高等院校计算机教材系列

Java 程序设计大学教程

Java Programming for the College Students

刘艺 等编著

机械工业出版社
China Machine Press

高等院校计算机教材系列

Java 程序设计大学教程

Java Programming for The College Students

刘艺 等编著

机械工业出版社
China Machine Press

本书以Java语言为载体，通过讨论Java程序设计的一般过程和方法，重点讲述程序设计基础、面向对象程序设计、算法与数据结构、GUI程序设计和Web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。本书同时详细分析了Java作为通用程序设计语言的本质特征和语法规则，并以大量Java程序实例演示说明有关应用程序的设计过程，介绍主流的程序设计思想方法，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。全书采用案例教学法，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。

本书是在多年教学基础上编写的，不但结合国内计算机教学改革的最新成果，还参照美国ACM和IEEE/CS最新开发的课程体系规范《Computing Curricula 2004》。本书适合作为计算机程序设计课程或Java程序设计的基础教材，是高等院校计算机专业本科教学的首选用书，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，本书也是一个极好的参考。

图书在版编目（CIP）数据

Java程序设计大学教程 / 刘艺等编著. －北京：机械工业出版社，2006.2
（高等院校计算机教材系列）
ISBN 7-111-18279-0

Ⅰ.J… Ⅱ.刘… Ⅲ.JAVA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（2005）第160082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：武恩玉
北京慧美印刷有限公司印刷 · 新华书店北京发行所发行
2006年2月第1版第1次印刷
787mm × 1092mm 1/16 · 20.25印张
印数：0 001-5 000册
定价：29.00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Java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着一个重要的角色，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可以既不具有程序设计经验，也没有面向对象技术的概念和Web程序设计知识，甚至没有太多的计算机知识。即使是一个对计算机一无所知的人，也能通过本书而获取所有有关的基本知识，了解和掌握程序设计。如果读者是一位很有经验的程序员，已在其他程序设计语言中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知识给实际编程带来的便利和优势。

为什么会这样?一方面原因是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，还把如何编写规范

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

程序设计课程主要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Java程序。还有误人子弟的教材，不用面向对象的思想、方法去讲Java，而是用变量、过程、函数等老概念来硬套Java语法，使学生不会利用面向对象语言的优势来解决问题。这些都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Java的捷径应该是抓住最核心的思想方法：面向对象方法。为实现这个目的，我们把面向对象分析和设计作为重点，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，并结合案例教学，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Java作为学习程序设计的语言，是因为Java不但继承了C语言简洁优美的风格，而且还具有面向对象语言的真正优势。更可喜的是Java还在继续发展，不断吸取现代编程语言的精华。这一切使得Java具备现代通用程序设计语言的主流特征，特别适合跨平台的编程使用。因此学习Java语言，掌握Java程序设计方法是本课程的另一个重要任务。

本书虽然以Java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，而不是对Java语言及其类库的全面介绍。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。该课程是理工科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。全书由程序设计基础、面向对象程序设计、算法与数据结构、Java应用程序与applet程序设计、程序设计高级话题等5个部分组成本书的核心知识，并涉及计算机基础、数据和控制、程序设计理论、软件工程知识等四大知识领域，汇集18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，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的迭代，循序渐进，不断深入的。

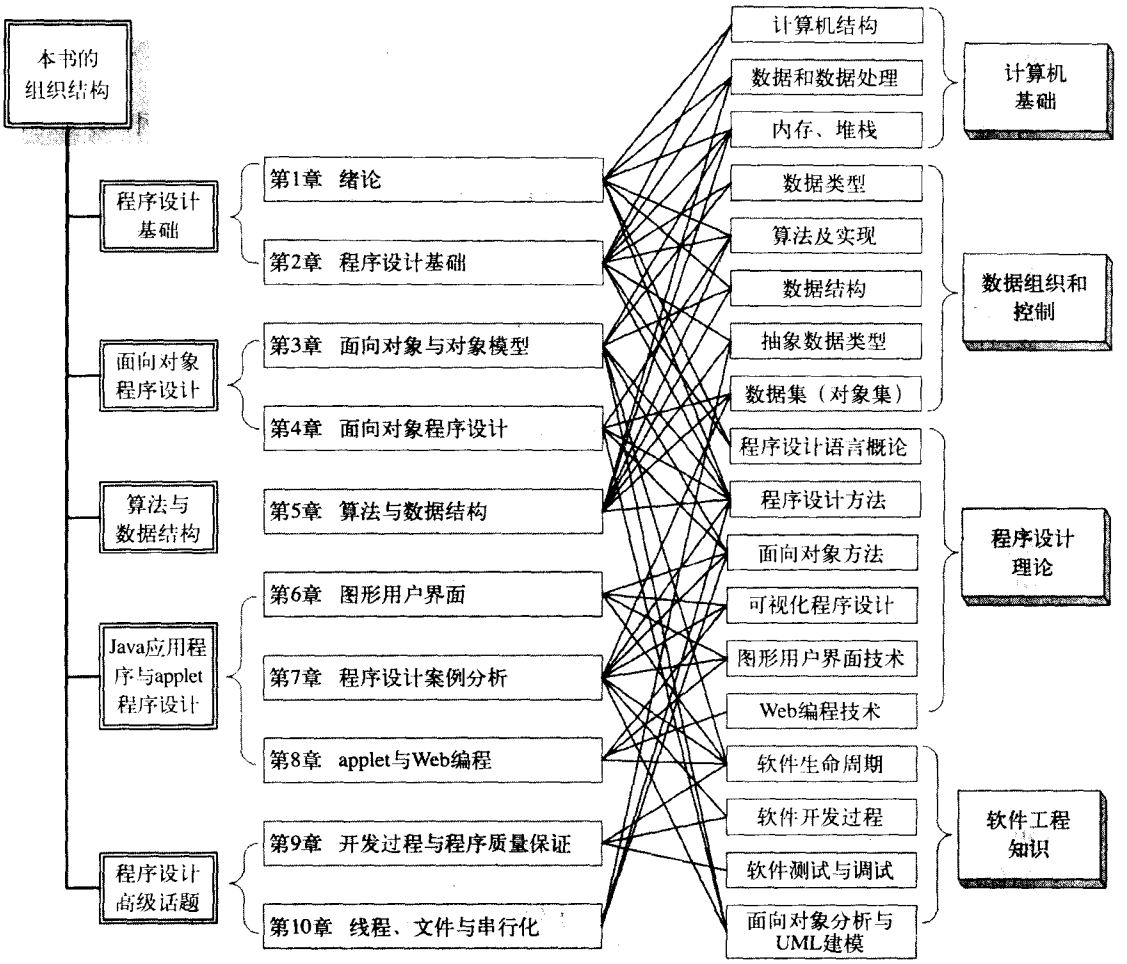


图1 本书组织结构与知识点

本书分为10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，初步认识Java。重点帮助读者搞清计算机程序、程序设计、程序设计语言等基本概念。同时介绍Java程序的编写、编译和运行，以及相关的环境设置和工具使用。
- **第2章 程序设计基础** 这一章首先通过一个简单的Java程序来了解程序的组成结构、语言要素和编写规范，建立程序的基本概念，然后以数据和运算作为程序设计的基础，通过讲解数据和数据类型、变量和常量、表达式和运算符以及流程控制，开始Java程序设计语言的探索之旅。
- **第3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Java对象模型中的核心部分：类及类的成员。使读者学会如何创建和使用Java对象。
- **第4章 面向对象程序设计** 本章将站在面向对象程序设计原则和方法的高度，围绕抽象性、继承性、多态性和封装性4个特点讲解面向对象程序设计的基本方法。重点讨论继承、多态和接口的编程。
- **第5章 算法与数据结构** 介绍算法的概念及常用算法，并通过数组、链表、栈、队列等数据结构以及Java对象容器，讨论算法的应用及算法的Java程序实现。

- **第6章 图形用户界面** 讲解Java图形界面应用程序的一般设计方法，包括如何创建窗体、设计界面、管理布局、绘制图形、使用组件、事件编程等。通过这一章的学习可以掌握图形用户界面应用程序的设计方法和编程技巧。
- **第7章 程序设计案例分析** 通过剖析和研究一个典型的Java应用程序设计案例，读者不仅可以对窗体、菜单、组件、事件、布局等的设计有一个感性的综合了解，还可以进一步理解Java应用程序的基本架构和完整设计过程，掌握使用开发工具（NetBeans IDE）完成程序设计项目的一般过程和方法，积累实际编程经验。同时还可学会利用NetBeans IDE的可视化设计功能提高程序设计效率。
- **第8章 applet与Web编程** 本章详细讲述applet的原理、特性、安全机制以及编程方法，并讨论applet在Web编程中的应用。同时还介绍了Web编程的一些有用知识。
- **第9章 开发过程与程序质量保证** 介绍软件的开发过程及过程的实施管理，从程序质量保证的高度讨论了程序的调试与测试，重点讲述了Java程序的调试方法、程序中的异常处理以及单元测试方法。
- **第10章 线程、文件与串行化** 本章将讨论有关线程及输入/输出的一些高级话题。通过本章的学习，读者可以了解多任务、进程和线程、线程模型、流、文件、输入/输出、对象串行化等诸多概念和编程方法，为Java的高级应用打下基础。

尽管本书包含了以上章节内容，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实际技能及对所讨论问题的熟悉程度。教学课时建议安排在50~80课时之间。

本书的读者对象

- 程序设计入门学生，选用本书作为程序设计基础教材，希望掌握一门优秀、实用、主流的程序设计语言，为深入学习其他计算机课程奠定基础。显然Java作为一种跨平台的面向对象编程语言能胜任教学的需要，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Java编程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Java编程经验，但没有系统学习过相关专业知识。希望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，已在其他编程语言及软件环境中掌握了一定的开发技能，但没有使用过Java语言或对Java语言一知半解，希望在本书中系统学习Java程序设计，发现Java与其所熟悉语言的不同点，并由此掌握Java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，概念与模型的结合可以帮助读者更好地掌握所学内容。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。为此，我们在全书中贯穿了软件工程的思想，使用了诸如UML建模、单元测试等与程序设计相关的软件工程实用方法。

代码编写和工具使用能力并重

初学Java程序设计的人，大多数会被Java强大的功能所吸引，同时也会被Java的代码编写难度所吓倒。传统的教材让学生使用记事本那样的简单文本编辑器编写Java程序，并在控制台中通过命令行编译和调试程序，无形之中增加了初学者学习的难度，这种方法在本教材中并不提倡。选用优秀的Java集成开发环境（IDE）作为教学平台，一方面可以帮助初学者发现代码错误（包括难以察觉的拼写错误），方便程序的调试和编译；另一方面可以让学生了解软件的实际开发环境，学会利用工具来提高学习和编程的效率。

鉴于快速应用开发（RAD）的思想正在改变程序设计的方法，而高效率的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Java程序是通过基于字符的编辑器键入一条条语句的方法，现在某些优秀的Java IDE也能像Delphi、VB那样支持我们交互式地在窗体上单击和拖放（click-and-drop）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Java 图形界面应用程序，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。Java IDE工具的出现和不断发展改变了这种模式。本教材中，我们采用SUN公司的开源IDE工具NetBeans，在强调编程能力的同时，我们鼓励学生使用IDE工具来提高程序设计的效率和质量。

鼓励学生使用IDE工具并不是削弱他们编写代码的能力，更不是宣扬那种无须写任何程序就可以用控件组装应用程序的神话，而是为了让学生把精力用于学习程序设计上，而不是浪费在繁琐的配置、设置、调试和重复输入字符上。其实这也是IDE工具进化的目标之一，即让程序员面对真正的程序设计任务，而不是浪费在窗口的几个GUI组件上或繁琐的手工调试上，程序员的创造力应该体现在结构优良，性能出众，稳定可靠，易于扩展的程序设计上。

真正的程序设计需要很多知识、技能和创造力。基于IDE工具的可视化程序设计只是手段，不是重点。即使是在GUI界面的设计中，我们也同时采用了可视化程序设计和非可视化程序设计两种实践，以便读者学习、比较。因为本书的重点仍然是学习程序设计语言的本质，培养代码的编写能力。这与强调代码编写和工具使用能力并重并不矛盾。

图文并茂

阅读本书后将会发现本书图文并茂。全书有100多幅精心设计的图片，这些图片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范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JDK 1.5中编译运行通过，建议读者在学习时选择Java 2标准版JDK 1.5的版本。

习题

每一章的结尾都包括本章习题。习题包括了三部分内容：复习题、测试题和练习题。

- 复习题：测试本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。

本教材配有专用的习题解答及课程设计教辅书籍《Java程序设计大学教程习题解答与课程设计》。

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（the 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。

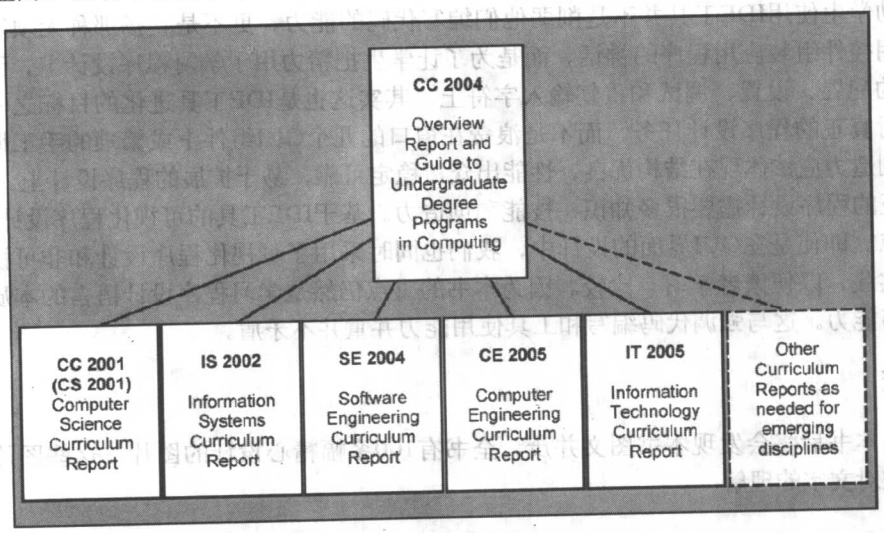


图2　CC 2004结构图

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，并通过多年来对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

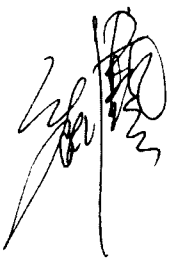
本书是在作者多年科研和教学基础上编写的，主要参考了作者已发表的文章和著作以及教学中积累的资料。书中还参阅了其他中外文教材、资料，由于无法在此一一列举，现谨对这些教材和资料的作者表示衷心的感谢。

参与本教材编写工作的人员还有昆明理工大学的刘迎春，海军工程大学的王永斌、周安栋、

段立、罗兵、李启元、杜军、吴苗、曹旭峰，南京航空航天大学无人机研究所的吴英，太原师范学院计算机中心的刘星，以及杨德刚、刘藩、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

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。帮助读者搞清计算机程序、程序设计和程序设计语言等基本概念。

Java作为我们要学习的程序设计语言，是本章要介绍的重点。我们会沿着Java的发展历史，探索这门应用广泛的计算机语言，并讨论Java是什么，又不是什么。

本章我们还要介绍如何下载JDK搭建Java平台，如何编写、编译和运行Java程序，如何使用Java程序的开发工具。总之，通过本章的学习，我们将为开始Java程序设计的探索之旅做好最充分的准备。

1.1 什么是程序设计

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的编排和优化。程序设计有着比计算机更长的历史，只不过计算机的出现使得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了。18世纪的音乐盒和19世纪末与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，法国发明家约瑟夫-玛丽·雅卡尔发明了由穿孔卡片控制的编织机，随着这一发明，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，编织机的各个部分必须进行机械定位。为了使这个过程自动化，雅卡尔使用一张纸质卡片代表编织机的一个定位，用卡片上的孔来指示该执行编织机的哪个操作。整条花毯的编织可被编码到一叠这样的卡片上，同样的一叠卡片每次使用都会编出相同的花毯图案。在这种可编程的编织机使用中，有的复杂程序需要由24 000多张卡片组成。

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，但从未完全制造完成。这台叫做分析机的机器，使用和雅卡尔的编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。这种机器几乎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。分析机的制造从未完成，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，她被认为是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年首先提出来的。冯·诺伊曼的想法是使用计算机的存储器既存储数据又存储程序。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，而且每次用户输给计算机的选项和数据不同，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。

- *存储器* 存储器是用来存储的区域，计算机在处理过程中存储器用来存储数据和程序，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，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* 输入子系统负责从计算机外部接受输入数据和程序；输出子系统负责将计算机的处理结果输出到计算机外部。输入/输出子系统的定义相当广泛，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。当一个磁盘用于存储处理后的输出结果，我们一般就可以认为它是输出设备，如果你是从该磁盘上读取数据，该磁盘就被认为是输入设备。

如果不关心计算机的内部物理结构，我们可以简单地认为计算机是一个“黑盒”。但是，仍然需要通过定义计算机所完成的工作来区别它和其他黑盒之间的差异。这里我们提供两种常见的计算机模型。

(1) 数据处理器

可以认为计算机是一个数据处理器。依照这种定义，计算机就可以认为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狭窄。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）。

图1-1 数据处理模型

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设计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。尽管如此，计算机作为一个当今使用的术语，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我们对计算机定义的模型来反映当今计算机的现实。

(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的模型如图1-2所示。图中添加了一个额外的元素——程序到计算机内部。程序是用来告诉计算

图1-2 可编程数据处理器模型

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，应用程序则是计算机语言所编写的一系列指令的集合。

在这个新模型中，输出数据依赖两方面因素的结合作用：输入数据和程序。对于相同的数据输入，如果改变程序，则可能产生不同的输出结果。类似地，对于同样的程序，如果改变输入内容，其输出结果也将不同。最后，如果输入数据和程序保持不变，输出结果也不变。

冯·诺伊曼模型的主要特征在于其存储程序的概念。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打开闭合和配线的改变。编程是在数据实际开始处理之前由操作员和工程师完成的一项工作。

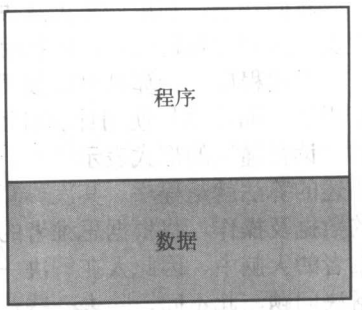


图1-3　内存中的程序和数据

冯·诺伊曼模型改变了“程序”的含义。在这种模型中，程序有了两个方面的含义。

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

其次，模型中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。这段程序包含了5条指令代码。

示例程序1-1　由多条指令组成的程序

```
1:   write('请输a,b两个整数: ');
2:   readln(a);
3:   readln(b);
4:   c=a+b;
5:   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，程序设计将会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，从而使得程序设计变得相对简单。程序员可以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使用计算机方面的因素。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。对于一些不同的问题，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。

在计算机时代的开端，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。但是随着程序的逐渐增大，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并不相同于计算机中所用的符号。这样计算机语言的概念诞生了。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；相对而言，计算机语言只有比较有限的符号和单词。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将是通过计算机使用算法解决问题所不可回避的问题。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。更特别的是，算法接收一组输入数据，同时产生一组输出数据。

算法是一组明确步骤的有序集合，它产生结果并在有限的时间内终结。因此我们应该从这几个方面理解算法。

- 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白的定义。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，不能用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，否则该算法也就没有意义。结果集可以是被调用的算法返回的数据或其他效果（如打印）。
- 有限的时间内终结　算法必须能够终结。如果不能（例如，无限循环），说明不是算法。显然，任何可解问题的解法形式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。实际上已经证实，算法必定是由顺序、选择和循环（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。

- 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，则可以继续顺序往下执行指令；假如结果为假，即条件不满足，程序将从另外一个顺序结构的指令继续执行。这就是所谓的选择结构。
- 循环结构　在有些问题中，相同的一系列顺序指令需要重复，那么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是这种结构的例子。