高等院校计算机教材系列

Delphi 程序设计大学教程

刘艺　罗兵　周安栋　编著

机械工业出版社
China Machine Press

高等院校计算机教材系列

Delphi 程序设计大学教程

刘艺　罗兵　周安栋　编著

机械工业出版社
China Machine Press

本书以Delphi语言为载体，通过讨论程序设计的一般过程和方法，重点讲述程序设计基础、算法与结构化程序设计、面向对象程序设计、Windows程序设计和数据库程序设计的知识，并涉及计算机科学基础、数据和控制、程序设计理论、软件工程等四大知识领域。本书同时详细分析Delphi作为通用程序设计语言的本质特征和语法规则，并以大量Delphi程序实例演示应用程序的设计过程，介绍主流的思想方法，培养读者的代码编写能力。

本书内容深入浅出，覆盖面广，图文并茂，独具特色。既有丰富的理论知识，也有大量的实战范例，更提供了精心设计的课后练习。

本书适合作为计算机及其相关专业本科教学用书，也可用作其他专业的计算机公共课基础教材。对于自学程序设计的计算机爱好者以及从事软件开发和应用的科技人员，本书也是极佳的参考。

图书在版编目（CIP）数据

Delphi程序设计大学教程/刘艺等编著. -北京：机械工业出版社，2005.6
（高等院校计算机教材系列）
ISBN 7-111-16421-0

Ⅰ.D…　Ⅱ.刘…　Ⅲ.软件工具-程序设计-高等学校-教材　Ⅳ.TP311.56

中国版本图书馆CIP数据核字（2005）第029626号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
策划编辑：温莉芳
责任编辑：朱　劼
北京诚信伟业印刷有限公司印刷·新华书店北京发行所发行
2005年6月第1版第1次印刷
787mm×1092mm 1/16·20.75印张
印数：0 001-5 000册
定价：30.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

前　言

欢迎进入Delphi的世界学习计算机程序设计课程。这将是一次美妙和激动人心的探索，可能会为你今后从事充满挑战和令人兴奋的职业奠定软件编程的基础。众所周知，计算机在我们的日常生活中扮演了一个重要的角色，而且在未来仍将继续扮演这一重要角色。

计算机科学是一个充满挑战和发展机遇的年轻学科，而计算机程序设计则是这门学科的重要基础。随着计算机在各行各业的广泛应用，很多非计算机专业也把计算机程序设计列为公共基础课之一。

既然是作为基础课的教材，那么本书所假定的读者就是不具有程序设计经验，也没有面向对象技术概念和Windows程序设计知识的人。即使是一个对计算机一无所知的人，也能通过学习本书而获取所有有关的基本知识，学习程序设计。如果读者是一位很有经验的程序员，已经用过其他程序设计语言，并掌握了一定的开发技能，也能在本书中发现很多有用的信息。

本书与程序设计课程

计算机程序设计既是一门概念复杂、知识面广的理论课，也是一门面向实战、需要动手的实践课。几乎所有的编程初学者都梦想着有朝一日能在计算机上驰骋，让一行行程序在自己敲击键盘的手下源源不断地流出，真正成为驾驭计算机的主人。然而，学完程序设计课程后，实际开始编写程序时，却往往会觉得难以下手、无所适从。尽管自己刻苦学习，高分通过考试，但并不能体会到所学的知识给实际编程带来的便利和优势。

为什么会这样?一方面是因为我们的学生在学习时没有掌握程序设计的一般过程，没有深入了解通用程序设计语言的本质规律；另一方面是我们的教学体制僵化、教材陈旧，教学思想和内容跟不上时代的发展，与软件开发实际情况脱节。

计算机程序设计语言是一种实现对计算机进行操作和控制的人造语言，与人类的自然语言有一定差距。程序设计语言仅仅是程序设计的手段和途径而并不是程序设计的全部。因此，掌握程序设计语言并不意味着就精通程序设计，就能写出优秀的程序。实际上，程序设计所涉及的领域、知识和技能要远远超出我们的想象。因此本教材对于程序设计课程在一些方面有着自己独特的理解。

程序设计首先是一个过程

程序设计过程通常分为问题建模、算法设计、编写代码和编译调试4个阶段。不同阶段的任务是相对独立的，不能混为一谈。即使是一个比较简单的程序，我们也应该养成先分析，再下手，最后调试的习惯，严格遵循程序设计过程。因为在缺乏对问题深入、全面分析的情况下，就匆匆动手编写程序，将会增加失败的风险，带来后期修改、维护的麻烦。因此，要学习程序设计，不但不能回避程序设计过程，更要从软件开发过程和软件生命周期的高度来了解和掌握程序设计过程，从一开始就要养成遵从程序设计准则的良好习惯。有别于其他程序设计教材，本书强调程序设计过程和软件开发过程的重要性，为读者介绍了有关软件建模与测试的基本原理和技术。特别考虑到现代软件开发依赖于集体合作和项目管理，是汇集了很多程序设计过程的更大的过程。因此，除了在书中增加有关软件过程实施和管理的介绍外，还把如何撰写规范的程序代码作为重要一节，使得读者在学习程序设计之初就了解程序设计的规范，注重编写程序的规范性、正确性和可靠性，对于培养将来参与大型软件开发所需要的分工合作团队成员十分重要。

程序设计还是一种解决问题的方法和能力

程序设计课程的目标是学习用计算机解决问题的思考方法，培养编程应用能力，而不是仅仅学会某个程序设计语言的语法规则。很多学生能弄清楚循环、if-else结构以及算术表达式，但很难把一个编程问题分解成结构良好的程序。这暴露了程序设计教学中偏重语法细节，忽略总体思想方法和整体过程实现的问题。

尽管程序设计理论的发展为解决问题提供了很多有效方法，但对于初学者而言，学习的捷径应该是抓住最核心的思想方法，即结构化方法和面向对象方法。为实现这个目的，我们既把结构化算法分析和设计作为教材重点，也把面向对象分析和设计作为重点。对于前者，我们以顺序结构、选择结构和循环结构这三种基本结构为基础，讲解常用的结构化算法；对于后者，我们则围绕面向对象的抽象性、继承性、多态性和封装性这4个本质特点阐述面向对象程序设计的基本方法。通过强调基本概念、基本方法、基本应用，为初学者奠定扎实的程序设计基础，树立良好的编程思想。通过大量的实例分析和范例程序设计过程演示，我们力图给初学者建立完整印象，培养其从整体上思考问题和解决问题的编程能力。

程序设计最终是对程序设计语言的应用

程序设计和程序设计语言存在着有趣的辩证关系。程序设计可以用不同的程序设计语言来实现，但是不同的程序设计语言又决定着能使用怎样的程序设计思想方法和技术技巧，制约着程序设计的实现能力和效率。本书使用Delphi作为学习程序设计的语言，并不是因为Delphi有强大的可视化编程功能，而是因为Delphi不但继承了Pascal语言完美的结构化风格，而且还具有面向对象语言的真正优势。更可喜的是，Delphi还在继续发展，不断吸取现代编程语言的精华。这一切使得Delphi具备现代通用程序设计语言的主流特征，特别适合教学使用。因此学习Delphi语言，掌握Delphi程序设计方法是本课程的另一个重要任务。

本书虽然以Delphi语言为背景介绍程序设计语言的相关知识，但是重点强调的是一些通用的思想方法，而放弃了Delphi的一些奇技淫巧。读者应该注意到，不同的程序设计语言其语法和风格可能迥异，但无论哪一种语言，都是以数据（类型）、操作（运算）、控制（逻辑流程）为基本内容。更进一步讲，学习一门程序设计语言，应该超越语言的具体表述格式，不拘泥于繁芜的语法现象，而是站在抽象的高度，掌握程序设计的基本概念，深入了解程序设计语言的本质规律。这样将会为深入学习其他程序设计语言带来便利。

本书的结构

本书是为计算机程序设计课程编写的。该课程是理工类大学的公共基础课，它通过讲授一门具体的计算机语言，来帮助学生掌握程序设计的基础知识和基本应用。同时，对于未接触过计算机科学的学生，本书还涉及和介绍了与程序设计相关的计算机科学知识。程序设计基础、算法与结构化程序设计、面向对象程序设计、Windows程序设计这4个部分组成了本书的核心，并涉及计算机基础、数据和控制、程序设计理论、软件工程知识等4大知识领域，汇集20个相关知识点。虽然在本书中讨论的内容有一定的理论性，但这些理论都是用实际程序问题表达的，这是为了帮助读者建立完整的程序设计知识体系结构。

本书的组织结构如图1所示，其中有些知识点是通过各章节迭代，从而循序渐进，不断深入的。

本书共有11章，各章的主要内容如下：

- **第1章 绪论** 介绍程序设计的基本概念，并初步认识Delphi。重点帮助读者搞清计算机程序、程序设计和程序设计语言等基本概念。
- **第2章 程序设计基础** 以数据和数据处理作为程序设计的基础，通过了解变量、常量和数据类型开始探索Delphi程序设计语言。通过建立第一个Delphi程序介绍Delphi语言要素，说明如何撰写规

范的程序代码。

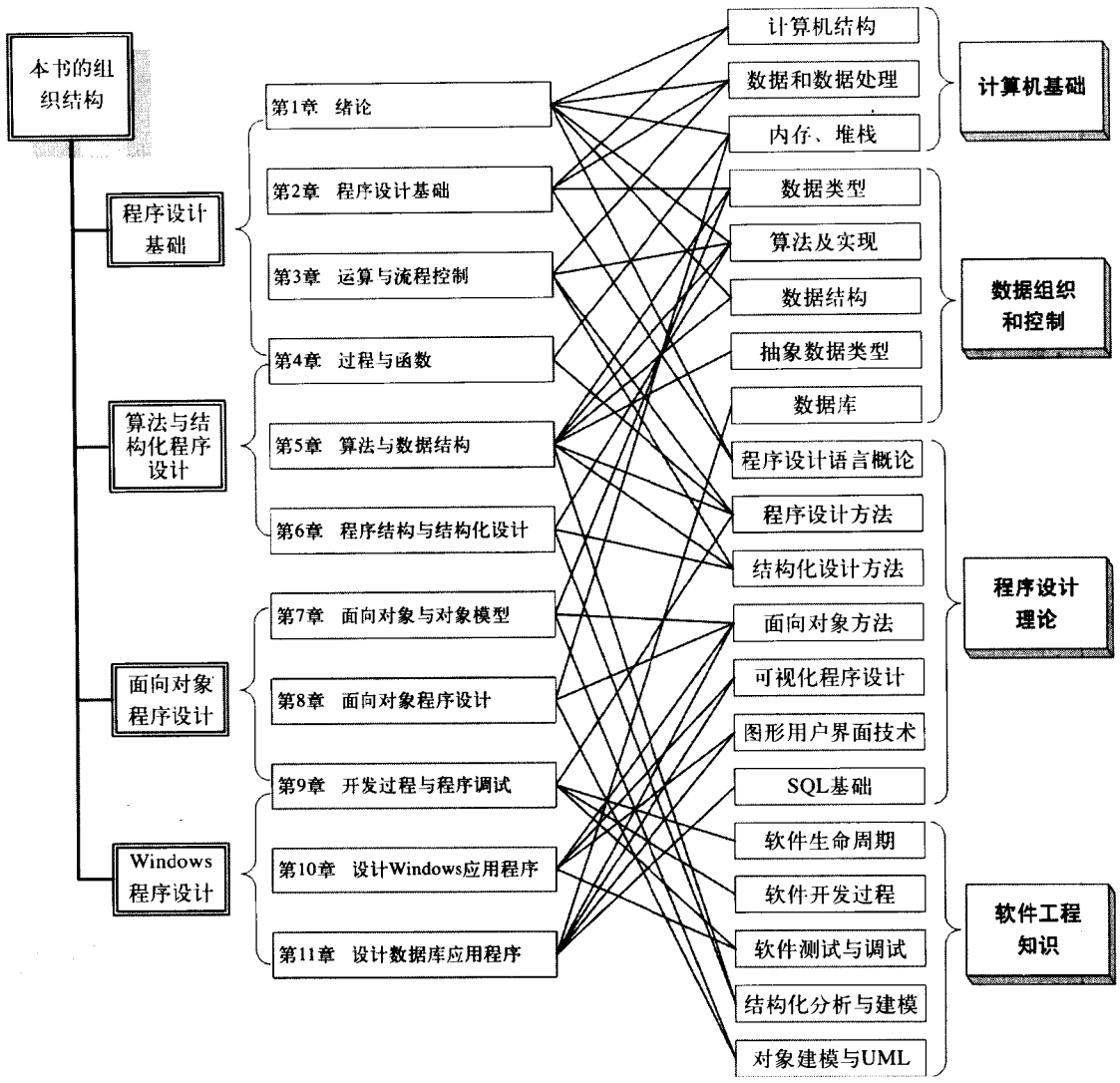


图1 本书组织结构与知识点

- **第3章 运算与流程控制** 介绍Delphi运算符和表达式。讲解表达式语法、各种算术、逻辑和其他运算符。通过介绍实现顺序、选择、循环结构的常用Delphi语句，帮助读者完整地理解如何控制程序的逻辑流程。
- **第4章 过程与函数** 讲解Delphi过程和函数的声明与实现，介绍参数类型与传递方式。最后讨论过程和函数的使用。
- **第5章 算法与数据结构** 介绍算法和数据结构的概念及常用算法。通过集合、数组、链表、栈、队列等数据结构，讨论算法的应用及算法的Delphi程序实现。
- **第6章 程序结构与结构化设计** 剖析Delphi程序的基本结构，介绍结构化程序设计方法。引入单元概念并且展示如何从较简单的程序块构造复杂的结构化程序。最后通过一个结构化设计的应用实例，演示结构化程序设计的完整实现过程。

- **第7章　面向对象与对象模型**　介绍面向对象的概念和对象建模的方法，讲解对象模型中的核心部分：类及类的成员，使读者学会如何用Delphi语言声明和实现一个类以及如何使用方法和属性。
- **第8章　面向对象程序设计**　围绕面向对象的抽象性、继承性、多态性和封装性这4个特点讲解面向对象程序设计的基本方法。重点讲授如何使用对象和对象接口，如何在程序设计中使用继承、合成和多态等面向对象技术。
- **第9章　开发过程与程序调试**　介绍软件的开发过程及过程的实施管理，从软件质量的高度讨论程序的调试与测试，重点讲述Delphi程序的调试方法和程序中的异常处理。
- **第10章　设计Windows应用程序**　讲解Windows应用程序的一般设计方法，包括如何创建窗体、设计界面、使用控件、处理事件等。读者可以学到Delphi在可视化快速开发Windows应用程序方面的强大功能。
- **第11章　设计数据库应用程序**　读者将学习如何用Delphi创建关系数据库应用程序，同时还介绍一些数据库理论基础。除了学会使用Delphi功能强大的数据库组件，本章还通过实例重点讲解和演示了ADO和SQL数据库编程。

尽管本书包含以上内容，但实际的教学进度和授课内容可以灵活确定，因为这要取决于课堂教学的安排或读者的实际技能及对所讨论问题的熟悉程度。教学课时数建议安排在20~60课时之间。其中标记*的章节仅供有能力的学生选学。

本书的读者对象

- 没有任何程序设计基础的学生。选用本书作为程序设计基础教材，读者可以掌握Delphi程序设计语言，同时为深入学习其他计算机课程奠定基础。显然Delphi无论是在结构化还是面向对象，或者是可视化Windows编程方面都能胜任教学的需要，从而为学习程序设计提供更全面的知识结构体系。
- 因工作或科研需要希望迅速掌握一门程序设计语言以完成不太复杂的编程任务的非专业编程人员。
- 非计算机专业的编程爱好者，改行从事程序员工作，有一些实际的经验，但没有系统学习过相关专业知识。可以通过本书重温程序设计知识，补习相关概念和理论。
- 有一定经验的程序员，已经学习过其他编程语言并在软件开发中掌握了一定的开发技能，但没有使用过Delphi语言或对Delphi语言一知半解。可以利用本书系统学习Delphi程序设计，发现Delphi与其所熟悉语言的不同点，并由此掌握Delphi语言。

本书的特色

本书的一些特色不仅使得本书与众不同，同时也特别有助于入门者的学习。

概念和知识面

贯穿本书，我们始终强调概念要比数学模型更重要，我们认为对概念的理解必然左右对模型的理解。同时，我们还特别注意开阔读者的知识面，使读者能够站在现代软件开发和软件工程这个比较开阔的层面上了解程序设计，而不是局限于繁琐的程序设计语言规则。

代码编写能力

初学Delphi程序设计的人，大多数会被Delphi强大的可视化编程功能所吸引。的确，快速应用开发(RAD)的思想正在改变程序设计的方法，而可视化程序设计实际上已经重造了开发者的工作平台。以前编写程序是通过基于字符的编辑器键入一条条语句的方法，现在我们可以交互式地在窗体上点击和拖放组件，并使用短小精悍的代码段连接它们。过去，即使是开发很小的Windows应用程序，也需要很多细

心而且繁杂的基础工作，先写出非常长的源代码文件，然后才可编译和测试其结果。Delphi改变了这种模式。它首先创建一个Windows应用程序的初始框架代码，即缺省的用户界面，而无须写任何程序。程序员的创造力被用于真正的程序设计任务，而不是浪费在窗体的几个组件上。

可是，千万不要被容易的、可视化的、基于组件的程序设计所迷惑，把程序设计变成了点击和拖放组件，然后在窗体上重新排列组件的工作。如果是这样学习程序设计，我们培养的是拖拉现成组件的“拖拉员”而不是创造代码的程序员。真正的程序设计需要很多知识、技能和创造力。为此，我们的教材不采用大多数Delphi程序设计教材那种以可视化程序设计为重点的编写模式。为避免过早地引入可视化组件的使用，分散学习程序设计语言基础知识的注意力，我们在本书第10章“设计Windows应用程序”之前尽量以控制台程序为示例讲解程序设计方法，重在学习程序设计语言的本质，培养代码的编写能力。

图文并茂

本书有大量精心设计的插图，这些插图可以帮助读者增进对文字的理解。

示例程序

本书尽可能地运用示例程序来表述概念和模型，同时尽量提供完整的示例程序和程序设计过程。本书所有示例程序和习题中的程序都已在Windows环境下Delphi 7中编译运行通过，建议读者在学习时选择Delphi 7。

小结和习题

每一章的结尾都包括“本章小结”和“本章习题”。“本章小结”简明概括该章中所有关键内容和知识点，是复习时的参考。“本章习题”包括三部分内容：复习题、测试题和练习题。

- 复习题：测试该章中所有的要点和概念，帮助学生复习巩固重点内容。
- 测试题：通过选择题客观地测试学生对所学知识的理解和掌握程度。
- 练习题：通过课后练习题，检查学生能否运用掌握的概念和知识独立思考，解决问题。

本书为教师准备了电子教案。另外，本书的部分源代码可以从http://www.liu-yi.net以及http://www.hzbook.com获得。

CC 2004课程体系

从1990年开始，美国电气和电子工程师协会计算机社团（Computer Society of the Institute for Electrical and Electronic Engineers，简称IEEE-CS）和计算机学会（Association for Computing Machinery，简称ACM）就着手开发新的本科生计算机课程体系。1991年联合推出了Computing Curricula 1991（简称CC 1991），当时仅限于Computer Science（计算机科学）和Computer Engineering（计算机工程）两个专业的课程。1998年秋季开始，IEEE-CS和ACM联合投入新的力量更新该课程体系，并在2001年开发出Computing Curricula 2001（简称CC 2001），并将该计算机课程体系扩大到Computer Science（计算机科学）、Computer Engineering（计算机工程）、Software Engineering（软件工程）、Information Systems（信息系统）等多个专业。在CC 2001的实施中，专家们发现，计算机课程所涉及的学科专业和教学范围正在不断扩大，而且在内容和教学方面的变化也日新月异。IEEE-CS和ACM意识到10年一次的Computing Curricula修订已经难以满足要求，于是联合国际信息处理联合会（International Federation for Information Processing，简称IFIP）、英国计算机协会（British Computer Society，简称BCS）等更多的组织开发了Computing Curricula 2004（简称CC 2004），使之成为开放的、可扩充的、适合多专业的、整合了计算机教学相关原则体系观点的课程体系指南。其结构参见图2。

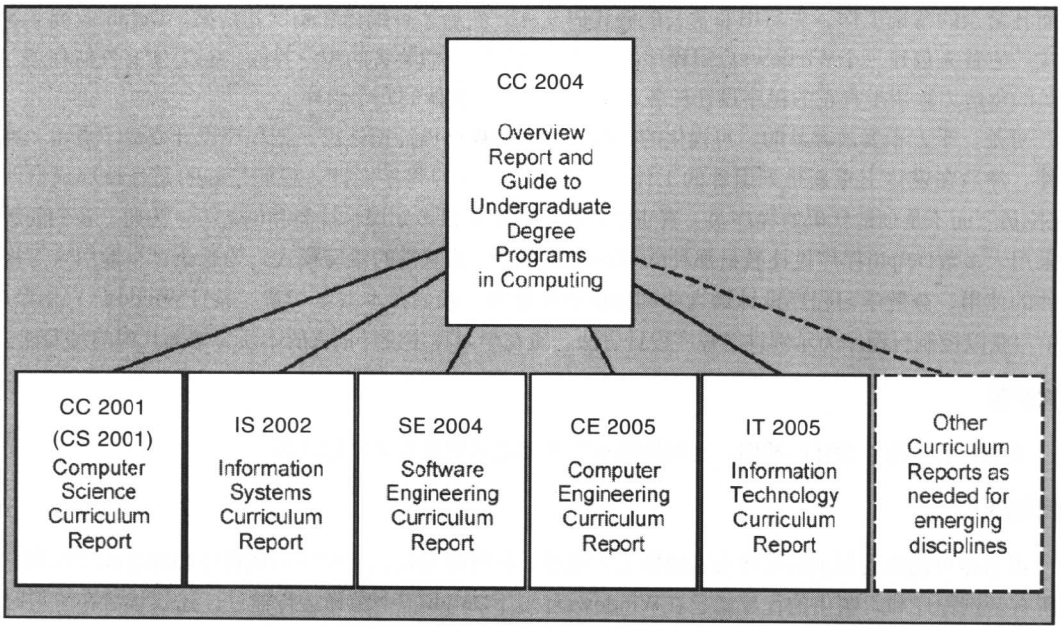


图2　CC 2004课程体系

为了进一步反映当代计算机科学技术的发展水平，与国际主流计算机教育思想接轨。通过多年来对IEEE-CS和ACM的Computing Curricula课程体系的跟踪研究，我们在本教材的编写中，借鉴了CC 2004课程体系的最新研究成果，同时吸取了国外同类教材的优秀经验，目的是进一步推动教材和课程改革，培养有竞争力的人才。

致谢

本书是作者在多年科研和教学基础上编写的，主要参考了作者已发表的文章和著作以及教学中积累的资料。书中还参阅了其他中外文教材、资料，由于无法在此一一列举，现谨对这些教材和资料的作者表示衷心的感谢。

参与本教材编写工作的人员还有网通吉林市分公司的孙滔，太原师范学院计算机中心的刘星，海军工程大学的段立、李启元、杜军、吴苗、曹旭峰，南京航空航天大学无人驾驶飞机研究所的吴英，以及杨德刚、郭巍、刘潘、吴永逸、洪蕾等。

一本书的出版离不开许多人的支持，尤其是这本书。为此感谢我们的家人和朋友。我们在忍受写作之苦的同时，牺牲了与他们共享天伦之乐的宝贵时光。

由于作者水平有限，本书中难免有疏漏和不妥之处，恳请各位专家、同仁和读者不吝赐教，并在此表示特别感谢!

http://www.liu-yi.net

2005年2月17日于南京

目　录

第1章 绪 论

计算机程序设计对于很多初学者来说，充满了神秘的诱惑。本章通过介绍计算机程序设计和程序设计语言的基础知识，揭开了程序设计神秘的面纱，帮助读者理解计算机程序、程序设计和程序设计语言等基本概念。

Delphi作为我们要学习的程序设计语言是本章要了解的重点之一。我们会沿着Delphi的发展历史，了解Delphi是什么，Delphi程序是如何编写、编译和运行的。总之，通过本章的学习，我们将为开始Delphi程序设计的探索之旅做好最充分的准备。

1.1 程序与程序设计

程序是指按照时间顺序依次安排的工作步骤，而程序设计则是对这些步骤进行编排和优化。程序设计的历史比计算机更长，只不过计算机的出现使程序设计有了更专用的领域——计算机程序设计，并得到空前的发展。计算机程序设计又称为编程（programming），是一门设计和编写计算机程序的科学和艺术。

1.1.1 程序与计算机

人们用程序的形式存储一系列指令已经有几个世纪的历史了。18世纪的音乐盒和19世纪末20世纪初的自动钢琴，就可以播放音乐程序。这些程序以一系列金属针或纸孔的形式存储，每一行（针或孔）表示何时演奏一个音符，而针或孔则表明此时演奏什么音符。19世纪初，随着法国发明家约瑟夫-玛丽·雅卡尔的由穿孔卡片控制的编织机的发明，人们对物理设备的控制变得更加精巧。在编织特定图案的过程中，编织机的各个部分要进行机械定位。为了使这个过程自动进行，雅卡尔使用一张纸质卡片代表织机的一个定位，用卡片上的孔来指示应该执行织机的哪个操作。整条花毯的编织可以用一叠这样的卡片编码，每次使用这叠卡片都会编出相同的花毯图案。使用这种可编程的编织机时，有的复杂程序需要使用24 000多张卡片。

世界上第一台可编程的机器是由英国数学家和发明家查尔斯·巴比奇设计的，但他并未将之制造完成。这台叫做分析机的机器，使用和雅卡尔的织机类似的穿孔卡片来选择每个步骤应执行的具体算术运算。插入不同的卡片组，就会改变机器执行的运算。这种机器能在现代计算机中找到类似的对应物，只不过它是机械化的，而非电气化的。分析机的制造没有完成，是因为制造它所需要的技术当时不存在。

供分析机使用的最早的卡片组程序是由诗人拜伦勋爵的女儿——英国数学家奥古斯塔·埃达·拜伦开发的。由于这个原因，她被公认为世界上第一位程序员。

现代的内部存储计算机程序的概念是由美籍匈牙利数学家约翰·冯·诺伊曼于1945年首次提出来的。冯·诺伊曼的想法是使用计算机的存储器来存储数据和程序。这样，程序可被视作数据，可像数据一样被其他程序处理。这一想法极大地简化了计算机中的程序存储与执行的任务。

计算机程序是指导计算机执行某个功能或功能组合的一套指令。要使指令得到执行，计算机必须执行程序，也就是说，计算机要读取程序，然后按准确的顺序实施程序中编码的步骤，直至程序结束。一个程序可多次执行，而且每次用户给计算机输入的选项和数据不同，就有可能得到不同的结果。

现代计算机都是基于冯·诺伊曼模型结构的，此模型着眼于计算机的内部结构，定义了处理机的运行过程。该模型把计算机分为四个子系统：存储器、算术/逻辑单元、控制单元和输入/输出单元。下面

分别介绍这四个子系统:

- **存储器** 存储器是用来存储的区域，计算机在处理过程中用存储器来存储数据和程序，我们会在后面讨论存储数据和程序的话题。
- **算术/逻辑单元** 算术/逻辑单元是用来进行计算和逻辑操作的地方。如果是一台数字处理用的计算机，那么它应该能够进行数字运算（例如进行一系列的数字相加运算）。当然它也应该可以对数据进行一系列逻辑操作（例如，找出两个数字中的小的一个）。
- **控制单元** 控制单元是用来对存储器、算术/逻辑单元、输入/输出等子系统进行控制操作的单元。
- **输入/输出单元** 输入单元负责从计算机外部接受输入数据和程序；输出单元负责将计算机的处理结果输出到计算机外部。输入/输出单元的定义相当广泛，它们还包含辅助存储设备，例如，用来存储处理所需的程序和数据的磁盘和磁带等。若一个磁盘用于存储处理后的输出结果，我们就可以认为它是输出设备，如果是从该磁盘上读取数据，那么该磁盘就被认为是输入设备。

冯·诺伊曼模型中要求程序必须存储在内存中。这和早期只有数据才存储在存储器中的计算机结构完全不同。在早期的计算机中，完成某一任务的程序是通过操作一系列的开关或改变其配线来实现的。

现代计算机的存储单元主要用来存储程序及其响应数据。这意味着数据和程序应该具有相同的格式，实际上它们都是以位模式（0和1序列）存储在内存中的。

冯·诺伊曼模型中的程序是由一组数量有限的指令组成。按照这个模型，控制单元从内存中提取一条指令，解释指令，接着执行指令。换句话说，指令一条接一条地顺序执行。当然，一条指令可能会向控制单元发出请求，以便跳转到其前面或者后面的指令去执行，但是这并不意味着指令没有按照顺序来执行。

如果不考虑计算机的内部物理结构，我们可以简单地把计算机看作一个黑盒。但是，仍然需要通过定义计算机所完成的工作来区别其和其他黑盒之间的差异。下面我们介绍两种常见的计算机模型，一种是数据处理器，另一种是可编程数据处理器。

1. 数据处理器

可以把计算机看作一个数据处理器。依照这种定义，计算机就是一个接受输入数据，处理数据，产生输出数据的黑盒（如图1-1所示）。尽管这个模型能够体现现代计算机的功能，但是它的定义还是太片面。按照这种定义，便携式计算器也可以认为是计算机（按照字面意思，它也符合定义的模型）。

另一个问题是这个模型并没有说明它处理的类型以及是否可以处理一种以上的类型。换句话说，它并没有清楚地说明一个基于这个模型的机器能够完成操作的类型和数量。它是专用机器还是通用机器呢?

这种模型可以表示为一种用来完成特定任务的专用计算机（或者处理器），比如用来控制建筑物温度或汽车油料使用的计算机。但在当今世界上，计算机是一种通用的机器，它可以完成各种不同的工作。这表明我们需要改变对计算机定义的模型来反映当今计算机的现实。

图1-1 数据处理模型

2. 可编程数据处理器

一个相对较好的适用于具有通用性的计算机的模型如图1-2所示。图中添加了一个新的元素，即程序到计算机内部的部分。程序是计算机对数据进行处理的指令集合。在早期的计算机中，这些指令是通过对配线的改变或一系列开关的打开闭合来实现的。今天，应用程序则是用计算机语言所编写的一

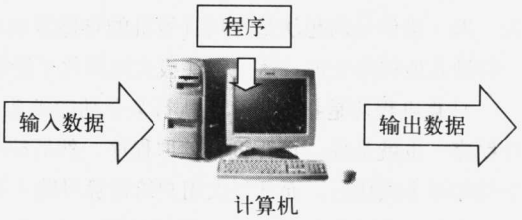


图1-2 可编程数据处理器模型

系列指令的集合。

在这个新模型中，输出数据与输入数据和程序的综合作用有关。对于相同的输入数据，如果改变程序，则可能产生不同的输出。类似的，对于同样的程序，如果改变输入内容，其输出结果也将不同。如果输入数据和程序保持不变，那么输出结果也不变。

冯·诺伊曼模型的主要特征在于其存储器的概念。尽管早期的计算机没有使用这种模型，但它们还是使用了程序的概念。编程在早期的计算机中体现为对一系列开关的开闭合和配线的改变。编程是在实际开始处理数据之前由操作员和工程师完成的一项工作。

冯·诺伊曼模型改变了“程序”的含义。在这种模型中，程序有两个方面的含义。

首先，程序必须是可存储的。在冯·诺伊曼模型中，这些程序被存储在计算机的内存中，内存中不仅仅存储数据，还要存储程序（参见图1-3）。

其次，该模型还要求程序必须是有序的指令集。每一条指令操作一个或者多个数据项。因此，一条指令可以改变它前面指令的作用。例如，示例程序1-1演示了输入a、b两个整数，将它们相加后显示出结果的程序。这段程序包含5条指令代码。

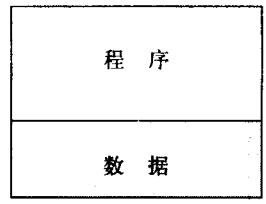


图1-3　内存中的程序和数据

示例程序1-1　由多条指令组成的程序

```
write('请输a,b两个整数: ');
readln(a);
readln(b);
c:=a+b;
writeln('a+b=' +IntToStr(c));
```

也许有人会问，为什么程序必须由不同的指令集组成？答案是重用性。如今，计算机要完成成千上万的任务，如果每一项任务的程序都相对独立而且和其他的程序之间没有任何的公用段，那么程序设计将变成一件很困难的事情。冯·诺伊曼模型通过仔细地定义计算机可以使用的不同指令集，从而使程序设计变得简单。一个程序员通过组合这些不同的指令来创建任意数量的程序。每个程序可以是不同指令的不同组合。

前面的要求使得程序设计变得可能，但也带来了另外一些计算机使用方面的问题。程序员不仅要了解每条指令所完成的任务，还要知道怎样将这些指令结合起来完成一些特定的任务。遇到不同的问题时，程序员首先应该以循序渐进的方式来解决问题，接着尽量找到合适的指令（指令序列）来解决问题。这种按步骤解决问题的方法就是所谓的算法。算法在计算机科学中起到了重要的作用，我们会在后面详细讨论算法的知识。

在计算机时代的早期，并没有计算机语言。程序员依靠写指令的方式（使用位模式，即直接写二进制代码指令）来解决问题。但是随着程序越来越大，采用这种模式来编写很长的程序变得单调乏味。计算机科学家们研究出利用符号来代表二进制格式指令。就像人们在日常生活中用符号（单词）来代替一些常用的指令一样。当然人们在日常生活中所用的一些符号和计算机中所用的符号并不相同。于是，计算机语言的概念应运而生。自然语言（例如英语）是一门丰富的语言，并有许多正确组合单词的规则；而计算机语言只有有限的符号和单词。后面我们还会介绍一些计算机语言的知识。

1.1.2　算法与数据结构

1. 计算机程序

程序是程序设计中最基本的概念，也是软件中最基本的概念。程序是对计算任务的处理对象和处理

规则的描述。所谓计算任务是指所有通过计算来解决实际问题的任务。处理对象是数据，如数字、文字和图像等。处理规则一般指处理动作和步骤。在低级语言中，程序是一组指令和相关的数据。在高级语言中，程序一般是一组说明和语句，它包括了算法和数据结构。

我们知道，利用计算机解决问题需要使用程序对问题的求解进行描述。这种解决问题的过程类似于人脑的解题过程，即利用一些规则或方法去处理特定的对象，从而解决问题。

通过程序，计算机可以按照人所规定的算法对数据进行处理。首先，人类凭借自然语言进行思维，而计算机使用计算机语言进行“思维”，控制计算机解题过程的算法必须以计算机能够“读得懂”的形式表示出来，也就是需要用计算机语言将算法描述出来，这种以计算机语言描述的算法就是程序。其次，算法只是描述人类思维时对数据的处理过程，人类思维时所用到的数据及其操作，将根据思维者的教育背景以人们无法直接看到的某种方式自然而然地存储在思维者的大脑中，因此人在解决一个具体问题时往往不会过多地考虑所涉及的数据。然而计算机解决问题的方式与此不同，除去一些基本的操作可以由计算机系统提供外，即便是看起来很简单的操作也需要进行专门定义和实现，而且那些“书写”在人脑中，常常被使用的数据，在使用计算机解决问题时将变得不再简单。如何将它们放置在计算机中是通过计算机使用算法解决问题时不可回避的问题。因此，使用计算机解决问题时，除了需要使用计算机语言描述算法，还必将涉及数据结构。从这个意义上讲，程序是建立在数据结构基础上使用计算机语言描述的算法，因此简单地讲，程序也可以表示成：算法＋数据结构。

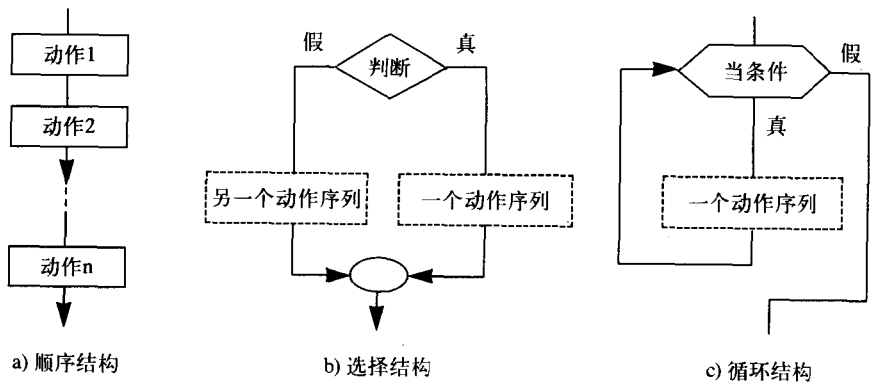
2. 算法

算法是一种逐步解决问题或完成任务的方法。算法完全独立于计算机系统。而且，算法接收一组输入数据，同时产生一组输出数据。

算法的定义如下：算法是一组有明确步骤的有序集合，它产生结果并在有限的时间内终结。因此我们应该从以下几个方面理解算法：

- **有序集合** 算法必须是一组定义完好且排列有序的指令集合。
- **明确步骤** 算法的每一步都必须有清晰明确的定义。例如，某一步是将两数相加，那么必须定义相加的两个数和加法符号，不能将一个符号在某处用作加法符号，而在其他地方用作乘法符号。
- **产生结果** 一个算法必须产生一个结果，否则该算法没有任何意义。结果集可以是被调用的算法所返回的数据或其他效果（例如，打印）。
- **在有限的时间内终结** 一个算法必须能够终结。如果不能终结（例如，无限循环），则说明不是算法。显然，任何可解问题的解法都可表示为一个可终结的算法。

计算机专家们为算法定义了三种结构。实际上，算法必定是由顺序、选择和循环（参见图1-4）这三种基本结构组成，其他结构都是不必要的。仅仅使用这三种结构就可以使算法容易理解、调试或修改。



a) 顺序结构　b) 选择结构　c) 循环结构

图1-4 算法的三种基本结构

- **顺序结构**　一个算法，甚至整个程序，都是一个顺序的指令集。它可以是一个简单指令或是其他两种结构之一。
- **选择结构**　有些问题只用顺序结构是不能够解决的。有时候需要检测一些条件是否满足。假如测试的结果为真，即条件满足，则可以继续顺序执行后面的指令；假如结果为假，即条件不满足，程序将转到另外一个顺序结构的指令处继续执行。这就是所谓的选择结构。
- **循环结构**　在有些问题中，需要重复相同的一系列顺序指令，此时就可以用循环结构来解决这个问题。例如，从指定的数据集中找到最大值的算法就使用了这种结构。

3. 数据结构

算法直观上表现为对各种数据的操作，那么什么是数据？数据又是如何组织的呢？

由于算法作用的对象是数据，而数据的定义又必须是计算机所能够识别、存储和处理的符号集合，并最终通过计算机加以实现和运行，所以数据对象需要通过一定的数据结构来组织。数据的基本单位是数据元素，数据元素可能具有底层结构，即每个数据元素由一个或多个数据项组成，数据项（又称为字段、域）是具有独立含义的最小单位数据，虽然一些数据元素具有底层结构，但是在使用它时总是将它看作一个整体。在算法中更常见的是处理多个具有相同性质的数据元素，因此又常把由一个或多个性质相同的数据元素组成的集合称为数据对象，数据对象是一个具有底层结构的实体，常常被作为一个整体加以引用。比如，存储在内存中的通讯录可称为数据对象，该数据对象由每个同学的通讯地址这个数据元素组成，而每个同学的通讯地址又由姓名、电话、住址、邮政编码等数据项组成。

因此，一个数据对象中的各数据元素的存在不是孤立的，相互之间存在着某些联系，通常把这些联系统称为数据结构。具体地说，数据结构由数据元素之间的逻辑结构、数据的存储结构以及在这些数据元素上定义的操作组成。

(1) 数据的逻辑结构

数据的逻辑结构抽象地反映出数据元素之间的逻辑关系。逻辑结构与数据在计算机中的存储方式无关，它所体现出的数据元素之间的关系完全是抽象的。例如，数字1，2，3，虽然它们可能书写在纸面上的不同位置，甚至数字3出现在数字1之前，但这并不能改变1小于3这种数值之间的关系。也就是说，它们之间的数值关系与写在纸面上的不同位置或存储在计算机内不同存储单元的顺序无关。

数据可以根据其是否具有底层结构划分成初等类型（也称基本类型）和构造类型两类，而常见的初等类型有5种:

- **整数类型**　计算机所定义的其值属于一定范围的整数。
- **实数类型**　又称浮点数类型，计算机所定义的其值属于一定范围的小数。
- **逻辑类型**　取值为真和假，通常用非0整数和0表示，或表示为true和false。
- **字符类型**　取值为计算机所采用的字符集的元素。
- **指针类型**　取值为内存中某存储单元地址，该单元存有某种类型的数据。

构造类型由初等类型或构造类型通过某种方式组合而成，不同的组合方式得到的构造类型不同，例如上文所述的通讯录等。

(2) 数据的存储结构

算法中出现的数据最终将被存储于计算机中，就像在纸面上书写数据要考虑在什么位置书写、一个数据占几格一样，在计算机中存储数据也要考虑将数据存储在内存中的哪个位置，占用多大的存储空间。数据的存储结构提供了数据的逻辑结构在计算机存储器中的映像，根据数据的存储结构将逻辑上相联系的数据元素存储在相应的存储单元中。也就是说，数据的存储位置和读写方式体现了数据的逻辑结构。常见的存储映像方式如下: