Data Structure and Algorithm

数据结构与算法教程

章炯民　窦　亮　黄国兴◎著

华东师范大学出版社

数据结构与算法教程

章炯民　窦亮　黄国兴　著

华东师范大学出版社

图书在版编目(CIP)数据

数据结构与算法教程/章炯民等著. —上海:华东师范大学出版社,2007.4
ISBN 978-7-5617-5356-9

Ⅰ.数… Ⅱ.章… Ⅲ.①数据结构-高等学校-教材 ②算法分析-高等学校-教材 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字(2007)第 056385 号

数据结构与算法教程

著　　者　章炯民　窦　亮　黄国兴
策划组稿　大中专教材事业部
项目编辑　朱建宝
文字编辑　姚清耘
版式设计　蒋　克

出版发行　华东师范大学出版社
社　　址　上海市中山北路 3663 号　邮编 200062
电　　话　021-62450163 转各部　行政传真 021-62572105
网　　址　www.ecnupress.com.cn　www.hdsdbook.com.cn
市 场 部　传真 021-62860410　021-62602316
邮购零售　电话 021-62869887　021-54340188

印 刷 者　上海市印刷三厂
开　　本　787×1092　16 开
印　　张　16.75
字　　数　386 千字
版　　次　2007 年 7 月第一版
印　　次　2007 年 7 月第一次
印　　数　3100
书　　号　ISBN 978-7-5617-5356-9/O·193
定　　价　26.00 元

出 版 人　朱杰人

前　言

计算机从其诞生至今短短的几十年间，得到了令人瞩目的飞速发展和广泛应用，在很多方面都深刻地改变了人类的活动方式和思维习惯，计算机技术在经济和社会发展中的地位越来越重要。

在用计算机解决实际问题时，首先要对现实世界建立数学模型，然后将这个模型存储在计算机中进行加工和处理，最后再把计算机的运算结果根据数学模型映射成现实问题的解。在建立数学模型时，可以用符号，即数据表示现实世界中的各种对象。但是，现实世界中的各个对象并不是孤立的，它们相互之间可能存在着各种复杂的关联关系。所以，为了准确地反映现实世界，在所建立的数学模型中，还需要包括有关的关联关系。粗略地说，这种带有关系的数据就是“数据结构”。现如今，计算机应用已由纯粹的数值计算发展到用来处理各种具有错综复杂关系的对象。为了编写出一个“好”的程序，必须深入分析和研究相应的数据结构。这就是“数据结构”这门学科形成和发展的背景。

“数据结构”是计算机专业的基础课程，是编译程序、操作系统、数据库系统等专业课程的前导课程。《中国计算机科学与技术学科教程(2002)》为计算机专业本科教学规定了16门核心课程，“数据结构”就是其中之一。《中国计算机科学与技术学科教程(2002)》对“数据结构”的具体教学知识点提出了明确的要求，希望通过这门课程的学习使学生掌握最常用的基本数据结构和算法，并能学以致用。

本教材正是按照《中国计算机科学与技术学科教程(2002)》的指导思想及其关于“数据结构”知识点的具体要求而编写的。本书作者从事“数据结构”课程教学多年，对这门课程本身有比较深刻的理解，对学生们学习该课程的需要和难点所在有着比较清晰的认识。根据作者所积累的丰富经验，本书在内容的安排和对知识点教学的处理方面作了一些尝试，力求体现以下特点：

1. 内容全面，本教材涵盖了《中国计算机科学与技术学科教程(2002)》所指出的“数据结构”课程应该包含的核心教学知识点。

2. 数据结构和算法密不可分，本书把数据结构的内容与算法设计及分析融合在一起讨论，在叙述数据结构的内容时，穿插算法设计与分析的内容。

3. 删繁就简，突出核心内容，并适当控制难度，使本书既可用于本专科教学，也可用作自学参考书。

4. 叙述简练明了，但对难点剖析详尽。整体思路清晰，条理清楚，使学生容易理解，也

便于教师讲授。

5. 提供例题较多，以期启发思路，帮助学生提高分析问题和解决问题的能力。

全书共分8章。第1章“绪论”，介绍数据结构和算法的基本概念、算法的渐近分析、算法设计的基本策略；第2章“线性表”，讲述线性表的概念、存储结构、操作算法及其应用；第3章“栈和队列”，介绍栈和队列的基本概念及操作、存储结构以及应用实例；第4章“数组、矩阵和串”，介绍另一些常见的线性表的处理方法，包括数组的顺序存储方法、矩阵的压缩存储方法，以及串的概念、存储结构和操作算法；第5章“树”，讲述树和二叉树，介绍了它们的存储结构、操作算法；第6章“图”，介绍图的基本概念、存储结构、操作算法以及应用；第7章“查找”，讲述了组织集合的几种方法，包括线性表、查找树、B-树、散列表，主要介绍了各种查找、插入和删除算法；第8章“排序”，介绍了几种最常用的排序算法。

虽然作者对本书的内容作了精心的取舍，但所涉及的内容仍然比较多，各学校的教师和学生的情况也不完全一样，所以，在采用本书作为“数据结构”课程教材时，建议根据实际情况和教学要求进一步作裁剪处理，适当地增减内容，合理地分配教学课时，本书在内容编排上也已经考虑到了这样的需要。

在本书的编写过程中，作者与计算机教育界的同行就“数据结构”的内容和形式进行了诸多有益的讨论，他们提出了许多建设性的意见和建议，在此一并致谢。

由于计算机科学技术发展迅速，加上编者水平有限，书中错误或不妥之处难免，恳请批评指正。

编　者

2007年2月

目　录

第1章 绪 论

自从1946年第一台计算机问世以来，计算机技术的飞速发展远超出了人们的预料。如今，计算机已渗透到现代社会生活的方方面面，并在一定程度上改变了人类的活动方式和思维习惯，计算机的应用也不再局限于科学计算，而是更多地用于控制、管理、娱乐等其他领域。与此相应，计算机处理的对象也从单纯的数值型数据发展到各种不同形式的数据，如字符、表格、声音、图像等。与纯粹的数值型数据不同，非数值型数据往往带有一定的“结构”，即数据之间的关联，它们是数据所代表的客观对象之间的联系的对应物。计算机在存储和处理这类数据时，也必须同时存储和处理这些“结构”。只有这样才能在计算机中完整地再现客观世界，并进而实现用计算机来控制或管理的目标。此外，利用现代计算机速度快、容量大的特点，存储和检索信息已成为计算机的重要应用之一，如何合理地组织和存储信息，以支持高效的信息处理，这也是计算机科学的研究重点之一。研究各种数据及其相互之间的关系、研究如何有效地存储数据、研究如何设计出好的处理数据的程序等，这些都是“数据结构”课程所要讨论的。

“数据结构”是计算机专业的核心基础课程之一。十分明显，数据结构与计算机软件有着密不可分的关系，它是程序设计的基础之一，是设计和实现编译程序、操作系统、数据库管理系统等系统软件和应用软件的重要基础。在研究数据结构时，往往还涉及计算机硬件，如各种存储器、存取方法等。此外，为实际问题建立模型、分析模型、分析算法的效率等都需要一定的数学工具和方法。所以，数据结构是介于数学、计算机硬件和计算机软件三者之间的一门综合性课程。

1.1 数据结构的概念

所谓数据结构，简单来说就是数据及其相互之间的关系，其中，数据间的不同关系种类决定了不同种类的数据结构。本课程将介绍四种基本数据结构：集合、线性表、树和图。

在计算机科学中，研究一种数据结构主要从以下四个方面来进行：

① 数据结构的逻辑结构：特定的数据结构的定义；

② 数据结构的物理结构：在计算机中存储这种数据结构的方法；

③ 该数据结构上常用的操作算法；

④ 该数据结构的应用。

在许多情况下，也把上述四个方面中的前三个方面合称为数据结构，而在另一些场合下，

数据结构又单指其逻辑结构或物理结构，所以名词“数据结构”的确切含义取决于上下文。

本书将从第 2 章开始，从上述四个方面讨论常用的基本数据结构。本章是一个简单的前导，主要讨论各种数据结构的共性问题，介绍研究数据结构的方法、工具以及其他相关问题。

1.1.1 逻辑结构

数据(data)是对客观事物的符号表示，在计算机科学中是指所有能输入到计算机中并被计算机程序处理的符号总称。这是计算机程序加工的“原料”。例如，利用数值分析方法解代数方程的程序，其处理对象是实数；编译程序或文字处理程序的处理对象是字符串等。因此，对计算机科学而言，数据的含义极为广泛，图像、声音等都可以通过编码而纳入数据的范畴。

数据元素(data element)指的是数据的基本组织单位，在计算机程序中通常将它作为一个整体进行考虑和处理。例如，在如图 1.1(a)所示的组织结构中，“树”中的每个方框所表示的数据是一个数据元素，图 1.1(b)中的每一个圆圈所表示的数据也被看作为一个数据元素。有时一个数据元素又可由若干个其他数据元素组成。例如，可以把本书看作一个数据元素，它由八章组成，其中的每一章都是一个数据元素。所以，数据元素从它们的组成结构来看是分层次的，其中不可分割的最小的数据组织单位称为**数据项**(data item)。注意，所谓“不可分割的最小的数据组织单位”只有相对意义，没有绝对的意义，一个数据元素是否是不可分割的，取决于具体的应用及其背景。当数据元素由若干个其他数据项组成时，称能唯一标识数据元素的数据项为**关键字**(key)，简称为**键**。注意，关键字也可以由多个数据项组合而成。本书中，为叙述方便起见，在表示一个数据元素时，往往只给出该数据元素的关键字，而省略其他的数据项。

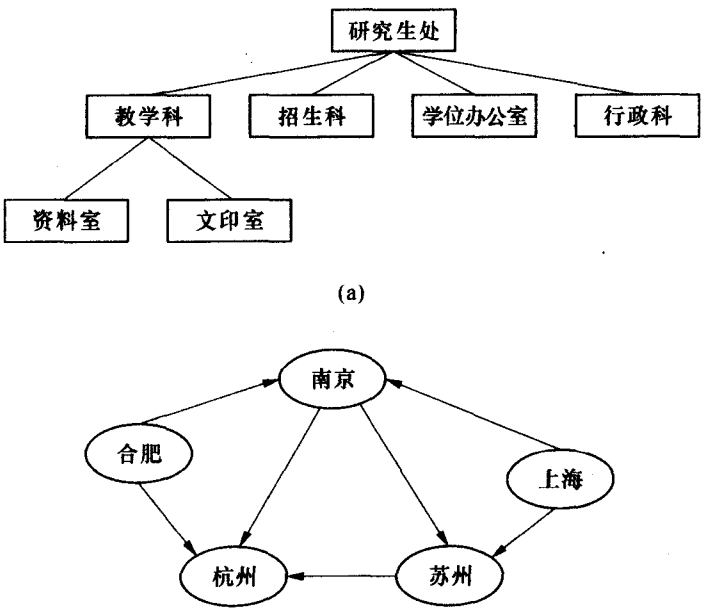

图 1.1　两个数据结构举例

在一个学生成绩管理系统中，有一张学生成绩表，如表 1.1 所示。其中的每一行都是一个数据元素，由学号、姓名、各科成绩及平均成绩等数据项组成，其中学号是关键字。若有必要，也可以把姓名进一步分解成多个字符，即不把姓名看作数据项，而把它看作是由多个字符(数据项)组成的数据元素。

表 1.1　学生成绩表

学　号	姓　名	数据结构	C 语言	高等数学	平均成绩
00122001	张　三	90	85	95	90
00122002	李　四	80	85	90	85
00122003	王　五	70	84	86	80
00122004	马　六	91	84	92	89
…	…	…	…	…	…

由于数据元素所代表的客观对象在现实中并不是孤立存在的，它们之间存在着或多或少的联系，所以数据元素之间也存在着某些关联，这种数据元素之间的关联称为**结构**(structure)，常用集合上的关系来描述。**数据结构**(data structure)是相互之间存在一种或多种特定关系的数据元素的集合，换言之，数据结构就是数据及其相互之间的关联的综合体。一个数据结构可以形式地表示成一个二元组(即有序偶对)(D, R)，其中 D 是数据元素的集合，R 是 D 上的关系(即由 D 中的元素组成的有序偶对)的集合(在本书所涉及的各种数据结构中，R 中只有一个关系)。

图 1.1(a)所示的“树”是一个数据结构，其中 $D=\{$研究生处，教学科，招生科，学位办公室，行政科，资料室，文印室$\}$，$R=\{r\}$，$r=\{$(研究生处，教学科)，(研究生处，招生科)，(研究生处，学位办公室)，(研究生处，行政科)，(教学科，资料室)，(教学科，文印室)$\}$。图 1.1(b)所示的“图”也是一个数据结构，其中 $D=\{$上海，南京，苏州，杭州，合肥$\}$，$R=\{r\}$，$r=\{$(上海，南京)，(上海，苏州)，(南京，杭州)，(南京，苏州)，(苏州，杭州)，(合肥，南京)，(合肥，杭州)$\}$。

设有一个数据结构 (D, R)，$R=\{r\}$，a 和 b 是 D 中的数据元素。若 (a, b) 是 r 中的一个有序偶对，则称 a 是 b 的**直接前驱**(immediate predecessor)，有时也简称为**前驱**(predecessor)；称 b 是 a 的**直接后继**(immediate successor)，有时也简称为**后继**(successor)。通常，在数据结构的图形表示中，为了表示数据元素之间的关系 r，我们从前驱到后继画一个箭头。

如图 1.1(b)所示，“上海”是“南京”和“苏州”的前驱，“南京”和“苏州”都是“上海”的后继；“合肥”也是“南京”的前驱，但“合肥”不是“苏州”的前驱。

注意，在图 1.1(a)中，数据元素之间的连线上没有箭头，这是因为在用图形表示“树”时，我们约定把前驱画在上方，把后继画在下方，并省略连线上的箭头，默认为所有的连线上都有一个从上方指向下方的箭头(参见第 5 章)。

通常，根据数据元素之间的关系(结构)的不同特性，可以把数据结构划分成不同的种

类:集合、线性表、树和图。这四种基本数据结构的主要特点如下:

① 集合:集合结构中的数据元素之间除了"同属于一个集合"的性质(关系)外,别无其他关系。

② 线性表:线性表中的数据元素之间存在"一对一"的关系,除始点无前驱,终点无后继外,其余各个数据元素都有唯一的一个前驱和唯一的一个后继。表 1.1 给出了一个线性表,其中,"李四"所在的数据元素的前趋和后继分别是"张三"和"王五"所在的数据元素。

③ 树:树中的数据元素之间允许存在"一对多"的关系。一个数据元素可能有多个后继,但却至多只有一个前趋。图 1.1(a)给出了一棵"树",其中,"教学科"的唯一前驱是"研究生处",而后继有两个:"资料室"和"文印室"。

④ 图:图中的数据元素之间允许存在多对多的关系。一个数据元素可能有多个前驱和多个后继,也可以没有前驱和后继。图 1.1(b)给出了一个图,其中,"南京"有两个前驱:"合肥"和"上海",并有两个后继:"杭州"和"苏州"。

从上面的描述中可以看出,线性表是特殊的树,树是特殊的图,集合也是特殊的图,图是最一般的数据结构。注意,集合并不是特殊的线性表。

1.1.2 抽象数据类型

数据类型(data type)是一个值的集合以及定义在这个集合上的一组操作的总称(综合体)。例如,C 语言中的整数类型,其值的集合由某个区间中的整数构成(在不同的计算机上,这个区间的大小可能不一样),定义在这个集合上的操作有:加、减、乘、取模、除等。

数据类型与数据结构之间有着密切的联系。我们可以把数据结构中的数据元素和关系都看为"值",把数据结构中的操作看作定义在这个"值"的集合上的操作,从而把一种数据结构当作一个数据类型。

注意,数据类型的概念不涉及值的具体存储方式,也不涉及操作的具体实现,所以数据类型只是一个逻辑上的概念,与数据类型的实现无关。事实上,根据具体的软硬件环境以及具体应用的需要,一个数据类型可以有多种不同的实现方法。

抽象数据类型(abstract data type),简称 ADT,是一个数学模型,其中定义了一组操作,这组操作的行为刻画了这个数学模型的逻辑特性。与数据类型一样,ADT 也是一个逻辑概念,与其在计算机上的表示和实现无关。无论具体如何表示和实现一个 ADT,如果它的数学特性(行为)不变,则对用户来说都是相同的。抽象数据类型和数据类型实质上是相似的概念,只是 ADT 的范畴更广,不再局限于某种具体的程序设计语言,可用来定义用户所需要的任何数据类型。

在计算机科学中,解决复杂问题的基本策略是:将复杂问题简单化。传统的方法是把一个复杂的问题分解为若干个较简单的子问题,然后分别解决这些子问题。在这个过程中,还可以对较复杂的子问题作进一步的分解。这就是传统的自顶向下的程序设计方法,是面向过程的。现代的 OOP(面向对象的程序设计)方法则采用抽象化的手段,通过把问题划分成多个抽象层次来达到简化问题的目的。在每个层次上,只需要关注与该层次有关的关键点,

而忽略不必要关心的细节。在使用某个 ADT 的层次上，不必关心该 ADT 的内部实现细节，只需关心 ADT 的外在的、抽象的逻辑性质，从而将注意力集中于当前这个层次的关键问题。在具体实现 ADT 时，则只要关注 ADT 的内部实现细节，而不必过多地考虑 ADT 的应用。通过封装，ADT 内部的、与其逻辑性质无关的细节对其用户（应用）是隐藏的，不能访问的。

数据结构的逻辑特性可以表述为 ADT，即数据结构的逻辑结构和其上的操作的逻辑特性可以用 ADT 来描述，但不涉及具体的实现细节。根据 OOP 的思想，可以将一种数据结构视为一个 ADT，并预先实现常用的基本数据结构，将它们封装成独立的模块。在构造复杂程序时，可以把它们当作构件块来使用，而不必再考虑它们的实现细节，从而简化问题的求解过程，减少程序开发的工作量，提高效率。

1.1.3 物理结构

讨论数据结构的目的之一是为了在计算机中实现对它的存储和处理，因此必须研究如何在计算机中存储（表示）和操作数据结构。

数据结构在计算机中的表示（存储）称为数据的**物理结构**（physical structure），又称为数据结构在计算机中的**映象**或**存储结构**（storage structure），包括数据元素的表示和关系的表示。

数据元素在计算机中的映象是结点。结点在计算机中表示成**位串**（bit string），如用一个字长的位串表示一个整数，用八位二进制数表示一个字符等，通常称这个位串为**结点**（node）或**记录**（record）。当数据元素由若干其他数据元素或数据项组成时，位串中对应于各个数据元素或数据项的子位串称为**数据域**、**域**或**字段**（field）。比如某个学生结点，由学号、年龄、成绩等字段组成。有些时候，我们把数据元素也称作结点，所以术语“结点”的确切含义需视上下文而定。

出于表示数据元素之间的关系的需要或出于提高处理效率的需要等，人们创造了许多存储数据结构的方法（存储结构），以适应千变万化的应用要求。这些存储结构中最基本的大致有两种：顺序存储结构和链式存储结构。在顺序存储结构中，数据元素存储在整块连续的存储区域中，数据元素之间的逻辑关系借助于元素在存储器中的相对位置来表示，即数据元素间的关系由它们的相对物理存储位置隐含地表示。在链式存储结构中，数据的存储区域可以是不连续的，数据元素间的关系由指针（地址）显式地表示。两种存储结构各有特点。例如，顺序存储结构往往空间利用率较高，但更新操作的效率较低，适合于静态场合；链式存储结构则相反，往往空间利用率较低，但更新操作的效率较高，适合于动态场合。对于某种特定的数据结构，应该采用哪种方法取决于具体的数据结构和具体的应用场合，通常必须进行综合平衡才能取得较好的效果。

如何描述存储结构呢？虽然存储结构涉及数据元素及其关系在存储器中的物理位置，但由于我们是在高级语言的层次上讨论数据结构的操作，因此不能直接以内存地址来描述存储结构，但可以借用高级语言中提供的“数据类型”来描述它们。例如，可以用“一维数组”来描述顺序存储结构，用“指针”来描述链式存储结构。

1.2 算法的概念和描述

为了用计算机解决某个问题，需要制定出适合在计算机上执行的详细步骤，即算法，然后用某种程序设计语言实现这个算法，即编制计算机程序，最后在计算机上运行这个程序，解决预定的问题。数据结构上的各种操作都是通过算法来完成的。

算法在计算机科学中具有十分重要的核心地位，许多计算机界的前辈大师都对此有精辟的论述。例如，D. E. Knuth 曾经指出，“计算机科学是有关算法的学问”，“对所有的计算机程序设计来说，算法的概念总是最基本的”。

计算机算法与数据结构是紧密联系在一起的。一方面，数据结构中包含操作，需要为它们制定算法以便在计算机上实现；另一方面，任何算法都必须处理或操作某些数据，因而必须基于一定的数据结构。事实上，我们可以把程序看作在某种数据结构基础上对抽象算法的具体表达。N. Wirth 的一本有名的专著就以“算法＋数据结构＝程序”作为书名。

一般意义上的算法指任何解决问题的方法，但本书不研究这种一般的算法，本书仅局限于讨论计算机算法，即在计算机上解决问题的方法。

1.2.1 算法的概念

如果把问题抽象为数学模型，那么它就是从输入到输出的**函数**(function)，也称为**映射**。不同的输入所对应的输出可以是不同的，但是对于相同的输入，每次计算函数所得到的输出必定相同。而算法则以某些最基本、最简单的操作(指令)为基础，给出了把输入转换为输出的方法(操作步骤、“过程”、“工序”)。显然，一个问题可以有多种算法，但一个给定的算法只解决一个特定的问题。更精确地说，**算法**(algorithm)是指令的有限序列(显然，对无限的指令序列，我们无法编写计算机程序来实现它。另外，这里的指令是指某些一般的操作，不要求必须是计算机指令)，并具有下列五个性质：

① 有穷性。算法中的每条指令都可在有限的时间内执行完毕，且算法对任何预定的合法输入(在问题的定义域中的对象)都在执行有限条指令之后结束。

② 确定性。算法中的每条指令都有明确的含义，没有二义性；执行完一条指令后，接下来应执行哪条指令也是明确的。这样，算法在相同的输入下必定得到相同的输出。

③ 可行性。算法中的每条指令所指示的操作都是可执行的。粗略地来讲，所谓可执行就是指可以由人用一张纸和一支笔来完成(假定纸和笔都是足够的)。例如，没有要求执行 3÷0 之类的操作的指令。

④ 输入。有零个或多个输入，它们取自某个特定的数据集合。本书中，我们用 C 语言来描述算法，算法的输入常表示为 C 语言的形式参数。

⑤ 输出。有一个或多个输出，通常是对输入进行加工所得到的结果。若用 C 语言来描述算法，则输出一般表示为返回值或指针类型的形式参数。

显然，计算机**程序**(program)是计算机指令的有限序列，满足算法的性质②③和④。通常，程序也满足算法的性质⑤，因为一个没有输出的程序是没有任何意义的。有些程序的运

行效果可能肉眼看不出来，如修改磁盘上的某个文件等，但这些都是对外界的改变，都是输出。如果一个程序还满足算法的性质①，则它也是一个算法。然而，并不是所有的程序都是算法。不是算法的程序很多，也很常见，而且，许多这样的程序也是有实际用途的。例如，交互式的操作系统（如 Windows）就是一个这样的程序，它不满足算法的性质①，因为它永远不会终止（除非关闭计算机），永远在等待使用者的输入。但是，操作系统中的某一段程序，独立地来看，很可能满足算法的五个性质，因而是算法。

通常，程序被认为是对一个抽象算法（以各种非程序的形式，如自然语言表示）的具体实现或表示（使用某种程序设计语言）。但是，算法本身是独立于实现算法的程序的，一个算法可以有多个程序实现。比如，由不同的程序员使用不同的程序设计语言实现同一个算法，可以得到多个实现该算法的程序。尽管这样，由于我们的最终的目标是在计算机上求解问题，所以，在描述算法时，必须提供足够的细节，以便必要时转换为程序。

抽象的算法不能在计算机上实际运行，但我们有时仍然会说“执行”某个算法，这时我们隐含地是指执行该抽象算法的某个具体实现（程序）。

尽管“算法”和“程序”是两个独立的概念，但由于它们之间的密切联系，本书常常不加区别地使用这两个术语。

1.2.2 算法的描述——C 语言回顾

算法有多种表示方法，常见的表示方式有：使用自然语言、使用各种具体的程序设计语言、使用某种形式的伪代码等。通常，用自然语言描述的算法比较简洁，容易读懂，但不精确，可能有二义性，不能在计算机上运行；用程序设计语言描述的算法是精确的，没有二义性，可以直接在计算机上运行，但涉及较多的细节，是面向计算机的，不容易读懂；用伪代码描述的算法也不能直接在计算机上运行，其他性质则界于前两者之间。本书中，我们将同时使用上述三种描述方法。对比较复杂的算法，首先用自然语言或伪代码（自然语言与 C 语言的混合物）描述算法的基本思想和步骤，然后给出 C 语言描述的程序；对比较简单的算法，则直接给出 C 语言描述的程序。

本书主要采用 C 语言来描述算法，C 语言是本课程的先修课程。为了便于读者学习，本小节对 C 语言的主要内容作一个简要的回顾。熟悉 C 语言的读者可以跳过本小节。

(1) C 语言回顾

一种语言之所以能存在和发展，并具有生命力，总是有其不同于（或优于）其他语言的特点。C 语言的主要特点如下：

① 简洁、紧凑，使用方便、灵活。C 语言一共只有 32 个关键字，9 种控制语句，程序书写形式比较自由。

② 具有丰富的运算符。C 语言共有 34 种运算符，把括号、赋值、强制类型转换等都当作运算符来处理，从而使运算类型极其丰富，表达式类型多样化，灵活地使用各种运算符常可以非常简洁地表达某些比较复杂的操作。

③ 具有丰富的数据结构，包含现代高级语言所应具备的各种数据结构。C 语言的数据