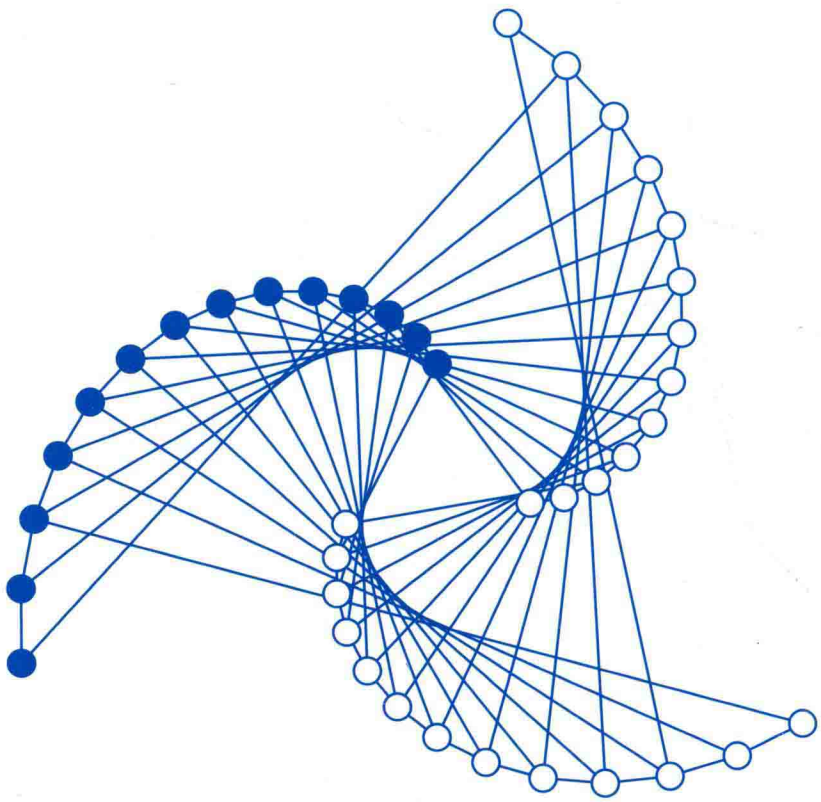
教育部高等学校电子信息类专业教学指导委员会推荐教材

Data Structure

数据结构

彭波 主编
Peng Bo



清華大學出版社

Data Structure

数据结构

彭波　主编
Peng Bo

清華大學出版社
北京

内 容 简 介

本书系统地介绍数据结构基础理论知识及算法设计方法，第1～7章从抽象数据类型的角度讨论各种基本类型的数据结构及其应用，主要包括线性表、栈和队列、串、数组和广义表、树和二叉树及图；第8章和第9章主要讨论查找和排序的各种实现方法及其综合比较；第10章介绍数据结构课程实验的目的、步骤及内容；附录给出全书习题的参考答案。全书采用类C语言作为数据结构和算法的描述语言，随书配备电子教案。

本书在内容选取上符合人才培养目标的要求及教学规律和认知规律，在组织编排上体现“先理论、后应用、理论与应用相结合”的原则，并兼顾学科的广度和深度，力求适用面广。本书具有结构严谨、层次清楚、概念准确、深入浅出、描述清晰等特点。

本书可以作为计算机类专业和信息类相关专业的本科或专科教材，也可以供从事计算机工程与应用工作的科技工作者参考。

图书在版编目(CIP)数据

数据结构/彭波主编. —北京：清华大学出版社，2016
ISBN 978-7-302-42214-3

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

中国版本图书馆CIP数据核字(2015)第278888号

责任编辑：盛东亮
封面设计：李召霞
责任校对：梁　毅
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，010-62795954

印 装 者：北京密云胶印厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：21.75　　**字　　数**：545千字
版　　次：2016年3月第1版　　**印　　次**：2016年3月第1次印刷
印　　数：1～2000
定　　价：45.00元

产品编号：061551-01

前言
PREFACE

“数据结构”课程是计算机、电子信息类及相关专业的专业基础。它在整个课程体系中处于承上启下的核心地位：一方面扩展和深化在离散数学、程序设计语言等课程学到的基本技术和方法；另一方面为进一步学习操作系统、编译原理、数据库等专业知识奠定坚实的理论与实践基础。本课程在教给学生数据结构设计和算法设计的同时，培养学生的抽象思维能力、逻辑推理能力和形式化思维方法，增强分析问题、解决问题和总结问题的能力，更重要的是培养专业兴趣，树立创新意识。本教材在内容选取上符合人才培养目标的要求及教学规律和认知规律，在组织编排上体现“先理论、后应用、理论与应用相结合”的原则，并兼顾学科的广度和深度，力求适用面广泛。

全书共分 10 章。第 1 章综述数据、数据结构和抽象数据类型等基本概念及算法描述与分析方法；第 2～7 章主要从抽象数据类型的角度分别讨论线性表、栈和队列、串、数组和广义表、树和二叉树、图等基本类型的数据结构及其应用；第 8 章和第 9 章讨论查找和排序的各种方法，着重从时间性能、应用场合及使用范围方面进行分析和比较；第 10 章介绍数据结构课程实验的目的、步骤及内容。本书对数据结构众多知识点的来龙去脉做了详细解释和说明；每章后面配有难度各异的适量习题，并在附录中给出习题的参考答案，供读者理解知识及复习提高之用。随书配备电子教案。

全书采用类 C 语言描述数据结构和操作算法。类 C 语言是 C 语言的一个精选子集，同时又采用了 C++对 C 非面向对象的增强功能，使本书对各种抽象数据类型的定义和与数据结构相关的操作算法的描述更加简明清晰，可读性更好，既不拘泥于 C 语言的细节，又容易转换成能够上机执行的 C 程序或 C++程序。

从课程性质上讲，“数据结构”是高等院校计算机科学、电子信息科学及相关专业考试计划中的一门专业基础课；其教学要求是学会分析研究计算机加工的数据结构的特性，以便为应用涉及的数据选择适当的逻辑结构、存储结构及其相应的算法，并初步掌握算法的时空分析技术。从课程学习上讲，“数据结构”的学习是复杂程序设计的训练过程；其教学目的是着眼于原理与应用的结合，在深化理解和灵活掌握教学内容的基础上，学会把知识用于解决实际问题，书写出符合软件工程规范的文件，编写出结构清晰及正确易读的程序代码。可以说，“数据结构”比“高级程序设计语言”等课程有着更高的要求，它更注重培养分析抽象数据的能力。

在本书的构思与编写过程中，得到了北京师范大学孙一林副研究员、北京建筑大学邱李华副教授等同志的帮助，在算法的实现与调试过程中，得到了杨志军、吴娟、黄健翔、史春雷

等研究生的帮助，在此表示感谢。本书可以作为计算机类专业和电子信息类相关专业的本科或专科教材，也可以供从事计算机工程与应用工作的科技工作者参考。本书结构严谨、层次清楚、概念准确、深入浅出、通俗易懂、便于自学。由于作者水平有限，教材中不当之处敬请读者提出批评和建议，作者电子邮件地址：pengbo_cau@126.com。

编　者

2016 年 1 月

目录

CONTENTS

第1章 绪 论

CHAPTER 1

主要知识点

- 数据结构的范畴及相关的概念
- 算法的基本概念及描述方法
- 算法的时间复杂度及空间复杂度的分析方法

使用计算机求解任何问题都离不开程序设计，而程序设计的实质就是数据表示和数据处理。数据要能被计算机处理，首先必须能够存储在计算机的内存中，这项任务称为数据表示，数据表示的核心任务是数据结构的设计；一个实际问题的求解必须满足各项处理的要求，这项任务称为数据处理，数据处理的核心任务是算法设计。因此，数据结构主要讨论数据表示和数据处理的基本问题。

1.1 数据结构的范畴

数据结构起源于程序设计。随着计算机应用领域的扩大和软件与硬件技术的飞速发展，计算机的应用已远远超出了科学和工程计算的范围，并广泛地应用于情报检索、信息管理、系统工程，乃至人类社会活动的一切领域。与此同时，计算机的处理对象也从简单的纯数值性数据发展到非数值性和具有一定结构的数据，例如文本、图形、图像、音频、视频及动画等，处理的数据量也越来越大，这就给程序设计带来一个问题：应该如何组织待处理的数据以及数据之间的关系(结构)。

1.1.1 计算机处理问题的分类

计算机处理的问题可以分为数值性问题和非数值性问题。

1. 数值性问题

众所周知，20世纪40年代，电子计算机问世的直接原因是解决弹道计算问题。早期的电子计算机的应用范围只限于科学和工程计算，处理对象是纯数值性数据，通常人们把这类问题称为数值性问题。例如，线性方程求解问题涉及的运算对象是简单的整型、实型或布尔型数据。程序设计者的主要精力集中于程序设计的技巧，不需要重视数据结构。

2. 非数值性问题

根据统计，当今处理非数值性问题占用了90%以上的机器时间，这类问题涉及的数据结构更为复杂，数据元素之间的相互关系一般无法用数学方程式描述。因此，解决此类问题

的关键已经不再是数学分析和计算方法，而是需要建立问题的数学模型和设计相应的算法，才能有效地解决问题。

1.1.2 非数值性问题的求解

1968年，美国唐纳德·克努特(Donald. E. Knuth)教授开创了数据结构的最初体系，他所著的《计算机程序设计艺术(第Ⅰ卷)·基本算法》是第一本比较系统地阐述数据逻辑结构和存储结构及其操作的著作。1976年，瑞士计算机科学家尼古拉斯·沃斯(Niklaus Wirth)教授(艾伦·麦席森·图灵(Alan Mathison Turing)奖获得者)提出：数据结构＋算法＝程序。斗转星移至今，尽管新的技术方法不断涌现，这句名言依然焕发着无限的生命力，它借助面向对象知识的普及，使数据结构技术更加完善和易于使用。由此，也说明了数据结构在计算机学科中的地位和不可替代的独特作用。

例 1-1 学生情况管理问题。

用计算机来完成学生管理就是用计算机程序来处理学生情况登记表，实现增加、删除、修好、查找等功能。图1-1是一张简单学生情况登记表。在学籍管理问题中，计算机的操作对象是每个学生的情况信息(档案表项)，各档案表项之间的关系可以用称为线性表的数学模型来描述。在该数学模型中，计算机处理的数据之间存在的是“一个对一个”的线性关系，这类数学模型可称为线性的数据结构。

姓名	性别	年龄	籍贯	班别	成绩			
					数学	物理	化学	外语
孙臣	男	18	北京	6003	95	90	92	96
钱晓	女	20	上海	6002	90	95	85	80
常依	女	19	长沙	6004	85	90	80	75
吴伟	男	19	湖北	6001	85	80	75	90
…	…	…	…	…	…	…	…	…

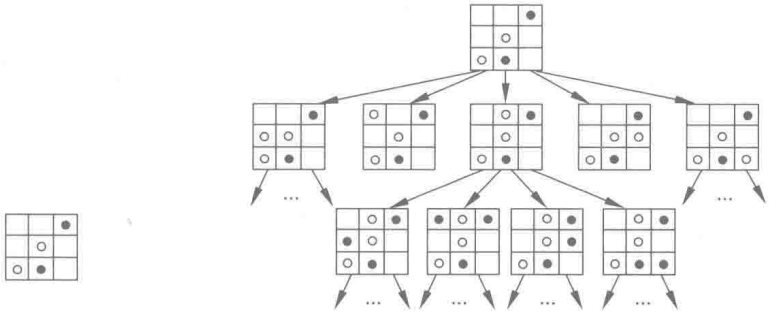
图 1-1 学生情况登记表

例 1-2 人机对弈问题。

计算机之所以能够和人对弈，是因为对弈的策略实现已经存入计算机内。在对弈问题中，计算机的操作对象是对弈过程中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。因为从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1-2(a)所示为井字棋(又称三子连珠，由两个人对弈，棋盘为3×3的方格，当一方的三个棋子占同一行、同一列或同一对角线时便为胜方)的一个格局，从该格局出发可以派生出五个新格局，从新格局出发，还可以再派生出新格局，如图1-2(b)所示。格局之间的关系可以用称为树的数学模型来描述。在该数学模型中，计算机处理的数据之间存在“一个对多个”的层次关系，这类数学模型可称为树的数据结构。

例 1-3 城市最小造价通信网问题。

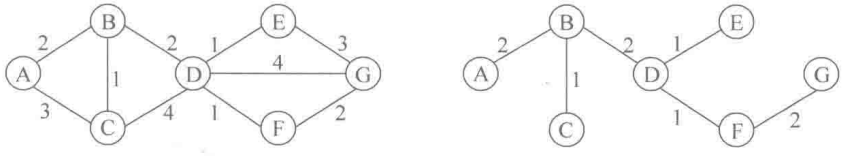
用计算机求解城市最小造价通信网问题就是由计算机程序在已知某些城市之间直接的通信线路预算造价的情况下，求出n个城市中任意两个城市之间直接或间接的通信线路，使网络造价最低。图1-3(a)所示为7个城市预算造价通信网，图1-3(b)所示为7个城市最小



(a) 井字棋的一个格局　　(b) 对弈树的局部

图 1-2　对弈问题中格局之间的关系

造价通信网。在城市最小造价通信网问题中，如果用圆圈表示一个城市，两个圆圈之间的连线表示对应城市之间的通信线路，连线上的数值表示该通信线路的造价，则城市之间的通信关系可以用称为图的数学模型来描述。在该数学模型中，计算机处理的数据之间存在的是“多个对多个”的任意关系，这类数学模型可称为图的数据结构。



(a) 7个城市预算造价通信网　　(b) 7个城市最小造价通信网

图 1-3　通信网问题中城市之间的关系

由上述三个例子可以看出，描述这些非数值问题的模型已经不再是数学方程，而是线性表、树、图等的数据结构。在抽象出问题的模型之后，数据结构的任务还包括对这个模型进行求解。因此，简单说来，数据结构是一门研究非数值性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等的学科。

1.2　数据结构发展的概况

数据结构随着程序设计的发展而发展。程序设计经历了三个阶段：无结构阶段、结构化阶段和面向对象阶段。相应地，数据结构的发展也经历了三个阶段。

1. 无结构阶段

20 世纪 40 年代至 60 年代，计算机的应用主要是针对科学计算，程序设计技术以机器语言及汇编语言为主，程序处理的数据是纯粹的数值，数据之间的关系主要是数学公式或数学模型。在这一阶段，人类的自然语言与计算机编程语言之间存在着巨大的鸿沟，程序设计属于面向计算机的程序设计，设计人员关注的重心是使程序尽可能被计算机接受并按指令正确执行，至于程序能否让人理解并不重要。

2. 结构化阶段

20 世纪 60 年代至 80 年代，计算机开始广泛应用于非数值处理领域，数据表示成为程

序设计的重要问题，人们认识到程序设计规范化的重要性，提出了程序结构模块化，并开始注意数据表示与操作的结构化。数据结构及抽象数据类型就是在这种背景下形成的。数据结构概念的引入对程序设计的规范化起到了重大的作用。从沃斯提出的著名公式“数据结构＋算法＝程序”可以看到，数据结构和算法是构成程序的两个重要组成部分，一个软件系统通常是以一个或几个关键数据结构为核心而组成的。

随着软件系统的规模越来越大、复杂性不断增加，人们不得不对结构化技术进行重新评价。软件系统的实现依赖于关键数据结构，如果这些关键数据结构的一个或几个有所改变，则会涉及整个系统，甚至导致整个系统彻底崩溃。

3. 面向对象阶段

面向对象技术（首先是面向对象程序设计）开始于 20 世纪 80 年代初，是目前最流行的程序设计技术。在面向对象技术中，问题世界的相关实体视为一个对象，对象由属性和方法构成，属性用于描述实体的状态或特征，方法用于改变实体的状态或描述实体的行为。一组具有相同属性和方法的对象的集合抽象为类，而每个具体的对象都是类的一个实例。例如，“学生”是一个类，“张三”、“李四”等对象都是“学生”类的实例。

由于对象（类）将密切相关的属性（数据）和方法（操作）定义为一个整体，从而实现了封装和信息隐藏。使用类时，不需要了解其内部的实现细节，如果数据（结构）修改了，只需要修改类内部的局部代码，而软件系统的其余部分不需要修改。

数据结构主要强调两个方面的内容：一个是数据之间的关系；另一个是针对这些关系的基本操作。这两个方面实际上蕴含着面向对象的思想：类重点描述实体的状态与行为，而数据结构重点描述数据之间的关系及其基本操作，数据及其相互关系构成了对实体状态的描述，针对数据元素之间关系的操作构成了对实体行为的描述。由此可见，类与数据结构之间具有对应关系，如图 1-4 所示。

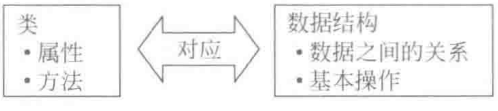


图 1-4　类和数据结构之间的对应关系

值得注意的是，数据结构的发展并未终结。其一，数据结构将继续随着程序设计的发展而发展；其二，面向各个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，各种实用的高级数据结构被研究出来，各种空间数据结构也在探索中；其三，从抽象数据类型的观点来讨论数据结构已经成为一种必然的趋势。

1.3　数据结构相关的概念

在本节中将先对以后各章节中反复出现的数据结构相关概念赋以确定的含义，以便在后面的学习中与读者取得“共同的语言”。

1.3.1　数据的概念

1. 数据

数据（Data）是信息的载体，在计算机科学中指所有能输入到计算机中并能被计算机程序识别、存储和处理的符号集合。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求解代数方程的程序，其处理对象是整数和实数；一个编

译程序或文字处理程序，其处理对象是字符串；一个影视作品制作程序，其处理对象是文、图、声、像等。对于计算机及其相关科学而言，数据的含义极为广泛，例如图像、声音、动画、视频等都可以通过编码而归之于数据的范畴。

2. 数据元素

数据元素(Data Element)是数据基本单位，也称元素(Element)、结点(Node)、顶点(Vertex)、记录(Record)，在计算机程序中常作为一个整体进行考虑和处理。一个数据元素可以是不可分割的原子，称为原子项；也可以由若干个数据项组成，称为组合项。数据项是数据不可分割的最小单位。

例如，包括书名、作者名、分类号、出版单位及出版时间等的一条书目信息在计算机图书管理程序中作为一个数据元素，而书名、分类号等称作数据项。数据元素具有广泛的含义，一般来说，能独立、完整地描述问题世界的一切实体都是数据元素。

3. 数据对象

数据对象(Data Object)是性质相同的数据元素的集合，是数据的一个子集。

例如，整数数据对象的集合可以表示为 N={0，±1，±2，…}；大写字母字符数据对象的集合可以表示为 C={'A'，'B'，…，'Z'}；银行业务处理系统中的数据对象是全体储户及全体贷款客户资料；电话号码查询系统中的数据对象是全体电话用户。

4. 数据关系

数据关系(Data Relation)反映了数据对象中数据元素所固有的一种结构。在数据处理领域，通常把数据之间的这种固有关系简单地用前驱和后继来描述。

例如，在编写家庭族谱时，数据对象是家庭中的所有成员，对家族中某成员的描述就是一个数据元素，各数据元素之间存在着血缘关系；父亲是儿子的前驱，儿子是父亲的后继，小孩子没有后继。一张按照名次排列的成绩表，数据对象是全班同学，对某个同学属性(姓名、成绩等)的描述就是一个数据元素，各数据元素之间存在着名次关系；第一名没有前驱，其后继是第二名，第二名的前驱是第一名，其后继是第三名。

1.3.2 结构的概念

1. 数据结构

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。在任何问题中，数据元素都不是孤立存在的，而是在它们之间存在着某种关系，这种数据元素相互之间的关系称为结构。

它研究的内容包括数据的逻辑结构和数据的物理(存储)结构。

2. 逻辑结构

数据的逻辑结构(Logical Structure)是对操作对象的一种数学描述，换句话说，是从操作对象抽象出来的数学模型。结构定义中的“关系”描述的是数据元素之间的逻辑关系，与数据的存储无关，是独立于计算机的。

根据数据元素之间逻辑关系的不同特性，通常有四类基本逻辑结构。

1) 集合(Set)

结构中的数据元素之间除了“同属于一个集合”外别无其他的关系，即只有数据元素而无任何关系，如图 1-5(a)所示。

2) 线性结构(Linear Structure)

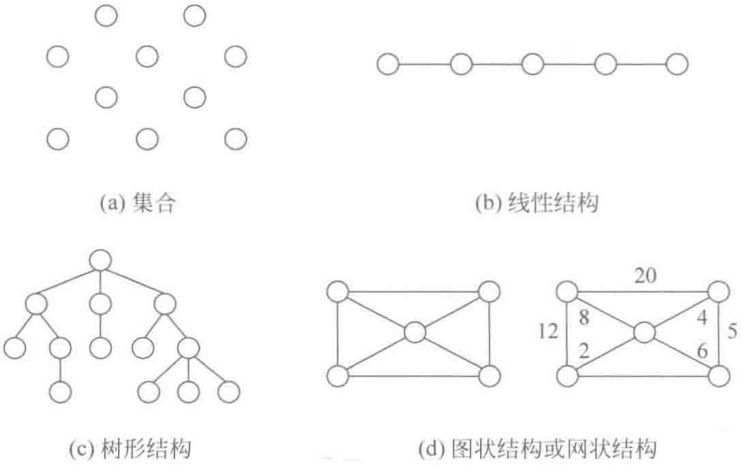
结构中的数据元素之间存在着“一个对一个”的关系,即除了第一个数据元素无前驱、最后一个数据元素无后继外,其他相邻数据元素均具有唯一的前驱和后继,如图 1-5(b)所示。

3) 树形结构(Tree Structure)

结构中的数据元素之间存在着“一个对多个”的关系,即除了一个根数据元素外,其他各元素均有唯一的前驱,所有数据元素都可以有多个后继,如图 1-5(c)所示。

4) 图状结构或网状结构(Graph or Net Structure)

结构中的数据元素之间存在着“多个对多个”的关系,即所有数据元素都可以有多个前驱或多个后继,如图 1-5(d)所示。



(a) 集合　(b) 线性结构

(c) 树形结构　(d) 图状结构或网状结构

图 1-5　数据的四类基本逻辑结构

由于“集合”是数据元素之间关系极为松散的一种结构,是数据结构的一种特例,因此也可以用其他结构来表示它。

例 1-4　描述一周七天数据的逻辑结构。

该数据的逻辑结构可以用一个数据元素的集合 D 和定义在该集合上的一个关系 R 来表示,如图 1-6 所示,其中:

D = {Sun, Mon, Tue, Wed, Thu, Fri, Sat}

R = {< Sun, Mon >, < Mon, Tue >, < Tue, Wed >, < Wed, Thu >, < Thu, Fri >, < Fri, Sat >}

图 1-6　一周七天数据的逻辑结构

3. 物理结构

数据的物理结构(Physical Structure)又称为存储结构(Storage Structure),是数据及其逻辑结构在计算机中的表示。换言之,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。

数据元素之间的关系在计算机中有两种不同的表示方法:顺序表示和非顺序表示。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。

1) 顺序存储结构(Sequential Storage Structure)

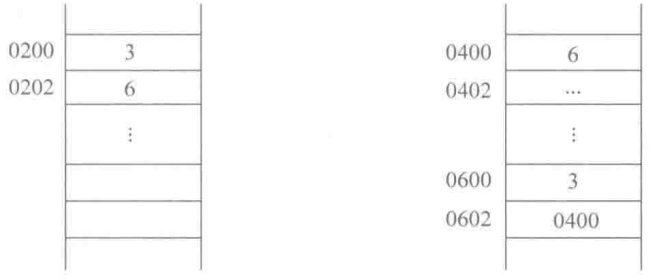
顺序表示的特点是借助数据元素在存储器中的相对位置来表示数据元素之间的逻辑关系,由此得到的结构称为顺序存储结构。

2) 链式存储结构(Linked Storage Structure)

非顺序表示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,由此得到的结构称为链式存储结构。

通常,在程序设计语言中,顺序存储结构借助于数组描述,链式存储结构借助于指针描述。在实际程序设计过程中,一种数据的逻辑结构到底选择哪种存储结构会影响到具体算法的实现。也就是说,在实现具体算法之前,必须先确定存储结构。

例 1-5 分别用顺序存储结构和链式存储结构表示 Y=3+6,每个数据元素占 2 个字节,如图 1-7 所示。



(a) 顺序存储结构　　(b) 链式存储结构

图 1-7　Y=3+6 存储结构示意图

数据的逻辑结构和物理结构是密切相关的两个方面,在后面的学习中可以了解到,任何一个算法的设计取决于选定的逻辑结构,而算法的实现依赖于采用的存储结构。

1.3.3 类型的概念

1. 数据类型

数据类型(Data Type)是和数据结构密切相关的一个概念,最早出现在高级程序语言中,用来刻画(程序)操作对象的特性。类型明显或隐含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值上允许进行的操作。因此,数据类型是一个值的集合和定义在这个值集上的一组操作的总称。

例如,C 语言中的"整数类型",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、除和取模等算数运算。

按照"值"的不同特性,高级程序语言中的数据类型可以分为两类:

(1) 原子类型:其值是不可分解的。例如 C 语言的整型、实型、字符型等,以及指针类型。原子类型通常由语言直接提供。

(2) 结构类型:其值可分解为若干个成分(或称为分量),并且它的成分(分量)可以是非结构的,也可以是结构的。例如 C 语言数组的值由若干分量组成,每个分量可以是整数,也可以是数组等。结构类型通常借助于语言提供的描述机制来定义。

2. 抽象数据类型

抽象数据类型(Abstract Data Type,ADT)是指一个数据模型以及定义在该模型上的

一组操作。抽象数据类型的定义仅取决于它的一组逻辑特性，而与其在计算机的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。

抽象数据类型可以理解为对数据类型的进一步抽象，其区别仅在于：数据类型指的是高级程序语言中已经实现的数据结构；而抽象数据类型的范畴更广，它不再局限于各处理器中已经定义并实现的数据类型(基本数据类型)，还包括用户在设计软件系统时自己定义的数据类型。

随着计算机科学的不断发展，特别是面向对象的程序设计语言的研究和发展，提出了抽象数据类型的概念。为了提高软件复用率，在近代程序设计方法学中指出，一个软件系统框架应该建立在数据之上，而不是像传统软件设计方法学，将一个软件系统框架建立在操作之上。也就是说，在构成软件系统每个相对独立的模块中定义一组数据和施与这些数据之上的一组操作，并在模块内部给出它们的表示和实现细节，在模块外部使用的只是抽象的数据和抽象的操作。显然，所定义数据类型的抽象层次越高，含有该抽象数据类型软件模块的复用率也就越高。

3. 抽象数据类型的表示

一个抽象数据类型的定义不涉及它的实现细节，在形式上可繁可简，本书对抽象数据类型的定义采用如下格式描述：

```
ADT 抽象数据类型名 {
    数据对象(Data)：<数据对象的定义>
    逻辑关系(Relation)：<数据关系的定义>
    基本操作(Operation)：<基本操作的定义>
} ADT 抽象数据类型名
```

其中，基本操作的定义格式为：

```
基本操作名(参数表)
    初始条件：<初始条件描述>
    操作结果：<操作结果描述>
```

基本操作有两种参数：赋值参数只为操作提供输入值；引用参数以 & 开头，除了可以输入值之外，还将返回操作结果。“初始条件”描述了操作执行之前数据结构和参数应该满足的条件；若不满足，则操作失败，并返回相应的出错信息；若初始条件为空，则省略它。“操作结果”说明操作正常完成之后，数据结构的变化状况和应该返回的结果。

例 1-6 复数的抽象数据类型定义。

```
ADT Complex {
    Data:
        D = {e1,e2|e1,e2∈RealSet}
    Relation:
        R = {<e1,e2>|e1 是复数的实数部分,e2 是复数的虚数部分}
    Operation:
        InitComplex(&z,v1,v2)
            初始条件：无
            操作结果：构造并返回复数 z,其实部和虚部分别赋予参数 v1 和 v2 的值。
        GetReal(z,&RealPart)
```