

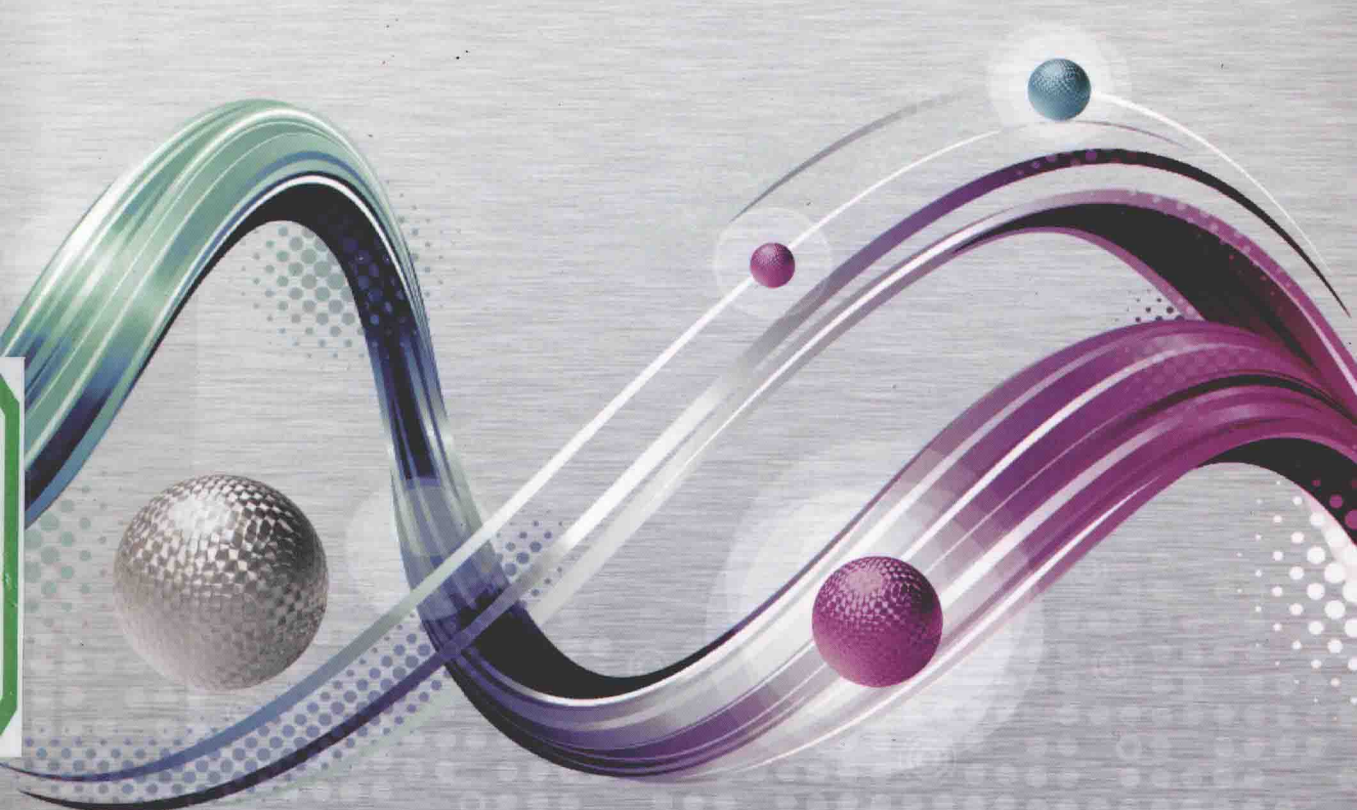


高等学校电子信息类“十三五”规划教材  
应用型网络与信息安全工程技术人才培养系列教材

# 数据结构

*Data Structure*

主 编 王祖俪  
副主编 王 翔 蔺 冰 吴春旺



西安电子科技大学出版社  
<http://www.xduph.com>

高等学校电子信息类“十三五”规划教材  
应用型网络与信息安全工程技术人才培养系列教材

# 数 据 结 构

主 编 王祖俪

副主编 王 翔 蔺 冰 吴春旺

西安电子科技大学出版社

## 内 容 简 介

本书以 C 语言为程序设计语言,通过算法、代码、流程图等多种表现形式详细介绍了数据结构的基本概念、逻辑特性和物理特性,对各种结构定义了相应的抽象数据类型,并给出了应用实例。在各章节末尾,还提供了习题供读者练习。

本书可作为高等院校计算机及相关专业数据结构课程的教材,也可作为计算机应用开发人员的参考书。

## 图书在版编目(CIP)数据

数据结构/王祖俪主编. —西安:西安电子科技大学出版社, 2016.7

高等学校电子信息类 “十三五”规划教材

ISBN 978-7-5606-4119-5

I. ① 数… II. ① 王… III. ① 数据结构—高等学校—教材 IV. ① TP311.12

中国版本图书馆 CIP 数据核字(2016)第 139052 号

策 划 李惠萍

责任编辑 李惠萍 宁晓蓉

出版发行 西安电子科技大学出版社(西安市太白南路 2 号)

电 话 (029)88242885 88201467 邮 编 710071

网 址 [www.xduph.com](http://www.xduph.com) 电子邮箱 [xdupfxb001@163.com](mailto:xdupfxb001@163.com)

经 销 新华书店

印刷单位 陕西华沐印刷科技有限责任公司

版 次 2016 年 7 月第 1 版 2016 年 7 月第 1 次印刷

开 本 787 毫米×1092 毫米 1/16 印 张 20

字 数 468 千字

印 数 1~3000 册

定 价 36.00 元

ISBN 978 - 7 - 5606 - 4119 - 5/TP

**XDUP 4411001-1**

\*\*\*如有印装问题可调换\*\*\*

# 序

进入 21 世纪以来,信息技术迅速改变着人们传统的生产和生活方式,社会的信息化已经成为当今世界发展不可逆转的趋势和潮流。信息作为一种重要的战略资源,与物资、能源、人力一起已被视为现代社会生产力的主要因素。目前,世界各国围绕着信息的获取、利用和控制的国际竞争日趋激烈,网络与信息安全问题已成为一个世纪性、全球性的课题。党的十八大报告明确指出,要“高度关注海洋、太空、网络空间安全”。党的十八届三中全会决定设立国家安全委员会,成立中央网络安全和信息化领导小组,并把网络与信息安全列入了国家发展的最高战略方向之一。这为包含网络空间安全在内的非传统安全领域问题的有效治理提供了重要的体制机制保障,是我国国家安全体制机制的一个重大创新性举措,彰显了我国政府治国理政的战略新思维和“大安全观”。

人才资源是确保我国网络与信息安全第一位的资源,信息安全人才培养是国家信息安全保障体系建设的基础和必备条件。随着我国信息化和信息安全产业的快速发展,社会对信息安全人才的需求不断增加。2015 年 6 月 11 日,国务院学位委员会和教育部联合发出“学位[2015]11 号”通知,决定在“工学”门类下增设“网络空间安全”一级学科,代码为 0839,授予工学学位。这是国家推进专业化教育,在信息安全领域掌握自主权、抢占先机的重要举措。

建国以来,我国高等工科院校一直是培养各类高级应用型专门人才的主力。培养网络与信息安全高级应用型专门人才,高等院校同样责无旁贷。目前,许多高等院校和科研院所已经开办了信息安全专业或开设了相关课程。作为国家首批 61 所“卓越工程师教育培养计划”试点院校之一,成都信息工程大学以《国家中长期教育改革和发展规划纲要(2010—2020 年)》《国家中长期人才发展规划纲要(2010—2020 年)》、《卓越工程师教育培养计划通用标准》为指导,以专业建设和工程技术为主线,始终贯彻“面向工业界、面向未来、面向世界”的工程教育理念,按照“育人为本、崇尚应用”“一切为了学生”的教学教育理念和“夯实基础、强化实践、注重创新、突出特色”的人才培养思路,遵循“行业指导、校企合作、分类实施、形式多样”的原则,实施

了一系列教育教学改革。令人欣喜的是，该校信息安全工程学院与西安电子科技大学出版社近期联合组织了一系列网络与信息安全专业教育教学改革的研讨活动，共同研讨培养应用型高级网络与信息安全工程技术人才的教育教学方法和课程体系，并在总结近年来该校信息安全专业实施“卓越工程师教育培养计划”教育教学改革成果和经验的基础上，组织编写了“应用型网络与信息安全工程技术人才培养系列教材”。该套教材总结了该校信息安全专业教育教学改革成果和经验，相关课程有配套的课程过程化考核系统，是培养应用型网络与信息安全工程技术人才的一套比较完整、实用的教材，相信可以对我国高等院校网络与信息安全专业的建设起到很好的促进作用。该套教材为中国电子教育学会高教分会推荐教材。

信息安全是相对的，信息安全领域的对抗永无止境。国家对信息安全人才的需求是长期的、旺盛的。衷心希望该套教材在培养我国合格的应用型网络与信息安全工程技术人才的过程中取得成功并不断完善，为我国信息安全事业做出自己的贡献。

高等学校电子信息类“十三五”规划教材

应用型网络与信息安全工程技术人才培养系列教材

名誉主编（中国密码学会常务理事）

何大可

二〇一五年十二月

## 中国电子教育学会高教分会推荐

高等学校电子信息类“十三五”规划教材  
应用型网络与信息安全工程技术人才培养系列教材

### 编审专家委员会名单

名誉主任: 何大可(中国密码学会常务理事)

主 任: 张仕斌(成都信息工程大学信息安全学院副院长、教授)

副 主 任: 李 飞(成都信息工程大学信息安全学院院长、教授)

何明星(西华大学计算机与软件工程学院院长、教授)

苗 放(成都大学计算机学院院长、教授)

赵 刚(西南石油大学计算机学院院长、教授)

李成大(成都工业学院教务处处长、教授)

宋文强(重庆邮电大学移通学院计算机科学系主任、教授)

梁金明(四川理工学院计算机学院副院长、教授)

易 勇(四川大学锦江学院计算机学院副院长、成都大学计算机学院教授)

杨瑞良(成都东软学院计算机科学与技术系主任、教授)

编审专家委员: (排名不分先后)

|     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|
| 范太华 | 叶安胜 | 黄晓芳 | 黎忠文 | 张 洪 | 张 蕾 | 贾 浩 |
| 赵 攀 | 陈 雁 | 韩 斌 | 李享梅 | 曾令明 | 何林波 | 盛志伟 |
| 林宏刚 | 王海春 | 索 望 | 吴春旺 | 韩桂华 | 赵 军 | 陈 丁 |
| 秦 智 | 王中科 | 林春蕾 | 张金全 | 王祖俪 | 蔺 冰 | 王 敏 |
| 万武南 | 甘 刚 | 王 燚 | 闫丽丽 | 昌 燕 | 黄源源 | 张仕斌 |
| 李 飞 | 王海春 | 何明星 | 苗 放 | 李成大 | 宋文强 | 梁金明 |
| 万国根 | 易 勇 | 杨瑞良 |     |     |     |     |

## 前 言

数据结构是计算机及相关专业的一门专业基础课程，也是重要的核心课程之一。通过数据结构课程的学习，学生应掌握各种基本数据结构的类型、存储方式以及它们的基本操作，同时该课程也培养学生利用所学的概念和理论知识，针对实际问题找到简洁适当的数据结构、存储方式和方法的能力。掌握数据结构知识是对程序设计人员最重要也是最基本的要求。

数据结构一直给人复杂、难懂的印象，初学者往往对基本结构的操作或者实际问题的处理思路不明确，不知如何下手，导致望而却步。为了减少读者学习数据结构时的畏难情绪，本书用尽量浅显易懂的语言描述基本的概念和基础知识，除了用传统的算法描述基本操作以外，对每个章节的重点操作都引入了 C 语言的代码描述，读者可以通过直接运行代码查看算法运行的结果，了解操作的过程。本书在介绍理论知识的同时，还引入了典型的实际问题，帮助读者了解具体问题如何用所学的理论知识来解决。算法、代码、流程等多种形式的表现，都旨在提高初学者的程序分析和设计能力。每章节后还附有习题，以便读者进一步练习并检验学习效果。

本书由长期从事数据结构课程教学的老师在总结了多年的教学实践经验的基础上编写而成，旨在提高学生的实践动手能力和理论联系实际的能力。本书根据常规课时限定，删去了部分非本科学生必须掌握的内容，并增加了实际应用内容，教师可在实际教学中进行调整。

本书可作为高等学校计算机及相关专业数据结构课程的教材，也可以作为计算机应用开发人员的参考书。对于计算机及相关专业，可讲授 52~64 学时，另进行 12 学时左右的上机练习；对于其他专业，课时可适当压缩，讲授 32~48 学时。

本书第 1、8 章由蔺冰编写，第 2、4 章由王祖俪编写，第 3 章由吴春旺编写，第 5、6、7 章由王翔编写，王祖俪负责校阅各章，并对全书统稿、定稿。

由于作者水平有限，加上计算机科学技术发展迅速，书中难免有不妥和遗漏之处，恳请广大读者赐教。

编 者

2016 年 3 月

# 目 录

|                          |    |
|--------------------------|----|
| 第 1 章 绪论.....            | 1  |
| 1.1 什么是数据结构.....         | 1  |
| 1.1.1 数据结构基本概念.....      | 1  |
| 1.1.2 数据结构图形表示.....      | 2  |
| 1.2 什么是算法.....           | 3  |
| 1.2.1 算法概念.....          | 3  |
| 1.2.2 算法设计要求.....        | 3  |
| 1.2.3 算法复杂度.....         | 4  |
| 1.3 C 语言要点回顾.....        | 7  |
| 1.3.1 基本数据类型.....        | 7  |
| 1.3.2 其他复合数据类型.....      | 8  |
| 1.3.3 指针数据类型.....        | 10 |
| 1.3.4 常用结构及函数.....       | 12 |
| 习题.....                  | 15 |
| 第 2 章 线性表.....           | 17 |
| 2.1 线性表的逻辑结构.....        | 17 |
| 2.1.1 线性表的概念.....        | 17 |
| 2.1.2 线性表的抽象数据类型.....    | 18 |
| 2.2 线性表的顺序结构及基本运算实现..... | 19 |
| 2.2.1 线性表的顺序表示.....      | 19 |
| 2.2.2 顺序表的基本运算.....      | 20 |
| 2.3 线性表的链式结构及基本运算实现..... | 32 |
| 2.3.1 单链表的表示.....        | 32 |
| 2.3.2 单链表的操作实现.....      | 34 |
| 2.3.3 单循环链表.....         | 44 |
| 2.3.4 双向链表.....          | 48 |
| 2.3.5 静态链表.....          | 49 |
| 2.4 线性表综合运用.....         | 52 |
| 2.4.1 一元多项式的加减法.....     | 52 |
| 2.4.2 约瑟夫环.....          | 55 |
| 习题.....                  | 58 |

|                            |     |
|----------------------------|-----|
| 第3章 栈和队列 .....             | 59  |
| 3.1 栈的基本概念 .....           | 59  |
| 3.1.1 栈的定义 .....           | 59  |
| 3.1.2 栈的抽象数据类型 .....       | 60  |
| 3.2 栈的表示与实现 .....          | 60  |
| 3.2.1 栈的顺序表示 .....         | 60  |
| 3.2.2 栈的链式表示 .....         | 63  |
| 3.3 栈的应用 .....             | 65  |
| 3.3.1 整数的数制转换 .....        | 65  |
| 3.3.2 判断字符串是否为回文 .....     | 69  |
| 3.4 队列 .....               | 70  |
| 3.4.1 队列的基本定义与抽象数据类型 ..... | 70  |
| 3.4.2 链队列 .....            | 72  |
| 3.4.3 循环队列 .....           | 80  |
| 习题 .....                   | 87  |
| 第4章 数组 .....               | 88  |
| 4.1 数组的概念 .....            | 88  |
| 4.2 数组的顺序存储 .....          | 89  |
| 4.3 矩阵的压缩存储 .....          | 89  |
| 4.3.1 对称矩阵 .....           | 90  |
| 4.3.2 稀疏矩阵 .....           | 91  |
| 习题 .....                   | 100 |
| 第5章 树 .....                | 101 |
| 5.1 树的相关基本概念 .....         | 101 |
| 5.1.1 树的定义与基本术语 .....      | 101 |
| 5.1.2 树的抽象数据类型定义 .....     | 103 |
| 5.1.3 树的存储结构表示 .....       | 105 |
| 5.2 二叉树 .....              | 109 |
| 5.2.1 二叉树的定义 .....         | 109 |
| 5.2.2 二叉树的性质 .....         | 112 |
| 5.2.3 二叉树的存储结构 .....       | 114 |
| 5.3 二叉树常用操作 .....          | 117 |
| 5.3.1 二叉链表结构下的常用操作 .....   | 117 |
| 5.3.2 顺序存储结构下的常用操作 .....   | 139 |
| 5.3.3 反推二叉树结构 .....        | 150 |
| 5.4 线索二叉树 .....            | 151 |
| 5.4.1 线索二叉树原理 .....        | 151 |

|            |                        |            |
|------------|------------------------|------------|
| 5.4.2      | 线索二叉树的结构实现 .....       | 152        |
| 5.4.3      | 线索二叉树的遍历 .....         | 154        |
| 5.5        | 树、森林和二叉树的转换 .....      | 156        |
| 5.5.1      | 树转换为二叉树 .....          | 157        |
| 5.5.2      | 森林转换为二叉树 .....         | 157        |
| 5.5.3      | 二叉树转换为树 .....          | 158        |
| 5.5.4      | 二叉树转换为森林 .....         | 158        |
| 5.5.5      | 树和森林的遍历 .....          | 159        |
| 5.6        | 哈夫曼树及其应用 .....         | 159        |
| 5.6.1      | 最优二叉树(哈夫曼树)的定义 .....   | 159        |
| 5.6.2      | 最优二叉树(哈夫曼树)的应用 .....   | 160        |
| 5.6.3      | 最优二叉树(哈夫曼树)的创建 .....   | 163        |
| 习题         | .....                  | 171        |
| <b>第6章</b> | <b>图 .....</b>         | <b>174</b> |
| 6.1        | 图的基本概念 .....           | 174        |
| 6.1.1      | 图的定义和术语 .....          | 174        |
| 6.1.2      | 图的抽象数据类型定义 .....       | 179        |
| 6.2        | 图的存储结构 .....           | 181        |
| 6.2.1      | 邻接矩阵表示法 .....          | 181        |
| 6.2.2      | 邻接表/逆邻接表表示法 .....      | 186        |
| 6.2.3      | 十字链表表示法 .....          | 190        |
| 6.3        | 图的遍历 .....             | 192        |
| 6.3.1      | 深度优先遍历 .....           | 193        |
| 6.3.2      | 广度优先遍历 .....           | 199        |
| 6.4        | 最小生成树 .....            | 202        |
| 6.4.1      | 普里姆(Prim)算法 .....      | 203        |
| 6.4.2      | 克鲁斯卡尔(Kruskal)算法 ..... | 209        |
| 6.5        | 有向无环图及其应用 .....        | 215        |
| 6.5.1      | 拓扑排序问题 .....           | 215        |
| 6.5.2      | 关键路径问题 .....           | 219        |
| 6.6        | 最短路径 .....             | 224        |
| 习题         | .....                  | 230        |
| <b>第7章</b> | <b>查找 .....</b>        | <b>233</b> |
| 7.1        | 查找表及其相关概念 .....        | 233        |
| 7.2        | 顺序表的查找 .....           | 235        |
| 7.3        | 有序表的查找 .....           | 237        |
| 7.4        | 索引表的查找 .....           | 243        |

|                         |            |
|-------------------------|------------|
| 7.5 二叉排序树 .....         | 245        |
| 7.5.1 二叉排序树的查找 .....    | 246        |
| 7.5.2 二叉排序树的插入和创建 ..... | 248        |
| 7.5.3 二叉排序树的删除 .....    | 250        |
| 7.5.4 二叉排序树的总结 .....    | 257        |
| 7.6 平衡二叉树 .....         | 257        |
| 7.6.1 平衡二叉树实现原理 .....   | 259        |
| 7.6.2 平衡二叉树的实现代码 .....  | 264        |
| 7.7 哈希查找 .....          | 271        |
| 7.7.1 哈希查找概述 .....      | 271        |
| 7.7.2 哈希函数的构造方法 .....   | 273        |
| 7.7.3 处理哈希冲突的方法 .....   | 275        |
| 7.7.4 哈希查找的性能分析 .....   | 279        |
| 习题 .....                | 280        |
| <b>第8章 排序</b> .....     | <b>282</b> |
| 8.1 排序概述 .....          | 282        |
| 8.2 插入排序 .....          | 282        |
| 8.2.1 直接插入排序 .....      | 282        |
| 8.2.2 希尔排序 .....        | 284        |
| 8.3 交换排序 .....          | 286        |
| 8.3.1 冒泡排序 .....        | 286        |
| 8.3.2 快速排序 .....        | 288        |
| 8.4 选择排序 .....          | 290        |
| 8.4.1 简单选择排序 .....      | 290        |
| 8.4.2 堆排序 .....         | 292        |
| 8.5 归并排序 .....          | 297        |
| 8.6 基数排序 .....          | 299        |
| 8.7 排序方法的总结 .....       | 306        |
| 习题 .....                | 306        |

# 第1章 绪 论

随着计算机技术的迅速发展,计算机的应用已不再局限于数学科学运算,而是更多地用于控制、管理和数据处理等非数值计算的处理工作。与此相对应,计算机处理的对象由纯粹的数值发展为字符、表格、图像等各种具有一定结构的数据,这就给程序设计带来一些新的问题。有效地分析和处理对象的特性以及对象之间存在的关系,就是数据结构学科形成和发展的背景。

~~~~~

## 1.1 什么是数据结构

### 1.1.1 数据结构基本概念

数据结构(Data Structure)是一门研究非数值计算的程序设计问题中计算机的操作对象以及它们之间的关系和操作的学科。

在 Windows 操作系统中,在查找自己较长时间以前编辑过的文件时,通常会用到 Windows 自带的搜索功能。Windows 文件夹是一个树形结构,如图 1-1 所示,搜索即是在这棵树里查找是否有同名文件或文件夹的过程。

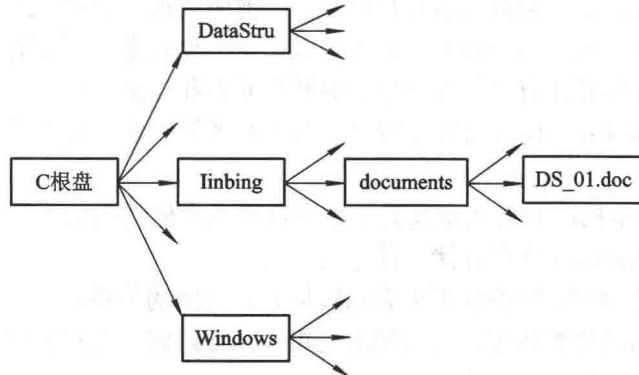


图 1-1 Windows 文件搜索树

描述非数值计算问题的数学模型不再是数学方程式,而是线性表、树、图类的数据结构。数据结构在计算机科学中是一门综合性的专业基础课,不仅涉及计算机硬件(特别是编码理论等)的研究,还和计算机软件的研究密切相关。在研究解决问题时,必须考虑数据如

何组织, 如何查找和存取较为方便等。因此, 可以认为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核心课程。

在此, 先了解相关概念和术语的定义。

(1) 数据(Data): 信息的载体, 对客观事物的符号表示, 是所有被计算机程序处理的符号的总称(如数字、字符串、图像、声音等)。

(2) 数据元素(Data Element): 数据的基本单位, 可由若干个数据项组成, 数据项是数据的不可分割的最小单位。数据元素可以是单一数据, 也可以是一条记录。

(3) 数据对象(Data Item): 性质相同的数据元素的集合, 是数据的一个子集(如整数集合、学生信息集合)。

(4) 数据结构(Data Structure): 相互之间存在某种特定关系的数据元素的集合。其中, 结构是指数据元素之间的关系。

### 1.1.2 数据结构图形表示

数据结构包括两方面的内容:

(1) 数据的逻辑结构: 数据元素之间的逻辑关系, 也可以看作是从具体问题抽象出来的模型, 描述操作对象之间的关系, 它与数据的存储方式、位置无关。

数据的逻辑结构包括集合结构、线性结构、树形结构、图形(网状)结构, 如图 1-2 所示。

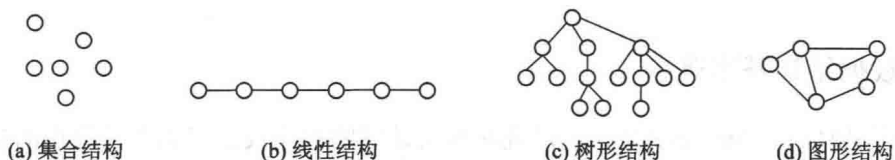


图 1-2 数据的逻辑结构关系图

集合结构: 数据元素仅仅同属于一个集合, 没有其他关系。

线性结构: 数据元素之间存在一对一的关系, 若结构非空, 则有且仅有一个开始数据元素和结束数据元素, 其他数据元素有且仅有一个前驱数据元素和一个后继数据元素。

树形结构: 数据元素之间存在着一对多的关系, 若结构非空, 则有且仅有一个称为根的数据元素, 其他数据元素有且仅有一个前驱数据元素和零至多个后继数据元素。

图形结构(网状结构): 数据元素之间存在着多对多的关系, 所有数据元素之间均有关系或无关系。

(2) 数据的存储结构: 数据元素及其关系在计算机存储器内的表示。数据的存储结构包括顺序存储、链式存储和索引存储三种方式。

顺序存储: 借助存储器中的位置来表示数据元素之间的关系; 链式存储: 对逻辑上相邻的元素不要求其物理位置相邻, 元素间的逻辑关系通过附设的指针字段来表示; 索引存储: 建立附加索引标识数据元素地址。

数据的逻辑结构与存储结构的关系: 存储结构是逻辑关系的映像与元素本身的映像; 逻辑结构是数据结构的抽象, 存储结构是数据结构的实现, 两者综合起来建立了数据元素之间的结构关系。

如在线性结构中其存储结构有顺序存储和链式存储两种方式, 设存放 5 个字符 a、b、

c、d、e 的线性结构，其顺序存储如图 1-3 所示，链式存储如图 1-4 所示。

|   |        |
|---|--------|
| a | 8f3b90 |
| b | 8f3b91 |
| c | 8f3b92 |
| d | 8f3b93 |
| e | 8f3b94 |

图 1-3 线性结构的顺序存储

|        |        |
|--------|--------|
| ...    |        |
| e      | 363bd8 |
| 0      |        |
| ...    |        |
| d      | 363c20 |
| 363bd8 |        |
| ...    |        |
| c      | 363c68 |
| 363c20 |        |
| ...    |        |
| b      | 363cb0 |
| 363c68 |        |
| ...    |        |
| a      | 363cf8 |
| 363cb0 |        |
| ...    |        |

图 1-4 线性结构的链式存储

## 1.2 什么是算法

### 1.2.1 算法概念

算法(Algorithm)是对特定问题求解步骤的一种描述，是指令的有限序列。

算法有五个重要特性：

(1) 有穷性——一个算法必须总是在执行有穷步之后结束，且每一步都在合理可接受的时间内完成。

(2) 确定性——算法中每一条指令必须有确切的含义，读者理解时不会产生二义性，并且在任何条件下，算法只有唯一的执行路径，即对于相同的输入只能得到相同的输出。

(3) 可行性——一个算法是可行的，即算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现的。

(4) 输入——一个算法有零个或多个输入，这些输入取自于某个特定的对象的集合。

(5) 输出——一个算法有一个或多个输出，这些输出是同输入有着某些特定关系的量。

算法的含义和程序十分类似，但也有区别。一个程序可能不需要满足上述第一点(有穷性)，如操作系统，只要整个系统正常运行，则操作系统程序就不会结束。另外，程序是用机器可执行的语言完成的，算法没有这种限制。

### 1.2.2 算法设计要求

一个好的算法应考虑满足以下设计要求：

(1) 正确性——算法应当满足具体问题的需求。

(2) 可读性——算法主要是为了人的阅读与交流，其次才是机器执行。可读性好有助于人的理解；晦涩难懂的程序易于隐藏较多错误，难以调试和修改。

(3) 健壮性——当输入数据非法时，算法也能适当地作出反应或进行处理，而不会产生莫明其妙的输出结果。

(4) 效率与低存储量需求——效率指算法执行的时间。对于同一个问题如果有多个算法可以解决，执行时间短的算法效率高。存储量需求指算法执行过程中所需要的最大存储空间。效率和低存储量需求都与问题的规模有关。

### 1.2.3 算法复杂度

#### 1. 算法效率的度量

算法执行时间需通过依据该算法编制的程序在计算机上运行时所消耗的时间来度量。而度量一个程序的执行时间通常有两种方法：算法的事后统计方法和算法的事前分析估算方法。

##### 1) 事后统计方法

在 C 程序中可以使用如下代码获得系统时间。

**例 1-1** 获取系统时间判断算法的效率。

```
#include <stdio.h>
#include <time.h>
#include <sys/timeb.h>
```

```
int main(void)
```

```
{
```

```
    int i, j;
```

```
    double k = 0;
```

```
    struct timeb tp;
```

```
    struct tm *tm;
```

```
    //在算法处理前获取系统时间：时、分、秒、毫秒
```

```
    ftime(&tp);
```

```
    tm = localtime(&(tp.time));
```

```
    printf("%02d:%02d:%02d:%03d\n", (tm->tm_hour), (tm->tm_min), (tm->tm_sec), (tp.millitm));
```

```
    //算法：模拟延时
```

```
    for (i = 0; i < 10000; i++)
```

```
    {
```

```
        for (j = 0; j < 1000; j++)
```

```
        {
```

```
            k = k + 1.5;
```

```
    }  
}  
  
//在算法处理后再次获取系统时间  
ftime(&tp);  
tm = localtime(&(tp.time));  
printf("%02d:%02d:%02d:%03d\n", (tm->tm_hour), (tm->tm_min), (tm->tm_sec), (tp.millitm));  
  
return 0;  
}
```

运行结果如图 1-5 所示。

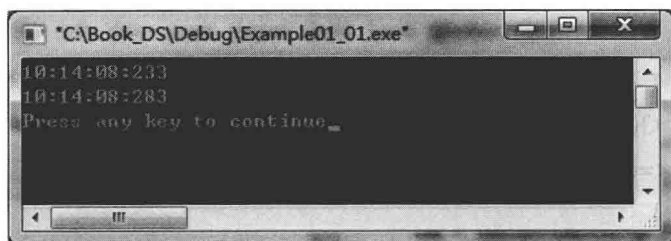


图 1-5 获取系统时间代码运行结果

事后统计方法有缺陷, 统计量依赖于计算机的硬件、软件等环境因素。同样算法的程序在不同的硬件环境或不同的软件编译环境中的运行时间都可能不同, 更不用说不同的算法。只有将不同的算法在相同的软、硬件环境中分别进行统计, 才可能区分出不同算法的优劣。

## 2) 事前分析估算方法

用高级语言编写的算法的执行时间主要取决于下列因素:

- 算法选用的策略。不同思路算法的效率可能不相同。
- 问题的规模。处理的数据量越大, 处理的时间越长。
- 编写程序的语言。语言的级别越高, 执行效率越低。
- 编译程序所产生的目标代码的质量。编译程序优化代码越好, 程序效率越高。
- 机器执行指令的速度。CPU 性能直接影响效率。

程序语言、编译器和机器执行速度不同, 算法的效率也不相同, 这是软、硬件差异造成的, 不在本书中讨论。所以不同算法(即算法策略不同)依赖不同规模表现出不同的效率, 即算法的效率可以表示为规模(通常用整数量  $n$  表示)的函数。

一个算法的时间复杂度  $T(n)$  是该算法的时间开销, 是求解问题规模  $n$  的函数。当  $n$  趋于无穷大时, 时间复杂度  $T(n)$  的数量级称为算法的渐近时间复杂度, 记为

$$T(n) = O(f(n))$$

它表示随问题规模  $n$  的增长, 算法执行时间的增长度和函数  $f(n)$  的增长率是相同的。

**例 1-2** 设用两个算法 A 和 B 求解同一问题, 它们的时间复杂度分别是  $T_A(n) = n^2/2$ ,  $T_B(n) = 2n \lg n$ , 试讨论其效率。

当输入量较小( $n < 15$ )时, 有  $T_A(n) < T_B(n)$ , 前者花费的时间较少。随着问题规模  $n$  的增长, 两算法的时间消耗之比  $n/(4 \lg n)$  逐渐增大。也就是说, 当问题规模较大时, 算法 A