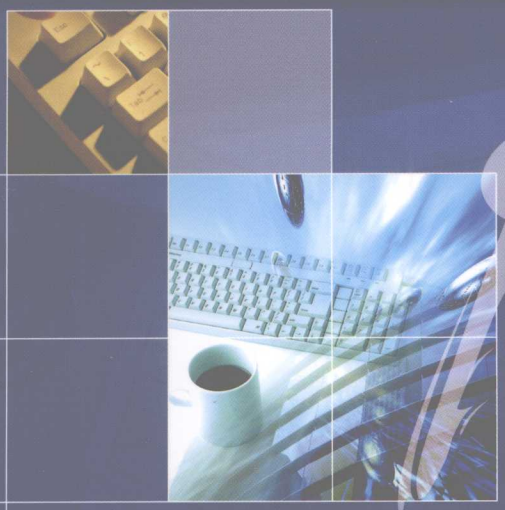
21st CENTURY
实用规划教材

21世纪全国应用型本科计算机案例型规划教材

C++程序设计基础案例教程



Programming

主　编　于永彦　王志坚　娄渊胜

北京大学出版社
PEKING UNIVERSITY PRESS

21 世纪全国应用型本科计算机案例型规划教材

C++程序设计基础案例教程

主　编　于永彦　王志坚　娄渊胜

北京大学出版社
PEKING UNIVERSITY PRESS

内 容 简 介

本书为“C++程序设计”课程体系的基础教材，主要讲述C++语言的基本概念与基本应用，包括数据类型、运算符、表达式、流程控制语句、函数、数组、链表、类与对象、继承与派生、多态性及文件流操作等。全书以一个实用的“简易学生管理系统”为研究载体，将整个系统工程划分为若干个模块，每个模块归纳为一个核心问题，为每个问题设计一个“子工程模型”，集中对应于一个章节。

本书适用于理、工类大中专院校的“C++程序设计”课程，也可供程序设计爱好者和工程技术人员参考使用。

图书在版编目(CIP)数据

C++程序设计基础案例教程/于永彦，王志坚，娄渊胜主编. —北京：北京大学出版社，2009.1
(21世纪全国应用型本科计算机案例型规划教材)
ISBN 978-7-301-14510-4
Ⅰ.C… Ⅱ.①于… ②王… ③娄… 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(2009)第005924号

书　　名：C++程序设计基础案例教程
著作责任者：于永彦　王志坚　娄渊胜　主编
策划编辑：李　虎　孙哲伟
责任编辑：孙哲伟
标准书号：ISBN 978-7-301-14510-4/TP·0989
出 版 者：北京大学出版社
地　　址：北京市海淀区成府路205号　100871
网　　址：http://www.pup.cn　http://www.pup6.com
电　　话：邮购部62752015　发行部62750672　编辑部62750667　出版部62754962
电子邮箱：pup_6@163.com
印 刷 者：涿州星河印刷有限公司
发 行 者：北京大学出版社
经 销 者：新华书店
787毫米×1092毫米　16开本　20.5印张　470千字
2009年1月第1版　2009年1月第1次印刷
定　　价：33.00元

信息技术的案例型教材建设

(代丛书序)

刘瑞挺

北京大学出版社第六事业部在2005年组织编写了《21世纪全国应用型本科计算机系列实用规划教材》，至今已出版了50多种。这些教材出版后，在全国高校引起热烈反响，可谓初战告捷。这使北京大学出版社的计算机教材市场规模迅速扩大，编辑队伍茁壮成长，经济效益明显增强，与各类高校师生的关系更加密切。

2008年1月北京大学出版社第六事业部在北京召开了“21世纪全国应用型本科计算机案例型教材建设和教学研讨会”。这次会议为编写案例型教材做了深入的探讨和具体的部署，制定了详细的编写目的、丛书特色、内容要求和风格规范。在内容上强调面向应用、能力驱动、精选案例、严把质量；在风格上力求文字精练、脉络清晰、图表明快、版式新颖。这次会议吹响了提高教材质量第二战役的进军号。

案例型教材真能提高教学的质量吗？

是的。著名法国哲学家、数学家勒内·笛卡儿(Rene Descartes，1596—1650)说得好：“由一个例子的考察，我们可以抽出一条规律。(From the consideration of an example we can form a rule.)”事实上，他发明的直角坐标系，正是通过生活实例而得到的灵感。据说是在1619年夏天，笛卡儿因病住进医院。中午他躺在病床上，苦苦思索一个数学问题时，忽然看到天花板上有一只苍蝇飞来飞去。当时天花板是用木条做成正方形的格子。笛卡儿发现，要说出这只苍蝇在天花板上的位置，只需说出苍蝇在天花板上的第几行和第几列。当苍蝇落在第四行、第五列的那个正方形时，可以用(4，5)来表示这个位置……由此他联想到可用类似的办法来描述一个点在平面上的位置。他高兴地跳下床，喊着“我找到了，找到了”，然而不小心把国际象棋撒了一地。当他的目光落到棋盘上时，又兴奋地一拍大腿：“对，对，就是这个图”。笛卡儿锲而不舍的毅力，苦思冥想的钻研，使他开创了解析几何的新纪元。千百年来，代数与几何，井水不犯河水。17世纪后，数学突飞猛进的发展，在很大程度上归功于笛卡儿坐标系和解析几何学的创立。

这个故事，听起来与阿基米德在浴池洗澡而发现浮力原理，牛顿在苹果树下遇到苹果落到头上而发现万有引力定律，确有异曲同工之妙。这就证明，一个好的例子往往能激发灵感，由特殊到一般，联想出普遍的规律，即所谓的“一叶知秋”、“见微知著”的意思。

回顾计算机发明的历史，每一台机器、每一颗芯片、每一种操作系统、每一类编程语言、每一个算法、每一套软件、每一款外部设备，无不像闪光的珍珠串在一起。每个案例都闪烁着智慧的火花，是创新思想不竭的源泉。在计算机科学技术领域，这样的案例就像大海岸边的贝壳，俯拾皆是。

事实上，案例研究(Case Study)是现代科学广泛使用的一种方法。Case包含的意义很广：包括Example例子，Instance事例、示例，Actual State实际状况，Circumstance情况、事件、境遇，甚至Project项目、工程等。

我们知道在计算机的科学术语中，很多是直接来自日常生活的。例如Computer一词早在1646年就出现于古代英文字典中，但当时它的意义不是“计算机”而是“计算工人”，

即专门从事简单计算的工人。同理，Printer 当时也是“印刷工人”而不是“打印机”。正是由于这些“计算工人”和“印刷工人”常出现计算错误和印刷错误，才激发查尔斯·巴贝奇(Charles Babbage，1791—1871)设计了差分机和分析机，这是最早的专用计算机和通用计算机。这位英国剑桥大学数学教授、机械设计专家、经济学家和哲学家是国际公认的“计算机之父”。

20 世纪 40 年代，人们还用 Calculator 表示计算机器。到电子计算机出现后，才用 Computer 表示计算机。此外，硬件(Hardware)和软件(Software)来自销售人员。总线(Bus)就是公共汽车或大巴，故障和排除故障源自格瑞斯·霍普(Grace Hopper，1906—1992)发现的“飞蛾子”(Bug)和“抓蛾子”或“抓虫子”(Debug)。其他如鼠标、菜单……不胜枚举。至于哲学家进餐问题，理发师睡觉问题更是操作系统文化中脍炙人口的经典。

以计算机为核心的信息技术，从一开始就与应用紧密结合。例如，ENIAC 用于弹道曲线的计算，ARPANET 用于资源共享以及核战争时的可靠通信。即使是非常抽象的图灵机模型，也受到二战时图灵博士破译纳粹密码工作的影响。

在信息技术中，既有许多成功的案例，也有不少失败的案例；既有先成功而后失败的案例，也有先失败而后成功的案例。好好研究它们的成功经验和失败教训，对于编写案例型教材有重要的意义。

我国正在实现中华民族的伟大复兴，教育是民族振兴的基石。改革开放 30 年来，我国高等教育在数量上、规模上已有相当的发展。当前的重要任务是提高培养人才的质量，必须从学科知识的灌输转变为素质与能力的培养。应当指出，大学课堂在高新技术的武装下，利用 PPT 进行的“高速灌输”、“翻页宣科”有愈演愈烈的趋势，我们不能容忍用“技术”绑架教学，而是让教学工作乘信息技术的东风自由地飞翔。

本系列教材的编写，以学生就业所需的专业知识和操作技能为着眼点，在适度的基础知识与理论体系覆盖下，突出应用型、技能型教学的实用性和可操作性，强化案例教学。本套教材将会有机融入大量最新的示例、实例以及操作性较强的案例，力求提高教材的趣味性和实用性，打破传统教材自身知识框架的封闭性，强化实际操作的训练，使本系列教材做到“教师易教，学生乐学，技能实用”。有了广阔的应用背景，再造计算机案例型教材就有了基础。

我相信北京大学出版社在全国各地高校教师的积极支持下，精心设计，严格把关，一定能够建设出一批符合计算机应用型人才培养模式的、以案例型为创新点和兴奋点的精品教材，并且通过一体化设计、实现多种媒体有机结合的立体化教材，为各门计算机课程配齐电子教案、学习指导、习题解答、课程设计等辅导资料。让我们用锲而不舍的毅力，勤奋好学的钻研，向着共同的目标努力吧！

刘瑞挺教授　本系列教材编写指导委员会主任、全国高等院校计算机基础教育研究会副会长、中国计算机学会普及工作委员会顾问、教育部考试中心全国计算机应用技术证书考试委员会副主任、全国计算机等级考试顾问。曾任教育部理科计算机科学教学指导委员会委员、中国计算机学会教育培训委员会副主任。PC Magazine《个人电脑》总编辑、CHIP《新电脑》总顾问、清华大学《计算机教育》总策划。

前　言

对于计算机科学与技术、电子信息工程等专业的理工类学生来说，“C++程序设计”无疑是最重要的一门专业基础课程，对于后续课程的进一步学习、专业知识结构的完整构建都具有举足轻重的作用。系统学习并深入掌握 C++语言基础理论与实用技术已成为大多数学生的必修课程与必备技能。目前，一些典型的 C++语言编译工具，如 Visual C++、Borland C++等，在完全兼容结构化程序设计思想与方法的基础上，全面支持面向对象程序设计理论与技术，虽然较好地实现了功能抽象和数据抽象的有机统一，但也因此使得课程涵盖的学习内容非常庞大，知识点多，涉及面广，教、学难度都非常大，往往费了大量时间而达不到预期的教学效果。

俗话说：“学习的最好方法是实践。”北京大学出版社诸位同仁高瞻远瞩，因势利导，适时发起并组织的“21 世纪全国应用型本科计算机案例型规划教材”横空出世。北京大学出版社此举顺应了我国计算机教育迅猛发展的大好形势，配合了高等院校本科计算机教学改革和教材建设，在坚持“因材施教”教学原则的大前提下，注重理论联系实际，全面促进了高等院校教材建设，进一步提高了我国高校教材的质量。本书正是在这种大背景下，由淮阴工学院、河海大学部分骨干教师联合组织编写的。

本书是在原 C++程序设计教学讲义的基础上改编而成的。原讲义已在淮阴工学院等几所学校试用过 6 届，历时 8 年，中间修订过 4 次，并逐步完善了《实践教程》、《学习指导》、《习题册》等配套教辅材料。本次由北京大学出版社出版的“C++程序设计”专题教材，按设计的程序是否涉及可视化元素而分为相对独立的两个部分，本书主要介绍非可视化程序设计的基础知识，重点是 C++语言的基本概念与基本应用，包括数据类型、运算符、表达式、流程控制语句、函数、数组、链表、类与对象、继承与派生、多态性及文件流操作等。对于 C++的可视化程序设计技术，包括多媒体程序设计、数据库程序设计、网络编程技术等，均放在《Visual C++高级程序设计案例教程》中讲解，并可作为本书的后续教材与本书配合使用。

针对目前计算机语言教学中存在的侧重基本概念而忽视综合应用等问题，这里首次提出了“面向工程、面向应用”的教学思想和教学设计模式，就是以“课程教学”为起点，以“案例教学”为主体，以“工程教学”为归宿，内涵上逐层紧缩，在外延上不断扩大，本书以一个规模相当的实例工程为平台，围绕工程需求，组织、设计教学内容，引导学生针对工程中存在的问题，去书本中寻找答案，从实践中获得真知。

为此，本书在内容安排上具有以下特色。

“一个工程”：以“简易学生管理系统”为研究载体，设计一个实用系统来管理学生信息，包括学号、姓名、性别、出生年月、文化课成绩、专业课成绩、体育课成绩等信息。将整个工程划分为若干个模块，每个模块归纳为一个核心问题，为每个问题设计一个“子工程模型”。工程的求解过程就是“由点到面”不断学习的过程，学习内容呈“星状”辐射，向外层逐渐延伸，以点带面，逐层展开，注重培养工程理念。

“两种方法”：以数组、链表作为主要数据结构设计主体算法实现上述系统。通过讲述数组的概念、基本操作、典型案例及实际应用，引导读者主动“理论联系实际”，培养“学以致用”的学习方法和行为习惯；通过对比教学，深入讲解链表的存储原理、工作机制，并在实际系统中对照使用，让读者自己去体会其中的奥妙。

“三大基础”：结构化程序设计的核心就是 3 个基础，即数据类型、流程控制和函数。以数据存储为起点，以语句执行的流程控制为中心，以函数为程序结构，足以构筑一个完整的 C++应用程序。这是设计上述系统的基础，也是教学的重点内容。

“四个难点”：本书的 4 个难点分别是指针、引用、结构和类的概念。其中指针和引用是两种重要的参数传递方式，是 C++中的难点，是“数据结构”等后续课程的重要基础。“类”与“对象”是 C++语言的灵魂，其封装性、继承性与多态性是面向对象程序设计的核心和基础，也是本课程的知识难点，本书将通过具体的实例详细讲解。

“五大对比”：通过数组与结构、数组与链表、内存数据与外存数据、结构化与面向对象、具体与抽象的对比教学，让读者真正理解、掌握这些知识点，学会使用它们解决实际问题。

在本书的编写过程中，得到了很多热心人士的帮助与支持。感谢淮阴工学院计算机工程系的党政领导及各位同事，尤其感谢系副主任赵建洋博士，他为本书的顺利立项、组织与编写投入了大量精力，时刻关注书稿的进度，严把质量关，并为全书的整体性构思提供了许多建设性的建议。还要感谢于长辉、刘作军、任永峰、戴峻峰等诸位同事，感谢他们为本书的讲义提供了试用平台，并提出了许多宝贵的修改建议。还有其他为本书进行过文字校对、编辑排版的老师、同学，此处不再一一列出，谨此表示最诚挚的谢意。

虽然编者主观上做了最大的努力，但由于本身的水平有限，加上时间仓促，教学改革的力度又较大，难免存在这样或那样的不足。“他山之石可以攻玉”，真诚地希望使用或阅读本书的读者给予批评指正，不吝赐教。

作　　者

2008 年 9 月 28 日　　于淮安

本书内容安排及教学设计

我们编写的“C++程序设计”教材拟分为相互独立的两本书：《C++程序设计基础案例教程》、《Visual C++高级程序设计案例教程》。本书主要介绍非可视化程序设计的基础知识，重点是C++语言的基本概念与基本应用，包括数据类型、运算符、表达式、流程控制语句、函数、数组、链表、类与对象、继承与派生、多态性及文件流操作等。《Visual C++高级程序设计案例教程》则着重讲述C++语言的可视化程序设计技术，包括Visual C++的高级编程技术及典型应用，如Windows编程基础、基于对话框或文档/视图结构的程序设计、多媒体程序设计、数据库程序设计，以及网络编程技术等。

全书以一个实际的工程展开，根据“简易学生管理系统”涉及的基础概念、设计技术，从软件工程的角度，将整个工程划分为若干个模块，每个模块归纳为一个核心问题，为每个问题设计一个“子工程模型”。如图1所示为该系统的参考模型。

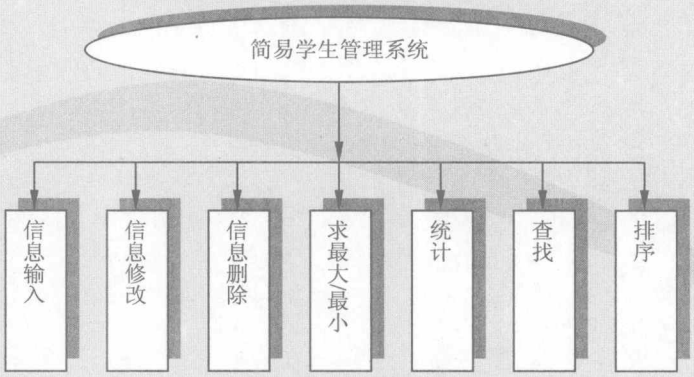


图1　简易学生管理系统功能模块

围绕每一个“子工程模型”，寻求、设置教学内容，设计教学模式。

表1中是“简易学生管理系统”所涉及的数据信息。

表1　简易学生管理系统中的学生信息

标　识	含　义
1	学号
2	姓名
3	性别
4	出生年月
5	文化课成绩
6	专业课成绩
7	体育课成绩

工程的求解过程就是“由点到面”不断学习的过程，学习内容呈“星状”辐射，向外层逐渐延伸，以点带面，逐层展开，注重培养工程理念，这就是“基于工程建构”的教学理论与教学模式。这种教学组织模式，强调以一个规模相当的实例工程为平台，围绕工程需求，组织、设计教学内容，引导学生针对工程中存在的问题，去书本中寻找答案，去图书室查阅文献资料，去向老师、同学寻求帮助，从实践中获得真知。这种教学模式更加适合于应用型人才的培养，暗合“实践出真知”的古训，也更能体现“学以致用”的教学精神，从长远来看，也充分反映素质教育在计算机教育中的具体要求。

目　录

第1章

绪　论

随着1946年世界上第一台电子计算机“ENIAC”(Electronic Numerical Integrator And Calculator，电子数字积分机和计算器)在美国宾夕法尼亚大学的横空出世，人类文明进入了一个崭新的时代。电子计算机是20世纪最伟大的发明之一，是第三次工业革命中最辉煌的成就，对国民经济、国防建设和科学文化事业的飞速发展起到了巨大的催化与推动作用。计算机的普及与应用水平已成为各行各业步入现代化的重要标志之一，具备计算机应用能力也成为现代化人才的基本素质之一。

一台计算机是由硬件系统和软件系统两大部分构成的，硬件是物质基础，而软件则是灵魂：没有软件，计算机是一台什么也不能干的“裸机”；有了软件，才是一台真正的“计算机”。而所有的软件，都是用计算机语言编写的。对于理工科学生而言，学习一门计算机语言，掌握一定的软件设计方法，对于科学精神的培养、抽象思维的锻炼、科研作风的养成，以及发现问题、分析问题、解决问题的能力训练，都是至关重要的。

1.1　软件设计基础

所谓软件，是指与计算机系统操作有关的计算机程序、规程以及与之有关的文件，包括程序和文档两部分。程序是指适合于计算机处理的指令序列以及所处理的数据，而文档是与软件开发、维护和使用有关的文字材料。人们平常也将程序简称为软件。

从不同的角度可以对软件进行不同的分类：按功能划分，可将软件分为系统软件、支撑软件和应用软件；按规模划分，可分为微型、小型、中型、大型及特大型软件等；按工作方式划分，可分为实时处理软件、交互式工作软件、分时工作软件等。上述划分主要是从使用者或开发者的角度出发，如果从计算机本身的处理能力出发，则可将软件分为数值计算型软件、逻辑推理型软件、人机交互型软件和数据密集型应用软件等。

软件设计是一个“把用户需要转化为软件需求，把软件需求转化为软件设计，用软件代码来实现软件设计，对软件代码进行测试，并签署确认它可以投入运行使用的过程”。从工程学角度出发，软件设计过程包括计划、分析、设计、编码、测试和维护共6个阶段。常用的软件设计方法有结构化方法、面向对象方法和专家系统方法等。

1.1.1　结构化方法

20世纪60年代中后期，软件越来越多，规模越来越大，而软件的生产基本上还是以

“英雄主义的各自为战”为主，缺乏科学规范的系统规划、测试与评估，其后果是耗费巨资建立起来的软件系统，由于含有不可预见的错误而瞬间崩溃。因而软件给人的感觉是越来越不可靠，以致有人断言“没有不出错的软件”。这一切问题的存在直接导致了软件危机的大规模爆发，极大地震动了计算机界。人们开始认识到，大型程序的编制不同于写小程序，它应该是一项技术，应该像处理工程一样处理软件研制的全过程，程序设计的正确性应该既易于保证又便于验证。于是，1969 年就有人提出了“结构化程序设计方法”，1970 年第一个结构化程序设计语言——Pascal 语言的诞生标志着结构化程序设计时代的开始。

结构化设计方法是一种围绕功能来组织软件系统的方法。在这种方法中，系统的基本构成要素是模块，它是一种实现系统某一功能的程序单元。模块具有输入、输出、内部数据和过程等基本特性。输入和输出分别是模块需要的和产生的数据，内部数据是仅供模块本身引用的数据，过程则是对模块具体处理细节的描述和表示。输入和输出是模块的外部特性，而内部数据和过程是模块的内部特性。

结构化设计方法通过按功能将问题分解抽象成模块，进而建立模块和模块之间的调用关系来进行软件开发。结构化程序设计的主要思想是功能分解并逐步求精。当一些任务十分复杂以致无法描述时，可以将它拆分为一系列较小的子任务，直到这些子任务小到易于理解的程度。

结构化程序设计方法成功地为处理复杂问题提供了有力的手段，然而到 20 世纪 80 年代末，它的缺点越来越突出，主要表现在两个方面，一是当数据量增大时，数据与处理这些数据的方法之间的分离，使程序变得越来越难以理解。对数据处理能力的需求越强，这种分离所造成的负面影响越显著；二是采用结构化程序设计方法的程序员发现，每一种相对于老问题的新方法都要带来额外的开销，与可重用性相对，通常称为重复投入。基于可重用性的思想是指建立一些具有已知特性的部件，在需要时可以插入到程序之中。这是一种模仿硬件组合方式的做法，当工程师需要一个新的晶体管时，他不用自己去发明，只要到仓库去找就行了。对于软件工程师来说，在面向对象程序设计出现之前，一直缺乏具备这种能力的工具。

1.1.2 面向对象方法

20 世纪 80 年代初，软件设计思想又产生了一次革命，其直接成果就是面向对象的程序设计方法。在此之前的计算机高级程序设计语言几乎都是面向过程的，程序的执行是流水线式的，在一个模块被执行完成前，人们不能干别的事，也无法动态地改变程序的执行方向。这和人们日常处理事物的思维方式是不一致的。人们总是希望发生一件事就处理一件事，也就是说，不能面向过程，而应是面向具体的应用功能，这就是对象(Object)。其方法就是软件的集成化，如同硬件的集成电路一样，生产一些通用的、封装紧密的功能模块，即软件集成块。这些模块与具体应用无关，但能相互组合，完成具体的应用功能，同时又能重复使用。对使用者来说，只关心它的接口(输入量、输出量)以及能实现的功能，至于如何实现，那是它内部的事，使用者完全不用关心。这方面的典型代表就是 C++程序设计语言。

面向对象方法是一种围绕真实世界中的事物来组织软件系统的全新方法。在这种方法中，系统的基本构成要素是对象。从软件开发人员的角度来看，对象是一种将数据和处理

这些数据的操作合并在一起的程序单元；从用户的角度来看，对象是一种具有某些属性和行为的事物。对象可以是具体的，如自行车；也可以是概念性的，如车辆通行方案。对象具有标识唯一性、分类性、多态性、继承性和封装性等基本特性。

面向对象方法是通过将存在于问题空间范围内的事物抽象成对象、建立对象和对象之间的通信联系来进行软件开发的。

1.1.3　专家系统方法

专家系统方法是一种围绕知识来组织软件系统的方法。它应用人工智能技术，根据人类专家提供的知识、经验进行推理和判断，模拟人类专家解决那些需要专家解决的复杂问题。在这种方法中，构成系统的基本要素是知识和应用这些知识的推理机制。

专家系统方法是通过对存在于问题空间范围内的知识和经验进行收集、整理和描述，建立知识和知识之间的逻辑推理关系来实现软件开发的。

从概念方面来看，结构化软件是功能的集合，通过模块以及模块和模块之间的分层调用关系实现；面向对象软件是事物的集合，通过对象以及对象和对象之间的通信联系实现；而专家系统软件是知识的集合，通过知识以及知识和知识之间的逻辑推理关系实现。

从构成方面来看，结构化软件＝过程＋数据，以过程为中心；面向对象软件＝(数据＋相应操作)的封装，以数据为中心；专家系统软件＝知识＋推理，以知识为中心。

从运行控制方面来看，结构化软件采用顺序处理方式，由过程驱动控制；面向对象软件采用交互式、并行处理方式，由消息驱动控制；专家系统软件采用交互式、并行处理方式，由数据驱动控制。

从开发方面来看，结构化方法的工作重点是设计；面向对象方法的工作重点是分析；专家系统方法的工作重点是知识的获取与表达。但是，在结构化方法中，分析阶段和设计阶段采用了不相吻合的表达方式，需要把在分析阶段采用的具有网络特征的数据流图转换为设计阶段采用的具有分层特征的结构图，在面向对象方法中则不存在这一问题。

从应用方面来看，相对而言，结构化方法更加适合数据类型比较简单的数值计算和数据统计管理软件的开发；面向对象方法更加适合大型复杂的人机交互式软件和数据统计管理软件的开发；专家系统方法更加适合逻辑推理型软件的开发。

从发展方面来看，面向对象方法是软件开发方法的发展方向。

1.2　计算机语言发展史

计算机是无知无觉无生命的机器，要使它能够按照人的意图工作，就必须使计算机懂得人的意图，接受人向它发出的命令和信息，这就要求人们用特定的计算机可以理解的语言与计算机交流，这就是计算机语言。计算机语言是人与计算机之间通信的语言，是人与计算机之间传递信息的重要媒介。计算机语言的发展，经历了从低级到高级、从单一到多元、从专用到大众化的发展历程。

1.2.1　低级语言阶段

低级语言阶段的计算机语言可以被机器直接理解，或经过简单“解释”后即能被计算

机接受，包括机器语言、汇编语言等。

1. 机器语言

自从有了计算机，也就有了计算机的编程。最初的计算机编程语言是所谓的机器语言(也称为第一代计算机语言)，机器语言是最低级的语言，直接使用机器代码进行编程。

机器语言就是机器指令的集合。每种计算机都有自己的指令集合，计算机能够直接执行用机器语言所编写的程序。机器语言包括指令系统、数的形式、通道指令、中断字、屏蔽字、控制寄存器的信息等。

使用机器语言编写程序是十分痛苦的事情。由于每台计算机的指令系统各不相同，在一台计算机上能被执行的程序，要想在另一台计算机上执行，必须另外编写程序，造成了重复工作。另外，由于使用的是针对特定型号计算机的语言，故而其运算效率是所有语言中最高的。

由于机器指令是用二进制数表示的，用机器语言编程最烦琐，往往消耗大量精力和时间，存在难记忆、易出错、难以检查程序和调试程序、工作效率低等固有的缺陷。

2. 汇编语言

为了减轻使用机器语言编程的痛苦，人们进行了一种有益的尝试，用一些简洁的英文字母、符号串来替代一个特定指令的二进制串，如用 ADD 代表加法，MOV 代表数据传递等。这样一来，人们很容易读懂并理解程序在干什么，编写、查错及维护都变得方便多了。这种改良的程序设计语言就是汇编语言，也称为第二代计算机语言。然而计算机本身是不认识这些符号的，这就需要一种专门的程序来负责将这些符号翻译成二进制数的机器语言，这种翻译程序称为汇编程序。

早期的计算机由于速度慢、内存小，衡量程序质量高低最重要的指标是机器执行的效率。但是，随着计算机技术的发展，机器硬件的性能大幅度提高，程序的复杂度也在增加，人们越来越要求把简单、重复性的工作交给机器去做，而设计人员则更多地去从事创造性的工作。程序的可读性和可维护性渐渐成为衡量程序质量高低的最重要的指标。很显然，汇编语言取代机器语言是一种趋势，是必然的结果。

但汇编语言同机器语言相比，并没有本质的区别，只不过是把机器指令用助记符号代替，所以汇编语言同样十分依赖于机器硬件，移植性不好。汇编语言提高了编程效率，改进了程序的可读性和可维护性。针对计算机特定硬件而编制的汇编语言程序，能够充分发挥计算机硬件的功能和特长，程序精炼而质量高。以致直到今天，仍然有一部分软件开发人员在用汇编语言编程。

同时也应该看到，虽然汇编语言将程序设计的编程效率提高了，但执行效率却降低了。因为汇编语言是一种符号语言，它几乎和机器语言一一对应，但在书写时却使用由字符串组成的助记符。例如，加法在汇编语言中用助记符 ADD 表示，减法用助记符 SUB 表示等。

所以，汇编语言仍然是低级语言，也被称为面向机器的语言，但它是机器语言向更高级语言进化的不可替代的桥梁。

1.2.2　高级语言阶段

通过前面的介绍可以知道机器语言和汇编语言都与具体的计算机相关联，计算机类型一旦改变，程序就得重写。为了进一步提高编程效率，改进程序的可读性与可维护性，又出现了许多高级程序设计语言，也称为第三代计算机语言。计算机语言由此进入了“面向人类”的高级语言阶段。

高级程序设计语言是一种面向过程的语言，它用一些符号和数字对实际问题进行描述，故也称为过程化语言。高级语言比低级语言更加抽象、简洁，是一种接近于人们使用习惯的程序设计语言。高级语言容易学习，通用性强，书写出的程序比较短，易于推广和交流，是很理想的一种程序设计语言。

一条高级语言的指令相当于几条机器语言的指令，用高级语言编写的程序同自然语言非常接近，易于学习。用高级语言编写程序并不需要具备某种计算机的专门知识。

高级语言的发展经过了诞生期、发展期、结构化期、多范型期 4 个阶段，在各个阶段都涌现出一批具有代表性的程序设计语言，如 FORTRAN 语言、COBOL 语言、LISP 语言、Pascal 语言、C/C++语言等。

1.2.3　超高级程序设计语言

高级语言是一种动用语言，要完成某一个简单的计算步骤，用户必须详细准确地给出每一条指令。如解决经营管理活动中天天都要碰到的财务清账、库存等问题，就须编写无数条程序，而情况一经变化，原有的设计程序则要修改，这样就使出错的可能性增大，工作效率大大降低。为了解决这个问题，出现了超高级程序设计语言，即“实用语言”，也称为第四代计算机语言。

第四代语言是 20 世纪 80 年代出现的一种非过程化程序设计语言。第三代计算机语言要求程序设计者具有较高的技术，不仅要告诉计算机做什么，还要告诉计算机如何做，而第四代语言则只要指出做什么就行了，只需指出所求问题、输入数据及输出形式，就能得出输出结果，无需对算法和计算过程进行描述。

第四代计算机语言主要包括数据库及其查询语言，如 SQL；应用生成程序；可执行规格说明语言。目前问世的第四代计算机语言有 ADF、MAPPER 等。

第四代计算机语言的特点是语言功能强、效率高、使用方便；开发应用系统修改方便、维护容易；系统复杂，不但要编译还要生成程序。

1.2.4　第五代计算机语言

第五代计算机语言利用可视化或图形化接口编程，从而生成一种原语言。这种原语言通常用第三代或第四代计算机语言编译器来进行编译。如微软、波兰、IBM 及其他一些公司就生产了一些可视化编程工具。这些工具可以用 java 语言来开发一些应用程序。

可视化编程可以使用户很容易地想象出面向对象编程的类层面，并且可以用一些拖拉式图标来装配程序组件。