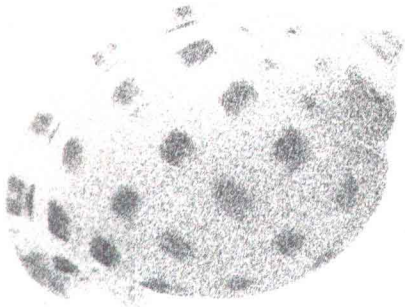
工业和信息化普通高等教育“十二五”规划教材立项项目
21世纪高等学校计算机规划教材
21st Century University Planned Textbooks of Computer Science
国家精品课程系列教材

大学程序设计技术概论

An Introduction to Programming Technology

董卫军 邢为民 索琦 编著
耿国华 主审

- 体系结构完善，强化能力培养
- 内容组织合理，促进理解消化
- 经典例题分析，便于知识巩固



高校系列

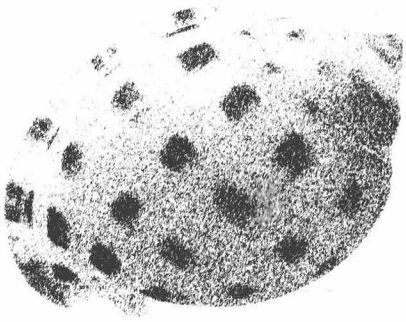
人民邮电出版社
POSTS & TELECOM PRESS

工业和信息[redacted]”规划教材立项项目
21世纪高等[redacted]家精品课程系列教材
21st Century University

大学程序设计技术概论

An Introduction to Programming Technology

董卫军 邢为民 索琦 编著
耿国华 主审



高校系列

人民邮电出版社
北京

图书在版编目（CIP）数据

大学程序设计技术概论 / 董卫军，邢为民，索琦编著. -- 北京 : 人民邮电出版社，2012.3
21世纪高等学校计算机规划教材
ISBN 978-7-115-27383-3

Ⅰ. ①大… Ⅱ. ①董… ②邢… ③索… Ⅲ. ①程序设计－高等学校－教材 Ⅳ. ①TP311.1

中国版本图书馆CIP数据核字(2012)第005069号

内 容 提 要

《大学程序设计技术概论》是国家精品课程“计算机基础”系列课程“大学程序设计技术概论”的主教材。教材以教育部计算机基础教育教学指导委员会关于高等学校计算机基础教育基本要求作指导，以培养大学生信息素质和计算能力为主线，以“程序设计方法+数据结构+软件工程+数据库基础”为知识体系，兼顾计算机等级考试要求，强化大学生程序设计素质和创造性思维能力的培养。

本书共5章，主要内容包括序言、程序设计概述、数据结构与算法、工程化程序设计、数据库基础等。每章后有小结和习题，以便读者巩固所学知识。

教材突出技术性，注重内容在应用上的层次性，兼顾整体在理论上的系统性。本书可作为高等学校“计算机基础”课程的辅助教材，也可作为全国计算机应用技术证书考试的培训教材或计算机爱好者的自学教材。

工业和信息化普通高等教育“十二五”规划教材立项项目

21世纪高等学校计算机规划教材

大学程序设计技术概论

◆ 编　　著　董卫军　邢为民　索　琦
主　　审　耿国华
责任编辑　贾　楠

◆ 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　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京昌平百善印刷厂印刷

◆ 开本：787×1092　1/16
印张　11　　2012年3月第1版
字数　287千字　　2012年3月北京第1次印刷

ISBN 978-7-115-27383-3

定价：29.80元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

大学生作为国家未来的建设者，是否具备相当的信息素养和掌握足够的信息技能已经成为影响国家竞争力的重要方面。传统的信息素质以信息获取和信息利用为核心，计算机作为信息时代的主要载体和工具，正担负着越来越重要的作用，计算能力的培养（使用计算机解决问题的能力）已成为信息素质的深层次要求。

本书是国家精品课程“计算机基础”系列课程“大学程序设计技术概论”的主教材。教材内容的组织以“程序设计方法+数据结构+软件工程+数据库基础”知识体系为基础，以培养大学生信息素质和计算能力为主线，逐步展开内容，从而使学习者获得完整的知识体系，在有限的时间内真正理解程序设计及其思想，同时对计算机的工作机制和实现过程有更为深入的理解。

本书体系合理、结构严谨，在编写时兼顾计算机等级考试的要求。

本书由多年从事计算机教育的一线教师编写，董卫军编写第 1 章、第 4 章~第 5 章及附录，邢为民编写第 2 章，索琦编写第 3 章。全书由董卫军统稿，西北大学耿国华教授主审。感谢教学团队成员的帮助。由于作者水平有限，书中难免有不妥之处，恳请读者指正。

编者

2011.12

目 录

第1章 序　言

大学生作为国家未来的建设者，是否具备相当的信息素养和掌握足够的信息技能已经成为影响一个国家竞争力的重要方面，培养大学生的信息素质是高等教育的基本目标之一。

1.1 信息素质的含义

1.1.1 信息素质的一般含义

1. 基本概念

信息素质来源于“信息素养”，该词最早是由美国信息产业协会主席保罗·泽考斯基在1974年提出来的。他认为信息素质是人们在工作中运用信息、学习信息技术、利用信息解决问题的能力。

计算机技术发展到今天，对信息素质最广泛的解释是：作为具有信息素质的人，必须具有一种能够充分认识到何时需要信息的能力，并有能力发现、检索、评价和利用所需要的信息，解决当前存在的问题。

在我国，信息素质通常被定义为：从各种信息源检索、评价和使用信息的能力，是信息社会劳动者必须掌握的终身技能。

2. 基本内涵

信息素质的内涵包括4个方面：信息意识、信息能力、信息道德、终身学习的能力。

（1）信息意识

信息意识指人们对情报现象的思想观点和人的情报嗅觉程度，是人们对社会产生的各种理论、观点、事物、现象从情报角度去理解、感受和评价的能力。具体来说它包含了对于信息敏锐的感受力、持久的注意力和对信息价值的判断力、洞察力。

（2）信息能力

信息能力也可以说是信息技能，包括确定信息需求的时机，选择信息源高效获取信息、处理评估信息、有效利用信息的能力。

（3）信息道德

信息道德指人们在信息活动中应遵循的道德规范，如保护知识产权、尊重个人隐私、抵制不良信息等。

（4）终身学习的能力

获得终身学习的能力是信息素质教育的目标。信息素质概念应该把焦点放在用户身上，即受

教育者或者被培训者身上，让用户学会学习，获得终身学习的能力。

1.1.2 信息素质的深层要求

传统的信息素质以信息获取和信息利用为核心。由于计算机作为信息时代的主要载体和工具，正担负着越来越重要的作用，计算能力的培养（使用计算机解决问题的能力）已成为信息素质的深层次要求。

1. 计算能力的概念

计算能力是指利用计算机解决问题的能力。传统的计算机操作包括能使用字处理作一些文字编辑，使用数据处理软件统计数据，使用因特网获取信息。计算能力则不同，它着眼于素质和能力的培养，是指理解计算机的内部结构和工作机制，理解计算机解决问题的原理和实现过程，并能将这些知识应用到以后的工作中，使计算机为工作提供深层次服务，而不是仅仅作为一种工具。

2. 计算机教育对思维品质的培养

计算机教育除了要培养思维素质外，其中最主要的还是对思维品质的培养，培养发现问题、思考问题和解决问题的能力。

（1）培养创造性思维

创造性思维最显著的特点是能够提供新颖独创而又有价值的思维成果，而传统的知识体系注重知识性内容的灌输，而忽略创造性思维的培养。而程序设计能极大地激发创造欲望。

（2）发展学生的抽象思维

程序设计是以抽象为基础的，要解决程序设计问题，首先要考虑适当的算法，通过对问题的分析研究，归纳出一般性的规律，然后再用计算机语言描述出来，将这个一般性的规律描述出来的过程就是一个高度抽象的过程。在程序设计中大量地使用了猜测、归纳、推理等思维方法，有利于培养抽象思维能力。

1.2 内 容 体 系

程序设计能力的培养绝非通过一门课可以解决，他们要通过多门课程的相互支撑才能实现。一般情况下，程序设计能力的培养至少需要通过“程序设计方法+数据结构+软件工程+数据库基础”的知识体系才能获得。

1. 程序设计方法

程序是计算机问题求解步骤的描述，是指挥计算机完成任务的指令序列。所以要让计算机解决问题，就需要告诉计算机问题是如何解决的，即需要编写计算机程序。

在编写计算机程序时，除了需要掌握某种程序设计语言外，还需要掌握相应的程序设计方法。学习程序设计方法的目标是设计出可靠、易读而且代价合理的程序。

2. 数据结构

程序可以简单地理解为：程序=算法+数据结构。要真正通过计算机解决问题，除了必须掌握一门程序设计语言外，还必须很好地理解数据结构和算法。“数据结构”是计算机存储、组织数据的方式；而“算法”是对解决问题的过程的描述。

在程序的设计中，数据结构的选择是一个基本的设计因素。许多大型系统的构造经验表明，系统实现的困难程度和系统构造的质量都严重依赖于是否选择了最优的数据结构。通过算法的学

习，主要达到如下两个目的。

（1）了解经典算法

（2）抽象思维能力的培养

算法是成百上千的数学家和工程师千锤百炼的精品，学习的过程便是学习前人的思考方法。当遇到具体问题的时候，可以尝试类比经典问题得到解答。

3. 软件工程

掌握了程序设计语言和数据结构后，已经具备了编写诸如字符编辑程序或报表打印程序等小型程序的能力。但是，如果需要研制一个大型的软件系统，例如，飞机订票系统或学校管理系统（包括教务、财务、人事、物资等各系科的全面管理），则会遇到许多困难，因此还必须通过“软件工程学”的学习，掌握多人协作的大型软件的开发方法。

用“工程化”的思想作指导，可以大大减少软件开发成本并提高软件质量。软件工程学研究的是：如何应用一些科学理论和工程上的技术来指导大型软件系统的开发，使其发展成一门严格的工程科学。

软件工程学的最终目的是：以较低的成本研制具有较高质量的软件。

4. 数据库原理

现代的计算机应用系统都离不开数据库的支持。所以，要开发具有实际应用价值的软件就必须了解有关数据库的基本知识。

数据库技术是计算机科学技术发展的重要内容，是构成信息系统的重要基础。数据库技术已广泛应用于各类管理信息系统（MIS）、决策支持系统（DDS）、计算机辅助软件工程（CASE）等领域。因此，学习和掌握数据库技术的基本知识和基本技能已成为大学生必备的素质。

第2章 程序设计概述

2.1 学习目标与要求

2.1.1 学习目标

掌握程序设计的基本规则，养成良好的程序设计习惯；掌握结构化程序设计的基本结构与特点；理解面向对象方法的一些基本概念。

2.1.2 学习要求

- 了解程序设计的基本方法，确立程序设计风格的原则。
- 了解结构化程序设计方法的基本原则。
- 了解面向对象的程序设计方法。

2.2 程序设计

计算机系统由硬件系统和软件系统组成。硬件系统是计算机的核心，是软件工作的基础。硬件就像人的躯体，软件系统就像人的灵魂。为了能让计算机解决一个具体的问题，不仅需要配备所需的计算机硬件设备，还要为其设计出能够解决这个问题的具体软件（简单地说，就是解决问题的计算机程序）。

2.2.1 程序设计中的基本概念

1. 程序概念

所谓计算机程序（Program），是计算机所执行的一系列指令的集合，通过这些指令集合，计算机可以实现数值计算、信息处理、信息显示等功能。也就是说，程序是为实现特定目标或解决特定问题而用计算机语言编写的命令序列的集合，为实现预期目的而进行操作的一系列语句和指令。

2. 程序设计语言

程序设计语言（Programming Language）是用于编写计算机程序的语言。语言的基础是一组记号和一组规则。根据规则由记号构成的记号串的总体就是语言。

3. 程序设计语言的分类

程序设计语言按级别可划分为低级语言和高级语言两大类，与高级语言相比，用低级语言开发的程序，其运行效率高。虽然运行效率高，但编写出来的程序可读性很差，且难以修改和维护，开发效率低。

① 机器语言：是特定的计算机系统所固有的语言。机器语言属于低级语言。

② 汇编语言：是机器语言的一种提升，它使用了一些助记符来表示机器指令中的操作码和操作数。但它仍然是一种和计算机机器语言十分接近的语言，使用起来仍然不太方便。同机器语言一样，汇编语言也属于低级语言。

③ 高级语言：与人们的自然语言比较接近，使用起来很方便，也极大地提高了程序设计效率。

程序设计语言按程序设计方法分为结构化程序设计语言和面向对象程序设计语言。

（1）结构化程序设计语言

结构化程序设计语言的结构特性主要有：一是用自顶向下逐步精化的方法编程；二是按照模块组装的方法编程；三是程序只包含顺序、选择（分支）及循环结构，而且每种构造只允许单入口和单出口。例如，C、PASCAl 等都是典型的结构化程序设计语言。

（2）面向对象程序设计语言

面向对象的程序设计在很大程度上应归功于从模拟领域发展起来的 Simula，Simula 提出了类和对象的概念。C++、Java 和 Smalltalk 是面向对象程序设计语言的代表。

2.2.2　程序设计风格

好的程序设计习惯和设计风格，有助于提高程序设计的效率，保障程序的正确性、程序的可阅读性，便于对程序进行调试和维护。主要有以下几个方面的原则。

1. 源程序文档化

① 标识符应按意取名（见名知意）。

② 程序应加注释。

注释使程序的读者或用户对程序的理解和使用起到帮助的作用。注释一般采用自然语言或形式化语言描述。它说明了程序的功能，特别在维护阶段，对理解程序提供了明确的指导作用。注释分为序言性注释和功能性注释。

序言性注释应置于每个模块的开始部分，主要内容如下。

① 说明每个模块的用途、功能。

② 说明模块的接口：调用形式、参数描述及从属模块的清单。

③ 数据描述：重要数据的名称、用途、限制及其他信息。

④ 开发历史：设计者、审阅者姓名，修改说明及日期。

功能性注释嵌入在源程序内部，说明程序段或语句的功能以及数据的状态。注意以下几点。

① 注释用来说明程序段，而不是每一行程序都要加注释。

② 使用空行、缩格或括号，以便很容易区分注释和程序。

③ 修改程序也应修改注释。

2. 数据说明规范化

在编写程序时，需要注意数据说明的风格，以便使程序中的数据说明更易于理解和维护。一般有以下几点需要注意。

① 数据说明顺序应规范。为了便于程序的理解、阅读和维护，数据说明次序应该相对固定，可以使数据的属性容易查找，有利于测试、排错和维护。

例如按以下顺序：常量命名、类型说明、全程量说明、局部量说明。

② 当一个语句说明多个变量时，变量最好按照字母顺序排序。

③ 对于复杂的数据结构，要加注释，说明在程序实现时的特点。

3. 语句结构简单化

语句设计的原则是：程序应该简单易懂，语句结构应该简单直接，不能为了追求效率而使代码复杂化。为了便于阅读和理解，尽可能不要在一行书写多条语句；不同层次的语句应采用缩进形式，使程序的逻辑结构和功能特征更加清晰；要避免复杂的判定条件，避免多重的循环嵌套；在表达式中使用括号以提高运算次序的清晰度等。

4. 输入和输出方便美观

输入和输出信息是用户直接关心的，输入操作步骤和输入格式应尽量简单，尽可能方便用户的使用，因为系统能否被用户接受，往往取决于输入和输出的风格。

① 输入操作步骤和输入格式尽量简单。

② 应检查输入数据的合法性、有效性，报告必要的输入状态信息及错误信息。

③ 输入一批数据时，使用数据或文件结束标志，而不要用计数来控制。

④ 交互式输入时，提供可用的选择和提示。

⑤ 当程序设计语言有严格的格式要求时，应保持输入格式的一致性。

⑥ 输出数据表格化、图形化。

5. 高效率

效率是指程序运行时对处理机和存储空间的使用情况，对效率的追求应遵守以下几点。

① 效率是一个性能要求，其目标在需求分析时给出。

② 追求高效率是建立在不损害程序可读性、可靠性的基础上，要先使程序正确、清晰，再提高程序效率，即当今程序设计的风格是“清晰第一，效率第二”。

③ 提高程序效率的根本途径在于选择良好的设计方法、良好的数据结构和算法，而不是单纯地在编程时对程序语句进行调整。

2.2.3 程序设计基本步骤

1. 分析实际问题

首先必须明确求解问题的意义和任务。对题目给出的已知条件和需要完成的任务进行详细的了解和分析，将一个实际问题转化为计算机可以处理的问题。对于接受的任务要进行认真的分析，研究所给定的条件，分析最后应达到的目标，找出解决问题的规律，选择解题的方法，完成实际问题。

2. 算法设计

所谓算法，简单地说就是计算机能够实现的有限的解题步骤。我们知道，计算机只能进行最基本的算术和逻辑运算，要完成较为复杂的运算和控制操作，必须选择合适的算法，这是正确编程的基础。

解决一个具体问题时通常有多种解题途径（算法）供选用，因此有必要知道哪一种算法是最好的，这就需要对算法执行效率进行分析。算法的复杂性是算法效率的度量，是评价算法优劣的重要依据。一个算法复杂性的高低体现在运行该算法所需要的计算机资源的多少，所需的资源越多，我们就说该算法的复杂性越高；反之，所需的资源越低，则该算法的复杂性越低。因此，算法的复杂性体现在时间效率和空间效率两个方面，分别称为时间复杂性（Time Complexity）和空间复杂性（Space Complexity）。

一般而言，对于任意给定的问题，设计出复杂性尽可能低的算法是我们在设计算法时追求的一个重要目标；另一方面，当给定的问题已有多种算法时，选择其中复杂性最低者，是我们选用算法应该遵循的一个重要准则。因此，算法的复杂性分析对算法的设计或选用有着重要的指导意义和实用价值。

3. 编写程序

编写程序是采用程序设计语言来实现已确定的算法。也就是说，使用计算机系统提供的某种程序设计语言，根据上述算法描述，将已设计好的算法表达出来。将非形式化的算法转变为形式化的由程序设计语言表达的算法，这个过程称为编写程序。程序在编写过程中需要反复调试，这样才能得到可以运行且结果“正确”的程序。

4. 程序的检验与调试

程序编写好以后，必须经过书面检查和上机调试，以便说明程序是否正确。检验时，应预先选择典型数据，运行可执行程序，得到运行结果，检查是否是预期结果。能得到运行结果并不意味着程序是正确的，要对结果进行分析，看它是否合理。若结果不合理要对程序进行调试，即通过上机发现和排除程序中的错误。

5. 编写文档说明

一个完整的软件应该有相应的说明文件，这不仅便于用户使用，也有利于对程序的维护和扩充。说明文件主要应包括程序名称、程序功能、程序的装入和程序启动的方法、运行环境、需要输入的数据、程序的基本结构和所采用的主要算法以及程序必要说明和注意事项等。

2.2.4　程序设计方法的发展

程序设计是一门技术，需要相应的理论、技术、方法和工具来支撑，就其发展而言，主要经过了早期程序设计阶段、结构化程序设计阶段和面向对象的程序设计阶段。

程序设计初期，由于计算机硬件条件的限制，运算速度与存储空间都迫使程序员追求高效率，编写程序成为一种技巧与艺术，而程序的可读性、可理解性和可扩充性等因素被放到第二位。随着计算机硬件与通信技术的发展，计算机应用领域越来越广泛，应用规模也越来越大，程序设计不再是一两个程序员可以完成的任务。在这种情况下，编写程序不再片面追求高效率，而是综合考虑程序的可靠性、可扩充性、可重用性和可理解性等因素。正是这种需求刺激了程序设计方法与程序设计语言不断向前发展。

1. 早期程序设计

早期出现的高级程序设计语言有 FORTRAN、COBOL、ALGOL、BASIC 等语言。

这一时期，由于追求程序的高效率，程序员过分依赖编程的技巧，不太注重所编写程序的结构，这时期可以说是无固定程序设计方法的时期。

存在的一个典型问题是程序中的控制随意跳转，即不加限制地使用 goto 语句，这样的程序对

别人来说是难以理解的，程序员自己也难以修改程序。

2. 结构化程序设计

随着程序规模与复杂性的不断增长，人们也在不断探索新的程序设计方法。证明了只用 3 种基本的控制结构（顺序，选择，循环）即可实现任何单入口/单出口的程序。Dijkstra 建议从一切高级语言中取消 goto 语句；Mills 提出程序应该只有一个入口和一个出口。这些工作导致了结构化程序设计方法的诞生。

PASCAL 语言是由 Niklaus Wirth 根据结构化程序设计方法开发出来的语言，其特点是提炼出程序设计共同的特征并能将这些特征编译成高效的代码，因而成为结构化程序设计的有力工具。C 语言也是一种广为流行的结构化程序设计语言，它具有灵活方便，目标代码效率高，可移植性好等优点。

3. 面向对象程序设计

面向对象方法是一种把面向对象的思想运用于软件开发过程中，指导开发活动的系统方法，简称 OO 方法，是建立在“对象”概念（对象、类和继承）基础上的方法学。对象是由数据和操作组成的封装体，与客观实体有直接的对应关系。

面向对象的方法起源于面向对象的编程语言。20 世纪 60 年代后期就出现了类和对象的概念，类作为语言机制用来封装数据和相关操作。70 年代前期，Smalltalk 语言，奠定了面向对象程序设计的基础，1980 年 Smalltalk-80 标志着面向对象的程序设计已进入实用阶段。进入 80 年代相继出现了一系列面向对象的编程语言，如 C++等。自 80 年代中期到 90 年代，面向对象的研究重点已经从语言转移到设计方法学方面，尽管还不成熟，但已陆续提出了一些面向对象的开发方法和设计技术。

2.3 结构化程序设计

20 世纪 60 年代出现的软件危机使得一大批软件的开发相继失败，软件危机使人们开始认真思考如何才能设计出结构合理、高质量的程序，于是提出了结构化程序设计（Structured Programming）的方法。结构化程序设计由迪克斯特拉（E.W.Dijkstra）在 1965 年提出，是以模块化设计为中心，将待开发的软件系统划分为若干个相互独立的模块，这样就使完成每一个模块的工作变单纯而明确，为设计一些较大的、复杂的软件提出了良好的解决思路。结构化程序设计是软件发展过程中的第 3 个重要里程碑，其影响比前两个里程碑（子程序、高级语言）更为深远。结构化程序设计的概念和方法及支持这些方法的一整套软件工具构成了结构化程序的核心。

2.3.1 结构化程序设计的基本思想

Dijkstra 的“结构化程序设计”思想，提出了一套方法，使程序具有合理的结构，以保证和检验程序的正确性。该方法规定程序设计者不能随心所欲地编写程序，而要按照一定的结构形式来设计和编写程序。其目的在于使程序具有良好的结构，这样程序就易于设计、阅读和理解，易于调试和修改，从而提高程序设计效率和程序维护程序的效率。

结构化程序设计的基本思想是采用的“自顶向下，逐步求精”的程序设计方法和“单入口单出口”的控制结构。自顶向下、逐步求精的程序设计方法从问题本身开始，经过逐步细化，将解

决问题的步骤分解为由基本程序模块组成的结构化程序框图；“单入口单出口”的思想认为一个复杂的程序，如果它仅是由顺序、选择和循环 3 种基本程序结构通过组合、嵌套构成，那么这个新构造的程序一定是一个单入口单出口的程序。在设计时要考虑什么时候输入数据、什么时候处理数据、什么时候输出数据等操作过程。据此就很容易编写出结构良好、易于调试的程序来。具体包括以下几个方面的概念。

（1）自顶向下的设计方法

程序设计时，应先考虑总体，后考虑细节；先考虑全局，后考虑局部目标。即先从最上层总目标开始设计，逐步使问题具体化。

（2）模块化的设计方法

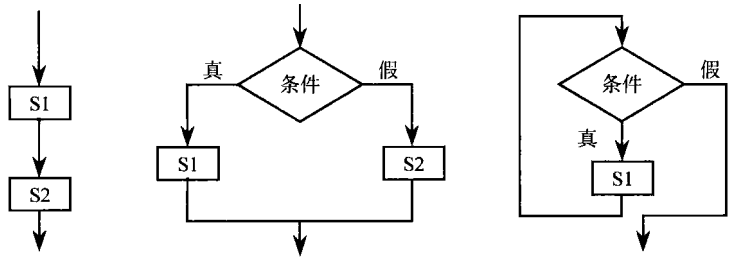
一个复杂问题，都是由若干个简单问题构成的。模块化即是将复杂问题进行分解，即将解决问题的总目标分解成若干个分目标，再进一步分解为具体的小目标，把每一个小目标称作一个模块。

（3）逐步求精的设计方法

在一个程序模块内，先从该模块功能描述出发，逐层细化，直到最后分解、细化成语句为止。这样就构成了实现模块功能的程序。在一个系统的设计与实现时，也采用逐步求精的方法进行设计、编码、测试。

（4）在程序编码中采用的 3 种基本控制结构

在结构化程序设计中，任何程序均可由 3 种基本结构构成，这 3 种基本结构分别是顺序结构、选择结构和循环结构，如图 2.1 所示。



（a）顺序结构　　（b）选择结构　　（c）循环结构

图 2.1　程序的 3 种基本结构

① 顺序结构。从前向后顺序执行 S1 语句和 S2 语句，只有当 S1 语句执行完后才执行 S2 语句。

② 选择结构。根据条件有选择地执行语句。当条件成立（为真）时执行 S1 语句，当条件不成立（为假）时执行 S2 语句。

③ 循环结构。有条件地反复执行 S1 语句（也称为循环体），直到条件不成立时终止循环，控制转移到循环体外的后继语句。

人们已经证明，以上 3 种基本结构可以派生出其他形式的结构。由这 3 种基本结构所构成的算法可以解决任何复杂的问题，结构化程序就是指由这 3 种基本结构所组成的程序。一般高级程序设计语言都具有这 3 种结构的控制语句。

（5）限制使用 goto 语句

从理论上讲，只用顺序结构、选择结构、循环结构这 3 种基本结构就能表达任何一个只有

一个入口和一个出口的程序逻辑。如果大量使用 goto 语句实现控制路径，会使程序的执行路径变得复杂而且混乱，增加出错的机会，降低程序的可读性和可靠性，因此要控制 goto 语句的使用。

2.3.2 结构化程序设计的优缺点

采用结构化程序设计方法，程序结构清晰，易于阅读、测试、排错和修改。由于每个模块具有单一功能，模块间联系较少，使程序设计比过去更简单，程序更可靠，而且增加了可维护性，每个模块可以独立编制、测试。

1. 优点

① 程序易于阅读、理解和维护。程序员采用结构化编程方法，将一个复杂的程序分解成若干个子程序结构，便于控制，降低程序的复杂性，因此容易编写程序，同时便于验证程序的正确性。

② 提高了编程工作的效率，降低了软件开发成本。由于结构化编程方法能够把错误控制到最低限度，因此能够减少调试和查错的时间。

2. 缺点

结构化程序设计方法是面向过程的。它把程序定义为“数据结构+算法”，程序中数据与处理这些数据的算法（过程）是分离的。这样，对不同的数据结构进行相同的处理或对相同的数据结构进行不同的处理，都要使用不同的模块，降低了程序的可维护性和可重用性。这种分离导致了数据可能被多个模块使用和修改，难以保证数据的安全性和一致性。因此，对于小型程序和中等复杂程度的程序来说，它是一种较为有效的方法，但对于复杂的、大规模的软件开发来说，会出现可维护性和可重用性差的问题。

2.4 面向对象程序设计

面向对象方法的本质，是主张从客观世界固有的事物出发来构造系统，提倡用人类在现实生活中常用的思维方法来认识、理解和描述客观事物，强调最终建立的系统能够反映问题域，即系统中的对象以及对象之间的关系能够如实地反映问题域中客观事物及其关系。

2.4.1 面向对象程序设计思想的产生

随着软件产业的蓬勃发展，软件系统变得越来越庞大、复杂，开发成本越来越高，而且在开发过程中容易出现一系列问题，例如 IBM360 操作系统历时 4 年才完成，并不断修改、补充，但仍存在大量错误。这种软件开发与维护过程中遇到的一系列严重问题被人们称为“软件危机”。出现软件危机的原因是多方面的，如软件需求变化频繁、开发工具落后等。人们尝试从不同角度、不同层次来解决软件危机问题，如严格确定软件需求、采用新的开发模型、采用计算机辅助工具等。

面向对象程序设计思想就是在这一大环境中产生的。在面向对象程序设计语言产生之后，面向对象程序设计逐步成为编码的主流，其中所蕴涵的面向对象的思想不断向开发过程的上游和下游发展，形成现在的面向对象分析、面向对象设计、面向对象测试，并逐步发展为面向对象软件开发方法。

1. 传统开发方法存在的问题

（1）软件可重用性差

重用性是指同一事物不经修改或稍加修改就可多次重复使用的性质。软件重用性是软件工程追求的目标之一。

（2）软件可维护性差

在软件开发过程中，应保证软件的可读性、可修改性和可测试性。实践证明，用传统方法开发出来的软件，维护成本高。

软件工程强调软件的可维护性，强调文档资料的重要性，规定最终的软件产品应该由完整、一致的程序和文档组成。在软件开发过程中，始终强调软件的可读性、可修改性和可测试性是软件的重要的质量指标。实践证明，用传统方法开发出来的软件，维护时其费用和成本仍然很高，其原因是可修改性差，维护困难，导致可维护性差。

（3）开发出的软件不能满足用户需要

用传统的结构化方法开发大型软件系统涉及各种不同领域的知识，在开发需求模糊或需求动态变化的系统时，所开发出的软件系统往往不能真正满足用户的需要。

用结构化方法开发的软件，其稳定性、可修改性和可重用性都比较差，这是因为结构化方法的本质是功能分解，从代表目标系统整体功能的单个处理着手，自顶向下不断把复杂的处理分解为子处理，这样一层一层分解下去，直到仅剩下若干个容易实现的子处理功能为止，然后用相应的工具来描述各个最低层的处理。因此，结构化方法是围绕实现处理功能的“过程”来构造系统的。然而，用户需求的变化大部分是针对功能的，因此，这种变化对基于过程的设计来说是灾难性的。用这种方法设计出来的系统结构常常是不稳定的，用户需求的变化往往造成系统结构的较大变化，从而需要花费很大代价才能实现这种变化。

2. 面向对象程序设计思想的产生及发展

发明面向对象程序设计方法的主要出发点是弥补面向过程程序设计方法中的一些缺点。为了克服传统软件开发方法的缺陷，需要一种新的程序设计思想，面向对象程序设计思想便是在这种情况下逐渐产生的。第一个面向对象的程序设计语言是 20 世纪 60 年代开发的 Simula-67，它提供了比子程序更高一级的抽象和封装，引入了数据抽象和类的概念，它被认为是第一个面向对象的语言。在类和对象的概念中，对象代表着待处理问题中的一个实体，在处理问题过程中，一个对象可以以某种形式与其他对象通信。从理论上讲，一个对象是既包含数据，又包含处理这些数据操作的一个程序单元。类用来描述特性相同或相近的一组对象的结构和行为。该语言还支持类的继承，可将多个类组成为层次结构，进而允许共享结构和行为。

20 世纪 70 年代初，出现了 Smalltalk 语言，Smalltalk 语言是第一个比较成功的面向对象语言，对后来面向对象语言的发展产生过重大影响。该语言丰富了 Simula 中类和对象的概念，信息也更加隐蔽，程序设计就是向对象发送信息。之后又开发出 Smalltalk-80。Smalltalk-80 被认为是最纯正的面向对象语言，它对后来出现的面向对象语言产生了深远的影响。随着面向对象语言的出现，面向对象程序设计应运而生，且得到迅速发展。之后，面向对象不断向其他阶段渗透，出现了面向对象设计、面向对象分析等概念。

20 世纪 80 年代以后，面向对象的程序设计语言广泛应用于程序设计，并且有许多新的突破。特别是随着操作系统和软件项目日益庞大，人们日益需要一种更高效的开发方式，这更加推动了面向对象语言的发展，出现了面向对象程序设计方法。