        | C++隐式类型异常的捕获与处理             | ( 473 ) |
| 13.2.5        | C++函数的异常类型限定表与处理            | ( 475 ) |
| 13.2.6        | C++异常处理的高层延伸求援              | ( 475 ) |
| 13.2.7        | C++创建对象时的异常处理               | ( 476 ) |
| *13.3         | 从对象化 C++退化到结构化 C++/C 语言编程示例 | ( 477 ) |
| 习题 13         |                             | ( 479 ) |
| <b>附录 A</b>   | <b>ASCII 表</b>              | ( 480 ) |
| <b>参考文献</b>   |                             | ( 482 ) |

# 第1章 引 论

以计算机科学技术为龙头和核心的 21 世纪,人类生产、工作、生活、休闲的基本方式和主要特征是“社会信息化,经济知识化,世界网络化,竞争全球化”。无疑,学好、用好计算机,掌握至少一种计算机语言的程序设计基本技能,是当今社会人人都必须拥有的主要通行证,也是当代社会各类人才必须具备的基本素质之一。为此,本书以“与时俱进”的精神,对传统程序设计入门课程进行深层改革、全面改进和源头创新,即以“同构化指导对象化,对象化涵盖结构化,结构化服务对象化”的首创科学方法,用“最新 VC++作为上机实习平台,主流 C++作为学生编程主体语言,也向下兼容魅力渐逝的 C 语言基本内容”的先进教学模式,走“算法制导程序,上机验证编程,学用促成应用”的编程新路。从而,为无任何计算机知识的初学者尽快迈入现代程序设计殿堂敞开了大门。

## 1.1 计算机及其进步概要

计算机系统简称计算机,由计算机的硬件系统与软件系统共同构成,是人脑的仿生与延伸——“人脑的物化,物化的人脑”,并在不断进步中。

### 1.1.1 计算机及其应用概要

计算机硬件如计算机的躯体,它由存储器、运算器、控制器、输入输出设备和外部存储设备构成。从 1946 年第一台电子计算机诞生以来,计算机硬件已迅速经历了 4 代(即:电子管计算机、晶体管计算机、集成电路计算机、大规模和超大规模集成电路计算机)的历史飞跃,目前正向着多元化的第 5 代(即:量子、激光、分子、生物、智能计算机等)稳步前进。计算机硬件技术进步的社会影响在于:新一代计算机的问世,总预示着老一代计算机过时的开始。

计算机软件是所有计算机程序及其相关技术文件的总称。它如计算机的灵魂,并分为系统软件(如:操作系统、计算机语言、数据库系统……)与应用软件(如:工具软件、杀毒软件、用户应用软件……)。没有思维的人,只能是不会生活的“植物人”;没有软件的计算机,只能称为不能工作的“裸机”。由此可见,计算机的软件系统,远比硬件系统深刻得多、根本得多、重要得多。

迅猛发展中的计算机及其应用,正日益广泛地深入到人类社会的经济、生产、生活的各个方面,已经并将不断地极大影响和改变人类的生产方式、工作方式、生活方式、休闲方式。计算机应用通常可分为五大类:数值计算,数据处理,自动控制,辅助工作(如:辅助设计,辅助制造,辅助教育,辅助测试……)和人工智能。

### 1.1.2 计算机语言、算法、程序、程序设计概要

计算机从诞生之日起,就有了自己的专用语言——计算机语言(即程序设计语言),它是人类为了控制计算机正确工作而特殊构造的“形式语言”。它实际上是“人脑”与“电脑”相互沟通的联系纽带,是“人”与“计算机”进行双向交流的语言桥梁,是当今人类社会新文化——“计算机文化”的精神文明工具。目前,它已走过了 4 代(即:机器语言、汇编语言、高级语

言、非过程语言)的发展历程,现在正向着第5代(即:智能语言)积极迈进。应当明确指出,与计算机硬件的科技进步相比,计算机语言的科技进步具有全然不同的性质:新一代计算机语言的产生,只表明它为更多的计算机用户提供了一种更为便捷、高效的新语言编程工具,但通常并不意味着其他计算机语言历史使命的结束,因为每一代计算机语言(哪怕是产生最早的机器语言)至今仍各有其特殊专长与用武之地。

计算机算法简称算法,是人们为了借助计算机解决给定问题,而命令计算机按照“人一(计算)机”系统所认可的操作方式和控制方式,具体施行的有穷操作的有序集合。算法恰似计算机的“解题思想”,它颇类似于人们解决给定问题而实施的方法、想法、思路、规程、处方,等等。

计算机程序简称程序,是人们在设计出能控制计算机正确解决给定问题的算法基础上,进一步用选定的某种计算机语言,把算法翻译成计算机可接受、理解和执行的“计算机语言表现形式的算法”。换言之,程序是人交给计算机具体执行的工作任务书和行为指南针。应当强调指出:如同一事物(例如:学生),由于可用不同自然语言(例如:中文、英文、法文、俄文……)表述而有不同说法(例如:学生、student, etudiant, студент)那样,同一个算法,完全可以根据设计者的实际需要和编程(即用计算机语言编写程序的简称)爱好而采用不同的计算机语言来进行编码(即翻译),并得到不同计算机语言表现形式的程序。这些不同计算机语言的程序,其执行效果是完全一致的(例如,同一算法可以采用C++, VB, Delphi, VFP语言进行翻译,而其产物是同一“原文”——算法的四种语言的不同“译文”:C++程序、VB程序、Delphi程序、VFP程序)。由此可见,算法远比程序深刻得多、根本得多、重要得多,“算法,是程序之母;程序,是算法之子”。

赋予计算机灵魂和生机的计算机程序设计简称程序设计,是为计算机设计算法及其程序的知识型、高智能创造性劳动。程序设计是“人”驾驭“计算机”的基础环节和核心工作,是使“计算机”服务于“人类”的工作重心和成败关键,是人类新文明——“计算机文明”的基本建设、基础工程和知识氛围。因此,学习并掌握至少一种主流计算机语言的程序设计,对构筑当今人才和未来英才的完整知识结构,具有根本重要性。

### 1.1.3 程序设计进步简评

程序设计,已历经“非结构化(Nonstructured)→结构化(Structured)→对象化(Objectized)”的三大历史性技术进步。

非结构化程序设计发祥最早,为计算机的诞生与早期发展作出了不可磨灭的历史贡献。但它滥用GOTO结构控制程序流程,也为日后令人震惊的“软件危机”埋下祸根。

20世纪60年代,为解决由非结构化程序设计导致的“软件危机”而提出的结构化和对象化(注:作者认为——从程序设计发展史来看,采用术语“对象化”比“面向对象”(Object-Oriented)更为简练、精确地刻画出所谓“面向对象程序设计”的技术特色和历史地位,故本书倡导并只用此术语)程序设计思想,是程序设计发展史上最有影响的两大思想方法。

结构化程序设计(Structured program design,简称结构程序设计)发展较早,其方法的精髓是“基于功能,面向数据,结构驱动”,宜于设计小规模程序。它可概括为公式“结构程序设计=算法+数据结构+结构化软件方法学+结构化编程语言”;式中,加号表示结合,等号表示统一。当年,它为解决软件危机立下汗马功劳。

对象化程序设计 (Objectized program design, 简称对象程序设计, 即旧称面向对象程序设计) 是在结构程序设计基础上发展起来的, 其思想核心是“基于(子)算法, 面向对象, 消息驱动”, 长于中、大规模的程序设计。它可概括为公式“对象=数据结构+算法, 对象程序设计=(对象+对象+...+对象)+对象化软件方法学+对象化编程语言”; 式中, 括号表示优先。如今, 它正为软件发展开辟广阔道路, 已发展成为世界各国所公认的现代程序设计基本方法与主流方向。

应当强调指出: 在对象程序设计中, 对象的重心是“施加于封装在对象内部的对象数据(即对象的数据特性属性, 简称数据属性或数据元素)之上的对象行为(即对象的行为特性属性, 简称行为属性或行为元素)”; 而用以描述、处理、实现对象行为的程序设计基本方法, 其实质仍然是结构程序设计。由此可见, 结构程序设计与对象程序设计的内在关系为“结构程序设计, 是对象程序设计的底层基石; 对象程序设计, 是结构程序设计的上层建筑”。这表明, 现今程序设计已发展成结构程序设计与对象程序设计的有机融合, 任何把两者割裂开来甚至对立起来的观点和做法, 在理论上是错误的, 在实践上是有害的。因此, 作者原创性的“把结构程序设计有机融入对象程序设计”的新观点、新思路和新方法, 为初学者开辟了一条可直达“学好、用好对象程序设计”的新捷径。

此外, 也应当说明: 在程序设计经济学看来, 开发中、大规模程序, 对象程序设计无疑具有结构程序设计不可比拟的巨大优越性。但在设计小规模程序时, 采用对象程序设计好比“杀鸡焉用牛刀”, 因为此时, 它原本胜过结构程序设计的技术优势表现得并不显著或难以发挥, 其时空效率、研制成本有可能反不及结构程序设计而变得并不经济, 这正如“打麻雀, 用鸟枪比用导弹更合算”。不过, 为了简明、节俭、高效地讲述对象程序设计的基本原理、技术与方法, 本书所有示例尽管统统为小规模程序(因而解决它们并不是“非用对象程序设计不可”), 但却特许它们可例外: 不受上述程序设计经济学原则的约束, 而几乎一律采用对象化方法进行设计。

## 1.2 类、对象及其初始化、善后化行为简介

为了让没有任何计算机知识的初学者从学习程序设计开始, 就能直接以对象程序设计新视角学习结构程序设计思想, 并最大限度降低和实现“零”入门门槛, 本书前10章将尽可能地只使用对象化方法的封装性(注: 继承性和多态性将留待第11、12章讲述)与下述最基本核心概念。

### 1.2.1 类

“类”(Class), 是同类事物的人脑抽象与电脑映象, 是把数据结构与基于它的子算法封装集成在一起的新型数据类型, 是类的构成属性(即: 用以描述类所对应事物数据特性的数据属性, 与作用于这些数据属性的、用以描述类所对应事物行为特性的行为属性)统一体, 是对象程序设计的基石。按照类的构成属性是否完整, 类可分为: 既有数据属性又有行为属性的完整类, 与不全有数据属性、行为属性的退化类; 按照类有无直系血亲(指: 派生它的父类与它派生的子类), 类可分为: 无父类和子类的单独类, 与有父类或子类的非独类(包括: 始祖类、中继类、末代类)。