工业和信息化部“十二五”规划教材

21世纪高等教育计算机规划教材

COMPUTER

数据结构与算法分析（C++语言版）

Data Structures and Algorithm Analysis (C++)

张琨 张宏 朱保平 编著

- 内容全面、循序渐进
- 案例详实、指导实践
- 图表丰富、阐述清晰

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

工业和信息化部“十二五”规划教材
21世纪高等教育计算机规划教材

COMPUTER

数据结构与算法分析（C++语言版）

Data Structures and Algorithm Analysis (C++)

张琨 张宏 朱保平 编著

人民邮电出版社
北京

图书在版编目（CIP）数据

数据结构与算法分析 : C++语言版 / 张琨, 张宏, 朱保平编著. -- 北京 : 人民邮电出版社, 2016.2
21世纪高等教育计算机规划教材
ISBN 978-7-115-40927-0

Ⅰ. ①数… Ⅱ. ①张… ②张… ③朱… Ⅲ. ①数据结构－高等学校－教材②算法分析－高等学校－教材③C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(2015)第263108号

内 容 提 要

本书是工业和信息化部“十二五”规划教材，全书借鉴了国内外高等院校《数据结构》相关教材，吸收了当代计算机领域最新成果。内容鸟瞰全貌，删减陈旧，反映新知。并在相关章节增加了典型习题。

本书介绍了数据结构的基本理论及方法，主要有绪论、线性表、栈和队列、串、数组和广义表、树和二叉树、图、查找、内部排序和算法设计与分析等内容。

本书可作为高等院校计算机科学与技术相关专业的教材，也可作为教师、研究生或软件技术人员的参考用书。

◆ 编　　著　张　琨　张　宏　朱保平
责任编辑　许金霞
责任印制　沈　蓉　彭志环
◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市潮河印业有限公司印刷
◆ 开本：787×1092　1/16
印张：19.25　　2016 年 2 月第 1 版
字数：507 千字　　2016 年 2 月河北第 1 次印刷

定价：45.00 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

前言

数据结构是计算机科学各专业以及其他相近专业的核心课程之一，也是计算机程序设计的重要理论基础。计算机的数据处理能力是计算机解决各种实际问题的关键。现实世界中的实际问题抽象成数学模型，然后得出反映事物本质的数据表示，这样才有可能被计算机处理。那么如何以计算机所能接受的形式来描述这些数据？又如何将这些数据以及它们之间的关系存储在计算机中？如何用有效的方法来处理这些数据？这些都是数据结构所要解决的问题。

作者根据多年在教学一线的经历和体会，编写了《数据结构与算法分析（C++语言版）》一书。本书是在深入研究国内外同类教材的基础上，结合多年“数据结构”课程的教学经验编写而成。本书为绝大多数抽象数据类型提供了完整的 C++ 类源代码和关键算法实现代码，并给出了大量的编程练习题来加深巩固对数据结构的理解。

全书共分为 10 章。

第 1 章是绪论部分，主要是对数据结构的初步认识，包括数据结构的发展、研究对象、概念及其相关专业术语、数据类型和抽象数据类型、算法和算法分析等内容。

第 2 章是线性表，主要介绍了线性表的概念、抽象数据类型定义、顺序存储结构和链式存储结构的表示及实现、线性表在一元多项式的表示和运算中的应用等内容。

第 3 章是栈和队列，主要介绍了栈和队列的基本概念、抽象数据类型定义、顺序和链式存储结构及其相关应用。

第 4 章是串，主要介绍了串的基本概念、表示与实现、BF 和 KMP 两种模式匹配方法。

第 5 章是数组和广义表，主要介绍了数组和广义表的概念、抽象数据类型定义及存储结构、特殊矩阵和稀疏矩阵的压缩存储等内容。

第 6 章是树和二叉树，主要介绍了树、森林和二叉树的相关概念、抽象数据类型定义、性质、存储结构、遍历算法、以及三者之间的相互转换，还有线索二叉树、堆、哈夫曼树和哈夫曼编码等内容。

第 7 章是图，主要介绍了图的基本概念和术语、抽象数据类型定义、存储结构、深度和广度优先遍历算法、构造最小生成树的 Prim 算法和 Kruskal 算法、AOV 网和 AOE 网及其应用、最短路径等内容。

第 8 章是查找，主要介绍了查找的基本概念，静态查找表中的顺序查找、有序表查找、分块查找、二叉排序树、平衡二叉树和 B_树等，以及哈希表的构造和冲突处理方法。

第 9 章是内部排序，主要介绍了直接插入、折半插入、表插入、希尔排序等插入排序，冒泡排序和快速排序两种交换排序方法，简单选择排序、树形选择排序和堆排序等选择排序，还有归并排序、基数排序等内容。

第10章是算法设计与分析，主要介绍了分治策略、回溯法、贪心算法、动态规划法和分支限界法的概念、实现及相关应用等。

本书具有以下特点。

（1）深入浅出，通俗易懂。对数据结构的基本概念、基本理论的阐述注重科学严谨。同时从应用出发，对新概念的引入从实例入手。对各种基本算法描述尽量详细，叙述清楚。

（2）为了巩固所学的理论知识，每章都围绕知识点和难点附有练习题，供学生书面练习和上机作业选用。习题题型多样，难度适中，既适合课堂教学，又便于读者自学时对基础知识的理解和掌握。

（3）用C++语言描述算法。本书的侧重点仍在“数据结构”上，使用C++语言作为算法描述的一种工具。而C++的类对象设计正好与数据结构的抽象数据类型ADT的实现相吻合。本书所有的算法和实例程序都在VC++ 6.0环境下编译通过并正常运行的。

（4）针对学生中普遍存在“只懂概念，不会编程”的问题，每章中都有若干个算法实现的C++源程序示例，供学生参考模拟，以提高学生程序设计的能力。

（5）在教材的基础上，重新设计教学过程、制作大量的教学动画，以直观的方式揭示教学的思想本质，有助于加深学生对数据结构基本概念、原理和方法的理解。

与以往教材相比，本书充分考虑到教师教学、学生学习与进一步深造的需要，在处理好数据结构的知识结构和强化算法的实践与应用的同时，使读者通过实现算法复杂的程序训练，编写出结构清晰、正确易读、符合软件工程规范的程序；教师更方便组织教学内容，教学过程完整清晰，内容循序渐进，插图丰富易于理解。本书适合作为高等院校信息类相关专业的“数据结构”教材。

本书第1章～第7章、第10章由张琨编写，第8章由张宏编写，第9章由朱保平编写。感谢冯新淇、刘健等在本书的编写过程中提供的帮助。

由于时间仓促，作者水平有限，书中错误在所难免，欢迎广大读者和专家批评指正，以便作者进行修订和补充。

编　者

2015年5月

目　录

第 1 章 绪论

计算机是一门研究利用计算机进行信息表示和处理的科学，它涉及信息的表示和信息的处理两个方面，而信息的表示又直接关系到处理信息的程序的效率。计算机早期主要用于数值计算，后来其处理逐渐扩大到非数值计算领域，并且能处理多种复杂的具有一定结构关系的数据。随着计算机的普及，信息量的增加，信息范围的拓宽，许多系统程序和应用程序的规模增大，结构变得更加复杂。因此，为了编写出一个“好”的程序，必须分析待处理数据的特征、数据间的相互关系以及数据在计算机内的存储表示，并利用这些特性和关系设计出相应的算法与程序，这就是数据结构所要研究的问题。

1.1 数据结构的概念

以下小节分别从数据结构的发展历史、具体什么是数据结构、数据结构的研究对象以及与数据结构相关的概念等方面对数据结构的基本概念做简单介绍。

1.1.1 数据结构的发展

数据结构作为一门独立的学科形成于 20 世纪中期，但在此之前有关内容已散见于编译原理和操作系统的教材之中。1968 年，美国的第九位“图灵奖”获得者 Donald Ervin Knuth 教授开创了数据结构的最初体系，他的著作《计算机程序设计艺术》第 1 卷《基本算法》是第一本比较系统地阐述数据的逻辑结构和存储结构及其操作的著作。20 世纪 60 年代末到 70 年代，计算机的应用领域已不再局限于科学计算（而更多地应用于控制、管理等非数值处理领域），软件也相对独立，结构程序设计成为程序设计的主要内容。因此，人们越来越重视数据结构。20 世纪 70 年代初，数据结构作为一门独立的课程开始进入大学课堂。20 世纪 70 年代中期到 80 年代初，各种版本的数据结构著作相继出现。

从我国计算机教学现状来看，数据结构不仅是计算机专业教学计划中的核心课程之一，而且已逐步成为非计算机专业的主要选修课程之一。数据结构与数学、计算机硬件和计算机软件有着十分密切的关系，如图 1.1 所示。

所以数据结构又是一门介于数学、计算机硬件和计算机软件之间的计算机科学领域的核心课程。在计算机科学中，数据结构不仅是一般非数值计算程序设计的基础，而且是高级程序设计语言、编译原理、操作系统、人工智能等课程的基础。同时，数据结构技术也广泛应用于信息科学、系统工程、应用数学以及各种工程技术领域。

目前，数据结构的发展并未终结，一方面，面向各专门领域中特殊问题的数据结构得到研究和发展，例如多维图形数据结构等；另一方面，从抽象数据类型和面向对象的观点来讨论数据结构已成为一种新的趋势，越来越为人们所重视。

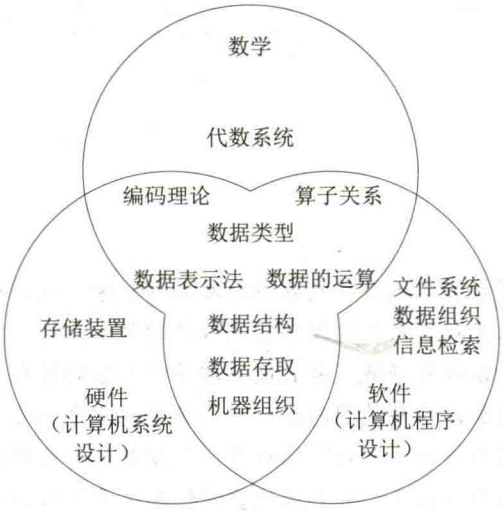


图 1.1 “数据结构”所处的地位

1.1.2 什么是数据结构

一般来说，计算机解决一个具体问题时，大致需要经过下列步骤：一、建立数学模型，二、构造求解方法，三、选择存储结构，四、编写程序，五、测试，如图 1.2 所示。在建立数学模型阶段，重点关注描述问题的共性（寻求数学模型的实质是分析问题，从中提取操作对象，并找出这些操作对象之间含有的关系）；在构造求解方法阶段，重点关注描述问题的求解方法；在选择存储结构的阶段，重点关注如何将问题涉及的数据存储到计算机中；在编写程序阶段，重点关注如何提高编程的技术；最后进行数据测试。在上述五阶段中，数据结构在第一阶段有助于更好地进行问题分析，在第二阶段有助于进行更为复杂的算法设计，在第三阶段有助于选择合理的存储结构，最终依据基于数据结构的设计，实现程序编写。

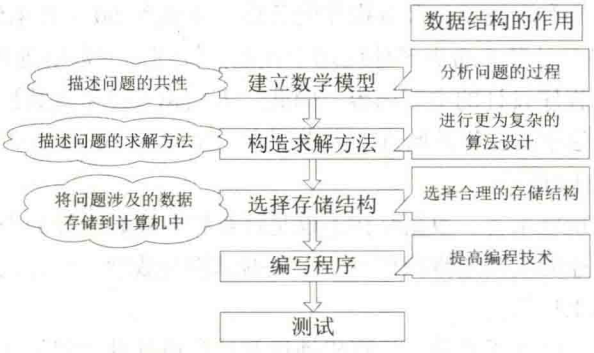


图 1.2 计算机解决问题的过程

著名的瑞士计算机科学家尼古拉斯·沃斯（Niklaus Wirth）教授指出：算法+数据结构=程序。算法是解决特定问题的步骤和方法，数据结构是问题的数学模型，程序则是计算机处理问题编制

的一组指令集。计算机算法与数据的结构密切相关，算法无不依附于具体的数据结构，数据结构直接关系算法的选择和效率。运算是由计算机来完成的，这就要设计相应的插入、删除和修改算法。也就是说，数据结构还需要给出每种结构类型所定义的各种运算的算法。所以，数据结构是研究程序设计中计算机操作的对象以及它们之间的关系和运算的一门学科。

以下是几种类型的数据结构实例。

如表 1.1 所示是一个学生选课系统中的课程信息表，其中，各个课程之间的关系可以用线性的数据结构来描述。

表 1.1　　学生选课系统

课程编号	课程名称	主讲教师	课程简介	课程总学时	课程学分	……
010115A05	自然辩证法概论	张鑫	点击进入	16	1	……
B101C001	微米/纳米技术基础	李军	点击进入	32	2	……
B101C003	火箭武器系统分析	王琳	点击进入	32	2	……
B101C004	现代测控技术导论		点击进入	32	2	……
……	……	……	……	……	……	……
B101C007	新型传感器技术	许文龙	点击进入	32	2	……
B101C008	推进系统结构强度理论	王长运	点击进入	32	2	……
B10Z002	导航定位及目标探测辨识	王磊	点击进入	32	2	……
B10Z007	新型传感器及校准技术	李娟	点击进入	32	2	……
B102Z001	污染控制原理	赵启明	点击进入	32	2	……
……	……	……	……	……	……	……

图 1.3 所示是 UNIX 文件系统目录结构，它表示的则是一种树形的数据结构。

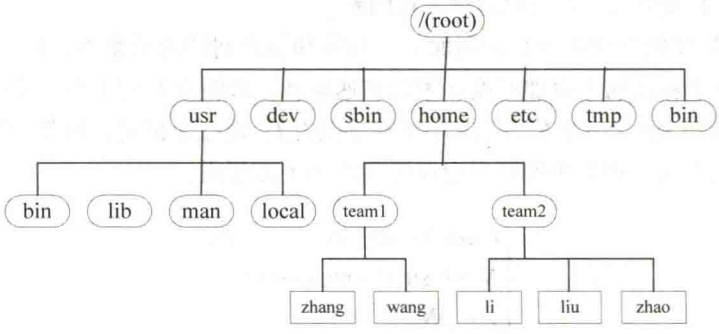


图 1.3　UNIX 文件系统目录结构

图 1.4 所示是一个由大量数据结点组成的连通网络拓扑图，它表示的是一种图状的数据结构。

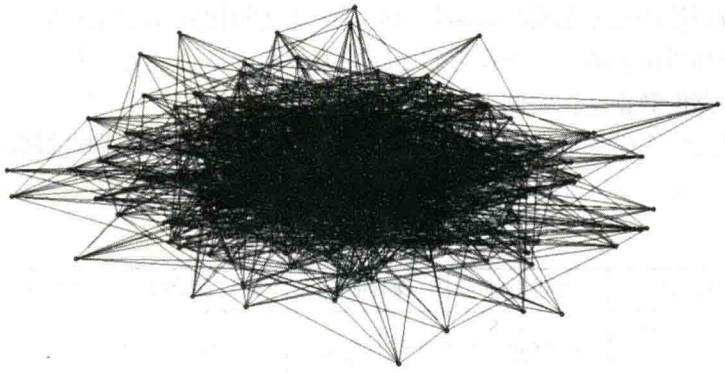


图 1.4　大量数据结点的拓扑结构

1.1.3　数据结构的研究对象

计算机程序对数据进行加工处理，一般情况下，这些数据之间都存在着一定的关系。当计算机程序所涉及的运算对象是简单的整型、实数型或布尔类型的数值型数据时，程序设计者的主要精力用于解决程序设计的技巧问题；当计算机处理非数值计算问题时，所涉及数据之间的关系可能非常复杂，甚至很多问题无法用数学方程式加以描述，因此数据结构的内容在非数值计算中显得尤为重要。

以下是几个典型的实例。

1．数值计算程序设计问题

例 1.1　一元二次方程求解问题

对于一元二次方程 $ax^2+bx+c=0$（$a\neq 0$），给定参数 a，b，c 的值，则可以使用数学公式法求解，它的根可以表示为：

$$x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

有些时候也写成：

$$x_{1,2}=\frac{2c}{-b\pm\sqrt{b^2-4ac}}$$

计算机可以根据输入结果利用公式求出问题的解，并输出结果。

例 1.2　应用牛顿第二定律求解力学方程问题

质量为 m 的物体放在倾角为 θ 的斜面上，物体和斜面间的滑动摩擦因数为 μ。如沿水平方向加一个力，使物体沿斜面向上以加速度 a 做匀加速运动，求外力 F 的大小。可选质量为 m 的物体作研究对象，受重力 mg，外力 F，支持力 N 和摩擦力 f。建立坐标时，以加速度方向即沿斜面向上方向为 x 轴的正方向，根据牛顿第二定律可列出如下方程组：

$$\begin{cases}F\cos\theta-mg\sin\theta-f=ma\\N-F\sin\theta-mg\cos\theta=0\\f=\mu N\end{cases}$$

上述方程组求解可得外力 F 的表达式为：

$$F=\frac{\mu mg\cos\theta+mg\sin\theta+ma}{\cos\theta-\mu\sin\theta}$$

以上两个例子均为数值计算问题，计算机程序可以直接利用数学公式或方程进行解答。

2. 非数值计算程序

例 1.3　电话号码查询系统

设有一个电话号码薄，记录了 n 个人的名字和其对应的电话号码，假定按如下形式安排：

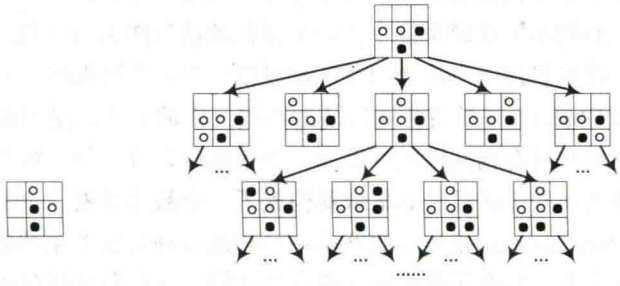
$$(a_1,b_1),(a_2,b_2),\cdots,(a_n,b_n)$$

其中 $(a_i,b_i)(i=1,2\cdots n)$ 分别表示某人的名字和其对应的电话号码。要求设计一个程序，当给定任何一个人的名字时，该程序能够打印出此人的电话号码，如果该电话簿中无此人，则该程序也能够报告相应的查找失败提示信息。

上述电话号码表中的数据是一种简单的线性关系，这类数学模型可称为线性的数据结构。

例 1.4　人机对弈问题

计算机之所以能和人对弈，是因为对弈的策略已存入计算机。在对弈问题中，计算机的操作对象是对弈过程可能出现的棋盘状态——格局，而格局之间的关系是由对弈规则决定的。因为一个格局可以派生出多个格局，所以，这种关系通常不是线性的。如图 1.5（a）所示为井字棋的其中一个格局，从该格局出发可派生出 5 个新格局，从新的格局出发，还可以再派生出新的格局，如图 1.5（b）所示。格局之间的关系可以用树形的数据结构来描述。



（a）井字棋的一个格局　　（b）对弈树的局部

图 1.5　人机对弈中格局之间的关系

例 1.5　铺设城市的煤气管道

图 1.6 所示为城市的各小区之间铺设煤气管道，对 n 个小区只需铺设 $n-1$ 条管线，由于地理环境等不同因素使各条管线所需投资不同（如图中线上所标识），如何使投资成本最低？这是一个关于图的最小生成树问题。

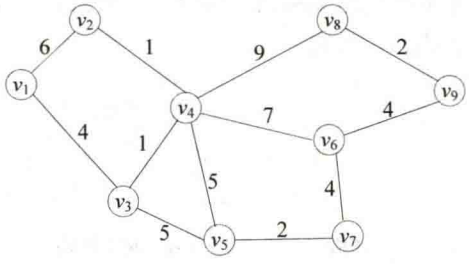


图 1.6　煤气管道的铺设示意图

从以上 3 个非数值计算的例子可以看出，描述这类问题的数学模型不再是数学方程，而是诸如表、树和图之类的数据结构。因此，概括地说，数据结构是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一门学科。

1.1.4 数据结构相关概念及术语

在系统的学习数据结构知识之前，应先对一些基本概念和术语赋予确切的含义。

数据（Data）是信息的载体，在计算机科学中是指能输入到计算机中并能被计算机程序识别和处理的符号集合，它是计算机操作对象的总称，是计算机处理的信息的某种特定的符号表示形式。数据可以分为两大类：一类是整数、实数等数值型数据；另一类是图形、图像、声音、文字等非数值型数据。数据是计算机程序加工的“原料”，例如，一个利用数值分析方法解决代数方程的程序，其处理的对象是整数和实数；一个编译程序或文字处理程序的操作对象是字符串。因此，从计算机科学的角度讲，数据的含义极为广泛，图像、声音等信息都可以通过编码而归结到数据的范畴中。

数据元素（Data Element）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，是数据结构中讨论的基本单位。数据元素具有广泛的含义，一般来说，能独立、完整地描述问题的一切实体都是数据元素。如例 1.4 中的“树”的一个棋盘格局，例 1.5 中的“图”的每一个圆圈都被称为一个数据元素。又例如整数“5”，字符“*N*”等是数据元素，同时也是不可分割的最小单位。

数据项（Data Item）是构成数据元素的不可分割的最小单位，一个数据元素可以由若干的数据项组成。如在表 1.1 的学生选课系统中，一个课程的信息作为一个数据元素，而课程信息中的每一项（如课程号、课程名称、主讲教师等）都可以作为一个数据项。

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。在实际应用中处理的数据元素通常具有相同性质，例如，学生选课系统中每个数据元素具有相同数目和类型的数据项，所有数据元素（课程）的集合就构成了一个数据对象。

数据结构（Data Structure）是指相互之间存在一种或多种特定关系的数据元素的集合。从 1.1.3 节中的几个例子可以看出，数据元素都不是孤立存在的，在它们之间存在着某种关系，这种数据元素之间的关系称为**结构**（Structure）。根据数据元素之间关系的不同特性，数据结构通常分为以下 4 类基本结构。

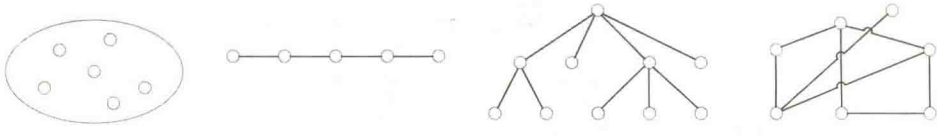
（1）**集合**：数据元素之间就是“属于同一个集合”，除此之外没有任何关系。

（2）**线性结构**：数据元素之间存在着一对一的线性关系。

（3）**树形结构**：数据元素之间存在着一对多的层次关系。

（4）**图状结构或网状结构**：数据元素之间存在着多对多的任意关系。

图 1.7 所示为上述 4 类基本结构的关系图。集合是一种数据元素关系松散的结构，因此在实际解决问题中，往往用其他结构来表示它。



（a）集合　（b）线性结构　（c）树形结构　（d）图状结构

图 1.7　4 类基本关系结构图

从上述数据结构的概念可知，一个数据结构有两个要素：数据元素的集合和数据元素之间关系的集合。在形式上，数据结构通常可以采用一个二元组来表示，记为：

$$Data\_Structure=(D, R)$$

其中，D 是数据元素的有限集，R 是 D 上关系的有限集。

数据结构包括逻辑结构和物理结构两个层次。

逻辑结构（Logical Structure）是指数据元素之间逻辑关系的整体。所谓的逻辑关系是指数据元素之间的关联方式或邻接关系。数据的逻辑结构可以看作是对操作对象的一种数学描述，换句话说，是从具体问题中抽象出来的数学模型，它与数据自身的存储无关。研究数据结构是为了在计算机中实现对它的操作，为此还需要研究如何在计算机中存储表示数据的逻辑结构。

物理结构（Structure）是指数据结构在计算机中的表示（即映像），又称**存储结构**（Storage Structure）。它研究的是数据结构在计算机中的表示方法，包括数据结构中数据元素的表示以及数据元素之间关系的表示。

位（Bit）是指在计算机中表示信息的最小单位，是二进制数的一位。在计算机中，可以用由若干位组合起来形成的位串来表示任何一个数据元素。如：数值 345 可以用位串 101011001 来表示，字母 B 可以用位串 001000010 来表示等，通常称这个位串为**元素**（Element）或**结点**（Node）。当数据元素由若干数据项组成时，位串中对应于各个数据项的子位串称为**数据域**（Data Field）。因此，元素或结点可看成是数据元素在计算机中的映像。

以下通过一个实例说明数据的逻辑结构。

例 1.6　用 3 个 4 位的十进制数 a_1(3214)，a_2(6587)，a_3(9345)表示一个 12 位数的十进制数 3214,6587,9345，则在数据元素 a_1、a_2 和 a_3 之间存在着“次序”关系$<a_1,a_2>$、$<a_2,a_3>$，任何两个数据元素之间的次序是无法颠倒的。若$<a_1,a_2>$的次序关系互换为$<a_2,a_1>$，则其表示的十进制数则变为 6587,3214,9345（$\neq$3214,6587,9345），即$<a_1,a_2,a_3>\neq<a_2,a_1,a_3>$。数据元素之间的这种次序关系就是一种逻辑结构。

例 1.7　在 2 行 3 列的二维数组中 6 个数据元素$\{a_1,a_2,a_3,a_4,a_5,a_6\}$之间存在两个次序关系：

a_1	a_2	a_3
a_4	a_5	a_6

行的次序关系 $row=\{<a_1,a_2>,<a_2,a_3>,<a_4,a_5>,<a_5,a_6>\}$

列的次序关系 $col=\{<a_1,a_4>,<a_2,a_5>,<a_3,a_6>\}$

若在 6 个数据元素$\{a_1,a_2,a_3,a_4,a_5,a_6\}$之间存在如下的次序关系：

$$\{<a_i,a_{i+1}>|i=1,2,3,4,5\}$$

则该逻辑关系表示的是一个 1 行 6 列的一维数组。

可见，相同数据元素之间的关系不同，即逻辑结构不同，则构成的数据结构不同。

数据的存储结构除了存储数据元素之外，必须隐式或显式地存储数据元素之间的逻辑关系。通常数据元素在计算机中有两种不同的表示方法：**顺序映像**（Sequential Mapping）和**非顺序映像**（Non-Sequential Mapping），并由此得到两种不同的存储结构：**顺序存储结构**和**链式存储结构**。

顺序映像的特点是借助数据元素在存储器中的相对位置来表示数据元素之间的逻辑关系，所以顺序存储就是把逻辑上相邻的数据元素存储在物理位置相邻的存储单元中，由此得到的存储表示称为顺序存储结构。顺序存储结构是一种最基本的存储表示方法，通常借助于程序设计语言中的数组来实现。

非顺序映像的特点是借助指示数据元素存储地址的**指针**（Pointer）表示数据元素之间的逻辑关系，所以链式存储就是用一组任意的存储单元存储数据元素，不要求其物理位置相邻，数据元素之间的逻辑关系通过附设的指针字段来表示，由此得到的存储表示称为链式存储结构，链式存储结构通常借助于C++语言中的指针类型来实现。

在不同的编程环境中，存储结构可有不同的描述方法。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，但由于本书是在高级程序语言的层次上讨论数据结构的操作，因此不能直接以存储地址来描述数据结构，但可以借助程序语言中提供的数据类型来描述它。如用一维数组类型来描述顺序存储结构，以C++语言提供的指针来描述链式存储结构。例如，用3个带有次序关系的整数表示一个长整数时，可利用整数数组类型，定义长整数为：int long_int[3]。

综上可以看出，数据的逻辑结构和存储结构是密切相关的两个方面。一般来说，一种数据的逻辑结构可以用多种存储结构来存储，而采用不同的存储结构，其处理数据的效率往往是不同的。一个算法的设计取决于选定数据的逻辑结构，而算法的实现则依赖于所采用的存储结构。

1.2 数据类型和抽象数据类型

数据类型和抽象数据类型都是和数据结构密切相关的概念，数据类型是一组值的集合以及定义在这个集合上的一组操作的总称，而抽象数据类型的定义则涉及数据结构。

1.2.1 数据类型

类型（Type）是指一组值的集合。例如，**布尔**（Boolean）类型由true和false这两个值组成。整数也构成类型，若采用2个字节，则整数表示范围在-32768～32767；若采用4个字节，则整数表示范围在-2147483648～2147483647。

数据类型（Data Type）则是一组值的集合以及定义在这个值集上的一组操作的总称。数据类型和数据结构密切相关，它最早出现在高级程序语言中，用以刻画程序中操作对象的特性。在用高级程序语言编写的程序中，每个变量、常量或表达式都有一个它所属的确定的数据类型。类型显式或隐式地规定了在程序执行期间变量或表达式所有可能取值的范围，以及在这些之上允许进行的操作。例如，C++语言中的整型变量，其值集为某个区间上的整数，区间大小依赖于不同的机器，定义在其上的操作为加、减、乘、除和取模等算术运算。

按“值”的不同特征，高级程序语言中的数据类型可分为两类：非结构的原子类型和结构类型。原子类型的值是不可分解的。例如，C语言中的整型、字符型、浮点型、双精度型等基本数据类型，分别用保留字int、char、float、double标识。除此之外，还包括枚举型、指针类型和空类型等。结构类型的值则是由若干成分按某种结构组成的，因此是可以分解的，并且它的组成部分可以是非结构的原子类型，也可以是结构类型。例如，数组的值是由若干分量组成，每个分量可以是整数，也可以是数组等其他数据类型。在某种意义上，数据结构可以看成是“一组具有相同结构的值”，则结构类型可以看成由一种数据结构和定义在其上的一组操作组成。非结构原子类型和结构类型这两类编程语言中已定义并实现的数据类型统称为**固有数据类型**。

实际上，在计算机中数据类型的概念并非局限于高级语言中，每个处理器都提供了一组原子类型或结构类型，包括计算机的硬件系统、操作系统、高级语言、数据库等。例如，一个计算机硬件系统通常含有“位”、“字节”、“字”等原子类型，它们的操作通过计算机设计的一套指令系

统直接由电路系统完成。而高级程序语言提供的数据类型，其操作需通过编译器或解释器转化成低层语言，即汇编语言或机器语言的数据类型来实现。从硬件的角度看，引入“数据类型”的概念是作为解释计算机内存中信息含义的一种手段；而对使用数据类型的用户来说，它实现了信息的隐蔽，即将一切用户不必了解的细节都封装在类型中。例如，用户在使用“浮点数”类型时，既不需要了解“浮点数”在计算机内部是如何表示的，也不需要知道其操作是如何实现的。当“两浮点数求和”时，程序设计者重视的仅仅是其在“数学上求和”的抽象特性，而不关心其硬件中的“位”操作是如何实现的。

1.2.2　抽象数据类型

抽象数据类型（Abstract Data Type，简称 ADT）则是指一个数学模型（数据结构）以及定义在该模型上的一组操作。所谓**抽象**（Abstract）就是抽出问题的本质特征而忽略非本质的细节，是对具体事务的一个概括。抽象数据类型的定义仅仅取决于它的一组逻辑特性，而与计算机内部如何表示和实现无关，即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部的使用。

抽象数据类型和数据类型实质上是一样的。例如，各种高级程序设计语言中都拥有“整数”类型，这是一个抽象数据类型，尽管它们在不同处理器上实现的方法不同，但对程序员而言是“相同的”，因为它们的数学特性相同。因此，从“数学抽象”的角度看，可称它为一个“抽象数据类型”。

另一方面，抽象数据类型的范畴更广，它不再局限于前述各种处理器中已定义并实现的数据类型，即固有数据类型（如整型、字符型、浮点型等数据类型），还包括用户在设计软件系统时自己定义的数据类型，为了提高软件的复用率，在近代程序设计方法学中指出，一个软件系统的框架应建立在数据之上，而不是建立在操作之上。即在构成软件系统的每个相对独立的模块上，定义一组数据和施加在这些数据上的一组操作，并在模块内部给出这些数据的表示及其操作的细节，而在模块外部使用的只是抽象的数据和抽象的操作。显然，定义的数据类型抽象层次越高，含有该抽象数据类型的软件模块的复用程度也就越高。

综上所述，抽象数据类型有两个重要特征：数据抽象和数据封装。

数据抽象（Data Abstraction）是指用 ADT 描述程序处理的实体时，强调其本质的特征、所能完成的功能以及它和外部用户的接口（即外界使用它的方法）。

数据封装（Data Encapsulation）是将实体的外部特性和其内部实现细节分离，并且对外部用户隐藏其内部实现细节。所以，抽象数据类型设计时，把类型的定义与实现分离开来。

数据结构包括数据元素以及元素之间的关系，因此抽象数据类型一般由数据元素、关系及操作 3 种要素来定义。和数据结构的形式定义相对应，抽象数据类型可用以下三元组表示：

$$ADT=(D,S,P)$$

其中，D 是数据对象，S 是 D 上关系集，P 是对 $D=\{e_1,e_2 \mid e_1,e_2 \in \mathrm{Re}alSet\}$ 的基本操作集。本书采用以下格式定义抽象数据类型：

ADT 抽象数据类型名{
数据对象：<数据对象的定义>
数据关系：<数据关系的定义>
基本操作：<基本操作的定义>
} **ADT 抽象数据类型名**