高等学校通识教育系列教材

用C++实现数据结构程序设计

马春江 编著

清华大学出版社

高等学校通识教育系列教材

用C++实现数据结构程序设计

马春江 编著

清华大学出版社
北京

内 容 简 介

本书的特色是在源码级别而不是算法级别上讨论数据结构。书中给出的程序源码能帮助学生掌握数据结构程序设计，提高综合运用数据结构的能力。全书共分 12 章，内容包括数据结构基础和递归思想、线性数据结构、非线性数据结构、查找、排序等应用。

本书对于数据结构的综合运用进行了较为深入的讨论，在线性表、栈、队列、索引结构和二叉树程序设计方面给出了源码的深入介绍；在游戏设计、MP3 歌曲文件和二维码等数据结构构造方面提供了较为详细的说明。全书提供的源码可大大加深学生对于数据结构程序设计过程的理解。

本书可作为高等院校研究型与应用型本科计算机相关专业教材，还适合高职高专各类学校计算机相关专业参考使用，也可作为计算机岗位培训和计算机爱好者自学用书。

图书在版编目(CIP)数据

用 C++实现数据结构程序设计/马春江编著. —北京：清华大学出版社，2019
(高等学校通识教育系列教材)
ISBN 978-7-302-52701-5

Ⅰ. ①用… Ⅱ. ①马… Ⅲ. ①C++语言－程序设计－高等学校－教材 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2019)第 063137 号

责任编辑：刘向威
封面设计：文　静
责任校对：焦丽丽
责任印制：宋　林

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：清华大学印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：19.25　　**字　　数**：467 千字
版　　次：2019 年 8 月第 1 版　　**印　　次**：2019 年 8 月第1次印刷
印　　数：1～1500
定　　价：59.00 元

产品编号：077138-01

前　言

高级语言程序设计、数据结构、算法设计与分析构成了程序设计理论底层的黄金三角形。高级语言程序设计偏重语法的描述和程序设计细节等能力培养，数据结构重点讨论程序设计中如何分析、规划、存储和实现相关的数据以及关系，算法设计与分析则偏重解题思路的实现。大部分算法设计首先依赖数据结构的构造，这将深远影响到程序的时间效率和空间效率，以及程序结构的合理性和程序阅读的简易性。计算机界著名学者尼古拉斯·沃思(Wirth N.)提出的“算法＋数据结构＝程序”的观点正说明数据结构的重要性，即使在面向过程转向面向对象程序设计的今天，对象中底层的函数实现依然能体现这句至理名言的正确性。万丈高楼平地起，楼高几何看地基。数据结构就是程序设计的地基，必须先学好它，以打下扎实的基础。

我从事数据结构教学已有三十多年，教学实践中深深体会到，“数据结构”是计算机专业一门很难的课程，学生普遍反映过于抽象和难以编程实现。这反映了几个现实问题：①高级语言课程所教授的内容与数据结构编程需求有一定的距离，高级语言教程讨论得更多的是数值计算方面的程序设计范例，对于离散结构的讨论相对偏少。②“算法设计”是数据结构的后续课程，很多设计思想在“数据结构”课程中暂时无法起到引导作用。③从数据结构本身看，教材书写方式有时成了学习过程的阻碍。部分数据结构教材偏重理论，多用所谓的算法表述数据结构程序设计思想，导致学生可以模仿编程的范例不够，实际效果不大理想；部分教材源于翻译国外教材，术语和描述生涩难懂；部分教材重点、难点不突出，整体结构不完整，缺乏应用层面的案例分析，增加了学生的学习困难。

2015 年，我在清华大学出版社作为主编出版了《用 C 实现数据结构程序设计》，收到了来自全国各地的大学生或程序开发工作者的来信。因为阅读了我编写的教材，他们能够以较快的速度学习到数据结构的程序设计，效果很理想，对考研也有较大帮助，希望我能推出更多平台上的数据结构源码设计。受到这些热情读者和编辑的鼓励，我开始着手编写数据结构的 C++ 版程序设计，最终完成这本教材，希望能帮助读者尽快学会数据结构程序设计。

学习数据结构是一种“痛并快乐着”的过程，在学习的时候大部分学生都会感到过于抽象和高深，但是如果能用程序具体实现，就会有成就感，会被计算机科学家的奇思妙想所震撼，体会到程序设计的引人入胜，整个过程是充满挑战和乐趣的。

本书特色是全面给出数据结构相关程序设计源码，并给出了运行界面图，使得学生有一个可以研究、探讨、模仿、提高的平台。本书的程序设计范例很多都具有可移植性、实用性和趣味性，覆盖了多种程序设计方法和界面设计风格。

全书体系结构完整，注重原理与实践结合，重点和难点突出，为学生搭建了全面的学习和研究平台，其目标是学生易于学习、老师易于组织教学。本书提供了菜单编制、多种数据提供方式、多种输出格式、程序反复运行、处理意外情况、颜色和标题栏、方向键控制等源码，提供了多种界面设计的范例。线性表之后先介绍查找和排序基础部分，体现了线性表的使用价值，而进阶部分涉及更复杂的数据结构，在大部分数据结构知识学习完成之后再分别讲解。

全书共分12章，内容涉及学习数据结构基础和递归思想；线性数据结构、非线性数据结构；线性表、栈、队列、字符串、二维数组、树和森林、二叉树、图；还涉及查找、排序等基础应用。为拓展数据结构知识，还介绍了文件的基础知识。

本书配有电子教案和程序源代码，便于老师教学和学生使用。采用C++语言编程实现，程序均在VC++ 6.0下调试并运行通过。

数据结构是程序开发的基础，是进入程序设计殿堂的敲门砖。本书是我多年来勤恳敬业、认真教学和深入思考的结晶，希望读者能分享和品味学习的乐趣。

我要感谢我的父母及家人，还有我的诸多朋友，他们的鼓励和支持让我不能忘怀。

我要感谢我多年前在清华大学研习“人工智能”研究生课程时的指导老师石纯一教授，他对专业的精通、做事的认真以及平易近人的态度让我一生都受益匪浅，那段时光是我一生的珍贵回忆和骄傲。

本书能够顺利出版，得益于清华大学出版社的全力支持和帮助，责任编辑的细致审稿和建议使本书增色不少，在此深表谢意。

由于能力所限及时间紧迫，书中难免存在疏漏，希望读者提出宝贵意见和建议，以便再版时改进和提高。

马春江　于湖北汽车工业学院计算机系

2019年1月

目 录

第1章　数据结构基础

本章介绍数据结构的背景和基本概念，为数据结构相关程序设计奠定基础。讲解逻辑结构分类、存储结构分类与基本操作的名称和特点，讨论算法和算法效率分析、递归的概念和相关范例。

1.1　面式思维和点式思维

人们学完高级语言程序设计之后，通常希望能尽快展示出自己的程序设计能力，开发出一些原创的软件作品，但却发现经常在程序设计一开始就陷入了不知所措的情形，发现自己好像根本不会编程。这是一个令人困惑的问题。

下面通过问题的讨论来看看这究竟是怎么回事。

在图 1-1 所示的模拟白板上有一批数据，请问哪一个是最大值？

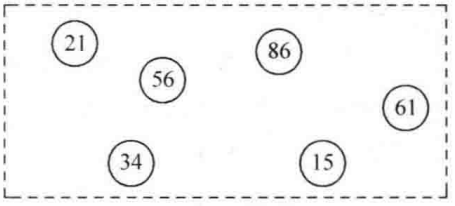


图 1-1　面式思维的原理示意图

很显然，读者一眼就可以看出最大值是 86。读者根据什么说最大值是 86？观察、判断、思考过程是什么？能否书写出完整的思考过程呢？

很明显答案是不同的，或者根本说不清楚是如果思考的，只是一眼就看出来而已。为什么会出现这种情况呢？

这是因为人类有了思维能力之后，经过训练有了大小比较和选择能力。人的双眼观察世界是立体的，观察到的信息以图像的方式进入大脑，简化这种模型后可以认为已经是平面关系。人类的大脑开始工作，此时相当于有很多部高速计算机在同时工作，一瞬间结果已经出现。但是实际上，从开始思考到说出结果这一段的工作过程是无法精确描述的。

把计算机能够做的所有工作称为“计算”，就是一种广义的计算。计算机在进行各种计算时是否和人的思维方式一样呢？

人类还没有真正了解人的大脑思维模式，无论是医学家、生物学家，还是心理学家、哲学家都无法说明人是如何记忆、如何思考的。但是人却提出了让机器来帮助进行计算这样的设想。怎样才能让机器做这些事呢？科学家给出了一种思路，那就是“模拟”机制，也就是不

管人类是如何完成这些工作的，只要机器最后做出来的结果和人类做的结果一样就可以了。

人类的记忆机制和思考机制虽然还不能完全搞清楚，但可以想象它是一个很复杂的网络结构，复杂到任何一个信息点都可以直接激活另外一个信息点，这就是“联想”，也可以失去任何联系，这就是“忘记”。在这个复杂的网络中，所有信息都是立体的，完成思考和记忆的过程也是立体的，同时人类在观察现实生活中的事物时也是立体的，这些立体信息一瞬间同时进入大脑，之后人类开始立体式思维过程和记忆过程。从上面求最大值问题来看，眼睛把所有数据一次看完，之后同期进入大脑，开始立体式网状思维。本书把这种思维方式称为“面式思维方式”（指的是多面组成的网状结构），简称“面式思维”。

对应记忆方面，计算机中使用的是“存储器”；对应思考方面，计算机中则使用的是“中央处理器”（CPU），它的特点是在某个时刻永远只能处理当前特定“一个”存储单元的信息。

虽然内存中可以同时存放很多数据，但是中央处理器实际上工作在具体的“点”上，而不是线条、平面图形、立体图形等，因此把计算机的思维方式称为“点式思维”。编程就是把“面式思维”转化为“点式思维”的过程，或者说如何用“点式思维”来模拟“面式思维”，所以在编程中对任何要处理的事物就不能再用人类的一般思维方式，而应该习惯去用计算机的思维方式。

针对求最大值问题，为了达成“点式思维”，就必须先把所有数据排成一行（或一列），然后确定求最大值的规则，其思路为：首先默认“第一个数据”就是最大值，然后把后面的数据“逐一”和当前最大值进行比较，如果有比当前最大值更大的，则记录该位置，设其为最大值，继续比较过程，直到比完“最后一个数据”，就求出了最大值的位置。图 1-2 为重新排列后的图 1-1 中的数据，在上述讨论中，“第一个”“逐一”“最后一个”等字眼就是点式思维最明确的象征，而在图 1-1 中，这些概念都是无法体现的。

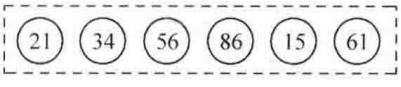


图 1-2　点式思维的原理示意图

这里的“一行”实际上就是后面将要介绍的数据结构“线性表”，而算法的名称可以归纳为“顺序比较与刷新法求最大值”。

了解“点式思维”的基本特点后，程序设计的很多思路都将变得相对简单，在数据结构的程序设计中更是处处体现“点式思维”的影响。

1.2　数据结构背景

计算机发展初期，人们主要使用计算机处理数值计算问题，如天气预报、军事领域、大型工程等计算量大的工作。解决这样一个具体问题需要经过几个步骤：首先从具体问题抽象出一个数学模型，然后设计或选择一个求解此数学模型的算法，最后编写程序、进行调试，直至得到最终的结果。

随着计算机理论和技术的发展，计算机的主要用途从数值计算转到非数值计算，也就是数据处理。解决这类问题的关键不再是数学分析和计算方法，而是要设计出合适的数据结构。这类非数值计算问题的数学模型不再是数学方程，而是诸如表、树、图之类的数据结构。“数据结构”课程主要研究非数值计算的程序设计问题中所出现的计算机操作对象以及它们之间的关系和操作。

1968年美国的唐纳德·克努特(Knuth D. E.)教授开创了数据结构的最初体系,他编著的《计算机程序设计艺术》(*The Art of Computer Programming*)第一卷《基本算法》是第一本较系统地阐述数据逻辑结构和存储结构及其操作的著作,此获得了1974年图灵奖。

数据结构发展史上还有一位不能不提的巨匠——Pascal语言的开发者尼古拉斯·沃思。他撰写的《算法+数据结构=程序》(*Algorithms+Data Structures=Programs*)是具有里程碑意义的优秀书籍之一。他是结构化程序设计的首创者,也是1984年图灵奖的获得者。

计算机科学是一门研究数据表示和数据处理的科学。数据是计算机化的信息,它是计算机可以直接处理的最基本和最重要的对象。在科学计算、数据处理、过程控制以及对文件的存储、检索和数据库技术等计算机应用领域中,都有对数据进行处理的过程。因此,要设计出一个结构良好、效率很高的程序,必须研究数据的特性、数据间的相互关系及其对应的存储表示,并利用这些特性和关系设计出相应的算法,进一步编出程序。

学习数据结构的目的是为了了解计算机处理对象的特性,将实际问题涉及的处理对象在计算机中表示出来,并对它们进行处理。与此同时,通过算法训练来提高逻辑思维能力,通过程序设计技能训练来促进综合应用能力和专业素质的提高。

程序设计的基本理论涉及三门重要的基础课:高级语言程序设计、数据结构、算法设计与分析。高级语言程序设计主要是解决上机编程的环境、语法要求和一些基本的程序设计技巧;数据结构主要是把数据的关系研究清楚并且确定其存储方案,以及基本操作的编程实现;算法设计与分析主要是深入研究算法的设计技术和时间、空间效率分析。这三者缺一不可。在算法设计与分析中会重点讨论时间效率分析,故本书中没有将时间复杂度的计算过程与深入分析作为重点,仅通过一些简洁分析确定其结论。

"数据结构"课程主要讨论软件开发过程中的设计阶段,同时涉及编码和分析阶段的若干基本问题也会被综合讨论。此外,为了构造出好的数据结构并实现,还需考虑数据结构及其实现的评价与选择。因此,数据结构的研究内容包括3个层次、5个要素,见表1-1。

表1-1　数据结构研究的主要内容

层　次	数据表示	数据处理
抽象层	逻辑结构	基本操作
实现层	存储结构	算法
评价层	各种存储结构的比较及算法分析	

1.3　数据结构的应用案例

在学习程序设计的过程中,必须体会到"量变引起质变"的重要性。当面临的任务或软件系统足够复杂时,很多无足轻重的"小事"都会演变成"大事",那么整体设计就成为首要的工作。在使用高级语言编程时如果不进行整体设计,而是边想边做,也不按照开发规范书写,一旦该开发任务涉及的数据之间的关系比较复杂,就可能出现多次返工或开发彻底失败的结局,所以在软件开发前必须先考虑好各种情况,尤其是数据的关系(即数据结构)的设计。

【应用案例 1-1】 人机对弈(如下棋等)程序设计。有些读者喜欢计算机可能是因为利用计算机可以玩游戏,而作为专业工作者需要更深层面的思考,如各种棋类的对弈是如何编程实现的。首先是界面的实现,如棋盘和棋子如何实现,显然不能是简单的图片,因为还要考虑棋子的移动。另外要对弈的话,计算机怎么知道哪些位置可以落子,对方的哪些棋子已经被吃掉,而我方的哪些棋子已经受到威胁,又如何进一步决策进攻或防守呢……。通过这个案例可以认识到,不论是界面还是功能都会涉及数据结构的大量知识,如果没有数据结构很难编出游戏或更复杂的软件。

【应用案例 1-2】 计算机计算表达式。计算机能计算表达式,似乎天经地义,否则为什么叫计算机呢?但当初科学家实现这个功能时却很困难。表达式是 2+1,计算机能开始计算吗?不行,因为可能是 2+11,所以必须往后读到回车或者另外一个运算符,但是遇到运算符还是不能开始计算,因为可能是 2+11×3。根据运算规则,必须先计算 11×3,可是表达式又可能是 $2+11\times3^2$,那么能不能先算平方呢?也不能,因为原来的表达式可能是 $2+11\times3^{(2+1)}$,既然有括号,而括号中又出现了加法,那么最初的问题就会重新出现,如 $2+11\times3^{(2+1\times2)}$。即使中间部分计算结果能先求出来,还是有次序问题,如原式为 $2+11\times3^{(2+1\times2)}-6$,那么现在该退回去做加法,还是往前走去做减法呢?已经读完的数据和运算符如何存储?后面暂时没有读到的数据又如何存储?如果这些数值全部用一些变量存储,那么启用多少个变量?变量之间的关系如何管理?这些问题如果不能解决,想让计算机进行"计算"完全不可能。幸运的是科学家们已经解决了这些问题,使用计算机能够做到数值计算的精确化和高速化,火箭、宇宙飞船等涉及大量计算也就有了保证。本范例说明计算机要实现的很多功能仅仅靠编程技巧很难解决,必须依赖数据结构的支撑。

【应用案例 1-3】 痕迹检测、DNA 检测、文字识别、人像识别等技术的实现前提。现代科技带来了很多前人闻所未闻、想也想不到的科技成果,如公安系统能够使用指纹或血液等信息追查罪犯,医院能够用 DNA 做亲子鉴定,计算机可以进行文字或语音识别等。这些技术面临着一个共同的难题,那就是必须存储超大量的数据,还要进一步进行快速和有效查找。有时数据总量远远超过想象,被称为海量数据。那么海量数据如何存储,数据之间的关系又如何管理,进一步又如何实现快速查找等功能呢?这个范例说明数据结构对数据的组织和查找技术将起到关键的作用。

1.4 数据结构基本概念

数据结构基本结构如下。

数据(Data)。一般人们谈及数据就会想到 0～9 组成的数字,实际上这些应称为数值。在现实生活中的"信息"这个概念一旦被计算机存储,就成为所谓的"数据"。要注意的是,"数据"并不能完全覆盖"信息",比如日常生活中的"眉目传情",情感是一种信息但是并不能被目前的计算机识别、存储从而处理,所以数据是信息的一种载体,它能够被计算机识别、存储和加工处理。

计算机科学中,数据可以是数值数据,也可以是非数值数据。数值数据是一些整数、实数或复数,主要用于工程计算、科学计算和商务处理等;非数值数据包括字符、文字、图形、图像、语音等,其中的文字又包括英文、中国的汉文和少数民族的文字、其他各国文字等。

数据元素(Data Element)。数据元素是数据的基本单位。在不同场合,可把数据元素称为元素、结点、顶点、记录等。

数据项(Data Item)。有时一个数据元素还可以由若干数据项组成。例如,人员信息的每一个数据元素就是一个人的记录,可能包括编号、姓名、性别、籍贯、出生日期、电话、手机、通信地址、电子邮件地址等数据项。这些数据项又被分为两种:一种称为基本项,如性别、籍贯等,这些数据项在数据处理时不再分割;另一种称为组合项,如出生日期可以分为年、月、日等更小的项。

数据对象(Data Object)。数据对象也可以称作数据元素类(Data Element Class),是具有相同性质的数据元素的集合。在某些具体问题中,数据元素具有相同的性质,属于同一数据对象,即数据元素是数据对象的一个实例。

数据结构(Data Structure)。数据结构是指互相之间存在一种或多种关系的数据元素的集合。在使用计算机处理任何问题时,我们不仅关注数据本身,也一定会关注数据元素之间的关系,因为这些"关系"就是我们需要的"信息",这种数据元素之间的关系就是"结构"。

数据结构的形式定义。数据结构是一个二元组

$$\text{Data\_Structure} = (D, R)$$

其中,D(Data 的首字母)是数据元素的有限集;R(Relation 的首字母)是 D 上关系的有限集。

数据的逻辑结构。数据的逻辑结构可以看作是从具体问题抽象出来的数学模型,是数据本身的关系,它与数据的存储并无关系,有点像纸上谈兵。但是研究数据结构的目的就是为了在计算机中实现对它们的存储并进行其他操作,为此还需要研究如何在计算机中存储逻辑结构。

数据的存储结构。计算机中的存储方法就是"存储结构",有时也称为数据的物理结构。它主要研究逻辑结构在计算机中的实现方法,包括元素的表示和元素之间关系的表示。所以对数据结构可以理解为两个层面,即逻辑结构和存储结构。

1.5　逻辑结构分类

根据数据元素之间可能存在的关系,逻辑结构可分为以下 4 种基本结构:线性结构、树状结构、图形结构和集合结构。

(1) 线性结构。该结构的数据元素之间(从一个方向而言)只有一维的关系,即线性关系,也称为"一对一"关系。

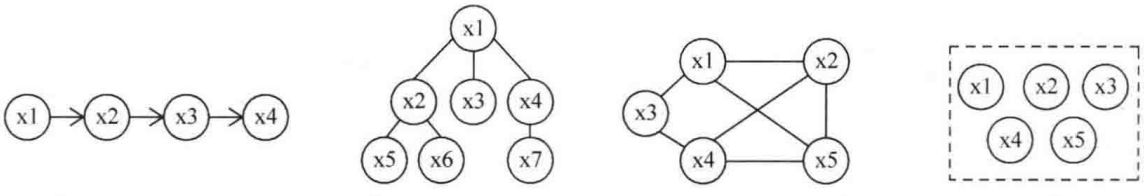
(2) 树状结构。该结构的数据元素之间存在分支的关系,也称为"一对多"关系。可以从一个军队的组成体系来理解:一个团有多个营,一个营有多个连,一个连有多个班,一个班有多名战士等。

(3) 图形结构。该结构的数据元素之间可以有任意的关系,也称为"多对多"关系,图形结构有时也称作网状结构。可以从城市的交通来理解,从一个地点到另外一个地点可能存在很多路径,也可能绕了一圈,又回到原地。

(4) 集合结构。该结构内部的数据关系很松散。集合结构主要强调元素的"整体性"和

元素的“存在性”，数据元素之间的关系基本忽略。离散数学课程中集合关注的是“某个元素是否存在于某个集合中”，而对应于程序设计，这实际上就是查找功能。本书第 3 章会指出这种结构也很重要，通过特殊的算法思想可以使得查找操作达到很快的速度。如在高级语言编译系统中，判断程序设计者使用了哪些变量，以及哪些是定义过的，哪些是没有定义过的，为检查语法错误提供信息，需要在尽可能短的时间内得到结果。

图 1-3 为 4 种逻辑结构的示意图。



(a) 线性结构　(b) 树状结构　(c) 图形结构　(d) 集合结构

图 1-3　4 种逻辑结构的示意图

1.6　存储结构分类

数据结构主要讨论数据的关系以及存储其关系的具体实现，任何一种数据结构都将通过存储结构来体现在计算机中是如何处理的，此时就必然涉及“地址”等概念。

存储地址就是一个存储单元的编号，通常可以认为从 1 开始，这样比较容易理解，但是高级语言的数组大多从 0 开始，更实用和更通用的还有其他进制，在程序设计中还可以降序使用。图 1-4 为各种存储单元地址编号对比示意图。

简单地址编号	1	2	3	4	5	6	7	8	9	10
数组下标地址	0	1	2	3	4	5	6	7	8	9
二进制地址	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
十六进制地址	0116	0117	0118	0119	011A	011B	011C	011D	011E	011F
十进制升序地址	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033
十进制降序地址	1000	0999	0998	0997	0996	0995	0994	0993	0992	0991

图 1-4　各种存储单元地址编号对比示意图

常用的数据存储结构有 4 种：顺序存储、链接存储、索引存储和哈希存储。

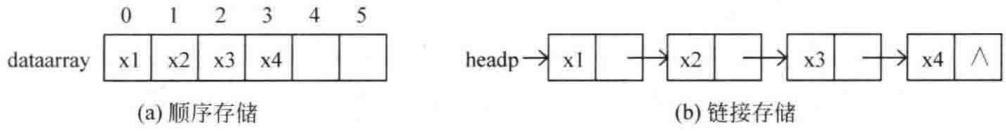
(1) 顺序存储。顺序存储方法是启用一批物理位置相邻的存储单元，然后将逻辑上相邻的元素依次存储在其中。顺序存储结构是一种最基本的存储表示方法，通常借助高级语言中的数组机制实现。

(2) 链接存储。借助高级语言中的指针机制可以实现链表，这种链接存储方法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指针来管理。

(3) 索引存储。这种存储方案的思路是在被处理的正常数据之外，增加一批管理数据来提高查找效率，其原理类似书籍的目录和内容。

(4) 哈希存储(也称为散列存储)。这是一种特殊的存储方案，主要用来提高查找效率。

图 1-5 为顺序存储和链接存储示意图。



(a) 顺序存储　　(b) 链接存储

图 1-5　顺序存储和链接存储示意图

对于某种逻辑结构，这 4 种存储结构并不一定都会采用。作为基本的存储方法，大多数逻辑结构会讨论用顺序存储和链接存储如何实现，而索引存储和哈希存储只有在特殊的情况下才会采用。

存储器一般分为外存和内存。外存的特点是存储容量大，价格相对低，读写速度慢，数据可以永久性保存；内存一般价格相对高，所以计算机配备的内存容量相比外存都小得多，内存读写速度相当快，但是在断电的情况下，信息会全部丢失。顺序存储用到的数组以及链接存储使用的链表都是内存中的存储方法。

1.7　数据结构基本操作

处理数据并不仅仅是为了把数据存入计算机，最终目的是要使用数据，必须通过各种方式来对其中的某些数据或整个结构进行处理，通常把这种处理称为“操作”或“运算”。真实的某个系统中对于数据的“操作”可能是非常多的，比如通过菜单可以看到常用的文字处理软件 Word 的功能就很强大。

以下最基本的操作构成了所有复杂操作的基础，通过“搭积木”式的方法可以演变出更复杂的操作。

基本操作有初始化数据结构、销毁数据结构、读取一个数据、查找一个数据、遍历所有数据、插入一个数据、删除一个数据、判断数据结构是否为空、判断数据结构是否为满、将数据进行排序等。

上述操作被分成两大类：静态操作和动态操作。静态操作指的是操作完成后对于原来的数据或信息量没有产生任何变化性的影响；动态操作指的是经过操作后，或者数据发生了变化，或者信息量发生了变化。进行这样的分类是因为动态操作有一定的危险性，可能会带来副作用，编程时一定要小心并且最好提供回退机制。如在使用某些软件中，想要进行动态操作，会弹出提示窗口，询问是否确定，如果确定就继续，否则就返回。静态操作则可以进行任意多次。

(1) 初始化数据结构。数据结构从无到有被建立的过程称为初始化数据结构，虽然此时其中没有数据，但是结构已经存在。另外也可以是原本有数据，根据某种情况要求清空后回到初始状态。

(2) 销毁数据结构。和上面的操作正好相反，它将一个结构彻底删除。注意，可能会有一些附加的工作，如向操作系统申请了存储单元，则要进行释放，以免造成“内存垃圾”。

(3) 读取一个数据。从一个特定的位置读取一个数据以供使用。

(4) 查找一个数据。目标是寻找一个数据是否存在于该数据结构中。

(5) 遍历所有数据。这是一个重要操作，它应该是每种数据结构(除个别特殊的)的基