高等院校计算机教育系列教材

数据结构

(第二版)

张永宝 主编

- 结构清晰，知识完整
- 入门快速，易教易学
- 实例丰富，实用性强
- 学以致用，注重能力

赠送实例代码和电子课件

清华大学出版社

高等院校计算机教育系列教材

数据结构

(第二版)

张永宝　主编

清华大学出版社
北京

内 容 简 介

本书用C语言描述数据结构。全书共分10章，具体内容包括数据结构的基本概念、线性表、栈、队列、字符串、二叉树、树和森林、图状结构、排序、查找，并作了适当延伸。全书内容安排合理，介绍力求透彻、全面，并对学生在编程中经常出现的一些错误给予了重点提示。本书各章中的示例代码均调试通过。书中每章最后都有习题，并提供电子答案。

本书既可作为高等院校计算机科学与技术专业以及软件工程专业本科生学习数据结构与算法课程的教材，也可作为从事计算机或软件系统开发人员的学习资料。

图书在版编目(CIP)数据

数据结构/张永宝主编；--2版.--北京：清华大学出版社，2013
(高等院校计算机教育系列教材)
ISBN 978-7-302-32820-9

Ⅰ.①数…　Ⅱ.①张　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(2013)第136207号

责任编辑：李春明　郑期彤
封面设计：杨玉兰
版式设计：北京东方人华科技有限公司
责任校对：李玉萍
责任印制：沈　露

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：北京密云胶印厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：18.75　　**字　　数**：456千字
版　　次：2002年1月第1版　2013年8月第2版　　**印　　次**：2013年8月第1次印刷
印　　数：1～3000
定　　价：37.00元

产品编号：048778-01

前　言

“数据结构”课程在整个计算机学科领域中具有重要地位和意义。教育部高校计算机科学与技术教学委员会指出：“数据结构与算法是计算机学科本科教学计划中的骨干基础课程，对学生基本的计算机问题求解能力的培养具有重要意义。作为一门必修课程，该课程既是对以往课程的深入和扩展，也可为将来更加深入学习其他专业课程打下基础。课程中所学习的排序问题的算法以及基本的树、图等数据结构，是计算机学科的基本功。B+树、散列等高级数据结构也是数据库、操作系统、编译原理、计算机网络等后续课程的基础。”

数据结构不仅为程序设计提供了方法论的指导，还在更高层次上总结了现实生活中复杂数据的计算机处理方法，是计算机软件开发、应用人员必备的专业知识。

数据结构是我国高等院校计算机专业的核心课程之一，也是其他信息类专业，如信息管理、通信工程、信息与计算科学等专业的必修课程之一。正因为它在计算机专业培养计划中的重要地位，计算机专业研究生入学考试将数据结构作为必考课程之一。数据结构理论的应用范围已经渗透到计算机专业的各领域中，如操作系统中要使用队列、存储管理表、目录树，数据库系统中要使用线性表、链表、索引树，编译系统中要使用栈、语法树，人工智能中要使用广义表、检索树、图等。在面向对象的程序设计、计算机图形学、多媒体技术、软件工程等领域，也会用到各种不同的数据结构。数据结构课程中自然而完美地整合了计算问题的数学分析建模、基本算法构建和基于高级程序设计语言的具体实现，从而体现出课程的重要意义和强大魅力。

本书是依据教育部制订的关于计算机科学与技术及相关专业的培养目标及教学大纲的要求，通过分析国内外多种同类书籍，结合作者长期的教学与程序设计经验而编写的。本书的算法使用 C 语言进行描述，并配有相关的解释，以帮助读者理解。

本书用 C 语言描述数据结构。全书共分 10 章，具体内容包括数据结构的基本概念、线性表、栈、队列、字符串、 二叉树、树和森林、图状结构、排序、查找，并作了适当延伸。全书内容安排合理，介绍力求透彻、全面，并对学生在编程中常见的一些错误给予了重点提示。书中各章后的示例代码均通过调试。书中每章都有习题，各习题都有电子答案。

本书既可作为高等院校计算机科学与技术专业以及软件工程专业本科生学习数据结构与算法课程的教材，也可作为从事计算机或软件系统开发人员的学习资料。

本书由张永宝执笔，参与本书编写和程序测试的人员还有余健、段德亮、张仁才、仇亚飞、田野、韩忠明、陈嗥、陈运来、代小华等。在本书编写过程中，借鉴了部分国内外优秀教材和相关材料，在此表示衷心感谢。

编　者

目　录

第 1 章　数据结构的基本概念

本章要点

- 数据结构的基本概念和术语。
- 数据的存储结构。
- 算法的描述与算法分析。
- C 语言预备知识，程序结构化。

1.1　数据结构的产生和发展

数据结构是随着电子计算机的产生和发展而发展起来的一门计算机学科。近年来，电子计算机技术飞速发展，这不仅体现在计算机本身运算速度的不断提高、信息存储量的日益扩大，而且更重要的是其应用范围的扩展。

早期的电子计算机主要用于科学计算，所处理的对象是纯数值性的信息。这类问题解题的算法较复杂，但数据量较少并且结构简单。因此早期计算机科学是以研究程序及所描述的算法为中心的。随着计算机广泛地应用于情报检索、事务管理、系统工程等领域，计算机加工处理的对象也从简单的纯数值性信息发展到数、字符、字符串、表、文件、图像、声音等各种复杂的、具有一定结构的数据。人们称前者为数值问题，而后者为非数值问题。

非数值问题要求用复杂的数据结构来描述系统的状态，它们的运算是实现对数据结构的访问或修改。要设计出效率高、可靠性强的非数值程序，要求程序设计人员不但要掌握一般的程序设计技巧，还必须研究计算机程序加工的对象，即研究各种数据的特性以及数据之间的关系，这就促进了数据结构这一学科的发展。然而，数据必须在计算机中进行处理，因此不仅要考虑数据本身的数学性质，还必须考虑数据在计算机内的存储方式和相应的运算，从而扩大了数据结构研究的范围。随着数据库系统、情报检索系统的不断发展，在数据结构的技术中又增加了文件结构，特别是增加了大型文件的组织和 B 树、B+树的知识，使得数据结构逐步成为了一门比较完整的学科。

“数据结构”作为一门独立的课程在国外是从 1968 年才开始设立的。美国唐 • 欧• 克努特教授在 1968 年开创了数据结构的最初体系。数据结构在计算机科学中是一门综合性的专业基础课，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。数据结构是计算机各专业必修的核心课程，是研究计算机程序设计的重要理论和技术的专业基础课程。

1.2 何谓数据结构

随着电子技术的发展，计算机逐渐深入到商业、制造业等人类社会各个领域，从而广泛地应用于数据处理和过程控制。与此相应，计算机能处理的数据也不再是简单的数值，而是字符串、图形、图像和语音等复杂数据。这些复杂数据不仅量大，而且具有一定的结构。数据结构所研究的就是这些有结构的数据，是研究数据对象的特征以及数据的组织和处理方式。在计算机科学中，数据结构(Data Structure)是计算机中存储、组织数据的方式，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。通常情况下，精心选择的数据结构可以带来最优效率(Optimal Efficiency)的算法，数据结构往往同高效的检索算法和索引技术有关。

一般而言，数据结构的选择首先会从抽象数据类型的选择开始。一个设计良好的数据结构，应该在使用尽可能少的时间与空间资源的前提下，为各种临界状态下的运行提供支持。数据结构可通过编程语言所提供的数据类型、引用(Reference)及其他操作加以实现。不同种类的数据结构适合于不同种类的应用，而有些甚至专门用于特定的作业任务。例如，当计算机网络依赖于路由表运作时，B 树高度适用于数据库的封装。

在许多类型的程序设计中，选择适当的数据结构是一个主要的考虑因素。许多大型系统的构造经验表明，封装的困难程度与最终成果的质量与表现都取决于是否选择了最优的数据结构。在许多时候，确定了数据结构后便能很容易地得到算法。而有些时候，方向则会颠倒过来。例如当某个关键作业需要特定数据结构下的算法时，会反过来确定其所使用的数据结构。然而，不管是哪种情况，数据结构的选择都是至关重要的。

建立数学模型的实质是对现实事物和问题进行分析，从中提取出对象以及对象之间特有的关系。下面来看具体的问题。

【例 1.1】企业员工信息管理问题。

各企业都有不同数量的员工，少则几十人，多则上万人。在计算机系统中应该怎样管理好这些员工的信息数据呢？员工信息中反映了与每个员工相关的数据，如工号、姓名、性别、部门、技术职称、出生年月、家庭住址等，如表 1.1 所示。

表 1.1 员工信息表

工 号	姓 名	性 别	部 门	技术职称	出生年月	家庭住址
GH20120001	张三	男	软件部	高级工程师	81/02/02	北京海淀区中关村大街 1 号
GH20120002	李四	女	硬件部	高级工程师	78/08/02	上海徐家汇 12 号
GH20120003	王五	女	行政部	无	82/01/15	北京海淀区西四环北路 14 号楼 101
GH20120004	陈亮	男	销售部	助理工程师	90/08/31	北京西城区三里河三区 8 栋 11 号
GH20120005	赵立国	男	销售部	无	87/08/31	北京市海淀区大钟寺 312 号
…	…	…	…	…	…	…

员工信息以表的形式存入计算机后，对员工信息的处理工作就转化为对数据表的处理。例如，查询员工信息的处理为在表中查询基本信息，对新入职员工的处理为在表中增加一行基本信息，对员工离职的处理为删除表中相应的行。因此，由计算机实现企业员工信息管理问题抽象出的描述模型就是：①建立包含每个员工基本信息的表；②对该表进行插入、删除、修改、查询等操作。

在这类数学模型中，每一行元素之间的关系是一种线性关系，这类数学模型是一种线性表数据结构。

【例 1.2】组织机构管理问题。

组织机构包括行政组织机构和企业组织机构。某企业组织机构如图 1.1 所示。

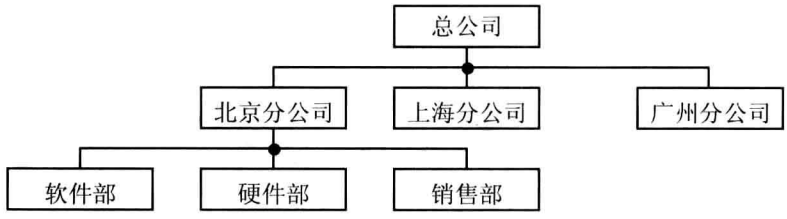


图 1.1　组织机构图

从图 1.1 中可以看出，组织机构管理问题中所描述的模型就是建立具有层次结构的组织机构。这类模型是具有层次结构的数据模型，是一种树形结构，适合于每个家庭中的家谱、行政组织机构、高校院系关系等。把它称为“树”是因为它看起来像一棵倒挂的树，也就是说它是根朝上而叶朝下的。

【例 1.3】学校排课问题。

学生在校期间都要按规定学习一系列的课程，有的课程需要另外的课程作为先修课。下面以软件专业为例，列出一个课程关系表，共 12 门，如表 1.2 所示。排课时怎样排才合理呢？当然，排一门课程的首要条件是其先修课全部排完。这是一种特殊的排序问题。

表 1.2　课程关系表

课程编号	课程名称	先 修 课
C_1	程序设计基础	—
C_2	C 语言	C_1
C_3	数据结构	C_1, C_2
C_4	汇编语言	C_1
C_5	计算机组成原理	C_3, C_4
C_6	数据库	C_{11}
C_7	编译原理	C_3, C_5
C_8	操作系统	C_3, C_6

续表

课程编号	课程名称	先 修 课
C_9	高等数学	—
C_{10}	线性代数	C_9
C_{11}	离散数学	C_9
C_{12}	数值分析	C_1, C_9, C_{10}

图 1.2 所描述的数据模型是表 1.2 所列出的课程依赖关系的图示，这种数据结构称为“图”。一个图看起来是由一些点(如 C_1，C_2 称为顶点)和连接这些圆点的直线或曲线(称为边)组成的。

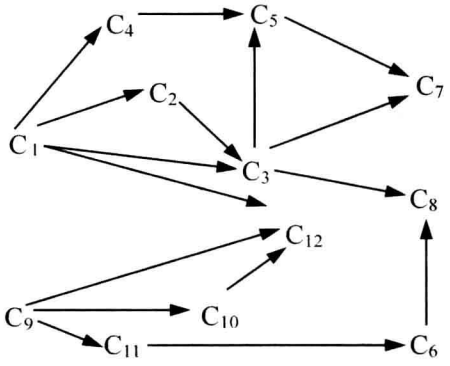


图 1.2 课程关系图

上述三个例子分别描述了线性表、树形和图的数据结构。人们将数据归纳为某种逻辑结构来进行相应运算，完成数据处理过程。

1.3 基 本 术 语

数据(Data)：计算机中的数据是广义的，包括了数、字符、字符串、表、文件等，声音、图形、图像都属于数据的范畴。数据指计算机加工的“原料”，是对客观事物采用计算机能够识别、存储和处理的形式所进行的描述。数据就是计算机化的信息，是计算机中符号化的特定表示形式。

数据元素(Data Element)：数据元素是数据的基本单位，是数据集合的个体，在计算机中通常作为一个整体进行考虑和处理。一个数据元素可由若干个数据项组成。此时的数据元素通常称为**记录**(Record)。如例 1.1 中描述的某一条员工信息。

数据项(Data Item)：数据项是不可分割的、含有独立意义的最小数据单位，有时也称为字段(Field)或域，如员工记录中的姓名、性别等。

数据对象(Data Object)：数据对象是性质相同的数据元素的集合。它是数据的一个子集。例如，整型数据对象是集合{0,±1,±2,…}，字符数据对象是{a,b,c,…}等。数据对象可以是有限的，也可以是无限的。

数据类型(Data Type)：数据类型是高级程序设计语言中的概念，是数据的取值范围和对数据进行操作的总和，如 C 语言中的整型及定义在其上的一组操作(加、减、乘、除等)。数据类型中定义了两个集合，即类型的取值范围和该类型可允许使用的一组运算。例如，高级语言中的整型、实型、字符型就是已经实现的数据类型的实例。从这个意义上讲，数据类型是高级语言中允许的变量种类，是程序语言中已经实现的数据结构(即程序中允许出现的数据形式)。在 C 语言中，整型(int)可能的取值范围是-32768～+32767，可用的运算符集合为加、减、乘、除、取模(+、-、*、/、%)。从硬件的角度来看，这些运算的实现涉及“字”、“字节”、“位”、“位运算”等；而从用户的观点来看，用户并不需要了解整数在计算机内是如何表示、运算细节是如何实现的，只需要了解整数运算的外部运算特性，就可以运用高级语言进行程序设计了。

抽象数据类型(Abstract Data Type，ADT)：抽象数据类型是指一些数据以及对这些数据所进行的操作的集合。数据之间有一定的关系，抽象数据类型不同，数据之间的关系就不同。对数据进行的操作由数据间的关系决定，数据间的关系不同，对数据的操作也不同。这些操作既向程序的其余部分描述了这些数据是怎么样的，也允许程序的其余部分改变这些数据。一个 ADT 可能是一个图形窗体以及所有能影响到该窗体的操作，也可以是一个文件以及对这个文件进行的操作，或者是一张保险费率表及相关操作等。

抽象数据类型是基于一类逻辑关系的数据类型以及定义在这个类型之上的一组操作。抽象数据类型的定义取决于客观存在的一组逻辑特性，而与其在计算机内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，都不影响其外部使用。从某种意义上讲，抽象数据类型和数据类型实质上是一个概念。整数类型就是一个简单的抽象数据类型实例。“抽象”的意义在于数学特性的抽象。一个抽象数据类型定义了一个数据对象，数据对象中各元素间的结构关系，以及一组处理数据的操作。抽象数据类型通常是指由用户定义用以表示应用问题的数据模型，通常由基本的数据类型组成，并包括一组相关的操作。

抽象数据类型包括定义和实现两方面，其中定义是独立于实现的。抽象数据类型的特征是使用与实现分离，实现封装和信息隐蔽，也就是说，在抽象数据类型设计时，类型的定义与其实现分离。另一方面，抽象数据类型的含义更广，不仅限于各种不同的计算机处理器中已定义并实现的数据类型，还包括设计软件系统时用户自己定义的复杂数据类型。所定义的数据类型的抽象层次越高，含有该抽象数据类型的软件的复用程度就越高。抽象数据类型定义该抽象数据类型需要包含哪些信息，并根据功能确定公共界面的服务，用户可以使用公共界面中的服务对该抽象数据类型进行操作。从用户的角度来看，只要了解该抽象数据类型的规格说明，就可以利用其公共界面中的服务来使用这个类型，而不必关心其物理实现，从而集中考虑如何解决实际问题。

抽象数据类型是近年来计算机科学中提出的最重要的概念之一，它集中体现了程序设计中一些最基本的原则：分解、抽象和信息隐藏。严格地用代数系统形式定义一个抽象数据类型，可以把抽象数据类型看成是定义了一组运算的数学模型。

ADT 的定义采用下述格式：

```
ADT<ADT 名>
{
```

```
        数据对象：<数据对象的定义>
        数据关系：<结构关系的定义>
        基本操作：<基本操作的定义>
    } ADT<ADT 名>
```

其中，数据对象和结构关系的定义采用数学符号和自然语言描述，而基本操作的定义格式为 C 语言基本函数的定义形式，如下：

```
函数类型  函数名([参数列表])
[类型定义列表]
{
    函数体
}
```

参数列表中的参数有两种：第一种参数只为操作提供待处理数据，又称值参；第二种参数既能为操作提供待处理数据，又能返回操作结果，也称变量参数。操作前提描述了操作执行之前数据结构和参数应满足的条件，操作结果描述了操作执行之后数据结构的变化状况和应返回的结果。抽象数据类型可用现有计算机语言中已有的数据类型，即固有数据类型来表示和实现。

数据结构(Data Structure)：数据结构是相互之间存在一种或多种关系的数据元素的集合。例如表 1.1 员工基本信息中数据元素集合是员工记录集，而数据元素集合上的关系就是它们在员工基本信息表中的前驱和后继的关系，这种数据之间的关系是一对一的，是线性的。数据元素相互之间的关系称为结构(Structure)。

在例 1.2 中描述的数据元素集合是组织机构的集合，而组织机构之间的关系就是数据元素之间的关系。这种数据元素相互之间的关系也是一种结构，称为层次结构或树形结构。

数据结构的形式定义为

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

其中，D 是数据元素的有限集；S 是 D 上关系的有限集。

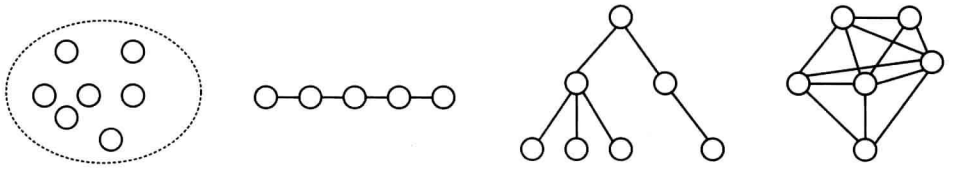
根据数据元素之间关系的不同特性，通常有下列 4 类基本的数据结构。

(1) 集合(Set)：该结构中的数据元素除了存在“同属于一个集合”的关系外，不存在任何其他关系，如图 1.3(a)所示。

(2) 线性结构(Linear Structure)：该结构中的数据元素存在着一对一的关系，如图 1.3(b)所示。

(3) 树形结构(Tree Structure)：该结构中的数据元素存在着一对多的关系，如图 1.3(c)所示。

(4) 图状结构(Graphic Structure)：该结构中的数据元素存在着多对多的关系，如图 1.3(d)所示。



(a) 集合　(b) 线性结构　(c) 树形结构　(d) 图状结构

图 1.3　基本的数据结构

有时树形结构和图状结构也称为非线性结构。

【例 1.4】某课题小组中有 1 位教授，可以带 1～3 名研究生，每名研究生可以带 1～2 名本科生。设计该课题小组的数据结构。

设课题小组的数据结构为

$$Group = (P, R)$$

其中，P 表示数据对象；R 表示数据对象中数据元素的关系。

P、R 的描述如下：

$$P = \{T, G_1, \cdots, G_n, S_{11}, \cdots, S_{nm}\} \quad (1 \leqslant n \leqslant 3,\ 1 \leqslant m \leqslant 2)$$

$$R = \{R_1, R_2\}$$

$$R_1 = \{<T, G_i> \mid 1 \leqslant i \leqslant n,\ 1 \leqslant n \leqslant 3\}$$

$$R_2 = \{<G_i, S_{ij}> \mid 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m,\ 1 \leqslant n \leqslant 3,\ 1 \leqslant m \leqslant 2\}$$

其中，T 表示教授；G 表示研究生；S 表示本科生。

严格地说，数据结构包括两方面的内容：数据的逻辑结构(Logical Structure)和数据的物理结构(Physical Structure)。例如，例 1.4 中 Group 定义的数据结构是对操作对象的一种数学描述，其中的“关系”是描述数据元素之间的逻辑关系，因此称为数据的逻辑结构。与逻辑结构相对应的是数据的物理结构。数据的物理结构又称为数据的存储结构，是指数据的逻辑结构在计算机中的映像，即数据结构在计算机中的存储方法。由于数据结构包括数据元素集合及数据元素之间的关系，所以数据的存储结构也应该包含这两部分内容，即包括数据元素的映像和元素间关系的映像两部分。数据元素之间的关系在计算机中有两种不同的表示方法，即顺序映像和非顺序映像，并由此得到两种不同的存储结构，即顺序存储结构和链式存储结构。顺序存储结构的特点是借助于元素在连续空间的存储器中的相对位置来表示数据元素之间的逻辑关系。链式存储结构的特点是借助于指示元素存储地址的指针(Pointer)来表示数据元素之间的逻辑关系。

任何一个程序都涉及数据的存储结构。因为对于逻辑结构上的同一种运算，其具体的实现方法依存储结构的不同而不同。所以，当描述问题的模型确定之后，首要问题就是确定其存储结构。

1.4　数据的存储结构

数据的存储结构是指建立一种由逻辑结构到存储空间的映射：对于逻辑结构(K,r)，其中 r ∈R，对它的节点集合 K 建立一个从 K 到存储器 M 的单元映射：K→M，其中每个节点 j ∈K 都对应一个唯一的连续存储区域 c ∈ M。

常用的数据存储结构有两类：一类是顺序存储结构；另一类是链式存储结构。

1.4.1　顺序存储结构

顺序存储结构是指逻辑上相邻的数据元素，其节点的物理位置也相邻，数据元素之间的关系由节点的邻接关系体现。如例 1.1 中给出的员工信息逻辑结构是线性的，各个员工

记录前后是物理相邻的。顺序存储把一组节点存放在按地址相邻的存储单元里，节点间的逻辑关系用存储单元的自然顺序关系来表达，即用一块存储区域存储线性数据结构。顺序存储为使用整数编码访问数据节点提供了便利。由于顺序存储的存储空间除了存储有用数据外，没有用于存储其他附加的信息，因此顺序存储结构一般也被称为紧凑存储结构。计算机的内存单元是一维结构，顺序存储结构很方便实现。

顺序存储结构具有以下特点。

(1) 节点中只存放数据元素本身的信息，无附加内容。

(2) 由于每个节点所占存储空间大小一致，并且按顺序存储，所以只要知道第 1 个节点的地址，就可以通过计算直接确定结构中第 i 个节点的地址，从而直接存取第 i 个数据元素。

(3) 由于可根据公式直接确定第 i 个节点的地址而直接存取第 i 个数据元素，所以数据元素的存取操作速度较快。

(4) 插入、删除数据元素时，由于需要保持数据元素之间的逻辑关系，必须移动大量元素，因此实现起来较慢。关于这一点，将在第 2 章介绍线性表时详细说明。

(5) 顺序存储结构有两种空间分配方式，一种是静态结构，另一种是动态结构。当用静态结构时，存储空间一旦分配完毕，其大小就难以改变。因此，当表中元素个数难以估计时，分配的空间大小也就难以确定。预分配的空间太大会造成浪费，空间太小又可能发生存放不下的溢出现象。

1.4.2 链式存储结构

链式存储结构是指在节点的存储结构中附加指针字段来存储节点间的逻辑关系。链式存储中数据节点包括两部分：数据字段存放节点本身的数据；指针字段存放指向其后继节点的指针。链式存储方法适用于那些需要经常进行增删节点的复杂数据结构。

在链式存储结构中，逻辑上相邻的数据元素，其节点的物理位置不一定相邻，因此节点之间是否邻接并不能反映数据元素的逻辑顺序。在这种情况下，存储结构要反映数据元素在逻辑结构中的关系，只有在节点中增加信息来指明其与其他节点之间的关系，这个增加的信息就是指针。前一个节点的指针指向后一个节点，多个节点的指针一起形成一个链，因此称这种存储结构为链式存储结构。链式存储结构既可用于实现线性数据结构，也可用于实现非线性数据结构。对于非线性数据结构来说，每个数据元素在逻辑上可能与多个数据元素相邻，而计算机内存的一维地址结构限制了每一个节点只能与前、后各一个节点相邻，节点的相邻反映不出一个元素与多个元素的相邻关系，所以非线性逻辑结构只能用链式存储结构来实现，树、图等就是这种非线性数据结构。在链式存储结构的每个节点中，数据域可分为两类：一类用于存放数据元素本身的信息，称为信息域；另一类用于存放指针，称为指针域。

链式存储结构的主要特点如下。

(1) 节点中除存放数据元素本身的信息外，还需存放附加的指针。

(2) 不能直接确定第 i 个节点的存储位置。要存取第 i 个节点的信息，必须从第 1 个节点开始查找，沿指针顺序取出第 i-1 个节点的指针域，再取出第 i 个节点的信息，存取速度

较慢。

(3) 链式存储结构的一个主要优点是插入、删除数据元素时不必移动其他数据元素，速度较快。因此，当数据元素个数变动较大、插入删除操作频繁时，可用链式存储结构来实现。

(4) 链式存储结构是一种动态存储结构，当数据元素数量增加时可随时申请所需的空间，删除数据元素时无用的空间可归还给系统，故空间利用率较高，也不存在预分配空间的问题。

一般来说，一种数据结构既可用顺序存储结构实现，也可用链式存储结构实现。究竟用哪种存储结构，应根据具体情况来选择。选择时的主要依据有两个：一个是考虑数据结构上要执行的主要操作；另一个是看能否估计出数据元素的数目。若执行的主要操作是插入或删除，则最好用链式存储结构，否则应该用顺序存储结构；若预先可以估计出元素的个数，则可以采用顺序存储结构，否则宜用链式存储结构。

1.4.3 其他存储结构

数据结构的存储结构可采用顺序存储结构或链式存储结构，也可采用二者相结合的方式。除此之外，索引存储是顺序存储的一种推广，用于大小不等的数据节点的顺序存储。索引存储通过建造一个由整数域 Z 映射到存储地址域的函数，把整数索引值映射到节点的存储地址，从而形成一个存储一串指针的索引表，每个指针指向存储区域的一个数据节点。而散列法作为索引存储的一种延伸和扩展，利用散列函数进行索引值的计算，然后通过索引表求出节点的指针地址。

1.5 算法及算法分析

1.5.1 算法

算法(Algorithm)是为解决特定问题而规定的一系列操作，是为完成某一特定任务的有限命令的集合。一个算法应该具备以下 5 个特性。

(1) 有穷性(Finity)。一个算法总是在执行有穷步之后结束，即算法的执行时间是有限的。

(2) 确定性(Unambiguousness)。算法的每一个步骤都必须有确切的含义，即无二义性，并且对于相同的输入只能有相同的输出。

(3) 输入(Input)。一个算法具有零个或多个输入，即输入是在算法开始之前给出的量。这些输入是某数据结构中的数据对象。

(4) 输出(Output)。一个算法具有一个或多个输出，并且这些输出与输入之间存在着某种特定的关系。

(5) 可实现性(Realizability)。算法中的每一步都可以通过已经实现的基本运算的有限

次运行来实现。

算法的含义与程序非常相似，但两者有区别。一个程序不一定满足有穷性。例如操作系统，只要整个系统不遭破坏，它将永远不会停止。另外，一个程序只能用计算机语言来描述，也就是说，程序中的指令必须是机器可执行的，而算法不一定用计算机语言来描述，如自然语言、框图、伪代码都可以描述算法。

1.5.2 算法的评价

算法和数据结构密切相关。对于一个特定的问题，采用的数据结构不同，其设计的算法一般也不同，即使在同一种数据结构下，也可以采用不同的算法。数据结构的选择直接影响着算法的效率。那么，对于解决同一问题的不同算法，选择哪一种算法比较合适，以及如何对现有的算法进行改进，从而设计出更适合于数据结构的算法，这就是算法评价的问题。

评价一个算法一般从正确性、可读性、健壮性、时间效率和空间效率几方面来考虑。

(1) 正确性(Correctness)。算法的执行结果应当满足预先规定的功能和性能的要求，这是评价一个算法的最重要也是最基本的标准。算法的正确性还包括对于输入、输出处理的明确而无歧义的描述。

(2) 可读性(Readability)。算法主要是为了人阅读和交流，其次才是机器的执行。所以，一个算法应当思路清晰、层次分明、简单明了、易读易懂。即使算法已转变成机器可执行的程序，也需要考虑人能较好地阅读理解。同时，一个可读性强的算法也有助于对算法中隐藏错误的排除和算法的移植。

(3) 健壮性(Robustness)。一个算法应该具有很强的容错能力，当输入不合法的数据时，算法应当能作适当的处理，使得不至于引起严重的后果。健壮性要求算法要全面细致地考虑所有可能出现的边界情况，并对这些边界情况作出完备的处理，尽可能使算法没有意外的情况。

(4) 时间效率(Running Time)。时间效率是指算法在计算机上运行所花费的时间，它等于算法中每条语句执行时间的总和。对于同一个问题，如果有多个算法可供选择，应尽可能选择执行时间短的算法。一般来说，执行时间越短，性能越好。

(5) 空间效率(Storage Space)。空间效率是指算法在计算机存储上所占用的存储空间，包括存储算法本身所占用的存储空间、算法的输入及输出数据所占用的存储空间和算法运行过程中临时占用的存储空间。算法占用的存储空间是指算法执行过程中所需要的最大存储空间，对于同一个问题，如果有多个算法可供选择，应尽可能选择存储量需求低的算法。实际上，算法的时间效率和空间效率经常是一对矛盾，相互抵触。在运用中要根据问题的需要进行灵活处理，有时需要牺牲空间来换取时间，有时需要牺牲时间来换取空间。

通常把算法在运行过程中临时占用的存储空间的大小称为算法的空间复杂度(Space Complexity)。算法的空间复杂度比较容易计算，它主要包括局部变量所占用的存储空间和系统为实现递归所使用的堆栈占用的存储空间。