高等学校**应用型特色**规划教材

数据结构

(C语言版)

赠送电子教案

陈 锐 主 编
扶 晓 刘 琨 郑春霞 副主编

清华大学出版社

TP311.12
C543-2

郑州大学 *04010747705Z*

特色规划教材

数据结构(C 语言版)

陈 锐 主 编

扶 晓 刘 琨 郑春霞 副主编

2012-06-04
藏书

清华大学出版社
北 京

TP311.12
C543-2

内 容 简 介

数据结构是计算机专业的核心课程。本书所有算法都采用C语言描述，书中不仅讲解了数据结构的基本理论知识，还提供了大量典型的应用实例，所有实例均能直接运行，以帮助读者充分理解和掌握知识点。全书共分9章，内容包括数据结构概述、线性表、栈和队列、串、数组和广义表、树和二叉树、图、查找、排序等。

本书内容实用，结构清晰，实例丰富，可操作性强，可作为本、专科院校的计算机及相关专业的数据结构教材，也可作为进行计算机软件开发、考研和软件资格(水平)考试人员的参考书。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目(CIP)数据

数据结构(C语言版)/陈锐主编；扶晓，刘琨，郑春霞副主编. --北京：清华大学出版社，2012
(高等学校应用型特色规划教材)
ISBN 978-7-302-27907-5

Ⅰ. ①数… Ⅱ. ①陈… ②扶… ③刘… ④郑… Ⅲ. ①数据结构—高等学校—教材 ②C语言—程序设计—高等学校—教材 Ⅳ. ①TP311.12 ②TP312

中国版本图书馆CIP数据核字(2012)第008891号

责任编辑：黄　飞　杨作梅
封面设计：杨玉兰
责任校对：王　晖
责任印制：何　芊

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
课 件 下 载：http://www.tup.com.cn,010-62791865

印 装 者：三河市李旗庄少明印装厂
经　　销：全国新华书店
开　　本：185mm×260mm　　印　　张：18.75　　字　　数：451千字
版　　次：2012年2月第1版　　印　　次：2012年2月第1次印刷
印　　数：1～4000
定　　价：35.00元

产品编号：041267-01

前　言

在多年来的教学与研究过程中，作者读过不少关于数据结构方面的教材，发现好多教材不是内容不够全面，就是语言表述方面偏于晦涩，看书时间久了就会比较辛苦。有鉴于此，作者采用通俗活泼的语言风格编著了本书，把这些年来学习数据结构的一些心得分享给大家，希望能对大家的学习有所帮助。

数据结构是计算机科学与技术专业的一门重要的专业基础课程。当用计算机来解决实际问题时，就要涉及数据与数据之间关系的表示与处理，而这正是数据结构研究的对象。通过对数据结构的学习，可为后续课程，特别是软件方面的课程打下坚实的知识基础。因此，数据结构在计算机相关专业中起着举足轻重的作用。

目前，数据结构不仅是计算机专业的必修课程，而且大多数高等院校的非计算机专业也将数据结构作为主干课程。数据结构不仅是计算机专业考研的必考科目之一，还是全国计算机等级考试、软件资格(水平)考试的主要考试内容。

本书全面介绍数据结构中的线性结构、树形结构、图结构及查找、排序技术，通过图和实例的形式分析了算法思想，以便读者理解和掌握。相信学完本书之后，读者将具备一定的抽象思维的能力和算法设计的能力。

本书的特点

1．内容全面，讲解详细

本书涵盖了数据结构中线性结构、树形结构和图结构的所有知识点，对于每一种数据结构，都使用了所有可能的逻辑结构和存储结构进行描述，并对算法的实现尽量多地采用多种实现方式，如递归和非递归、顺序存储和链式存储，从而使读者对算法的理解更加深刻。

2．层次清晰，结构合理

本书将数据结构按章、节划分知识点，将知识点细化，易于读者理解。在知识点的讲解过程中，循序渐进，由浅入深，先引出概念，然后用例子说明，最后是算法描述与程序实现。这样的层次十分易于读者的理解和消化。

3．结合图表，叙述简单

在每个概念提出后，都结合图表和例子加以说明以方便读者理解。在内容的描述上，普遍采用短句子、易于理解的语言，而避免使用复杂句子和晦涩难懂的语言。通过以上方式的描述，读者可以更加容易和轻松地学习数据结构。

4．例子典型，深入剖析

在例子的选取上，都是一些最为常见且涵盖知识点丰富的典型算法。在每章的最后，都会给出一个完整的程序，对算法进行剖析，并给出程序运行结果，以帮助读者分析、理解算法。

5．配有习题，巩固知识

在每章的最后，都有一个小结，对本章的知识点进行总结。为了让读者熟练编写算法，本书在每章的最后都会配有一定数量的实践题目，在学习每章的内容之后，可以通过这些习题试着编写算法，以巩固本章的学习内容。

本书的内容

第 1 章：如果读者刚接触数据结构，这一章将告诉您数据结构是什么，以及本书的学习目标、学习方法和学习内容；另外，还介绍了本书对算法的描述方法。

第 2 章：主要介绍了线性表。首先讲解线性表的逻辑结构，然后介绍线性表的各种常用存储结构，在每一节均给出了算法的具体应用。通过学习这一章，读者可以掌握顺序表、动态链表和静态链表的操作。

第 3 章：主要介绍操作受限的线性表——栈和队列，内容包括栈的定义，栈的基本操作及栈与递归的转化，队列的概念，顺序队列和链式队列的运算。

第 4 章：主要介绍一种特殊的线性表——串。首先介绍串的概念，然后介绍串的各种存储表示，以及串的模式匹配算法。

第 5 章：主要介绍数组与广义表。首先介绍数组的概念，数组(矩阵)的存储结构及运算，几种特殊矩阵；然后介绍广义表的概念，广义表的两种存储方式，广义表的操作实现。

第 6 章：主要介绍非线性数据结构——树和二叉树。首先介绍树和二叉树的概念，然后介绍树和二叉树的存储表示、二叉树的性质、二叉树的遍历和线索化、树、森林与二叉树的转换及哈夫曼树。

第 7 章：主要介绍非线性数据结构——图。首先介绍图的概念和存储结构，然后介绍图的遍历、最小生成树、拓扑排序、关键路径及最短路径。

第 8 章：主要介绍数据结构的常用技术——查找。首先介绍查找的概念，然后结合具体实例介绍各种查找算法，并给出了完整程序。

第 9 章：主要介绍数据结构的常用技术——排序。首先介绍排序的相关概念，然后介绍各种排序技术，并给出了具体实现算法。

本书由陈锐(高级程序员)担任主编，空军航空大学的扶晓、北京联合大学的刘琨、郑春霞担任副主编，大连外国语学院的李绍华、河南工业大学的沈献念、郑州科技学院的汪东芳和张思卿、重庆科创职业学院的李学国、西安文理学院的王悦、长安大学的黄美玲参编。全书由陈锐统稿、定稿。

在本书的出版过程中，得到了来自中原工学院的夏敏捷、山东师范大学的李忠、华中

试读结束：需要全本请在线购买： www.ertongbook.com

师范大学汉口分校的刘河、北京信息职业技术学院的李红、渤海船舶职业学院的邢容、湖南娄底职业学院的刘益洪、华北水利水电学院的徐艳杰等老师的支持，在此表示衷心感谢。

由于作者水平有限，书中难免存在一些不足之处，恳请读者批评指正。邮件地址：nwuchenrui@126.com。

编　者

目　录

第 1 章　数据结构概述

数据结构是随着计算机科学的发展而逐步形成的一门学科，目前已成为高等院校计算机类各专业的核心课程之一，同时也是统计类、经济类和工程软件设计类各专业的必修课程，是以后学习计算机软件和算法的重要基础。它主要研究数据在计算机中的存储表示和对数据的处理方法。在学习数据结构课程之前，需要先了解以下几个问题。

(1) 什么是数据结构。

(2) 数据结构的研究范围。

(3) 数据的存储方式。

(4) 算法的描述工具。

(5) 算法性能的评价。

本章将对这些问题及数据结构的相关概念进行介绍，这些内容是今后学习数据结构的理论基础，将贯穿于数据结构课程的整个学习过程。

1.1　数据结构的发展概况

在计算机刚诞生时，计算机所能处理的数据只能是由 0 或 1 组成的二进制数，那时还谈不上什么结构。后来虽然计算机可以处理十进制整数和十进制实数，也不过是由 0 到 9 这十个数字组成的序列，或再加上小数点，其数据的结构非常简单，没有研究的必要。

随着计算机技术的飞速发展和数据处理能力的不断提高，到 20 世纪 60 年代中期，各种高级程序设计语言相继出现，所能描述的数据类型逐渐丰富。计算机对信息的处理加工已从单一的数值计算发展到非数值处理，其加工处理的信息也由简单的数值发展到字符、图像、声音等具有复杂结构的数据。数据结构这门学科随着计算机数据的复杂化而产生并发展起来了。

数据结构作为一门课程的形成和发展主要是在 20 世纪 60 年代后期。首先，在 1968 年由美国计算机协会(ACM)颁发了建议性的计算机教学计划，其中规定数据结构作为一门独立的课程，这在世界上是第一次。同一年，美国计算机科学家 D.E.Knuth 教授在他的巨著《计算机程序设计的技巧》中详细论述了数据的逻辑结构和数据的存储结构，并对各种结构给出了典型算法，为数据结构作为一门课程奠定了理论基础。

20 世纪 70 年代初，随着大型程序以及大规模文件系统的出现，结构化程序设计成为程序设计方法学的主要内容。在程序设计中，数据结构受到了极大的重视。认为程序设计的实质就是对要处理的问题选择一种最为适合的数据结构，同时在此结构上施加一种好的算法。1976 年瑞士著名计算机科学家 N.Wirth 教授曾提出这样一个等式：算法+数据结构=程序，这个等式形象地描述了算法、数据结构和程序之间的关系。

从 20 世纪 80 年代开始，在我国高等院校的教学计划中已经将数据结构课程列为计算机类各专业的核心课程之一，在许多非计算机专业也把数据结构作为必修课或选修课程。

数据结构是一门介于数学、计算机硬件和计算机软件三者之间的计算机专业基础课，是程序设计方法学、数据库系统、操作系统、编译原理、软件工程学等课程的先修课程，是设计和实现大型应用软件的基础。

1.2 数据结构的基本概念

为了更好地理解数据结构这个概念，下面首先解释一下数据结构中的一些基本概念。

数据(Data)是能输入计算机中并能被计算机处理的符号的总称，是计算机程序的加工“原料”。需要说明的是，这里的“数据”绝对不能狭义地理解为仅仅是数学中的整数或实数，应作广义的理解。例如，一个有某种功能的源程序，一个文件，一张图画，一段声音等都可以通过编码而归纳为数据的范畴。

数据元素(Data Element)是数据的基本单位，在计算机中通常作为一个整体进行处理。由于数据的范围非常广泛，因此数据元素可大可小，小到一个字符；大到一本书或一张地图等。对于较大的数据元素还可进一步分成若干个“数据项”。

例如，表 1.1 是一张学生成绩表，一条学生成绩记录(201101001，孙菲，女，85，94，83)就是一个数据元素，其中，“201101001”又是该数据元素的数据项。

表 1.1 学生成绩表

学 号	姓 名	性 别	语 文	数 学	英 语
201101001	孙菲	女	85	94	83
201101002	李明	男	88	71	70
201101003	张艳	女	93	80	88

数据对象(Data Object)是具有相同性质的数据元素的集合，是数据的一个子集。例如，全体整数的集合 $\mathbf{Z}=\{0,\pm1,\pm2,\cdots\}$就是一个数据对象，全体复数的集合 $\mathbf{C}=\{<x,y>|x,y\in\mathbf{R}\}$也是一个数据对象。

数据结构(Data Structure)是数据元素之间存在的一种或多种特定关系的数据元素的集合。数据结构有逻辑上的数据结构和物理上的数据结构之分。

数据的逻辑结构是指数据元素之间的逻辑关系。常见的逻辑结构有集合结构、线性结构、树形结构和图结构。

(1) 集合结构(Set)：在该逻辑结构中，只有数据元素集 D，而关系集为空集合，即 $R=\{\}$。也就是说，结构中的数据元素除了同属于一个集合外，数据元素之间没有其他关系，如图 1.1(a)所示。

(2) 线性结构(Line Structure)：在该结构中，除了第一个元素(记录)外，其他各元素(记录)都有唯一的前驱；除了最后一个元素(记录)外，其他各元素(记录)都有唯一的后继。数据中各元素之间存在一对一的关系。在图 1.1(b)中，A 是 B 的前驱，B 是 A 的后继。

(3) 树形结构(Tree Structure)：在该结构中，除了一个根元素(结点)外，其他各元素(结点)都有唯一的前驱；所有数据元素(结点)都可以有多个后继。数据中各元素之间存在一对多的关系，如图 1.1(c)所示。树形结构也称为层次结构。这就像学校的组织结构图，学校下面是教学的院系、行政机构的部和处及一些研究所。

束：需要全本请在线购买： www.ertongbook.com

(4) 图结构或网结构(Graph Structure)：在该结构中，各元素(顶点)之间可以有多个前驱和多个后继。数据中各元素之间存在多对多的关系，如图 1.1(d)所示。图结构和树形结构统称为非线性结构。城市之间的交通路线图就是多对多的关系，可将 *A*、*B*、*C*、*D* 看作是 4 个城市，城市 *A* 和城市 *B*、*D*、*E* 都存在一条直达路线，而城市 *B* 也可以和 *A*、*C*、*E* 存在一条直达路线。

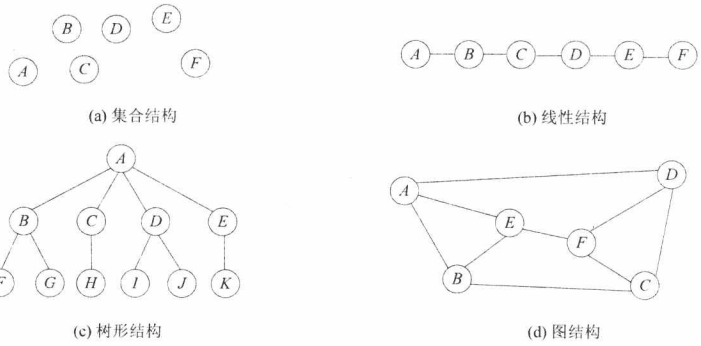

图 1.1　数据的逻辑结构

在现实生活中，学校的组织结构就是一种树形结构，城市之间的交通路线是一种图结构。

数据的物理结构(Data Physical Structure)又称数据的存储结构，是数据在计算机中的存储表示，它包括数据本身在计算机中的存储方式，以及数据之间的逻辑关系在计算机中的表示。因此，数据的物理结构是依赖于计算机的。

根据在存储器中表示数据的不同方法，通常有以下 4 种物理存储结构。

(1) 顺序存储结构(Sequential Storage Structure)：将逻辑上相邻的数据元素存储在物理位置上相邻的存储单元中，其数据元素间的逻辑关系和物理关系是一致的。顺序存储结构如图 1.2 所示。

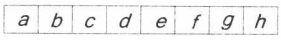

图 1.2　顺序存储结构

(2) 链式存储结构(Linked Storage Structure)：在存储器中不要求将逻辑上相邻的数据元素存储到相邻的物理存储位置中，而是通过附加指针字段来表示数据元素之间的逻辑关系。链式存储结构如图 1.3 所示。

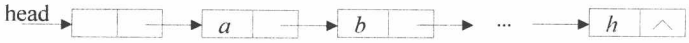

图 1.3　链式存储结构

(3) 索引存储结构(Index Storage Structure)：在存储数据元素信息的同时，还建立一个附加的索引表(比如字典中的索引表)，索引表中每一项称为索引项(关键字)，它是能够唯一标识一个数据元素的数据项。

(4) 哈希存储结构(Hash Storage Structure)：能根据数据元素的关键字直接算出该数据元素的存储地址。

数据类型是具有相同性质的计算机数据的集合及定义在这个数据集合上的一组操作的总称。例如，在C语言中，整数类型是集合 **Z**={0,±1,±2,±3,…}及定义在该集合上的加、减、乘、除等一组操作。

高级程序设计语言中的数据类型可分为简单类型和结构类型。简单类型是不可分的，如整型、实型等。结构类型是由若干个类型组合而成的，是可以再分解的。如C语言中的数组是一种结构类型，它由固定个数的同一类型的数据顺序排列而成；结构体也是一种结构类型，它是由固定个数的不同类型的数据顺序排列而成。例如：

```
typedef struct
{
    char name[20];
    int age;
    float score;
}STUDENT;
STUDENT stu1,stu2;
```

抽象数据类型(Abstract Data Type，ADT)是一个数学模型及定义在该模型上的一组操作。抽象数据类型描述的是一组逻辑上的特性，与在计算机内部如何表示和实现无关。不论内部结构如何变化，只要它的数学特性保持不变，都不会影响其外部使用。因此，抽象数据类型可实现信息隐藏和数据封装。从抽象数据类型的角度入手设计软件系统可大大提高软件构件的复用性。

抽象数据类型可以用数据集合和基本操作集合来描述。其中，数据对象集合定义了栈的数据元素及元素之间的关系，基本操作集合定义了在该数据对象上的一些基本操作。数据对象和数据关系的定义采用数学符号和自然语言描述，基本操作的定义格式为

基本操作名(参数表)：初始条件和操作结果描述

例如，栈的抽象数据类型描述如下。

1) 数据对象集合

栈的数据对象集合为$\{a_1, a_2, \cdots, a_n\}$，每个元素的类型均为DataType。栈是一种线性表，具有线性表的特点：除了第一个元素a_1外，每一个元素有且只有一个直接前驱元素，除了最后一个元素a_n外，每一个元素有且只有一个直接后继元素。数据元素之间的关系是一对一的关系。

2) 基本操作集合

栈的基本操作主要有如下几个。

(1) InitStack(&S)：初始化操作，建立一个空栈S。

初始条件：栈S不存在。

操作结果：构造了一个空栈S。

(2) StackEmpty(S)：判断栈是否为空。

初始条件：栈S已存在。

操作结果：如果栈为空，返回1；否则，返回0。

(3) GetTop(S,&e)：取栈顶元素。

初始条件：栈S存在且非空。

操作结果：返回栈S的栈顶元素给e。

(4) PushStack(&S,x)：入栈。

初始条件：栈 S 已存在。

操作结果：在栈 S 的顶部插入一个新元素 x，x 成为新的栈顶元素。

(5) PopStack(&S,&e)：出栈。

初始条件：栈 S 存在且非空。

操作结果：删除栈 S 的顶部元素。

注意：在算法的描述中，参数传递可以分为两种：一种是数值传递，另外一种是引用传递。前者仅仅是将数值传递给形参，而不返回结果；后者其实是把实参的地址传递给形参，可通过被调用函数修改后的变量值带回调用函数。通过在参数前加上&，表示引用传递，如果参数前没有&，表示数值传递。

1.3 算法的描述与算法的分析

数据结构与算法之间存在着本质的联系，在数据类型建立起来之后，就要对这些数据施加运算，从而建立起运算的集合，即程序。程序运行效率的高低直接取决于算法的好坏。

1.3.1 算法的定义与特性

算法是描述求解特定问题而规定的一系列操作步骤集合。要求解的问题可以是数值的，也可以是非数值的。解决数值问题的算法叫做数值算法，科学与工程计算方面的算法都属于数值算法，如求解数值积分，求解线性方程组，求解微分方程等；解决非数值问题的算法叫做非数值算法，数据处理方面的算法都属于非数值算法。例如，各种查找算法、排序算法、遍历算法等。

一个算法必须满足以下 5 个特性。

(1) 有穷性。一个算法应该包含有限个操作步骤，而不是无限个。对于合法的输入，算法能在执行有限次操作之后得到结果。有限次操作是指操作步骤为有限个，且每个步骤都能在规定的时间内完成。

(2) 确定性。算法的每个步骤都具有确定的含义，不会出现二义性。在一定条件下，算法只有唯一的一条执行路径，也就是说对于相同的输入只能得出相同的输出结果。

(3) 可行性。算法的每一个操作都可以通过已经实现的基本操作在规定的时间内执行有限次来实现。

(4) 有零个或多个输入。所谓输入是指在执行算法时需要从外界取得必要的信息或数据。一个算法可以没有输入也可以有有限个输入，这些输入应来源于某个特定的数据对象。

(5) 输出。算法的目的是为了求解，“解”就是输出结果。一个算法应该有一个或多个输出。输出的形式是多样的，算法的结果可以输出到文件中，也可以打印或显示输出，还可以返回一个或多个值。没有输出的算法是无意义的。

1.3.2 算法设计的要求

设计算法时，要考虑让算法实现以下目标。

1. 算法的正确性

算法的正确性是指算法应该满足具体问题的需求。其中，“正确”的含义大体上可以分为以下 4 个层次：

(1) 程序没有语法错误；

(2) 程序对于几组输入数据能够得到满足要求的结果；

(3) 程序对于精心选择的、典型、苛刻且带有刁难性的几组输入数据能得出满足要求的结果；

(4) 程序对于一切合法的输入都能得到满足要求的结果。

对于这 4 层含义，达到层次(4)是极为困难的，一般情况下，我们把层次(3)作为衡量一个算法是否合格的标准。

2. 可读性

一个好的算法首先应该便于人们阅读、理解和交流，其次才是计算机执行。可读性好的算法有助于人们对算法的理解，晦涩难懂的算法往往会使隐含的错误不易被发现，并且难以调试和修改。

3. 健壮性

当输入数据不合法时，算法应当恰当地作出相应处理，而不是产生异常或莫名其妙的结果。并且，处理出错的方法不应是中断程序的执行，而应是返回一个表示错误或错误性质的值，以便在更高的抽象层次上进行处理。

4. 高效率和低存储量

算法的效率通常指的是算法的执行时间。对于一个具体问题的解决通常可以有多个算法，执行时间短的算法效率高，执行时间长的效率低。存储量需求指的是算法在执行过程中需要的最大存储空间。设计算法应尽量选择高效率和低存储量需求的算法。

1.3.3 算法的描述

算法可以采用多种方式描述，常见的描述方式有：自然语言描述、程序流程图和程序设计语言。

1. 采用自然语言描述

自然语言描述是指使用自然语言描述问题的求解过程。下面举例说明。

问题：判断正整数 N 是否是质数。

使用自然语言描述的算法如下。

Step1：令 i=2；

Step2：判断 i 是否小于等于 $N/2$，若是，则转到 Step4；否则，转到 Step3；

Step3：判断 N 除以 i 的余数 r 是否为 0，若 r 等于 0，则转到 Step5；否则，i 加 1，转到 Step2；

Step4：输出“N 为质数”；

Step5：算法结束。

2. 采用程序流程图描述

采用流程图的形式描述“判断正整数 N 是否是质数”的算法如图 1.4 所示。

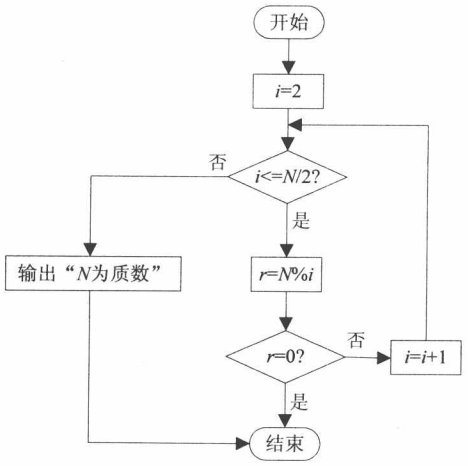

图 1.4　程序流程图

3. 采用程序设计语言描述

以 C 语言为例描述“判断正整数 N 是否是质数”的算法如下：

```
void IsPrime(int N)
{
    int i;
    for(i=2;i<=N/2;i++)
        if(N%i==0)
            break;
    if(i>N/2)
    printf("N是质数.");
}
```

除了以上 3 种形式外，还可以用类语言(如类 C 语言、类 Pascal 语言)描述问题的求解过程。自然语言描述可以是汉语或英语等文字描述；伪代码形式类似于程序设计语言形式，但是不能直接运行；程序流程图的优点是直观，但是不易直接转化为可运行的程序；程序设计语言形式采用像 C、C++、Java 等语言描述，可以直接在计算机上运行。

不管使用哪种形式描述算法，都必须能正确描述求解过程。本书中的所有算法都采用 C 语言描述。

1.3.4 算法分析

对于同一个问题可以构造不同的算法，那么，在众多的算法中该选取哪个算法呢？这就是如何评价一个算法好坏的问题。一个好的算法除了必须满足它的正确性等基本的设计要求外，还有一个指标就是它的效率。算法效率包括时间与空间两个方面，分别称为时间复杂度与空间复杂度。算法分析的目的是根据实际问题，从多种算法中选取一个最为适合的算法并从时间效率和空间效率两方面给出评价。

1. 算法的时间复杂度

衡量一个算法在计算机上的执行时间有事后统计和事前分析估算两种方法。

1) 事后统计方法

事后统计方法主要是通过设计好的测试程序和数据，利用计算机的计时器对不同算法编制好的程序比较各自的运行时间，从而确定算法效率的好坏。但是，这种方法有 3 个缺陷：

(1) 必须事先编制好程序，这通常需要花费大量的时间与精力；

(2) 依赖计算机硬件和软件等环境因素，有时会掩盖算法本身的优劣；

(3) 算法的测试数据设计困难，并且程序的运行时间往往还与测试数据的规模有很大的关系，效率高的算法在小的测试数据面前往往得不到体现。

2) 事前分析估算方法

事前分析估算方法主要是在编制计算机程序之前，对算法依据数学中的统计方法进行估算。算法的程序在计算机上的运行时间取决于以下因素：

- 算法采用的策略；
- 编译程序产生的机器代码质量；
- 问题的规模；
- 书写的程序语言。对于同一个算法，语言级别越高，执行效率越低；
- 机器执行指令的速度。

在以上 5 个因素中，算法采用不同的策略、不同的编译系统、不同的语言实现，在不同的机器上运行时，其效率均不同，所以，使用绝对时间单位衡量算法效率不合适。抛开以上因素，一个特定的算法运行效率依赖于问题的规模。

任何一个算法都是由控制结构(顺序、分支和循环结构)和基本语句(赋值语句、声明语句和输入/输出语句)构成的。一般情况下，算法的运行时间取决于两者执行时间的总和，可以将语句的执行次数作为算法效率高低的度量标准。语句的重复执行次数称为语句频度。

【例 1.1】 两个 n 阶矩阵相乘。

	该算法的语句频度
`for(i=0;i<n;i++)`	n
`  for(j=0;j<n;j++)`	n^2
`  {`	
`    c[i][j]=0;`	n^2
`    For(k=0;k<n;k++)`	n^3
`      c[i][j]=c[i][j]+a[i][k]*b[k][j];`	n^3
`  }`	

束：需要全本请在线购买： www.ertongbook.com