高等学校计算机应用规划教材

C 语言程序设计(第 2 版)

刘韶涛　潘秀霞　应　晖　编著

清华大学出版社

北　京

内 容 简 介

本书是在第 1 版的基础上，作者根据近年来实际工作过程中积累的教学经验与学生在学习和使用 C 语言进行程序设计过程中的心得体会、遇到的各种问题及各种反馈意见，进行了总结讨论和分析提炼，修改并进一步完善了第 1 版的基本内容，增加了扩展 C 程序设计的相关新章节。本书内容分为三个部分，分别是基础篇(第 1～9 章)、进阶篇(第 10～13 章)和提高应用篇(第 14 章)，可以满足不同学时、不同层次学生的要求。在提高应用篇中，将 C 语言应用到数据结构中几种典型的复杂数据类型的表示和实现中，希望能为学习 C 语言程序设计的读者进一步理解和掌握 C 程序设计的方法提供引导、思考和启发。

本书力求对 C 语言程序设计中涉及的基本概念、基本理论、典型应用和语法规则等的表述更为规范、科学和准确，文字叙述更加精炼通顺、实验数据更为准确。另外，本书还为全部习题和案例程序提供了完整的注释、运行结果分析和解题说明等。

在本书中，不仅仅局限于对 C 语言程序设计知识的描述，还阐述了与 C 语言程序设计相关的其他知识，特别介绍了 C 语言在其他交叉学科和相关领域中的新应用，让读者对 C 语言程序设计的整个学科体系、不同的软件开发环境、工程实践背景等都有一个较清楚的了解和认识。

本书既可作为高等学校 C 语言程序设计课程的教材，也可作为 C 语言程序开发人员的参考书。

图书在版编目(CIP)数据

C 语言程序设计 / 刘韶涛，潘秀霞，应晖 编著. —2 版. —北京：清华大学出版社，2020.1
高等学校计算机应用规划教材
ISBN 978-7-302-54458-6

Ⅰ. ①C…　Ⅱ. ①刘… ②潘… ③应…　Ⅲ. ①C 语言－程序设计－高等学校－教材　Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2019)第 264485 号

责任编辑：王　定
封面设计：孔祥峰
版式设计：思创景点
责任校对：成凤进
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**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：三河市铭诚印务有限公司
经　　销：全国新华书店
开　　本：203mm×260mm　　**印　　张：**22.5　　**字　　数：**547 千字
版　　次：2015 年 2 月第 1 版　　2020 年 1 月第 2 版　　**印　　次：**2020 年 1 月第 1 次印刷
定　　价：68.00 元

产品编号：079605-01

前　言

目前，国内外 C 语言程序设计的相关教材较多，但是大多数教材着重于对 C 语言基本语法规则和基本概念的阐述，学生学完之后，并不能真正掌握和灵活使用 C 语言来解决一些实际应用问题。特别是有些教材，是为了让学生学完后，参加全国或者省计算机等级考试而编写的，其内容完全是为了应付等级考试的考试内容。再加上目前计算机等级考试中的 C 语言程序设计的考试内容、考试方法等还存在很多不完善的地方，考试内容和考试成绩并不能反映考生真正运用 C 语言进行程序设计来解决实际问题的能力和水平。因此，编写高质量的 C 程序设计教材和辅导学习丛书，培养学生思考问题、分析问题和解决问题的能力和水平，提高其计算机的应用能力水平，对学生来说是非常必要，也是非常重要的。

本书根据初学者的特点，由浅入深，循序渐进，旨在帮助学生掌握 C 语言程序设计的基本方法，理解领会 C 语言的特点和本质，提高学生运用 C 语言解决实际问题的综合能力。并增加了一些典型的应用案例和模拟试题分析等，对基本概念和规则的表述更为科学、文字更为精炼通顺、数据更为准确。案例程序都给出完整的注释、运行结果和分析说明，所有习题和上机实践题也都提供了参考答案或分析和解答提示等，以利于学生在解题时参考和对比。对考试模拟试卷的分析，力求更为详尽，重点突出，让学生了解解决问题的思路和方法，起到举一反三的作用，更方便学生进行练习、自我检查，尽量做到一题多解，着重对学生的分析及思考能力进行培养和训练。

我们根据近年来实际教学过程中，学生使用初版教材遇到的各种问题和反馈意见，组织教师和授课教师，总结讨论，分析提炼，经过细心筛选和整理，重新编著了《C 语言程序设计》和《C 语言程序设计学习指导与上机实践》，修改和进一步完善了第 1 版中的内容，增加了和学科发展及知识更新相关的新章节。全书内容分为三个部分，分别是基础篇(第 1～9 章)、进阶篇(第 10～13 章)和提高应用篇(第 14 章)，可以满足不同学时、不同层次学生学习的不同要求。在提高应用篇中，将 C 语言应用到数据结构中几种典型的复杂数据类型的表示和实现中，希望能为学习 C 语言程序设计的读者进一步理解和掌握 C 语言程序设计的方法提供引导、思考和启发。在 C 语言程序设计的开发平台上，不再描述和使用 Turbo C，而是使用 VC++和 Dev C++进行描述和说明。另外，在本套书中，我们推行“不求全面，但求实用”的理念。尽量让读者都能寻找到各种知识点的方向和途径，指引出什么问题应该从哪些地方寻找答案。不再局限于 C 语言程序设计知识的描述，还阐述了与程序设计相关的其他知识，特别介绍了 C 语言在其他交叉学科和相关领域中的新应用，让读者对 C 语言程序设计的整个学科体系、不同的软件开发环境、工程实践背景等都有一个较清楚的了解和认识。

对于本书的编写，我们所追求的目标是：

(1) 既能作为一本学习 C 语言程序设计的学习教材，又能作为一本 C 语言程序设计的实验指导教材，也可作为一本探讨 C 语言程序设计学习和实践的艺术书籍。

(2) 突出 C 语言程序设计的应用重点和难点，不拘于具体语法细节的学习指导，而更注重 C 语言程序

设计的应用实践环节的上机实训。紧密联系教学实践，在教材中力求反映出学生学习相关知识的各类疑难问题，从学生学习的角度，对相关知识加以阐述和提炼。

(3) 引导读者良好的程序设计风格和程序设计思路，让读者能理解解决问题的方法，以达到触类旁通的效果。应用举例讲究经典实用而且丰富有趣，注重前后章节例题的连贯、一致和逐步深入。

(4) 主要面向初、中级读者群，又能兼顾高级读者的一些需求。内容既适合大多数初学者，又能满足少数高级读者深入学习的需求。先讲解基本知识，再探讨深层次的若干问题，以引起高级读者的兴趣。尽量把教学实践中学生学习中的问题反映到教材编写中，并加以解决，所以不但适用于学生读者，对教师读者而言，也有一定的参考价值。

(5) 进一步完善第 1 版教材中的学习指导内容和上机实践题目的设计，突出重点，加强应用，力求表述更为科学、阐述更加准确。所有例题、习题和上机操作题，都经过调试、运行和分析，以便于学生进行自我测试、自我检查和自我提高。

(6) 增加大量的例题、实验上机题和考试模拟试题等，对各个知识点进行详尽的分析、研究和探讨，旨在帮助学生通过学习和练习，真正理解和掌握 C 语言程序设计的理论知识和实践能力。设置各种不同层次等级的题目，以适用于不同的读者对象。

(7) 增加 C 语言程序设计在其他工程实践项目中的应用，让学生进一步理解 C 语言程序设计在各个工程领域中的应用实践情况，激发他们运用 C 语言解决专业问题的兴趣，切实提高他们应用 C 语言程序设计解决实际工程问题的能力和水平。

本书第 1 版由刘韶涛、潘秀霞、应晖编著，第 2 版中的所有章节内容(包括新增加的内容、习题和参考答案等)都由刘韶涛进行全面的修订、补充和完善。计算机科学与技术学院的缑锦院长、田晖副院长、王靖副院长、范慧琳副教授、余坚副教授等对教材的编写给予了全程的指导和关心，并给出了很多建设性的意见和建议。华侨大学教务处也对教材的编写和立项等工作给予了大力的支持，在此一并表示衷心的感谢！

由于时间仓促，加上编者水平有限，书中难免存在不妥之处，恳请广大读者批评指正，我们的联系邮箱是 shaotaol@hqu.edu.cn。

本书提供配套课件、教学大纲、教案和习题参考答案，可扫描下方对应二维码获取。

作者寄语　　课件　　教学大纲　　教案　　习题参考答案

编　者

2019 年 9 月

目　录

第1章　程序设计基础

在学习C语言程序设计之前，读者有必要熟知一些有关程序设计的基础知识，正所谓“工欲善其事必先利其器”。这些知识主要包括构成计算机系统的各个部件及其相互关系、计算机的基本工作原理、程序与程序设计的基本概念、算法及其表示方法、数据结构的基本概念、计算机中数据的表示方法以及结构化程序设计的基本思想等。

基本内容：

- 计算机系统基础知识
- 程序与程序设计语言基础
- 算法的概念和表示方法
- 数据结构基础
- 计算机中数据的表示
- 结构化程序设计概述

重点与难点：

- 硬件与软件基础
- 程序及程序设计语言
- 算法的表示和数据结构概念
- 计算机中数据的表示
- 结构化程序设计方法

1.1　计算机系统概述

程序是让计算机实现某种特定功能而编制的指令序列。程序设计需要了解计算机系统的基本组成构件及其相互关系。计算机系统是一个复杂的系统，从系统的整体构成上来讲，计算机系统是由硬件系统和软件系统构成的，可以认为，硬件系统是计算机工作的物质基础，软件系统则是计算机工作的思想和灵魂。

1.1.1　硬件基础知识

一个完整的计算机系统是由计算机硬件系统和计算机软件系统两部分组成的。硬件是计算机的实体，又称为硬设备，是所有固定装置的总称。它是计算机实现其功能的物质基础，其基本配置可分为主机和外设，主机主要包括中央处理器和内存储器，外设包括输入和输出

设备。

自 1946 年第一台电子计算机 ENIAC(Electronic Numerical Integrator And Computer)在美国诞生，至今已有 70 多年的历史。虽然电子计算机在外形、性能和应用领域等方面发生了巨大的变化，但是至今，电子计算机仍然沿用美籍匈牙利数学家冯 • 诺依曼等人提出的“存储程序与程序控制”的基本思想，该思想概括起来有如下一些要点：

(1) 计算机硬件由五大基本部件组成，即运算器、控制器、存储器、输入和输出设备。

(2) 采用二进制形式表示计算机的指令和数据。

(3) 将程序(由一系列指令组成)和数据存放在存储器中，使计算机在工作时能够自动高速地从存储器中取出指令加以执行。

这些概念奠定了现在计算机的基本结构，并开创了程序设计的时代。半个多世纪以来，虽然计算机结构经历了重大的变化，性能也有了惊人的提高，但就其结构原理来说，至今占有主流地位的仍是以存储程序原理为基础的冯 • 诺依曼型计算机，如图 1.1 所示。

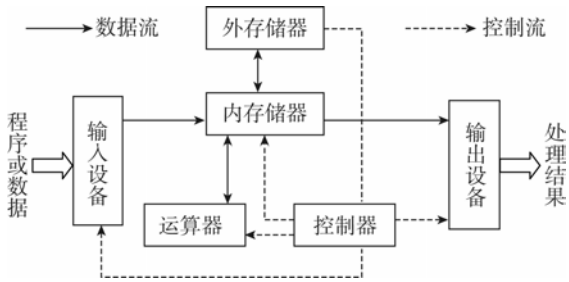


图 1.1　冯 • 诺依曼结构计算机

图 1.1 描述了计算机的工作流程，也就是应用计算机求解问题、执行程序的过程。首先将事先设计好的程序通过系统的输入设备，并在操作系统的统一控制下将程序或数据送入内存储器。控制器通过程序指令一步步执行来处理数据。执行的最终结果由输出设备输出。

1. 硬件的基本组成

计算机硬件系统是指所有构成计算机的物理实体，包括计算机系统中的一切电子、机械和光电设备等。计算机硬件系统的五大部件：控制器、运算器、主存储器、输入设备和输出设备以及其他辅助存储设备等。其中，控制器和运算器被称为中央处理器(CPU)。计算机硬件系统的基本组成如图 1.2 所示。

2. 中央处理器

在计算机系统中，中央处理器(CPU)是由计算机的两个重要部件——运算器和控制器组成的。中央处理器是计算机的控制和运算中心，它通过总线和其他设备进行联系。中央处理器的类型和品种非常多，各种中央处理器的性能差别也很大，有不同的内部结构、不同的指令系统。但由于都是基于冯 • 诺依曼结构，中央处理器的基本组成和工作原理也基本相似。

(1) 控制器。控制器是计算机系统的指挥中心，它把运算器、存储器和输入/输出设备等部件组成一个有机的整体，然后根据指令的要求指挥全机的工作。

图 1.2　计算机硬件系统基本组成

(2) 运算器。运算器是在控制器的控制下实现其功能的，运算器不仅可以完成数据信息的算术逻辑运算，还可以作为数据信息的传送通路。

基本的运算器组织包含如下几个部分：实现基本算术、逻辑运算功能的 ALU，提供操作数与暂存结果的寄存器组，有关的判别逻辑和控制电路等。将这些功能模块连接成一个整体时，运算器内的各功能模块之间的连接广泛采用总线结构，这个总线称为运算器的内部总线，ALU 和各寄存器都挂在上面。

运算器的核心部分是加法器。因为四则运算加、减、乘、除等算法都归结为加法与移位操作，所以加法器的设计是算术逻辑线路设计的关键。

(3) 寄存器。寄存器是 CPU 中的一个重要组成部分，它是 CPU 内部的临时存储单元。寄存器既可以用来存放数据和地址，也可以用来存放控制信息或 CPU 工作时的状态。在 CPU 中增加寄存器的数量，可以使 CPU 把执行程序时所需的数据尽可能地存放在寄存器中，从而减少访问内存的次数，提高其运行速度。但是寄存器的数目也不能太多，除了增加成本外，由于寄存器地址编码的增加也会增加指令的长度。CPU 中的寄存器通常分为存放数据的寄存器、存放地址的寄存器、存放状态信息的寄存器和其他寄存器等类型。

3. 指令系统

计算机系统包括硬件和软件两大组成部分。硬件是指构成计算机的中央处理机、主存储器、外围输入/输出设备等物理装置；软件则指软件厂家为方便用户使用计算机而提供的系统软件和用户用于完成自己的特定事务和信息处理任务而设计的用户程序软件。计算机能直接识别和运行的软件程序是由该计算机的指令代码组成的。

通常情况下，一条指令要由两部分内容组成，其格式为：

操作码	操作数地址

第一部分是指令的操作码。操作码用于指明本条指令的操作功能。例如，是算术加运算、减运算还是逻辑与、或运算功能，是否读、写外设操作功能，是否程序转移和子程

序调用或返回操作功能等，计算机需要为每一条指令配一个确定的操作码。

计算机的指令系统往往由几十条到几百条指令组成。一般包括如下指令：

(1) 算术与逻辑运算指令。

(2) 移位操作指令。

(3) 数据传送指令。

(4) 转移指令、子程序调用与返回指令。

(5) 特权指令。

(6) 其他指令。

第二部分是指令的操作数地址，用于给出被操作信息(指令或数据)的地址，包括参加运算的一个或多个操作数地址、运算结果的保存地址、程序的转移地址、被除数调用的子程序的入口地址等。

寻址方式解决的是如何在指令中表示一个操作数的地址，如何用这种表示得到操作数或怎样计算出操作数的地址。表示在指令中的操作数地址，通常被称为形式地址：用这种形式地址并结合某些规则可以计算出操作数在存储器中的存储单元地址，这一地址被称为物理(有效)地址。计算机中常用的寻址方式有如下几种：

(1) 立即数寻址。

(2) 直接寻址。

(3) 寄存器寻址、寄存器间接寻址。

(4) 变址选址。

(5) 相对寻址。

(6) 基地址寻址。

(7) 间接寻址。

(8) 堆栈寻址。

关于具体的寻址方式实现、主存和辅存、I/O 接口和 I/O 设备、控制器和运算器工作原理等计算机硬件知识，可参考《计算机组成原理》等相关书籍，这里不再赘述。

1.1.2　软件基础知识

人们用各种电路器件制造的计算机称为物理计算机或裸机。裸机使用的语言是二进制形式表示的机器语言。在机器语言的基础上，人们开始使用抽象层次更高、更接近人类自然语言的计算机语言，包括汇编语言和高级语言。汇编语言和高级语言需要进行翻译才能被计算机硬件识别。不同层次的用户使用不同层次的计算机语言与计算机进行交互，使计算机实现用户的要求。因此，可以把计算机看成是一个多层次的系统。裸机以上的用户所操作的计算机可以看成一台相应的虚拟计算机。

软件是指挥计算机运行的程序集，计算机软件系统是计算机运行时所需的各种程序、数据及其相关文档的总称。不管硬件系统体系结构如何，按照功能，计算机软件一般可以分为两类：系统软件和应用软件。

系统软件管理计算机资源，是硬件和用户间的接口，但一般不直接服务于用户的需求。用户对计算机使用的各种需求，通过计算机的应用软件来实现，即应用软件可以直接帮助用

户处理和解决某些具体的计算机应用问题。图 1.3 描述了计算机软件的一般分类。

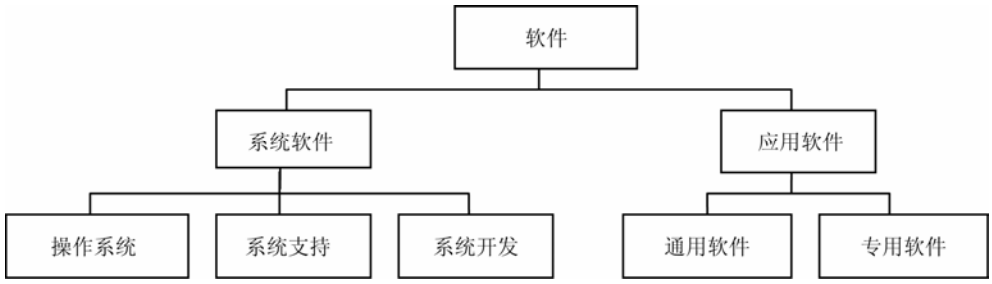


图 1.3　软件的类型

系统软件包括管理计算机硬件资源和执行信息处理任务时所需的程序。这些程序可分为：操作系统、系统支持和系统开发软件。操作系统提供了诸如用户界面、文件和数据库存取以及通信协议等功能。这种软件的主要目的是使系统以一种有效的方式运行，同时允许用户访问系统。系统支持(支撑)软件提供了系统工具和其他的运作服务。系统工具的实例包括排序程序和磁盘格式化程序等。运作服务则包括为保护系统和数据而给系统操作员和安全监控器提供系统性能统计数据的程序等。系统开发软件包括将程序翻译为可执行的机器语言的翻译器，保证程序免于错误的调试工具和计算机辅助软件工程(CASE)等。

应用软件分为两类：通用软件和专用软件。通用软件一般由软件开发商提供，可以用于多个应用。例如，字处理软件、数据库管理系统和计算机辅助设计系统等。它们被称为通用软件是因为它们能帮助用户解决多种计算机的普遍应用问题。专用软件仅被用于特定的目的和领域。例如设计人员使用的计算机辅助设计与制造(CAD/CAM)软件、会计师使用的财务会计辅助系统和建筑商使用的材料需求规划系统等。它们仅用于设计时指定的专业任务，而不能应用于其他一般性任务。

图 1.4 表述了系统软件和应用软件之间的关系。图中的圈表示界面。内部核心是硬件，用户位于外层。在使用系统时，一般用户使用的是应用软件，应用软件和位于系统软件层的操作系统交互，而系统软件则提供了和硬件直接交互的能力。同时，用户在需要时可通过底部开口直接和操作系统交互。

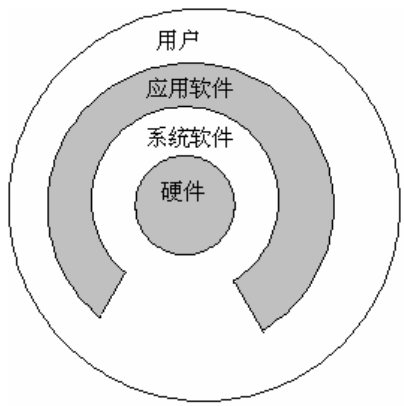


图 1.4　系统软件和应用软件间的关系

如果用户买不到能满足需要的软件，那么就需要定制开发他们需要的软件。本书讲授的 C 语言是当今众多软件开发工具中的一种重要语言工具。

1.2　程序与程序设计语言

在介绍计算机硬件和软件的基础知识之后，下面介绍作为软件重要基础的程序以及程序

设计的工具，即程序设计语言的相关知识。C 语言是一种面向过程的程序设计语言，那么程序到底是什么？程序设计语言又有哪些？它们之间存在什么样的关系？各种程序设计语言是怎么分类的，各自又具有什么样的特性呢？

1.2.1　程序的概念


程序概述

计算机程序(Computer Program)是一组计算机能识别和执行的指令，运行于电子计算机、满足人们某种需求的信息化工具。它以某些程序设计语言编写，运行于某种目标结构体系。打个比方，程序就如同以英语(程序设计语言)写作的文章，需要一个懂得英语(编译器)同时也会阅读这篇文章(结构体系)的人来阅读、理解、标记这篇文章。一般地，以英语文本为基础的计算机程序要经过编译、链接而成为人难以解读，但可轻易被计算机所解读的数字格式，然后进行运行。简单地说，程序是为了解决某一个具体问题而使用的一个指令序列(a set of instructions)。

问题的解决需要运行编写的计算机程序。为了使计算机程序得以运行，计算机需要加载代码，同时也要加载数据。从计算机的底层来说，这是由高级语言(例如C/C++，Java，C#，等等)代码转译成机器语言而被CPU所理解。

程序的运行是指为了得到某种结果而由计算机等具有信息处理能力的装置执行的代码化指令序列。

1.2.2　程序设计语言概述

人们用各种电路器件制造的计算机称为物理计算或裸机。裸机使用的语言是以二进制形式表示的机器语言。在机器语言的基础上，使用抽象层次更高、更接近于人类自然语言的计算机语言，包括汇编语言和高级语言。汇编语言和高级语言需要进行翻译才能被计算机硬件识别。不同层次的用户使用不同层次的计算机语言与计算机进行交互 ，使计算机实现用户的要求，因此，可以把计算机看成一个多层次的系统。裸机以上的用户所操作的计算机可以看成一台相应的虚拟计算机。

这些控制着计算机运行的程序，是由被称为“计算机程序员”的人员编写的。程序员在程序中指定了一系列的动作，计算机按照这一系列动作顺序执行，最终完成程序员所要解决的问题，获得结果。

长期以来，“编写程序”和“执行程序”是利用计算机解决问题的主要方法和手段。为计算机编写程序必须使用计算机语言。

多年来，程序设计语言已经从机器语言进化到接近自然语言的高级语言。图 1.5 是计算机语言进化的概括。

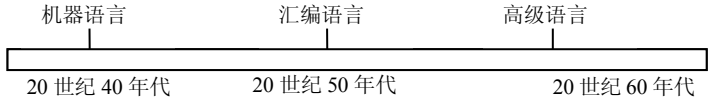


图 1.5　计算机语言的进化

1. 机器语言

机器语言是由 0、1 序列组成的指令码，它是计算机历史早期仅有的可以使用的语言。每种计算机都有自己的机器语言，特定的机器语言只能在特定的一类计算机上使用。因此，机器语言可移植性差。

机器语言是可以直接执行的语言，也就是说，计算机硬件能够理解的语言只有机器语言，因此执行速度快，效率高。但使用机器语言编写程序是很不方便的，它很难记忆，且要求使用者熟悉计算机的很多硬件细节。

典型的机器语言类似：1010 1111
0011 0111
0111 0110
……

2. 汇编语言

随着计算机硬件结构越来越复杂，指令系统也变得越来越庞大，对大多数程序员来说，用机器语言编写程序显然太慢，并且非常枯燥，一般工程技术人员难以掌握。为了取代计算机能够直接执行的数值串，程序员开始使用代表计算机基本操作的类英语符号或助记符来编写程序，如用 ADD 表示加法操作。这些符号(助记符)构成了汇编语言的基础。由于计算机不能理解这些助记符，需要将这些符号翻译成机器语言。

用汇编语言编写的典型的程序段如下：

汇编语言	机器语言
MOV　A，47	1010　1111
ADD　A，B	0011　0111
HALT	0111　0110
……	

汇编语言用程序员容易理解的方式组织程序，但是机器使用这些程序则较为困难，因此在机器运行它之前必须先进行翻译。这种方式是一个趋势的开始：编程语言变得越来越方便于程序员使用，而计算机用于翻译的时间却越来越长。

与机器语言程序相比，汇编语言源程序的阅读和理解都比较方便，但对一般人员来说，其描述问题方式与人类习惯还是相差甚远，而且通常要求编程人员对计算机硬件有深入的了解。

3. 高级语言

随着汇编语言的使用，大大提高了编程效率，计算机的用途也迅速扩大，但汇编语言仍然是面向机器的。即使完成最简单的任务仍然需要程序员关注程序运行的硬件。为提高程序员的效率，将其关注点转移到求解问题本身的期望上来，研究人员设计了一系列的高级语言。

用高级语言编写的指令类似于日常英语并且包含常用的数字符号，如“加”操作直接使用“+”。下面的程序是以 C 语言编写的两个整数的加法。

```
/*输入被加数 num1 和加数 num2，输出其和 sum*/
    #include<stdio.h>
    void main(void){
        int num1,num2, sum;
        printf("Enter num1 and num2:");
        scanf("%d%d", &num1, &num2);
        sum=num1+num2;
        printf("sum=%d\n",sum);
    }
```

高级语言使程序员从机器语言和汇编语言的细节中解放出来。当然，它同汇编语言一样不能直接被计算机硬件理解，必须被转换为机器语言。这个转换过程被称为编译(对有些高级语言，这个转换过程被称为“解释”)。把高级语言程序转换为机器语言的翻译程序被称为“编译器”。

高级语言种类繁多，常用的面向过程的语言有 BASIC、Pascal、FORTRAN 和 C 等。当前流行的面向对象程序设计语言有 C++、Java、C#和 Python 等。

1.3　算法及其表示

在着手编写程序解决问题之前，首先需要确定解决问题的策略，即具体的方法和步骤，这就涉及算法及其表示。算法是解决问题的关键，它是程序的思想和灵魂。

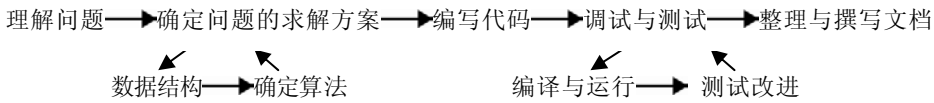
1.3.1　算法

算法和数据结构概述

程序是为使计算机完成一个预定的任务而设计的一系列语句或指令。程序设计是设计、书写及检查调试程序的过程。

为使计算机完成任务，解决问题。首先程序员要用计算机语言编写程序才能让计算机按着程序指令一步步地完成任务。计算机只是程序的执行者，并非程序的制造者。程序解决问题的方法是由程序员来处理的，程序员首先要自己找到解决问题的方法，并用程序设计语言正确地描述它，才可能让计算机去执行并完成任务。

程序员开发程序步骤如下：



在接到程序设计任务后，程序员首先要分析问题、理解(搞清楚)要计算机解决什么问题(需求分析)。然后确定问题求解方案，分析问题，从中提取操作的对象，并找出这些对象之间含有的关系，用数学语言进行描述，也就是确定问题的数据结构，再用算法描述对此问题的求解步骤，这是解决问题的关键。算法描述好并检查确认后，方可编写程序。最后对程序进行调试与测试改进，成功之后将相关文档进行整理以备后用。所以可以这样认为，程序=数据结

构+算法+程序设计方法+文档。

算法是解决某个问题的方法(策略)和具体步骤。因此，程序开发是一个多步骤的过程，学习和进行程序设计要按这个步骤一步一步地解决问题。掌握和学会问题求解方法，是学习高级语言的重点，也是最大难点。它将贯穿于学习和教学的整个过程，渗透到各个学习和教学环节中。

有的问题(数值型问题)可以用数学方程来描述，如预报人口增长情况的数学模型为微分方程、求解梁架结构中应力的数学模型为线性方程。但是还有更大量的非数值问题无法用数学方程等来描述，它的数学模型是用表、树和图之类的数据结构，如数据库文档的管理(表)、计算机与人对弈(树)、交通路径的选择(图)等。对于这些(复杂的非数值型)问题的解决，必然涉及描述问题(建模)的数据结构和解决问题的算法(基于数据结构)等知识。

为了介绍求解思路，我们先来看两个问题。

问题一：朋友要求你帮他布置一间屋子，你要怎样做呢？

(1) 要仔细了解你的朋友(客户)的需求。当确认自己已经完全理解后，再和客户讨论你的理解，进一步对问题进行确认无误的询问，如提出下面一些问题：

- 多大的屋子？屋子各部分的结构(关系)是怎样的？
- 屋子需要有哪些功能？
- 资金预算是多少？

……

(2) 一旦完全理解了问题，弄清了手上的问题，接着就需要给出求解方案。也就是给出解决问题的操作步骤，即算法。比如，可以实施以下操作，以回答上面的问题。

- 测量屋子，并画出其平面图。
- 屋子有厨房、卫生间、书房和卧室，弄清楚每部分的功能以及它们之间的关系等。
- 查询各种需要的物件的价格组合，以保证在预算的范围内。

……

将屋子分成厨房模块、卫生间模块、书房模块和卧室模块，分别进行布置，画出结构图和布置过程图等。

(3) 按照结构图和布置流程图一个模块一个模块、一个步骤一个步骤地完成房屋的布置。

(4) 检查、调整和完善整个布置。

问题二：求一元二次方程 $ax^2+bx+c=0$ 的根。

这是一个常见的数学问题。大家都很熟悉它的解题办法。

(1) 澄清三个常数 a、b、c 的值的关系。清楚什么时候有两个不等的实根，什么时候有相等的实根，什么时候没有实根。

(2) 给出求解方案。当 $d=b^2-4ac$ 大于 0 时，有两个不等的实根；当 $d=0$ 时，有两个相等的实根，否则有两个复数根，可以用算法的流程图表示。

(3) 将算法用某种高级语言编写成程序。在编写程序时，先从算法流程图的顶上方框开始，以从上而下的方式进行，按流程图自顶向下逐步求解，实现输出方程的根。

(4) 调试与测试程序。编完程序之后，必须对其进行测试。程序设计人员要了解程序内部的确切信息，必须确认每一条指令和每一种情况都被测试到。关于软件测试，可以进一步参考软件工程中软件测试部分的内容。

1.3.2　算法的特性

问题解决的过程是由一系列确定的、有限的步骤完成的，也就是算法。它是整个程序设计过程中的灵魂。

算法应具有下述特性：

(1) 有穷性。一个算法应当在执行有限步骤后结束，不应出现无终止的循环或永远执行不完的步骤。

(2) 确定性。算法中的每一步骤必须有确切的含义，不能有二义性(Ambiguity，即不能作两种或多种解释)，不应含混不清或模棱两可。

(3) 有 0 个或多个输入量或者初始值。

(4) 有 1 个或多个输出量。

(5) 可执行性。算法中的每一步是能够准确实现的。

1.3.3　算法的表示

可以用结构图、伪代码和流程图 3 种工具来描述算法。通常会使用结构图和伪代码或使用结构图和流程图。

结构图用于设计整个程序；而伪代码和流程图用于设计程序中的独立部分，这些部分在伪代码中被称为模块，在 C 语言中被称为函数。

1. 结构图

结构图，也可称为层次结构图，表示程序的功能流程。结构图说明了如何将程序分解为逻辑步骤，而每个步骤是一个独立的模块。结构图说明了所有部分(模块)之间的交互。

在编写程序之前，将操作步骤表示为结构图是非常重要的。这如同建筑师设计蓝图，建筑师在没有详细的计划之前是不会开始建造房屋的。

2. 伪代码

伪代码是英语、汉语和程序逻辑的混合语言，可实现细节上描述将要设计的程序的目的。我们需要足够详细的定义完成任务所需的步骤，以便之后可以顺利转换为计算机程序。例如将本节的问题二表示成伪代码，见算法 1-1。

算法 1-1　求一元二次方程根的伪代码算法表示

```
input a,b,c
disc=b²-4ac, d=sqrt(disc)
if disc≥0  begin
      if  disc=0  begin
                  x1,x2=-b/(2a)
            end
      else  begin
                x1=(-b+d)/(2a), x2=(-b-d)/(2a)
          end
print x1,x2
end
else  begin
    p=-b/(2a), q=-d/(2a), print p+q,"+",p-q,"i"
end
```

以上伪代码中的大多数语句是易于理解的，先输入三个参数(系数)，然后判断各种情况下的根，最后输出根。

3. 流程图

流程图是用一些图框来表示各种操作的程序设计工具。美国国家标准化协会ANSI(American National Standard Institute)规定了一些常用流程图符号，如图 1.6 所示，已为世界各国程序员普遍采用。

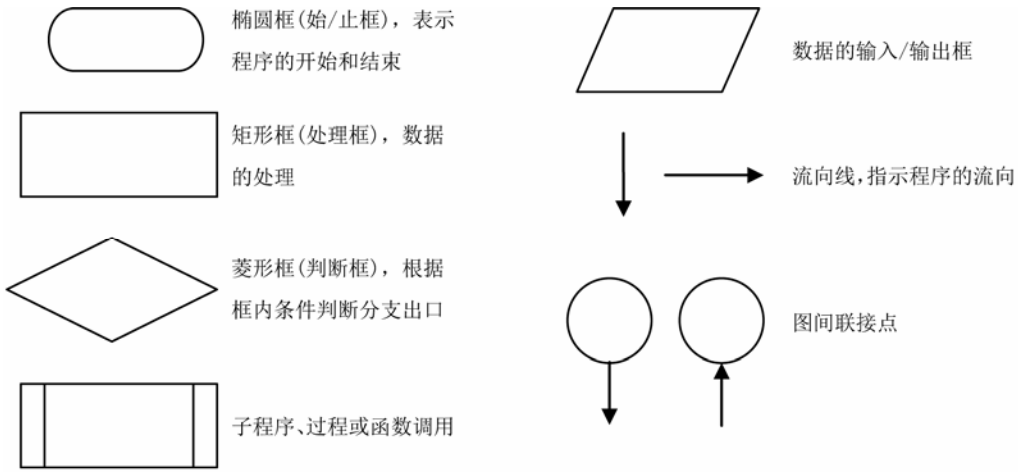


图 1.6　常用流程图符号

由于流程图是可视化的工具，对于初学者来说，使用流程图来学习程序设计比伪代码容易掌握。而专业程序员则更常用伪代码。问题二算法的流程图表示如图 1.7 所示。

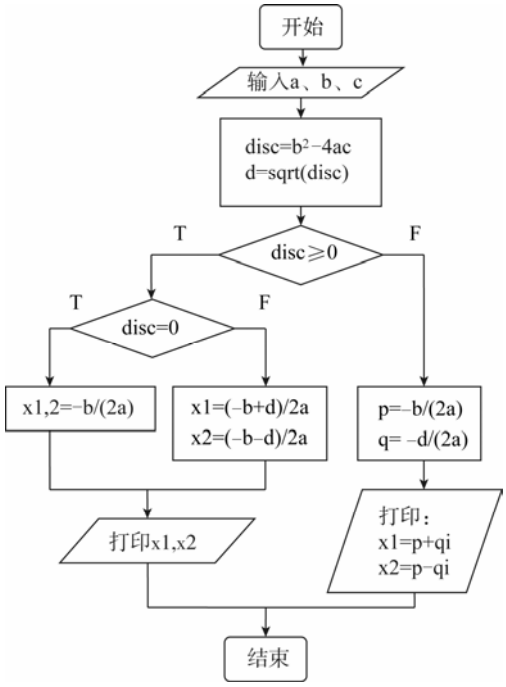


图 1.7　求一元二次方程根的算法流程图表示

还有一种表示算法的流程图，它将以上介绍的传统流程图的流程线去掉，算法的每一步用矩形框表示，并把它们按执行顺序连接起来对算法进行描述，这种描述的方法称为 N-S 图。N-S 图也被称为盒图或 CHAPIN 图，1973 年由美国学者纳斯(I. Nassi)和施内德曼(B. Shneiderman)提出，全部算法写在一个矩形框内，在框内还可以包含其他框，即由一些基本的框组成一个大的框。N-S 图包括顺序、选择和循环三种基本结构。

例如，将图 1.7 的算法表示成 N-S 图，如图 1.8 所示。

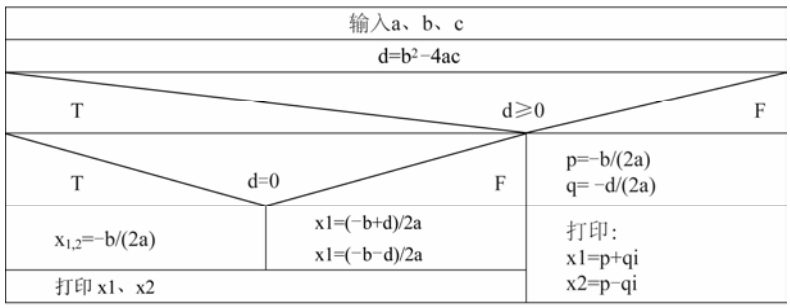
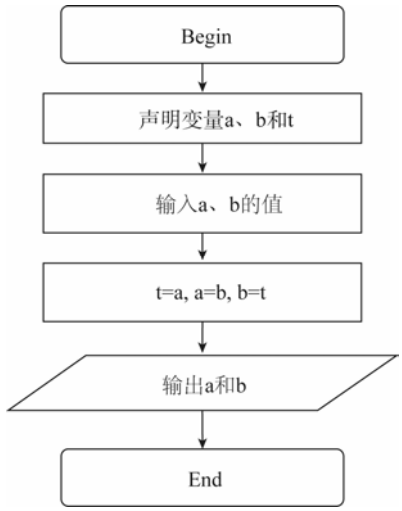


图 1.8　求一元二次方程根算法的 N-S 图表示

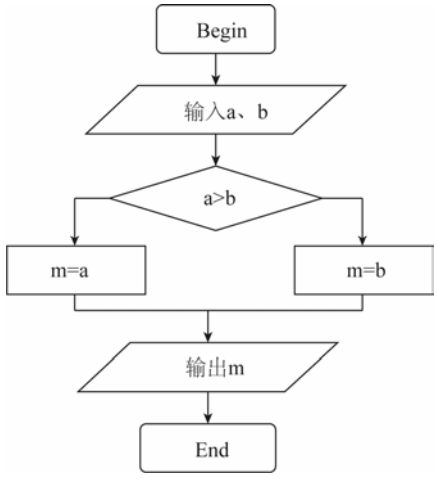
4. 算法表示示例

下面举几个简单算法表示的例子。

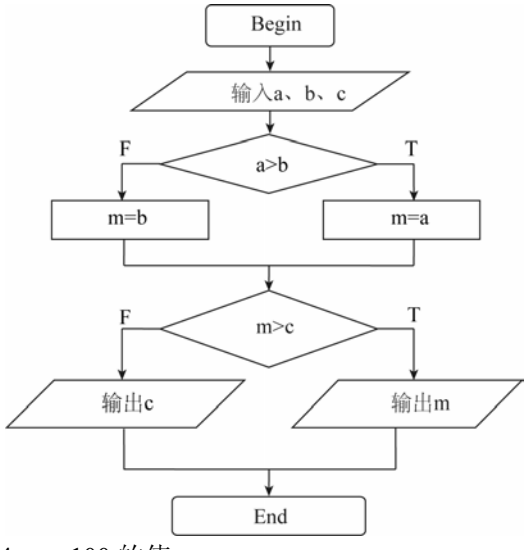
例 1-1　输入整数 a 和 b，将 a 与 b 的值互换并输出。



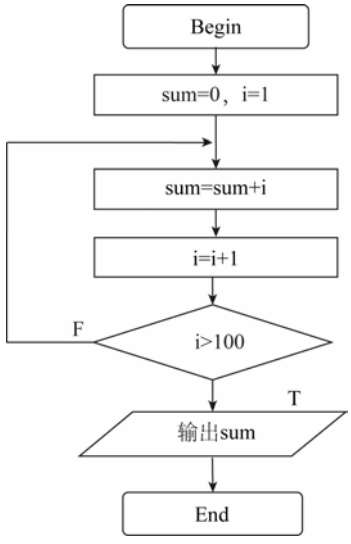
例 1-2　输入 a 与 b 两个值，若 a>b 则输出 a，否则输出 b。



例 1-3　输入 a、b、c 三个值，输出三个值中最大的一个。



例 1-4　求 1+2+3+4+…+100 的值。



1.4　数据结构概述

前面已经讲过，可以把计算机解决的问题分为两类：一类是数值问题，如求若干个数的最大值和最小值、求方程的根等；另一类是非数值问题，如最短路问题、机器下棋问题等。对于非数值问题的解决，往往比较复杂，关键在于对问题的描述(建模)和在此基础上的算法设计。即，首先要建立问题的数学模型，研究要处理的对象和它们之间的关系，以及如何在计算机内存中表示这些对象和它们的关系，然后设计相应的算法加以解决。

数据结构就是一门讨论“描述现实世界实体的数学模型(非数值计算)及其之上的操作在计算机中如何表示和实现”的学科。它与算法一样，都是进行复杂程序设计的基础。

1.4.1　与数据结构相关的基本概念

(1) 数据(Data)。数据是指所有能输入计算机中，且能被计算机处理的符号的总称，它是计算机程序加工的“原料”，如文字、字符、图形、图像、声音等。显然，数据结构中“数据”概念的内涵要比普通“数据”丰富得多，它不仅仅指数值。

(2) 数据元素(Data Element)。数据元素是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。如图书检索系统中的一本书的书目信息，下棋对弈状态树中的一个棋盘格局，煤气管道铺设图中的一个圆圈，等等。

(3) 数据项(Data Item)。数据项是数据元素的分量，数据项是数据不可分割的最小单位。

(4) 数据对象(Data Object)。数据对象是同类型数据元素的集合，如一个系的全体学生等。

1.4.2　数据结构的含义

目前，数据结构还没有被一致公认的定义。它具有三个层面的含义。

(1) 问题所涉及的数据对象，以及数据对象内部各个数据元素之间的特定关系——数据的逻辑结构(Logical Structure)。

(2) 全体数据元素以及数据元素之间的特定关系在计算机内部的表达——数据的存储结构(或称为物理结构，Physical Structure)。

(3) 为解决问题而对数据施加的一组操作——数据的运算(操作，Operations)集合。

逻辑结构是面向现实世界的，独立于计算机。存储结构是逻辑结构在计算机内存中的具体实现，它反映逻辑结构，是逻辑结构在内存中的镜像。存储结构是面向计算机的，相同的逻辑结构可以由不同的存储结构来表示。操作的实现依赖于所设计的存储结构。

从上面数据结构所涉及的三个层面可以看出，数据结构研究如何将现实世界中的对象及其关系，在计算机内存中进行描述和表示，以及基于这种存储结构上的具体操作实现方法(即算法)。

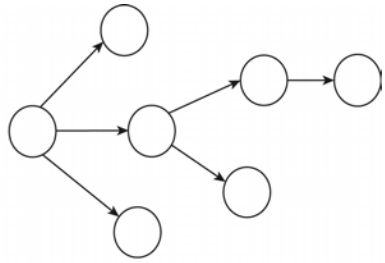
在进行复杂程序设计时，需要解决的首要问题是，如何在计算机世界中表达清楚客观世界中存在的对象及其关系(建模)，然后进一步分析、设计和实现在此模型上的各种操作实现，即算法的设计与实现。所以，数据结构涉及问题对象的逻辑结构、存储结构及其实现方法等，

在实现方法中还涉及实现方法效率的评估(算法的复杂度分析)等。合理的存储结构设计和高效的算法实现，是数据结构追求的目标。

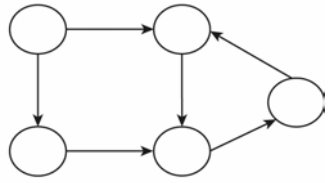
1.4.3　常用的逻辑结构

(1) 线性结构：数据元素之间存在一种简单的线性关系(1:1)，示例如下所示。

(2) 树形结构：数据元素之间存在一种层次的关系(1:n)，示例如下所示。



(3) 图形结构：数据元素之间存在一种多对多的图形关系(m:n)，示例如下所示。



1.4.4　常用的存储结构

数据的存储结构是指将问题所涉及的数据对象中的所有数据元素存入计算机，并且在计算机内部表达出数据元素之间存在的关系。常用的存储技术有顺序存储、链式存储、散列存储和索引存储等。

关于这几种存储结构技术的详细描述，可以参考数据结构等文献，本书在进阶篇和提高应用篇中也会作部分讨论和讲解，这里不再加以阐述。

1.4.5　数据的运算集合

数据的运算集合是指对数据进行加工和处理的一组算法。数据的运算，既面向问题(逻辑)，又面向计算机(物理)。操作集合的定义由问题决定；操作的实现与数据在计算机内的存储方式有关。

关于各种常用数据结构的表示和存储实现，请参考数据结构的书籍，这里不再详细讨论。本书的基础篇部分涉及的问题大多为简单的数值问题，这些问题的解决不需要数据结构的相关知识。此外，还会涉及非数性值问题(如数据的排序等)，可以直接采用 C 语言提供的构造数据类型(即简单的数据结构，如数组、结构体等)来解决。关于较为复杂的数据结构设计与实现，将在进阶篇和提高应用篇中进一步讲述。

1.5　计算机中数据的表示

编制程序是为了解决现实生活中的某些具体问题的。生活是丰富多彩，现实世界和生活中各种丰富的“数据”在计算机内部又是如何表示的呢？

1.5.1　数制及其转换

1. 数制的概念

在采用进位记数的数字系统中，如果用 r 个基本符号(例如 0，1，2，…，r−1)通过排列起来的符号串表示数值，则称其为基 r 数制，r 称为该数制的基。假定用 m+k 个自左向右排列的符号 $D_i(-k\leqslant i\leqslant m-1)$表示数值 N，即

$$N=D_{m-1}D_{m-2}\ldots D_1D_0\,D_{-1}D_{-2}\ldots D_{-k}$$

式中的 $D_i(-k\leqslant i\leqslant m-1)$为该数制采用的基本符号，可取值 0，1，2，…，r−1，小数点位置隐含在 D_0 与 D_{-1} 之间，则 $D_{m-1}D_{m-2}\ldots D_1D_0$ 为 N 的整数部分，$D_{-1}D_{-2}\ldots D_{-k}$ 为 N 的小数部分。

如果每个 D_i 的单位值都赋以固定的值 W_i，则称 W_i 为该位的权，此时的数制被称为有权的基 r 数制。此时 N 代表的实际值可表示为：

$$N=\sum_{i=-k}^{m-1}(D_i\times W_i)$$

如果该数制编码还符合“逢 r 进位”的规则，则每位的权(简称位权)可表示为：

$$W_i=r_i$$

式中的 r 是数制的基，i 为位序号。所以 N 代表的实际值也可以用下式表示：

$$N=\sum_{i=-k}^{m-1}(D_i\times r^i)$$

式中：r——数制的基；i——符号的列次序，即位序号；D_i——位序号为 i 的一位上的符号；r^i——第 i 位上的一个 1 所代表的值(位权)；$D_i\times r^i$ ——第 i 位上的符号所代表的实际值；N ——代表一个数值。

此时该数制称为 r 进位数制，简称 r 进制。下面是计算机中常用的几种进位数制。

二进制　　r=2，　　基本符号 0，1。

八进制　　r=8，　　基本符号 0，1，2，3，4，5，6，7。

十六进制　r=16，　基本符号 0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F。其中 A~F 分别表示十进制的 10，11，12，13，14，15。

十进制　　r=10，　基本符号 0，1，2，3，4，5，6，7，8，9，10。

如果每一数位都具有相同的基，即采用同样的基本符号集来表示，则称该数制为固定基数值，这是计算机内普遍采用的方案。在个别应用中，也允许对不同的数位或位段选用不同的基，即混合采用不同的基本符号集来表示，则该数制称为混合基数制。

图 1.9 示例了几种不同数制表示的数值。

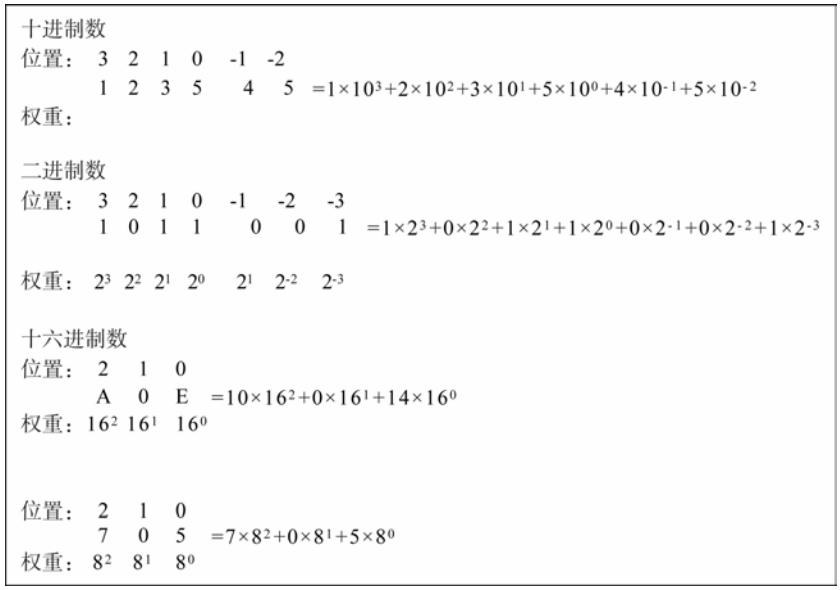


图 1.9　各种数制表示的数值

2. 十进制数到二进制、八进制、十六进制数的转换

计算机中常用几种不同的进位数制，包括二进制、八进制、十六进制和十进制。二进制数据容易用逻辑线路处理，更接近计算机硬件，能直接识别和处理电子化信息的使用要求，而十进制使人更容易接受。应熟练掌握两者之间的进制转换问题。

$$N=\sum_{i=-k}^{m-1}(D_i\times r^i)\ (\text{其中 } r^i \text{ 为 } D_i \text{ 对应的位权})$$

所确定的运算规则是不同进位计数制数据之间完成进制转换的依据。

十进制到二进制的转换通常要区分数的整数部分和小数部分，并分别按除以 2 取余数部分和乘以 2 取整数部分两种不同的方法来完成。

对整数部分，要用除以 2 取余数办法完成十进制到二进制的转换，其规则是：

(1) 用 2 除十进制的整数部分，取其余数为转换后的二进制数整数部分的低位数字。

(2) 再用 2 去除所得的商，取其余数为转换后的二进制数高一位的数字。

(3) 重复前一步，直到商为 0，结束转换过程。

对小数部分，要用乘以 2 取整数办法完成从十进制数到二进制数的转换，其规则为：

(1) 用 2 乘十进制数的小数部分，取乘积的整数为转换后二进制小数的最高位数字。

(2) 再用 2 乘上一步乘积的小数部分，取新乘积的整数为转换后二进制小数低一位数字。

(3) 重复前一步，直至乘积部分为 0，或已得到的二进制小数位满足要求，结束转换过程。

在小数进行转换的过程中，转换后的二进制已达到要求位数，而最后一次乘积的小数部分不为 0，会使转换结果存在误差，其误差值小于得到的最低一位的位权。

对既有整数部分又有小数部分的十进制数，可以先转换其整数部分为二进制数的整数部分，再转换其小数部分为二进制数的小数部分，把得到的两部分结果合并起来得到转换后的最终结果。例如，$(38.43)_{10} \approx (100110.0110111)_2$。

十进制到八进制和十六进制数的转换方法与前述方法类似，只是在乘除 8 和 16 时手工运算不太方便。

3. 二进制数到八进制、十六进制数的转换

用二进制表示一个数值 N，所用的位数 K 为 $\log_2 N$。如表示 4096，K 为 13，写起来位串较长。为此，计算机中也常常采用八进制和十六进制来表示数值数据。为表示 N，分别有如下对应关系：

$$N = \sum_{i=-k}^{m-1} (D_i \times 8^i)$$ (D_i 的取值为 0~7)

例如，$(7.44)_8 = 7\times8^0 + 4\times8^{-1} + 4\times8^{-2} = (7.5625)_{10}$。

$$N = \sum_{i=-k}^{m-1} (D_i \times 16^i)$$ (D_i 的取值为 0~9 和 A~F)

例如，$(1A.08)_{16} = 7\times16^1 + 10\times16^0 + 8\times16^{-2} = (26.03125)_{10}$。

在把二进制数转换成八进制或十六进制表示时，应从小数点所在位置分别向左、向右对每 3 位或每 4 位二进制位进行分组，写出每一组数对应的 1 位八进制数或十六进制数。若小数点左侧(整数部分)的位数不是 3 或 4 的整数倍，可以按在数的最左侧补 0 的方法处理；对小数点右侧(小数部分)，应按在数的最右侧补 0 的方法处理，否则容易转换错。对不存在小数部分的二进制数(整数)，应从低位开始向左，每 3 位划分一组，使其对应 1 个八进制位，或把每 4 位划分成一组，使其对应 1 个十六进制位。例如，$(10.101)_2$ 在转换成八进制时，应把它理解为$(010.101)_2$，是$(2.5)_8$，即八进制的 2.5。当把它转换为十六进制时，应先转换为$(0010.1010)_2$，是$(2.A)_{16}$，即十六进制的 2.A，而不是$(2.5)_{16}$。又如：

$(1100111.10101101)_2 = (147.532)_8$

$(1100111.10101101)_2 = (67.AD)_{16}$

二进制、八进制和十六进制之间的对应关系如表 1.1 所示。

表 1.1　二进制、八进制和十六进制之间的对应关系

二进制	八进制	二进制	十六进制	二进制	十六进制
000	0	0000	0	1000	8
001	1	0001	1	1001	9
010	2	0010	2	1010	A
011	3	0011	3	1011	B
100	4	0100	4	1100	C
101	5	0101	5	1101	D
110	6	0110	6	1110	E
111	7	0111	7	1111	F

八进制和十六进制之间的转换很少使用，它们经过二进制的中间结果进行转换比较方便。

4. 二进制、八进制、十六进制数到十进制数的转换

$$N = \sum_{i=-k}^{m-1} (D_i \times r^i)$$

对任意一种进位记数制表示的数都可以按照上式按权展开，从而计算出 N 的值。例如：

$$(1101.0101)_2 = 1\times2^3+1\times2^2+0\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}+0\times2^{-3}+1\times2^{-4}$$
$$=8+4+1+0.25+0.0625$$
$$=(13.3125)_{10}$$

从式中可以看到，由于二进制只用 0 和 1 两个符号，在计算二进制位串所代表的实际值时，只需把符号为 1 的那些位的位权相加即可，因此，熟悉地记清二进制数每位上的位权是有益的。当位号为 0~12 时，其各位上的位权分别为 1、2、4、8、16、32、64、128、256、512、1024、2048 和 4096。

八进制和十六进制到十进制的转换方法与前述方法类似，这里就不再赘述。

1.5.2　计算机中数据的表示

此部分重点讲解数据表示，即计算机内最常用的信息编码，包括逻辑型数据表示、中西文字符编码表示、数值型数据的编码表示。这些内容是程序设计人员要掌握的基本知识之一，要求能运用自如。

1. 原码、反码和补码

各种数据在计算机中表示的形式称为机器数，其特点是采用二进制计数制，数的符号用 0、1 表示，小数点则隐含表示而不占位置。机器数对应的实际数值称为数的真值。

机器数有无符号数和带符号数之分。无符号数表示正数，在机器数中没有符号位。对于无符号数，若约定小数点的位置在机器数的最低位之后，则是纯整数；若约定小数点的位置在机器数的最高位之前，则是纯小数。对于带符号数，机器数的最高位是表示正、负的符号位，其余位则表示数值。若约定小数点的位置在机器数的最低数值位之后，则是纯整数；若约定小数点的位置在机器数的最高数值位之前(符号位之后)，则是纯小数。

为了便于运算，带符号的机器数可采用原码、反码和补码等不同的编码方法，机器数的这些编码方法称为码制。

(1) 原码表示法。数值 X 的原码记为$[X]_{原}$。设机器字长为 n(即采用 n 个二进制位表示数据)，则最高位是符号位，0 表示正号，1 表示负号；其余 n−1 位表示数值的绝对值。数值零的原码表示有两种形式：$[+0]_{原}$=0 0000000，$[-0]_{原}$=1 0000001。

例如，若机器字长 n 等于 8，则：

$[+1]_{原}$=0 0000001，$[-1]_{原}$=1 0000001；

$[+127]_{原}$=0 1111111，$[-127]_{原}$=1 1111111；

$[+45]_{原}$=0 0101101，$[-45]_{原}$=1 0101101；

$[+0.5]_{原}$=0◇1000000，$[-0.5]_{原}$=1◇1000000，其中◇是小数点的位置。

(2) 反码表示法。数值 X 的反码记为$[X]_{反}$。设机器字长为 n，则最高位是符号位，0 表示正号，1 表示负号，正数的反码与原码相同，负数的反码则是其绝对值按位求反。数值零的反码表示有两种形式：$[+0]_{反}$=0 0000000，$[-0]_{反}$=1 1111111。

例如，若机器字长 *n* 等于 8，则：

$[+1]_{反}$=0 0000001，$[-1]_{反}$=11111110；

$[+127]_{反}$=01111111，$[-127]_{反}$=10000000；

$[+45]_{反}$=0 0101101，$[-45]_{反}$=11010010；

$[+0.5]_{反}$=0◇1000000，$[-0.5]_{反}$=1◇0111111，其中◇是小数点的位置。

(3) 补码表示法。数值 X 的补码记为$[X]_{补}$。设机器字长为 n，则最高位是符号位，0 表示正号，1 表示负号，正数的补码与其原码和反码相同，负数的补码则等于其反码的末尾加 1。在补码表示中，数值零有唯一的编码：$[+0]_{补}$=0 0000000，$[-0]_{补}$=00000000。

例如，若机器字长 n 等于 8，则：

$[+1]_{补}$=0 0000001，$[-1]_{补}$=11111111；

$[+127]_{补}$=01111111，$[-127]_{补}$=10000001，$[-128]_{补}$=10000000；

$[+45]_{补}$=0 0101101，$[-45]_{补}$=11010011；

$[+0.5]_{补}$=0◇1000000，$[-0.5]_{补}$=1◇1000000，其中◇是小数点的位置。

2. 定点数和浮点数

(1) 定点数。所谓定点数就是小数点的位置固定不变的数。小数点的位置通常有两种约定方式：定点整数(纯整数，小数点在最低有效数值位之后)和定点小数(纯小数，小数点在最高有效数值位之前)。

设机器字长为 n，各种码制表示下的带符号数的范围如表 1.2 所示。

表 1.2　各种码制下的带符号数范围

码　　制	定 点 整 数	定 点 小 数
原码	$-(2^{n-1}-1)\sim+(2^{n-1}-1)$	$-(1-2^{-(n-1)})\sim+(1-2^{-(n-1)})$
反码	$-(2^{n-1}-1)\sim+(2^{n-1}-1)$	$-(1-2^{-(n-1)})\sim+(1-2^{-(n-1)})$
补码	$-2^{n-1}\sim+(2^{n-1}-1)$	$-1\sim+(1-2^{-(n-1)})$

(2) 浮点数。当机器字长为 n 时，定点数的补码和移码可表示 2^{n-1} 个数，而其原码和反码只能表示 $2^{n-1}-1$ 个数(0 的表示占用了两个编码)，因此，定点数所能表示的数值范围比较小，运算中很容易因结果超过范围而溢出。所以引入浮点数，浮点数是小数点位置不固定的数，它能表示更大范围的数。

在十进制中，一个数可以写成多种表示形式。例如，83.125 可写成 0.083 125×10^3 或 0.008 312 5×10^4 等。同样，一个二进制数，也可以写成多种表示形式。例如，二进制数 1 011.101 01 可以写成 0.101 110 101×2^4，　也可以写成 0.001 011 101 01×2^6，等等。由此可见，一个二进制数 N 可以表示为更一般的形式：

$$N=2^E\times F$$

其中 E 称为阶码，F 叫作尾数。用阶码和尾数表示的数叫作浮点数，这种表示数的方法称为浮点表示法。

在浮点表示法中，阶码通常为带符号的纯整数，尾数为带符号的纯小数。浮点数的表示格式如下：

阶符	阶码	数符	尾数

很明显，一个数的浮点表示不是唯一的。当小数点的位置改变时，阶码也随之改变，因此可以用多种浮点形式表示同一个数。

浮点数所能表示的数值范围主要由阶码决定，所表示数值的精度由尾数决定。为了充分利用尾数来表示更多的有效数字，通常采用规格化浮点数。规格化就是将尾数的绝对值限定在区间[0.5,1)。关于这方面的详细内容，这里不再阐述。

3. 十进制数的编码

十进制数的每一个数位的基为 10，但到了计算机内部，出于存储与计算的方便，必须采用基 2 码对每个十进制数位进行重新编码，所需要的最少的基 2 码的位数为 $\log_2 10$，取整数为 4。用 4 位二进制代码表示 1 位十进制数，称为二—十进制编码，简称 BCD 编码。因为 4 位二进制可以有 16 种组合，而十进制数只有 0~9 这 10 个不同的数符，故有多种 BCD 编码。根据 4 位代码中每一位是否有确定的权来划分，可分为有权码和无权码两类。

应用最多的有权码是 8421 码，即 4 个二进制位的权从高到低分别为 8、4、2 和 1。无权码中用得较多的是余 3 码和格雷码。余 3 码是在 8421 码的基础上，把每个数的代码加上 0011 后构成的。格雷码的编码规则是相邻的两个代码之间只有一位不同。

常用的十进制数与 8421BCD 码、余 3 码、格雷码的对应关系如表 1.3 所示。

表 1.3　十进制数与 8421BCD 码、余 3 码、格雷码的对应关系

十进制数	8421BCD 码	余 3 码	格雷码	十进制数	8421BCD 码	余 3 码	格雷码
0	0000	0011	0000	5	0101	1000	1110
1	0001	0100	0001	6	0110	1001	1010
2	0010	0101	0011	7	0111	1010	1000
3	0011	0110	0010	8	1000	1011	1100
4	0100	0111	0110	9	1001	1100	0100

4. 非数值数据的表示

在计算机中除了数值数据外，还有另一大类的数据，那就是非数值数据，包括文字、声音和图形等信息。这些信息都必须经过数字化编号后才能被传送、存储和处理，因此掌握非数值数据的编码非常重要。

1) 逻辑数据的表示

逻辑数据是用来表示二值逻辑中的“是”与“否”或称“真”与“假”两个状态的数据。很容易想到，用计算机中的基 2 码的两个状态“1”和“0”恰好能表示逻辑数据的两个状态。例如，用“1”表示“真”，则“0”表示“假”。注意，这里的“1”和“0”没有数值有无或大小的概念，只有逻辑上的意义。在计算机内，可以用一位基 2 码表示逻辑数据，就是说，8 个逻辑数据可以存放在 1 个字节中，可用其中的每个 bit(位)表示一个逻辑数据。在实际应用中，也可以用一个字或一个字节表示单个的逻辑数。

正确地建立逻辑数据的概念是十分重要的。因为计算机内部的所有数字化信息，不论是数据、指令或控制信号，都是用基 2 码表示的，其存储与处理都是用逻辑线路实现的。可以说，逻辑线路是计算机硬件组成的基石，而逻辑线路处理的对象就是逻辑型数据，被称为逻辑变量，即实现输出逻辑变量与输入逻辑变量之间一定的逻辑运算关系。

2) ASCII 码

ASCII 码是计算机中使用最普通的字符编码，它是美国标准信息交换码的简称，该编码已被国际标准化组织(ISO)采纳，成为一种国际通用的信息交换使用的标准代码。

ASCII 码采用 7 个二进制位对字符进行编码，低 4 位组 $d_3d_2d_1d_0$ 用作行编码，高 3 位组 $d_6d_5d_4$ 用作列编码。而一个字符在计算机内实际上用 8 位表示，正常情况下，最高一位为 d_7 为“0”，在需要奇偶校验时，这一位可用于存放奇偶校验的值，此时称这一位为校验位。例如，字母 A 的 ASCII 码为 0100 0001。

根据 ASCII 码的构成格式，可以很方便地从对应的代码表中查出每一个字符的编码。基本的 ASCII 字符代码参见附录 A。

注意，这种字符编码中有如下两个规律：

- 字符 0~9 这 10 个数字符的高 3 位编码为 011，低 4 位为 0000~1001。当去掉高 3 位的值时，低 4 位正好是二进制形式的 0~9。这既满足正常的排序关系，又有利于完成 ASCII 码与二进制之间的类型转换。
- 英文字母的编码值满足正常的字母排序关系，且大小写英文字母编码的对应关系相当简便，差别仅表现在 b_5 一位的值为 0 还是 1，有利于大小写字母之间的编码变换。

3) 汉字编码

汉字处理包括汉字的编码输入、汉字的存储和汉字的输出等环节。也就是在计算机中处理汉字，必须先将汉字代码化，即对汉字进行编码。无论是拼音文字还是象形文字，它们的“意”都寓于它们的“形”和“音”上。目前，直接向计算机输入文字的字形和语音虽然可以实现，但还不够理想。在计算机内部直接处理、存储文字的字形和语音就更困难了，所以用计算机处理字符，尤其是在处理汉字字符时，一定要把字符代码化。

西文是拼音文字，基本符号比较小，编码比较容易，而且在计算机系统中，输入、内部处理、存储和输出都可以使用同一代码。汉字种类繁多，编码比拼音文字困难，而且在一个汉字处理系统中，输入、内部处理、存储和输出对汉字代码的要求不尽相同，所以采用的编码也不尽相同。汉字信息处理系统在处理汉字和词语时，关键的问题是要进行一系列的汉字代码转换。

在计算机中，通常用两个字节表示一个汉字。为了与西文字符的编码区别，把表示一

个汉字的两个字节的最高一个二进制 bit 位设定为 1，而通常用的西文字符 ASCII 码编码的最高一个二进制位总为 0。这种汉字编码方案的编码集最多编码数量为 128×128 个，而且它与西文传送中的把 ASCII 码的最高一个二进制位作为 1 个字符 7 位编码的奇偶校验位的用法有矛盾。

4) 检错纠错码

有关这方面的内容，这里也不再阐述，读者可以参考相关书籍。

5. 存储器中数据的存储特性

程序和数据都存储在存储器中。有关存储常常涉及的术语有以下几个。

- **位**(bit)：存储一个二进制代码 0 或 1 的最小单元称为位，简写为 b。
- **字节**(byte)：连续的 8 个位组成的存储单元称为字节，简写为 B。
- **字**(word)：连续多个字节组成的存储单元称为字。统一称 32 位二进制为一个“长字”，而称 16 位二进制为一个“短字”。
- **地址**：为了访问方便，每个字节都分配一个编号，称为“地址”。在多数计算机中，地址是从低到高连续编址的，最小从 0 开始，最大到实际内存结束，大多也是按字节编址的。

16 位系统的内存结构示例如图 1.10 所示，内存储器的单元很多，一般以字节计算。16 位系统一般有 64KB 的寻址空间。设占两字节的变量 x 在内存中安排了 2 个连续的地址：2000H 和 2001H。其中 2000H 是首地址，如图 1.10(a)所示；变量 x 中的值的二进制形式如图 1.10(b)所示，表示成一般的十进制形式如图 1.10(c)所示。

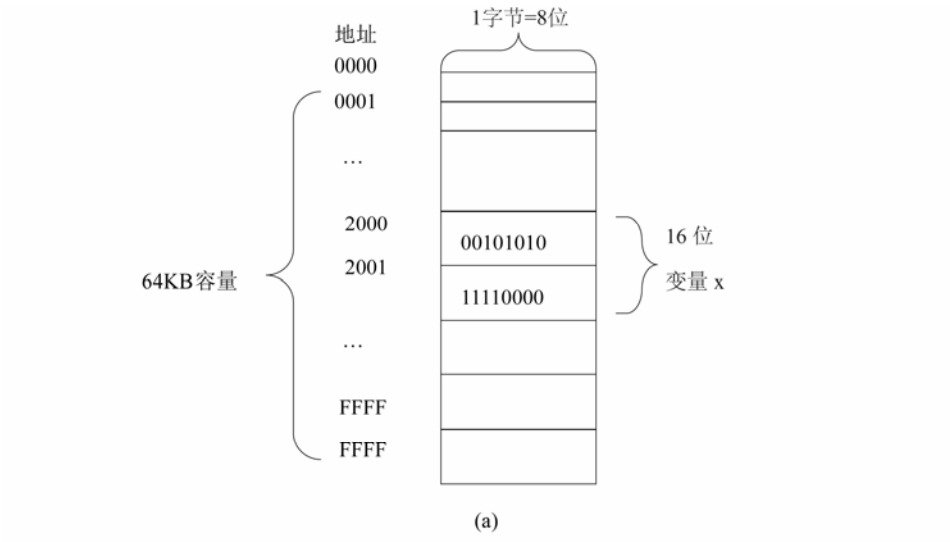


图 1.10　存储地址、变量和变量值

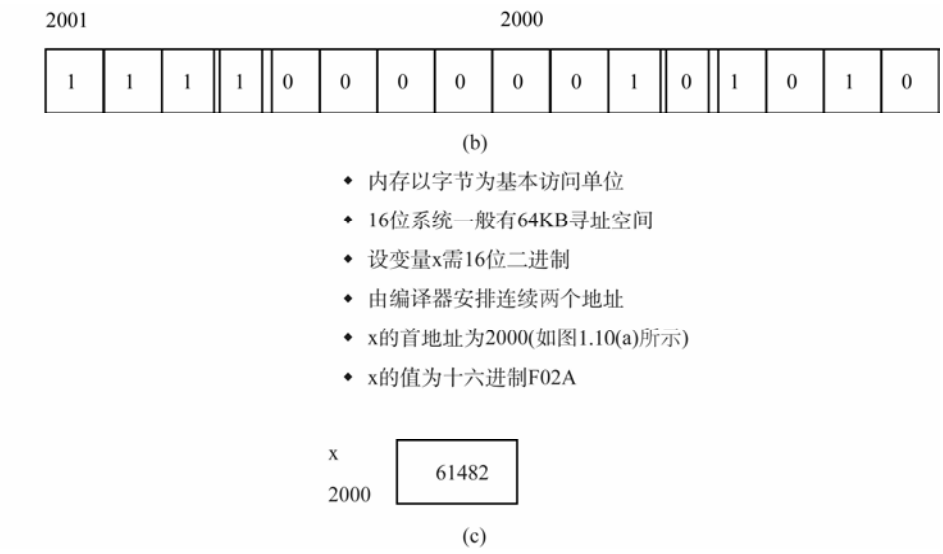


图 1.10　存储地址、变量和变量值(续)

要注意区分存储单元的地址与存储单元的值的不同。图中 2000H 和 2001H 是存储单元的地址，存储单元中的内容 61482 是该存储单元的值。在高级程序设计语言中涉及存储单元，必须把变量名、存储单元地址、存储单元值三个概念一同考虑。一旦定义了一个变量，就在内存中开辟了一个存储单元，具有确定的存储单元地址，但存储单元的值是不确定的，其所在的内存中有一个垃圾值(之前遗留的)或不定值(随机值)，用“?”表示。

以存放整型数据的存储单元 num 为例，讨论存数和取数的特点：当变量 num 定义后，其值不确定；当存入一个数 7 到变量 num 后，就有了一个确定的值；当从变量中取出数后，变量 num 的值仍保持是 7；只有当有一个新的值如 3 存入变量 num 中，变量 num 的值才改变为 3，取出的值为 3。变量的值可以反复地取用。

1.6　结构化程序设计概述


结构化程序设计

20 世纪 60 年代，许多大型软件开发工作遭到了重大的挫折(软件危机，Software Crisis)：软件开发进度被推迟，成本超出预算，产品质量不可靠。人们认识到软件开发比想象的要复杂得多。那个阶段的研究工作促使了“结构化程序设计”的诞生。结构化程序设计是编写清晰、正确和易于修改的程序的严格的方法。

结构化程序设计方法是要求程序员按照一定的规范、采用成熟的设计方法进行程序设计，而不允许程序员随心所欲地编写程序。它强调程序的风格、程序结构的规范化以及自顶向下、逐步细化和模块化的设计方法，其追求的主要目标是提高程序的易读性和易维护性。

1.6.1　结构化程序设计思想

结构化程序设计由艾兹格 • 迪科斯彻(E.W.Dijkstra)在 1969 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，这样使完成每一个模块的工作变得单纯而明确，为设计一些较大的软件打下了良好的基础。

结构化程序设计的中心思想，就是“分而治之”(Divided and Conquered)思想的体现。在面对复杂问题时，采用这种“分而治之”的思想，“化繁为简，各个突破”，这种思想不仅适用于平时的生活，也同样应用于软件开发过程的各个环节。

结构化程序设计中的模块分解和组合，不但可以“自顶向下”(Top-down)，也可以“自底向上”(Bottom-up)，即从两个不同方向进行，有时也可以“自顶向下”与“自底向上”混合着进行。

在结构化程序设计的具体实施中，要注意如下要素：

(1) 使用程序设计语言中的顺序、选择、循环等有限的控制结构表示程序的控制逻辑；选用的控制结构只准许一个入口和一个出口。

(2) 程序语句组成容易识别的块，每块只有一个入口和一出口；复杂结构应该用嵌套的基本控制结构进行组合嵌套来实现。

(3) 语言中所没有的控制结构，应该采用前后一致的方法来模拟；严格控制 goto 语句的使用。

1.6.2　三种基本程序结构

按照结构化程序设计的观点，任何算法功能都可以通过由程序模块组成的三种基本程序结构的组合：顺序结构、选择结构和循环结构来实现。

1. 顺序结构

顺序结构是指按顺序执行几个语句。如图 1.11 表示执行完 A 框所指定的操作后，必然接着执行 B 框所指定的操作。尽管用顺序符号表示的活动可能非常复杂，例如输入或输出操作，但逻辑流程必须是从顶部进入符号，并从底部流出符号。顺序符号不允许符号内有任何选择决策或流程转向。它由赋值、输入/输出、模块调用和复合语句构成。

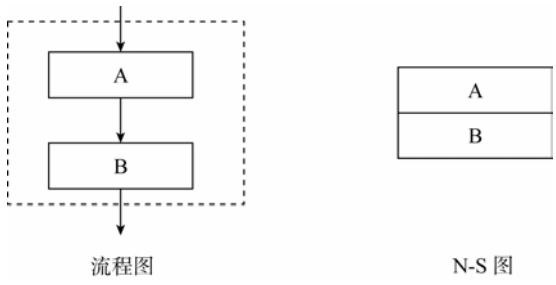


图 1.11　顺序结构

2. 选择结构

选择结构又称选取结构或分支结构，如图 1.12 所示。它与顺序结构不同，选择语句能使程序流程改变，它允许系统执行一些选定的语句，而跳过其他语句。结构化程序设计技术使用两

种选择语句：两路选择和多路选择。

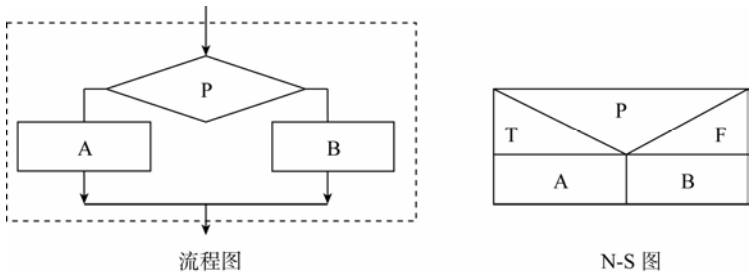


图 1.12　选择结构

此结构必包含一个判断框。根据给定的条件 P 是否成立而选择执行 A 框或 B 框。A 或 B 两个框中可以有一个是空的，即不执行任何操作。

3. 循环结构

循环结构又称为重复结构，即程序反复执行某个或某些操作，直到某条件为假(或为真)时才可终止循环。在循环结构中最主要的是：什么情况下执行循环？哪些操作需要循环执行？循环结构的基本形式有两种：当型循环结构(先判断后执行循环体)和直到型循环结构(先执行循环体后判断)，如图 1.13 所示。

这三种基本结构的共同特点是:

(1) 只有一个入口；只有一个出口。

(2) 结构内的每一部分都有机会被执行到。

(3) 结构内不存在“死循环”。

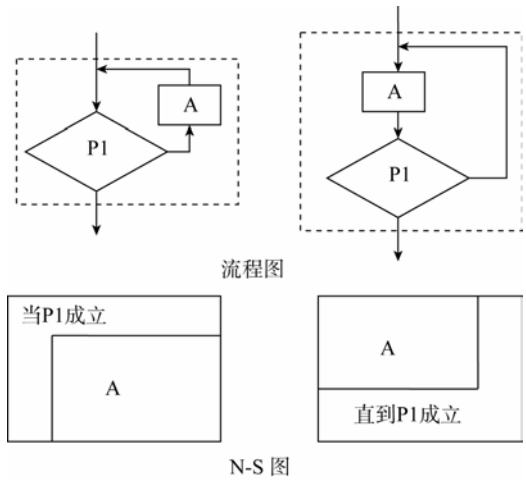


图 1.13　循环结构

1.6.3　结构化程序设计举例

下面举几个例子，说明这种自顶向下、逐步细化的结构化程序设计方法的使用。

例 1-5　打印 2000—2100 年中是闰年的年份。

分析：闰年的条件是：能被 4 整除但不能被 100 整除；或能被 100 整除且能被 400 整除。

解：先画出结构框图 1.14(a)，对其细化得到图 1.14(b)；再对图 1.14(b)中的 b1.1 细化，得到图 1.14(c)。

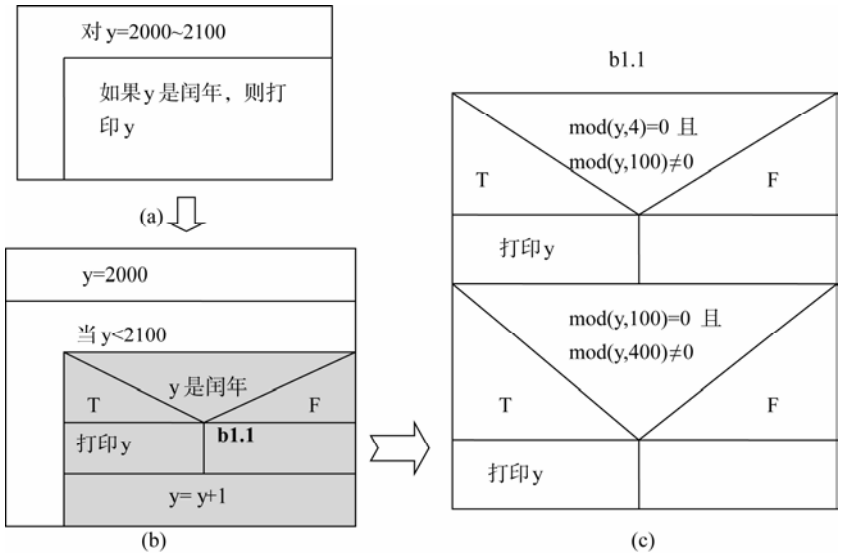


图 1.14　自顶向下，逐步细化

例 1-6　输入 n 个数，找出最大的一个数，并打印出来。

解：先画出结构框图 1.15(a)。考虑逐个读入数据，并把当前各数中的最大者保留下来，以便与后面读入的数比较。将图 1.15(a)细化后为图 1.15(b)，再细化为图 1.15(c)。

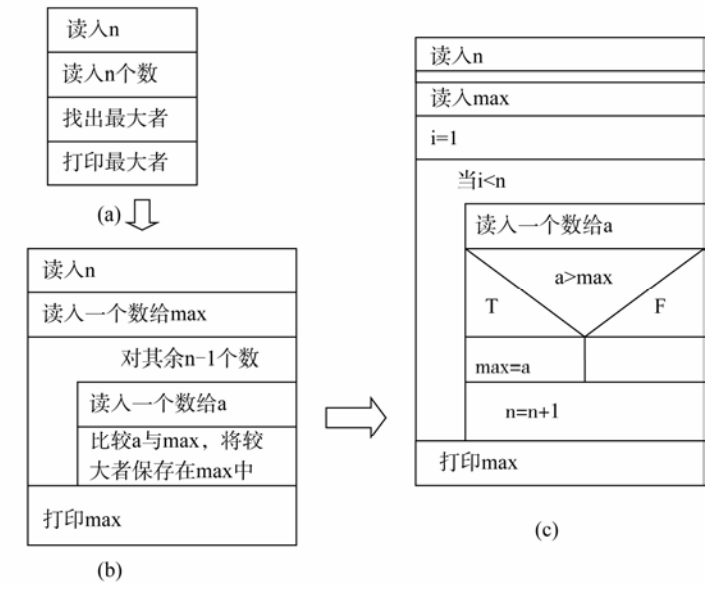


图 1.15　自顶向下，逐步细化

1.7　本章学习小结

本章是学习 C 语言程序设计的基础，内容比较繁杂，读者需重点掌握以下内容：算法及其表示，数制及其转换，三种常用的码制等。其他内容可以暂时不用理解得那么深入，需要时再回头学习或者查看相关的资料。学习这门课程的一个技巧，就是不要陷入语言语法的烦琐细节，应理解领会总体框架。在需要和使用时，带着目的再去学习和查阅相关资料，学习可能会更有效。

1.8　习　　题

1. 在计算机内部，数据和信息都是以_____表示。

2. 三种基本程序结构是指_____、_____和_____。

3. 计算机能直接识别的语言是_____。

4. 流行的主要程序设计方法有_____和_____。

5. 结构化程序设计方法的主要原则可以概括为_____。

6. 在结构化程序设计思想提出之前，程序设计中曾强调程序的效率。与程序的效率相比，现在人们更重视程序的(　　)。

A. 安全性　　B. 一致性　　C. 可理解性　　D. 合理性

7. 对建立良好的程序设计风格，下面的描述正确的是(　　)。

A. 程序应简单、清晰、可读性好　　B. 符号名的命名只要符合语法

C. 充分考虑程序的执行效率　　D. 程序的注释可有可无

8. 结构化程序设计主要强调的是(　　)。

A. 程序的规模　　B. 程序的易读性

C. 程序的执行效率　　D. 程序的可移植性

9. 进制转换和码制(用 1 字节表示)。

(1) $(123)_{10}$=(　　)$_2$=(　　)$_8$=(　　)$_{16}$

(2) $(-12)_{10}$=(　　)$_{原码}$=(　　)$_{反码}$ =(　　)$_{补码}$

10. 选用传统流程图、N-S 图或伪代码表示以下各题的算法。

(1) 读入 3 个数并分别存入 a、b、c 变量中，实现按从大到小的顺序把它们打印出来。

(2) 将 200～500 中的素数打印出来。

(3) 求 sum=1+2+3+…+100 。

(4) 求两个数中的最大公约数和最小公倍数。

(5) 输入 10 个数，找出最大的一个数，并打印出来。