高等院校信息技术规划教材

程序设计基础

——从问题到C语言程序

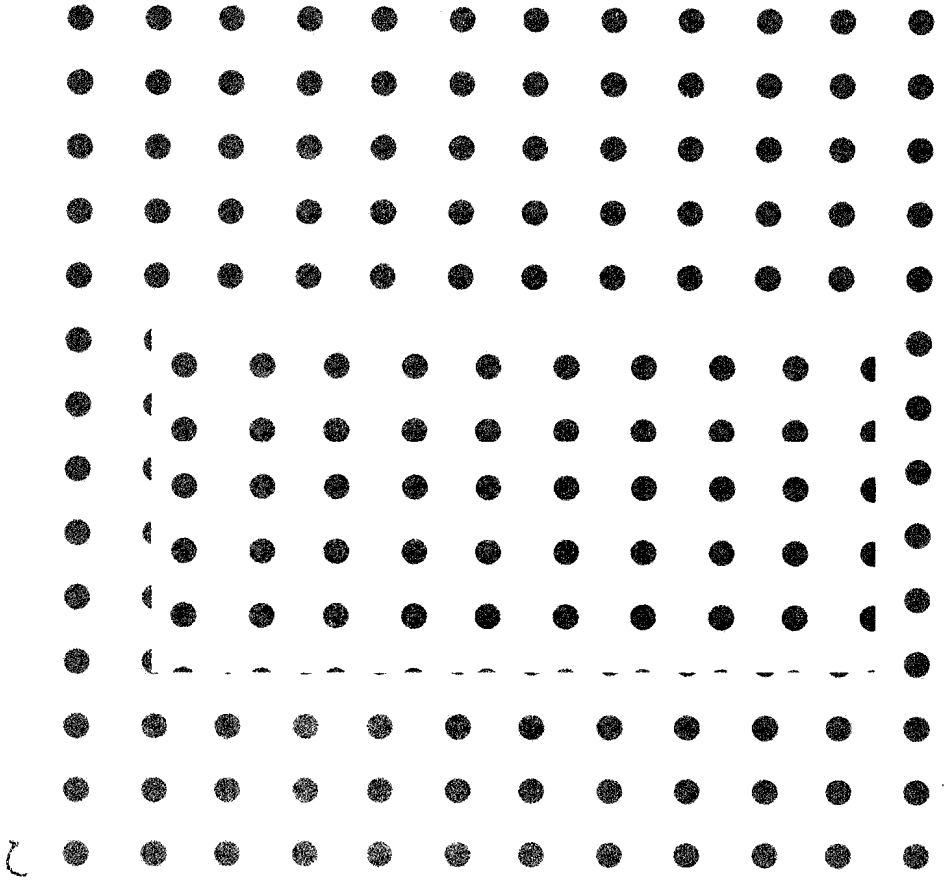
王创伟 蔡长安 编著

清华大学出版社

高等院校信息技术规划教材

程序设计基础
——从问题到C语言程序

王创伟 蔡长安 编著



清华大学出版社
北京

内 容 简 介

本书以C程序设计语言为工具，以程序设计为主线，编程应用为驱动，通过案例和问题引入内容，重点讲解程序设计的思想和方法，并结合相关的语言知识介绍，各章节中的实例经过集成后，最终构成一个完整的学生成绩管理系统应用程序。全书将基本知识与实际案例相融合，以侧重培养学生从实际问题空间向程序设计空间转换的能力和分析、解决问题的能力，让学生明白C语言知识只是一种工具，突出“程序设计”才是核心的理念。

本书适合作为高等院校计算机及相关专业的教材，也可供其他学习C程序设计语言的读者阅读。

图书在版编目（CIP）数据

程序设计基础——从问题到C语言程序 / 王创伟，蔡长安编著．—北京：清华大学出版社，2011.11
（高等院校信息技术规划教材）
ISBN 978-7-302-26754-6

Ⅰ．①程…　Ⅱ．①王…②蔡…　Ⅲ．①C语言－程序设计　Ⅳ．①TP312

中国版本图书馆CIP数据核字(2011)第183935号

责任编辑：焦　虹
责任校对：梁　毅
责任印制：李红英
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦A座
http://www.tup.com.cn　　**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
印 装 者：北京市清华园胶印厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：13.75　**字　数**：327千字
版　　次：2011年11月第1版　　**印　次**：2011年11月第1次印刷
印　　数：1～3000
定　　价：23.00元

产品编号：044184 01

前言 foreword

程序设计是高等学校重要的计算机基础课程，它以编程语言为平台，介绍程序设计的思想和方法。通过该课程的学习，学生不仅要掌握高级程序设计语言的知识，更重要的是在实践中逐步掌握程序设计的思想和方法，培养问题求解和应用能力。因此，这是一门以培养学生程序设计基本方法和技能为目标，以实践能力为重点的特色鲜明的课程。

C语言是得到广泛使用的程序设计语言之一，从1972年问世以来，至今已历经30多个年头。在此期间，信息技术得到迅猛发展，诞生了众多优秀的程序设计语言。然而，C语言仍是软件百花园中的奇葩，在系统开发、软件工程、软件测试等领域独领风骚。特别在程序设计、数据结构等教学方面已成为事实上的“标准”语言。C语言能保持经久不衰，是与其一系列突出的优点分不开的，它简洁、高效、灵活、可移植性好、应用面广，是第三代语言的杰出代表。

虽然目前介绍程序设计基础的教材很多，但在多年的教学实践中，我们发现适合程序设计入门课程教学要求的书并不多。现有的C语言教材一般围绕语言本身的体系组织内容，以讲解语言知识为主，特别是语法知识，辅以一些小例子的介绍，例子之间相互孤立，没有联系，不利于培养学生的程序设计能力和应用能力，缺少对学生思考能力的培养。

好的教材来源于教学改革和教学实践，能体现教学改革的成果。我们从2007年开始实施全方位的程序设计课程的教学改革，目的就是培养学生的程序设计能力，以适应新世纪人才培养的需求。经过多年的建设，在教学内容、教学方法、教学手段和考核方式上，已经基本形成一套比较完整的体系，本书充分展示了程序设计教学的改革成果。

本书以学生成绩管理系统的程序设计为主线，以编程应用为驱动，通过实例和问题引入内容，重点讲解程序设计的思想和方法，并穿插介绍相关的语言知识。每章在介绍实例时，首先从实际问题的角度思考问题的解决方法（问题空间），然后从问题空间转化为平台

空间，再从平台空间给出解决问题的算法，最后把算法映射为C语言的程序代码，给出程序运行结果，分析程序中用到的知识点。各章节中的实例经过集成后，最终构成一个完整的应用程序。本书将基本知识与实际案例相融合，以侧重培养学生从实际问题空间向程序设计空间转换的能力和分析、解决问题的能力，让学生明白C语言知识只是一种工具，突出“程序设计”才是核心的理念。

由于各专业在培养目标、课程设置及学时安排等方面存在差异，所以在采用本教材授课时，可以对内容酌情进行取舍，可以讲授全部内容，也可对一些章节内容重点讲解，一些章节内容由学生自学。

本书由王创伟、蔡长安两位编写完成。在编写过程中，汤克明老师提出了本书的总体框架设计，同时得到教学第一线专家、教师的许多宝贵意见和建议，在此表示衷心的感谢。

由于作者水平有限，书中难免有不妥甚至错误之处，恳请专家和广大读者批评指正。

作　者

目录 contents

第1章 chapter 1

绪　论

本章教学目标

(1) 理解什么是程序，程序如何描述。

(2) 了解什么是算法，算法有哪些特征。

(3) 了解程序设计的概念，什么是程序设计语言，计算机执行程序的过程。

(4) 了解C程序设计语言的发展及特点。

1.1 程序与程序设计

1.1.1 程序的概念

“程序”一词来自生活，通常指完成某些事务的既定方式和过程。比如早起活动按照如下的“程序”进行：

第一步，起床；

第二步，刷牙；

第三步，洗脸；

第四步，吃饭；

第五步，上学。

按顺序实施这些步骤，即完成了该项事务。

从上面的步骤可以看出它有一些基本特征[1]：(1)按部就班地进行；(2)开始与结束；(3)完成工作；(4)基本动作；(5)需要用某些记法形式描述；(6)不同的描述粒度(细节程序)等。

计算机程序是人们为解决某种问题用计算机可以识别的代码编排的一系列加工步骤，是人与计算机交流信息的基本方式，人通过程序指挥计算机的活动。

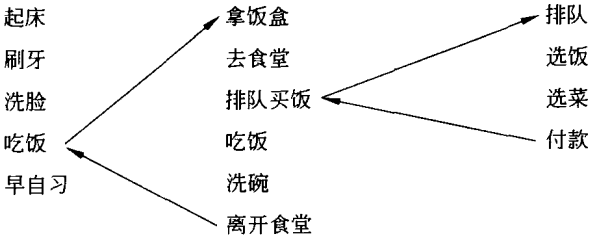
人们为了让计算机完成某一项任务，就必须要告诉计算机先做什么，然后再做什么，接着再做什么，直到计算机把这项任务完成。这一步步为机器事先安排的指令或者叫命令，就叫“程序”。计算机是听不懂我们所说的话，也识别不了我们的文字，它只能认识机器语言，就是“0”和“1”。可是人们如果用一长串的“0”和“1”来和机器说话，就会显得太麻烦，各种各样的“0”和“1”的组合表示着各种不同的意思，程序员很难记住。于是产生

了汇编语言，每一句机器语言都有一个相对应的助记符来表示，这就称为低级语言。由于编一个程序要使用成千上万条语句，如果用低级语言来写语句，就显得太慢了，于是就出现了高级语言，比如“C”语言，或者“B”语言等。高级语言的特点是：一条语句就相对应于几条，或者几十条机器语言。显然，编程时使用高级语言要比使用低级语言效率高得多；但是，它的缺点是计算机在执行高级语言写的程序前，先要把其转化为低级语言0和1的组合才能执行，因此执行速度不如用低级语言编写的程序高。

计算机程序，是指为了得到某种结果而可以由计算机等具有信息处理能力的装置执行的代码化的指令序列，或者是可以被自动转换成代码化指令序列的符号化指令序列或符号化语句序列。

1.1.2 程序设计的概念

人们描述（编制）计算机程序的工作称为程序设计或编程，其产品就是程序。由于计算机的本质特征，从它诞生之初就有了程序设计工作。要用计算机处理问题，写程序时就必须精确描述所需的全部细节，不能有一点含糊。但编写程序不应该从细节开始，应从问题需要出发，从高层开始设计程序，并逐步分解程序的功能，直至用程序语言实现。例如早起活动“程序”分解如下：



什么是程序设计？我们从一个著名的公式说起[2]：程序设计＝数据结构＋算法。数据结构即非数值计算的程序设计问题中的计算机的操作对象以及它们之间的关系和操作；算法是对特定问题求解步骤的一种描述，是针对指令的有序序列。实际上程序设计就像盖房子，数据结构就像砖、瓦，而算法就是设计图纸。你若想盖房子首先必须有原料（数据结构），但是这些原料不能自动地盖起你想要的房子，你必须按照设计图纸（算法）上的说明一砖一瓦地去砌，这样你才能拥有你想要的房子。程序设计也一样，虽然编译工具中有各种功能语句或基本结构（如Read/Write/Real/Boolean等），但它们不会自动排列成所需要的程序代码，必须按照程序规定的功能去编写，而程序功能的实现就是算法的具体体现。所以通俗地说：必须按照特定的规则，把特定的功能语句和基本结构按照特定的顺序排列起来，形成一个有特定功能的程序，即：程序设计＝数据结构＋算法。数据结构是程序设计这座大厦的基础，没有基础，无论设计有多么高明，这座大厦也是不可能建造起来的。算法则是程序设计的思想，是程序的灵魂！没有灵魂的程序不能叫程序，只是一堆杂乱无章的符号而已。在程序设计中，数据结构就像物质，算法就是意识。这就像哲学上所说的：意识依赖于物质而存在，物质由于意识而发展，双方相互依存、缺一不可！

程序设计（programming）是指设计、编制、调试程序的方法和过程。它是目标明确的

智力活动。由于程序是软件的本体，软件的质量主要通过程序的质量来体现，因此在软件研究中，程序设计的工作非常重要，内容涉及有关的基本概念、工具、方法以及方法学等。程序设计通常分为问题建摸、算法设计、编写代码和编译调试四个阶段。

1.1.3 程序设计的基本认知

1. 编程应该具有的基本素质

经常会有人问："(我)能不能学会编写程序"这样的问题。当一个软件被装在电脑里并开始运行时，人们便开始惊讶于程序员的厉害。"能不能学会编写程序"，甚至成了一些人对自己的智力考评，所以便有了这样的发问。仅就编写程序来说，实在是一件很简单的事，甚至可以说是一种劳力活。除了先天智障或后天懒惰者，其他人都是可以学会编写程序的。

学好编程，应具备一些基本素质：

1) 数学基础

从计算机发展和应用的历史来看，计算机的数学模型和体系结构等都是由数学家提出的，最早的计算机也是为数值计算而设计的。因此，要学好计算机就要有一定的数学基础，初学者有高中水平就差不多了。

2) 逻辑思维能力

学程序设计要有一定的逻辑思维能力，"思维能力"的培养要长时间的实践锻炼。要想成为一名优秀的程序员，最重要的是掌握编程思想。要做到这一点必须在反复的实践、观察、分析、比较、总结中逐渐积累。因此在学习编程的过程中，不必等到什么都完全明白了才去动手实践，只要大概明白，就要敢于自己动手去体验，谁都有第一次。有些问题只有通过实践后才能明白，也只有实践才能把老师和书上的知识变成自己的，高手都是这样成材的。

3) 良好的编程习惯

编程入门不难，但入门后不断学习是十分重要的，相对来说这个过程较为漫长。在此期间要注意养成一些良好的编程习惯。编程风格在很大程度影响着程序质量。良好的编程风格可以使程序结构清晰合理，且使程序代码便于维护。如代码的缩进编排、变量命令规则的一致性、代码的注释等。

4) 多问多学习

掌握编程思想必须在编程实际工作中去实践和体会，编程起步阶段要经常自己动手设计程序，具体设计时不要拘泥于固定的思维方式，遇到问题要多想几种解决的方案。这就要多交流。各人的思维方式不同、角度各异、各有高招，通过交流可不断学习别人的长处，丰富编程实践，帮助自己提高水平。亲自动手进行程序设计是创造性思维应用的体现，也是培养逻辑思维的好方法。

2. 程序设计的工具

目前常用的程序设计工具大致分为编程语言、数据库、可视化编程、专业系统以及客

户机/服务器等五类，分别为：

1）编程语言类

编程语言开发工具主要是指由传统编程工具发展而来的一类程序设计语言。如C语言、C++语言、BASIC语言、COBOL语言、PL/1语言、PASCAL语言、LISP语言等。

2）数据库类

数据库是管理信息系统最重要的组成部分，它是系统中数据存放、数据传递、数据交换的中心和枢纽。数据库管理系统是管理和操作数据库的主要工具。目前市场上提供的数据库管理系统大致有两类：一类是微机数据库管理系统，如dBASE、FoxBASE、Visual FoxPro等；另一类是大型数据库管理系统，如ORACLE、SYBASE、INGRE、INFOMAX等。

3）可视化编程类

Visual Basic开辟可视化程序设计的先河，以它为代表的一批可视化、面向对象的开发工具应运而生。如Visual FoxPro、Visual Basic、Visual C++、Power Builder等。这类开发工具的特点是利用图形工具和可重用部件来交互地编制程序。

4）专业系统类

专业系统类开发工具是在可视化、面向对象的开发工具基础上发展起来的，它不但具有这些工具的功能，而且更加综合化、图形化，因而使用起来更加方便。目前专业系统类开发工具主要有Excel、SDK、SQL、OPS等。

5）客户机/服务器类

客户机/服务器工具解决问题的思路很简单，它就是在原有开发工具的基础上，将原有工具改变为既可被其他工具调用，又可调用其他工具的“公共模块”。这样今后系统的开发工作就可以不局限于一种语言、一类工具，而是综合使用各类工具的长处，更快、更好地实现一个应用系统。

目前市场上的客户机/服务器类工具主要有Borland International Inc.公司的Delphi Client/Server，Powersoft Corp.公司的Power Builder Enterprise，Sysmantec Corp.的Team Enterprise Developer等。这类工具最显著的特点就是它们之间相互调用的随意性。另外，像Delphi Client/Server、Power Builder Enterprise和Team Enterprise Developer等工具，都是面向对象的工具，功能很强，能够支持SQL等对各种大型数据库管理系统的数据操作，所开发的系统能够实现客户机/服务器类型的程序调用关系，一般应用于管理软件、数据处理和网络系统的开发。

3. 程序设计的方法（从现实空间到计算空间）

当今社会是信息社会，信息社会的灵魂是作为“信息处理机”的电子计算机。从1946年第一台计算机ENIAC问世到今天的“深蓝”，电子计算机的硬件得到了突飞猛进的发展，程序设计的方法也随之不断进步。20世纪70年代以前，程序设计方法主要采用流程图，结构化设计（structure programming，SP）思想日趋成熟，20世纪80年代，SP是主要的程序设计方法。然而，随着信息系统的加速发展，应用程序日趋复杂化和大型化，传统的软件开发技术难以满足发展的新要求。20世纪80年代后，面向对象的程序设计

(object orient programming,OOP)和组件对象模型程序设计技术日趋成熟并逐渐为计算机界所理解和接受。

1) 结构化程序设计

早期的计算机存储器容量非常小,人们设计程序时首先考虑的问题是如何减少存储器开销,硬件的限制不容许人们考虑如何组织数据与逻辑。程序本身非常短小,逻辑简单,也无须人们考虑程序设计方法问题。与其说程序设计是一项工作,倒不如说它是程序员的个人技艺。但是,随着大容量存储器的出现及计算机技术的广泛应用,程序编写越来越困难。由于程序大小以算术级数递增,而程序的逻辑控制难度则以几何级数递增,因此人们不得不考虑程序设计的方法。

最早提出的方法是结构化程序设计方法,其核心是模块化[3]。1968 年 Dijskstra 发表文章,注意到了"结构化程序设计",之后,Wulf 主张"可以没有 GOTO 语句"。从 1975 年起,许多学者研究了"把非结构化程序转化为结构化程序的方法","非结构的种类及其转化","结构化与非结构化的概念"等问题。结构化程序设计逐步形成了既有理论指导又有切实可行方法的一门独立学科。SP 方法主张使用顺序、选择、循环三种基本结构来嵌套连接成具有复杂层次的"结构化程序",严格控制 GOTO 语句的使用。用这样的方法编出的程序在结构上具有以下效果:

(1) 以控制结构为单位,只有一个入口,一个出口,因此能独立地理解这一部分。

(2) 能够以控制结构为单位,从上到下顺序阅读程序文本。

(3) 由于程序的静态描述与执行时的控制流程容易对应,所以能够方便正确地理解程序的动作。

SP 的要点是:"自顶而下,逐步求精"的设计思想[4],"独立功能,单出、入口"的模块仅用三种(顺序、分支、循环)基本控制结构的编码原则。自顶而下的出发点是从问题的总体目标开始,抽象低层的细节,先专心构造高层的结构,然后再一层一层地分解和细化。这使设计者能把握主题,高屋建瓴,避免一开始就陷入复杂的细节中,从而使复杂的设计过程变得简单明了,过程的结果也容易做到正确可靠。"独立功能,单出、入口"的模块结构减少了模块的相互联系,使模块可作为插件或积木使用,降低了程序的复杂性,提高了可靠性。程序编写时,所有模块的功能通过相应的子程序(函数或过程)代码来实现。程序的主体是子程序层次库,它与功能模块的抽象层次相对应,编码原则使得程序流程简洁、清晰,增强了可读性。

在 SP 中,划分模块不能随心所欲地把整个程序简单分解成一个个程序段,而必须按照一定的方法进行。模块的根本特征是"相对独立,功能单一"。换言之,一个好的模块必须具有高度的独立性和相对较强的功能。模块的好坏,通常用"耦合度"和"内聚度"两个指标从不同侧面而加以度量。所谓耦合度,是指模块之间相互依赖性大小的度量,耦合度越小,模块的相对独立性越大。所谓内聚度,是指模块内各成分之间相互依赖性大小的度量,内聚度越大,模块各成分之间联系越紧密,其功能越强。因此在模块划分时应当做到"耦合度尽量小,内聚度尽量大"。

结构化程序与非结构化程序相比有较好的可靠性、易验证性和可修改性。结构化设计方法的设计思想清晰,符合人们处理问题的习惯,易学易用;模块层次分明,便于分工

开发和调试，程序可读性强。C语言就是其中的代表。

2）面向对象的程序设计

随着程序设计复杂性的增加，结构化程序设计方法就不够用了。不够用的根本原因是“代码重用”的时候不方便。于是面向对象的方法诞生了，它通过继承来实现比较完善的代码重用功能。

其实面向对象的思想和我们日常生活中处理问题的方式是吻合的。举例来说，我打算丢掉一个茶杯，怎么扔呢？太简单了，拿起茶杯，走到垃圾桶，扔！注意分析这个过程。先选一个“对象”——茶杯，然后向这个对象施加一个动作——扔。每个对象所能施加在它上面的动作是有一定限制的：茶杯，可以被扔，可以被砸，可以用来喝水，可以敲它发出声音；一张纸，可以写字，可以撕，可以烧。也就是说，一旦确定了一个对象，方法也就跟着确定了。我们的日常生活就是如此。

面向对象技术给软件设计领域带来极大的变化。它利用软件对象来进行程序开发。所谓对象是包含数据和对数据操作的代码实体，或者说是在传统的数据结构中加入一些被称为成员函数的过程，因而赋予对象以动作。而在程序设计中，对象具有与现实世界的某种对应关系，我们正是利用这种关系对问题进行分解。

从程序语言角度来看，在一个对象中代码和（或）数据可以是这个对象私有的，不能被对象外的部分直接访问。因而对象提供了一种高级保护以防止程序被无关部分错误修改或错误地使用了对象的私有部分。当从对象外部试图直接对受保护的内部数据进行修改时，将会被程序拒绝。只有通过对象所提供的对外服务函数才能够对其内部数据进行必要的加工，从而保证数据加工的合法性。因此，将这种代码和数据的联系称为“封装”。换句话说，封装是将对象封闭保护起来，从而将内部细节隐蔽起来。

在强调软件组件的重用方面，面向对象的技术与标准的工业设计规律有更多相似之处。在面向对象语言中，类是创建对象的关键。事实上类描述了一族对象的公共特征和操作，而对象则是具体实现的类。例如小汽车是一个基本概念，它具有颜色、几何尺寸、动力特性的特征。我们可以定义一个称为car的类，具有颜色、几何尺寸、动力特征等参数，以及描述汽车在外界条件下运动状态的成员函数。一辆具体的小汽车则是一个对象，在这个对象中有关参数均有具体数值，并可以通过输入说明变量（外界条件参数）获取该车的具体运动状态。

3）组件对象模型程序设计

随着硬件越来越快，越来越小，软件的规模却越来越大了，因此集体合作开发越来越重要，代码重用又出现了新的问题。组件对象模型（component object model，COM）程序设计是在面向对象程序设计技术的基础上发展起来的。它可以实现软件的功能模块化、编程语言的无关性、操作系统的无关性等，极大地提高了代码的可重用性、软件的可扩展性等。

同COM技术并行的另一个技术是公共对象请求代理体系结构（common object request broker architecture，CORBA）技术，即发展前景也非常广阔。CORBA的目的是简化开发分布式应用系统的复杂性及减少成本。CORBA使用了面向对象和组件的设计结构，允许软件对象在不同的操作系统平台和应用程序中重复使用。