计算机科学 1999Vol. 26№. 4

C++作为计算机专业入门语言的实践与探讨

The Practice and Discussion of Making C++ as an Elementary Programming Language for Computer Speciality Students

李文军 李师贤 周晓聪
（中山大学计算机科学系 广州 510275）

Abstract In this paper we propose that object-oriented programming can be and should be an elementary language for computer speciality students. We put forward the overall objects, knowledge units, instruction schedule and necessary accessories of this primer course, and analyse the effects of our instruction practice.

Keywords Computer instruction, Object-oriented programming

随着计算机硬件与通信技术的迅速发展，计算机应用的规模与复杂度不断提高，计算机软件的开发语言、工具与环境也在不断更新。在系统软件与应用软件领域中，面向对象程序设计（简称 OOP）语言 C++已成为主要开发工具，从微机、RISC 工作站、小型机到中、大型机的各种软件与硬件平台上均提供了 C++语言的开发环境；在管理信息系统、数据处理等结构化方法的传统应用领域中，越来越广泛地应用了客户/服务器体系结构、前端开发工具如 Powersoft 公司的 PowerBuilder、Borland 公司的 Delphi 等均支持 OOP；在热门的 Internet/Intranet 领域中，基于 C++语言发展起来的 Java 语言更是以 OOP 为指导思想；C++语言的主要竞争对手、由美国国防部支持的 Ada 语言也不甘落后，新修订的 Ada95 版本将 Ada 语言从基于对象的发展为面向对象的，此外，目前在软件产业界具有深远影响的规范标准 OLE2、CORBA 等也都以面向对象技术为基础。

由这种发展趋势可见，当前在工业界 OOP 已取代结构化程序设计，占据了程序设计领域的主流，作为计算机专业的学生必须学习并掌握 OOP 方法与 OOP 语言。"面向对象"不再是软件开发中的一个时髦名词，而是作为一种对软件开发人员的很基本的要求。

我系于 89 年开始在本科高年级开设 OOP 任选课。随着 OOP 技术的发展与成熟，该课程逐步过渡为指选课、必修课，各种大专、副修班也陆续开设了该课程，为普及面向对象技术起了很好的推动作用。在不断积累教学经验、加强师资力量、建设配套教材、完善实验条件的基础上，我系自 96 年开始对程序设计课程又作出重大改革：计算机软件、计算机及应用、管理信息系统 3 个专业一年级的程序设计入门课程改为学习 OOP，以 C++作为程序设计入门语言，从而将原来的两门高级语言程序设计课程合二为一。

1 OOP 作为计算机专业第一语言

程序设计是计算机专业非常重要的基础，学好程序设计入门课程对于数据结构与算法、软件工程、编译原理、形式语言与自动机、形式语义学等后续课程有很大帮助。程序设计课程不仅仅是讲授某种程序设计语言[5]，它应该包括两个相辅相成的方面：程序设计方法与程序设计语言。自从第一台计算机诞生以来，程序设计方法与程序设计语言在不断发展：70 年代是结构化程序设计的时代，而进入 80 年代以来 OOP 逐渐占据了程序设计的主流。工业界软件开发方法由结构化程序设计朝 OOP 的转变向教育界提出了新的要求，如果我们培养的学生不能适应这种转变，必将在日后的实际工作中落后于他人。

过去 10 多年来，计算机专业第一语言一般选用 Pascal 或 C 语言，并贯穿结构化程序设计思想。但随着 OOP 发展与成熟，从一开始就向学生传授 OOP 思想显得十分必要和有益。当前计算机教育界许多学者都认为应在本科教学中开设 OOP 作为必修课，但对于是否应将 OOP 作为第一语言还存在异议：OOP 是否有必要作为第一语言课程？OOP 是否有

可能作为第一语言课程？对此持观望态度的做法是在本科一年级仍将 Pascal 或 C 语言作为第一语言，学习结构化程序设计方法；到高年级再学习 OOP 方法与语言。然而我们从长期的 OOP 课程教学实践中认识到，OOP 不仅应该而且可以作为计算机专业的第一语言。

1.1 OOP 作为第一语言的必要性

OOP 与结构化程序设计是解决问题的两种不同思维方式，前者注重事物的结构，而后者注重事物表现的行为。这种思维方式的区别导致程序结构的完全不同，用 OOP 方法开发的程序结构与问题空间有更直接的对应关系，比结构化程序设计的结果更加稳定，因而提高了程序的可靠性、可维护性等质量要素。

一些习惯了用结构化方法设计程序、然后再学习 OOP 的学生反映他们设计出来的程序总是不象"面向对象的"，尽管他们也设法使用了类来组织程序。从结构化方法到 OOP 方法的转变并不象从 BASIC 到 Pascal 语言、或从 Pascal 到 C 语言的转变那样轻易，这种思维习惯的转变需要花更大的力气与更多的时间，因而有必要从一开始就让学生采用面向对象思维方式去解决实际问题，并让学生以 OOP 作为基本工具开展后续课程的学习。

在程序设计技术迅速发展的今天，程序设计课程的教学内容不能仍停留在 10 年前的水平。程序设计与软件工程研究不断取得的最新成果，如封装、信息隐藏、数据抽象、继承、多态性、异常处理等机制以及程序可靠性、可重用性、可维护性等概念将越来越多地包括在教学内容之中，导致教学内容不断更新与膨胀。然而随着我国 5 日工作制的实行，许多学生的日均学时数高达 6 学时、显然不利于学生个性的发展，不利于培养知识面广、动手能力强、综合素质高的学生。如果不对课程进行改革，势必走上"教材越编越厚、必修课越开越多、学生负担越来越重"的路子。OOP 作为第一语言课程避免了在本科期间开设两门高级语言程序设计课程。

1.2 OOP 作为第一语言的可行性

尽管 C++ 在理论上并不是最完善的 OOP 语言，但综合考虑该语言在软件产业界的主导地位、学生实习环境的建立、程序设计资源的可用性等因素，本课程仍选择 C++ 语言为主。

结构化程序设计是 OOP 的基础，如果 C++ 作为第一语言，必须首先向学生讲授结构化程序设计的部分内容、再介绍 OOP 的新思想与新机制、教学内容远比原来的程序设计入门课程庞大，因而 C++ 作为第一语言不能这样简单地加大教学内容或增加学时。解决这一可行性问题的途径应该是严格筛选教学内容、有效组织知识单元、合理安排学习进度、适当控制教学难度，并设计相应的上机实习题指导学生自己动手实践，我们通过这几个方面的有益尝试得出如下结论：学习与掌握 OOP 技术完全可在一门课程中完成。

例如学生掌握了 C++ 语言的输入/输出流类库后，就没有必要再学习 C 语言的输入/输出函数库中 printf()、scanf()等函数的用法；友元函数仅在讲授运算符重载时顺带介绍一下；诸如 goto 语句、volatile 对象、inline 成员函数、嵌套类、对象强制类型转换、一些特殊运算符重载等均可忽略不讲。这样学生可以用更多的时间来学习程序设计方法，并通过上机实践掌握程序设计技术，不必过多地陷于语言细节。此外，C++ 语言的许多灵活性对于初学者而言并不是必需的，教学中程序例子应尽量避免这些不严格性。

随着 PC 的普及和计算机广泛应用，许多一年级学生对计算机不再陌生；部分学生在高中已掌握了微机的基本操作；学校提供给学生的上机实习环境也不断改进，这些外部环境的变化都为课程改革提供了有利的条件。

2 OOP 课程建设目标

2.1 循序渐进、深入浅出

由于本课程作为本科一年级的入门课程，所以不能假设读者已掌握了结构化程序设计方法以及 C 或 Pascal 语言。教师必须从程序设计中最基本的概念开始，详细介绍基本数据类型与控制结构，逐步过渡到函数、类、继承、多态性、类属、输入/输出流等复杂机制，最后还包括提高程序可靠性的程序断言机制与异常处理机制、提高程序可重用性的构件库组织规范等高级内容。整个教学过程既要注意循序渐进，又要注意深入浅出。

2.2 尽早引入类的概念，开拓 OOP 思维方式

类是构造面向对象程序的基本单元，类中的操作是以函数来表达的，在本课程中函数更多地是以类中某个操作的形式出现，而不是作为构造程序的基本单位。提早引入类的概念有助于初学者采用面向对象的、而不是传统结构化的思维方式去解决实际问题，有助于构造良好的程序结构，为日后处理大型、复杂程序打好基础。因而本课程在介绍了所必需

的基本数据类型与基本控制结构后，即引入函数的概念，紧接着就引入类的概念。

2.3 不断疏导各种解决问题的策略

在引导本科生入门的OOP课程中，应该不断灌输人类解决问题的各种策略，如分而治之、自顶向下逐步求精、软件重用、程序可靠性与可维护性等。这是训练学生养成良好思维方式、提高学生综合能力的有效途径。ACM与IEEE/CS联合制订的《计算教学大纲1991》中提出了计算机科学中12个反复出现的核心概念，这些概念应该在本课程各个知识单元中反复强调[1]。

2.4 选择合适的高层次设计工具

图形工具不仅可作为设计结果的简单、直观表示，更可作为程序设计的有力工具。本课程中控制流的表示应放弃与结构化程序设计背道而驰的程序框图(即程序流图)，而选择诸如PAD图、N-S图、PDL伪码等结构化设计工具。

在对象、类及其间关系的表达或设计中，尽可能地采用图形表示。图形表示技术可选择较流行的Booch方法或OMT方法。

2.5 精选例题、练习题、实习题

丰富、典型的例题与练习思考题是帮助学生掌握本课程内容的一个关键手段。本课程中出现的程序可分为两大类：一类源于实际应用，一类专构造来演示某种程序设计机制的效果。在可以解释清楚程序设计机制的前提下应尽量选用前者，并培训学生自己设计后一类程序。由于本课程是入门课，所以不宜采用太多的数据结构例子。

程序设计风格对初学者影响甚大，所以教学中给出的程序例子应在程序版面、标识符命名、程序注释等方面仔细斟酌。不要为节省版面而放弃程序的书面风格。

2.6 专业学习与专业英语学习同步

由于主要的程序设计环境与日后进一步学习的许多资料都是英文所写，所以在本课程教学过程中出现程序设计专业术语时，均应向学生解释相应的英文单词。在每一门专业课的教学中都提倡这种做法将导致专业英语课程教学的改革——本科高年级不再开设专业英语课程，学生能通过考试即可取得学分。

3 知识单元组织与学时数安排

知识单元是组成课程的基本单位，而每个知识单元又包含若干个讲课专题、这些专题的深度和广度可由任课教师根据教学对象与学时进行调整。将课程划分为知识单元，利用知识单元描述教学计划中的公共要求，这样更利于课程建设的灵活性和适应性，便于各校结合自身的特点组织教学，设计并开出既具有自身特色、又不背离统一基本要求的课程。

3.1 课程概述

OOP是计算机专业的基础课程，通过讲授、练习和实验，使学生掌握OOP方法与OOP语言C++，并掌握利用计算机求解问题的一般策略、方法和步骤，为进一步学习计算机专业的其他课程打下坚实基础。

对于本科计算机专业学生，课内学时数共72个学时(其中习题课10个学时)，学生上机学时数共40个学时。对于已学习过C语言的非计算机专业学生，课内学时至少54个学时，上机学时数不少于36个学时，各知识单元中讲课专题的深度与广度由任课教师作相应调整。

3.2 知识单元内容与参考学时数

本课程共分12个知识单元，括号中列出了该知识单元的课内学时数/上机学时数(其中课内学时数不含习题课)：PR1 程序设计与C++语言初步(2/2)、PR2 基本数据类型(4/2)、PR3 基本控制结构(6/4)、PR4 函数(6/4)、PR5 类与对象(6/4)、PR6 复合数据类型(8/6)、PR7 继承机制(8/4)、PR8 多态性(6/4)、PR9 类属机制(4/4)、PR10 输入/输出流(6/4)、PR11 程序可靠性与可重用性(4/2)、PR12 程序设计风范与其他OOP语言(2/0)。各知识单元的讲课专题详见参考文献[2]。

4 课程改革的配套设施建设

4.1 师资建设

无论什么专业课程，要想建设一流的课程就必须有一支一流的教师队伍。这是在各种教学条件中最重要的条件。我们在本课程建设中除了要求任课教师精通C++语言外，还要求教师熟悉其他OOP语言(如Eiffel、Java、Ada95等)，提高程序设计理论基础的修养，并掌握好数据结构与算法、软件工程、编译原理等领域的知识，以帮助学生拓广他们的视野。此外，提供条件让教师参加一些较大规模系统软件或应用软件的实际开发工作。这些教师在课堂上结合讲课专题介绍实际开发经验与实例，深受同学们的欢迎。

4.2 教材建设

针对目前尚无合适教材可作为OOP入门课程

之选，我们根据上述知识单元与讲课专题编写了教材《面向对象程序设计基础》[2]（高等教育出版社1998年出版）与配套习题集《面向对象程序设计——实践与提高》[3]。该教材专为OOP作为第一语言的初学者编写，向他们介绍OOP方法与C++程序设计语言，内容大致可分为两部分：一部分从数据结构和控制结构两个角度介绍程序设计语言的一般概念，另一部分围绕类与对象介绍面向对象程序构造的基本思想。在介绍某些语言机制时，针对C++语言在OOP理论上的不完善之处参考了其他语言（例如类属、程序断言与异常处理等机制参考了Eiffel和Ada语言）。习题集不仅提供该教材练习与思考题、上机实习题的答案，更注重向读者展示解题的过程。此外，习题集还提供一些更复杂与更大型的练习题，并介绍部分在教材中未出现的OOP提高内容，让有兴趣的学生有更多的实际训练机会。编写教材一方面是对我们教学经验的总结，另一方面也促使我们开展教学研究。

4.3 计算机辅助教学研究

目前我们正在开展的计算机辅助教学研究不仅仅针对本课程，而是旨在利用我系的师资优势建立一个计算机专业的远程教学中心，为在校教育、继续教育、职业培训甚至家庭教育作出贡献。我们的研究目标是在Internet/Intranet上开发一个智能化的通用课件制作系统，并制订一套课件稿本设计规范[4]。该系统的智能化反映在可根据指导教师编写的规则库实行因材施教的个别化教育。除本系的课程建设可利用该系统外，还可成立专业的课件制作中心，为全校其他专业以至其他学校的教师服务。

4.4 程序设计后续课程改革

学生的程序设计入门课程改革为OOP后，后续课程也应作相应的改革或调整。例如数据结构与算法课程可用抽象数据类型来描述数据结构及其上操作，向学生讲授如何以类来实现这些抽象数据类型，程序设计语言也应从Pascal或C转向C++语言。软件工程课程除了传统的结构化方法外，还应该增讲面向对象分析与设计方法的基本概念与术语、表示技术、开发步骤、软件工具与环境等。编译原理课程也应增设OOP语言的编译技术。

5 教学评价调查问卷分析

课程结束后，我们安排专门时间对3个专业的130名学生发出调查问卷，学生以不记名方式答卷。统计结果表明，62%学生入学前已掌握微机基本操作，73%自备上机实习条件（其中46%条件很好），这说明程序设计课程改革的外部条件已成熟，学生可减少基础知识的学习时间，而将主要精力集中在程序设计上。安排学生上机实习的环境则可以从DOS平台上的Turbo C++3.0转向Windows平台上的C++开发环境。

编写教材是我们课程改革的重要组成部分。64%学生认为所用教材[3]的内容较好地做到循序渐进。56%学生认为教材难度适中，31%学生认为教材的某些知识点偏难（主要集中在指针、多态性、输入/输出流）。72%学生对教材的总体评价是良好或优秀，这些统计数字对我们是一种鼓励，但同时也促使我们进一步充实教材中的例题、更详细地解说一些学生认为是难点的知识单元。

有一种观点认为大学一年级开设程序设计课程会影响其他基础课的学习，但我们的统计结果表明，仅有8%学生反映OOP作为入门语言在时间与精力上很大程度影响了"数学分析"、"大学英语"等基础课程的学习，37%学生反映影响一般，而55%学生认为无影响或影响很小。学生对课程改革的认识将影响到他们的积极性，76%学生通过学习OOP后认识到课程改革是有必要的。

结束语 程序设计课程的改革顺应了计算机软件技术发展的大潮流，将为计算机专业的整个课程体系改革带来深远影响。通过调查分析我们更坚定了改革的信心，在未来的几年内仍将以OOP作为程序设计入门课程，同时加快数据结构与算法、软件工程等课程的改革步伐。

主要参考文献

1 中国计算机学会教育委员会，全国高等学位计算机教育研究会. 计算机学科教学计划1993. 电子工业出版社，1995

2 李师贤，李文军，周晓聪. 面向对象程序设计基础. 高等教育出版社，1998

3 李师贤，周晓聪，李文军. 面向对象程序设计——实践与提高. 中山大学计算机科学系讲义，1997

4 李文军，周晓聪，等. 基于Internet/intranet的智能化通用课件制作系统研究：[中山大学计算机科学系技术报告]. 1997

5 Knudsen J, Madsen O. Teaching Object-Oriented Programming is More Than Teaching Object-Oriented Programming Languages. ECOOP' 88, Lecture Notes on Computer Science, 1988, 322. 21～40