TURING 高等院校计算机教材系列

数据结构

基于C++模板类的实现

余腊生　等编著

人民邮电出版社
POSTS & TELECOM PRESS

數據結構

基於C++語言描述的實現

TURING 高等院校计算机教材系列

数据结构

基于C++模板类的实现

余腊生　等编著

人民邮电出版社
北　京

图书在版编目（CIP）数据

数据结构：基于C++模板类的实现 / 余腊生等编著．
北京：人民邮电出版社，2008.11
（高等院校计算机教材系列）
ISBN 978-7-115-18643-0

Ⅰ.数… Ⅱ.余… Ⅲ.①数据结构－高等学校－教材②C语言－程序设计－高等学校－教材 Ⅳ.TP311.12 TP312

中国版本图书馆CIP数据核字（2008）第122936号

内 容 提 要

本书采用能够自然体现抽象数据类型概念的C++语言作为算法描述语言，把数据结构的原理和算法分析技术有机地结合在一起。全书内容包括线性表、栈、队列、递归、广义表、字符串、数组、树、图、查找以及各种排序算法，并给出了相关的实验指导。书中还引入了一些比较高级的数据结构和相关的算法分析技术。

本书可作为高等院校计算机或相关专业的教材，也可以作为其他程序类课程的辅导教材，同时也适用准备参加研究生入学考试、自学考试和各类程序设计竞赛的人员阅读。

高等院校计算机教材系列

数据结构：基于C++模板类的实现

◆ 编　　著　余腊生等
责任编辑　杨海玲　刘　静

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京昌平百善印刷厂印刷

◆ 开本：787×1092　1/16
印张：22.75
字数：734千字　2008年11月第1版
印数：1－4 000册　2008年11月北京第1次印刷

ISBN 978-7-115-18643-0/TP

定价：39.00元

读者服务热线：(010)88593802　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

学生的头脑不是一个等待填满的容器，而是一支需要被点燃的火把。

——杨家福

我们生活在一个物质世界中，同时又时刻面对着一个数字的世界。如果将物质世界中的事与物数字化，那么它们在计算机中均表现为数据。这些数据来源于现实，具有具体的含义，而且在计算机中有着统一的表示方法，因而成为计算机程序处理的符号集合。研究数据在计算机中的表示方法、关联方法、存储方法以及典型处理方法，正是数据结构课程的主要内容。

早在20世纪80年代初，“数据结构”就已成为国内高校计算机专业教学计划中的核心课程。目前，ACM/IEEE计算学科课程计划（CC-2001）已将算法与数据结构类课程列为核心课程之首，数据结构愈益显现出其在信息学科理论中的重要地位。

在软件系统的开发过程中，数据结构的思维方法在本质上不同于常规数学训练的公理系统思维方法，而是一种算法构造性思维方法。所谓构造性思维方法就是要让学生理解、习惯并掌握算法的一种思维方法，也是本门课程教学的重要内容和主要难点。要成为专业的软件开发人员，仅仅知道开发工具的语言规则和简单使用过程是远远不够的，只有培养自身的数据抽象能力、算法设计能力，掌握创造性思维方法，举一反三，触类旁通，才能够具备应用知识解决复杂问题的能力。

本书是作者根据多年的教学经验，结合当前计算机科学技术的发展，并参考了众多的数据结构教材编著而成的。书中采用了能够自然体现抽象数据类型概念的C++语言作为算法描述语言，内容覆盖了数据结构课程的教学大纲，从数据类型角度系统地介绍了各种数据结构的逻辑特性、存储表示方法及基本操作算法，并针对常用的数据结构进一步讨论了各种应用算法及其实现方法。本书旨在培养学生分析计算机加工数据对象特征的能力，以便在实际操作中选择合适的数据结构和存储结构以及相应的算法，同时帮助学生掌握估算算法时间复杂度和空间复杂度的基本技巧。

本书将数据结构的原理和算法分析技术有机地结合在一起，使用了参数化的模板，提高了算法中数据类型的通用性，支持高效的代码重用。书中介绍和分析重点设计思想时，结合了大量图解和具体示例，使读者能够联系实际，掌握数据结构的实质内容。此外，本书还介绍了一些比较高级的数据结构及相关的算法分析技术。

习题的选择和设计是教材编写的一个重要方面，本书在这方面充分考虑了习题的完整性、系统性和典型性。概念题覆盖了每章基本内容要点，读者学习完每章后，应以这部分习题为提纲，透彻理解每章的基本内容以及相关的基本概念；算法设计题既是每章内容的延伸，也是对读者理解程度的检查。另外，数据结构课程要安排一定课时的上机实习，实习题就是专为此目的而设计的。

全书包括10章和一个附录。第1章综述数据、数据结构和抽象数据类型的基本概念，介绍算法分析和评价的基本思想；第2章讲述的线性表是一种最典型的线性结构；第3章介绍的栈和队列是一些常用的操作受限的特殊线性表；第4章介绍程序设计的基本技术（递归技术）以及线性表的推广（广义表）；第5章介绍应用广泛的字符串结构类型；第6章讨论的是工程中应用非常普遍的数组与矩阵类型；第7章讨论树形结构，包括树和二叉树结构；第8章讨论可以表达复杂数据关系并且应用非常广泛的图结构；第9章讨论查找方法及数据的组织结构；第10章介绍各种常见的排序方法；附录介绍数据结构

的实验环节，包括课程设计的规范和示例。

使用本教材授课约需54～70课时，其中包括10课时左右的上机实习。课时数较少时，书中标有“*”号的内容可不讲授。

本书可作为普通高等院校计算机相关专业数据结构课程的教材，也可作为信息类相关专业的教材和教学参考书，同时也是自学考试应试人员、程序设计竞赛参赛人员和软件开发人员的参考资料。本书对于准备参加研究生入学考试的人员以及从事计算机应用工作的科技工作者，也是一本实用的参考书。

在本书的成书过程中，我们得到了人民邮电出版社图灵公司的大力支持。中南大学信息学院教授、博士生导师陈松乔教授为本书的编写提出了不少建议。此外，中南大学信息学院计算机专业的研究生李徐、洪飞、王鹏人、徐漾等同学参与了书中有关代码的设计与调试工作，建立了中南大学数据结构精品课程网站。在此一并表示感谢。

由于作者水平所限，书中难免出现疏漏和错误之处，恳请广大读者指正。

作者

2008年7月

目　录

第1章 绪 论

计算机并不解决问题，它们只是执行解决方案。

——劳伦特·加瑟

学习目标

- 掌握基本概念，特别是数据的逻辑结构和存储结构之间的关系。
- 理解抽象数据类型的定义、表示和实现方法。
- 理解算法5个要素的确切含义。
- 掌握参数的传递方法。
- 掌握计算语句频度和估算算法时间复杂度的方法。

计算机科学是一门研究数据表示和数据处理的科学。数据是计算机可以直接处理的最基本和最重要的对象。无论是科学计算、数据处理和过程控制，还是对文件的存储和检索，都是对数据进行处理的过程。因此，要设计出一个结构好、效率高的程序，必须研究数据的特性、数据间的相互关系及与其对应的存储结构，并利用这些特性和关系设计出相应的算法。

1.1 数据结构的概念

美国计算机科学家Donald Ervin Knuth首次提出了数据结构和算法的概念，并在他所著的《计算机程序设计艺术 第1卷 基本算法》中首次较系统地阐述了数据的逻辑结构、存储结构及其操作。随后，美国的一些大学开始设立与数据结构相关的课程。同时，结构化程序设计逐渐成为当时程序设计的主流方法，人们也越来越重视数据结构。

数据结构作为计算机科学的一门分支学科，主要研究非数值计算的程序设计问题中计算机的操作对象、对象之间的关系和操作等。为了编写出“好”的程序，必须分析待处理对象的特性及各对象之间的关系。

数据结构的内容包括3个层次的5个“要素”，如表1-1所示。其中，3个层次分别为抽象、实现与评价。通过抽象，舍弃数据元素的具体内容，就得到逻辑结构表示，通过分解将处理要求划分成各种功能，再通过抽象舍弃实现细节，就得到基本操作的定义。上述两个方面的结合将问题变换为数据结构，这是一个从具体（即具体问题）到抽象（即数据结构）的过程。最后，通过增加对实现细节的考虑进一步得到存储结构和实现运算，从而完成设计任务，而这是一个从抽象（即数据结构）到具体（即具体实现）的过程。熟练掌握这两个过程是数据结构课程在专业技能培养方面的基本目标。在这个过程中需要对不同结构及其实现的性能进行评价，从而获得最佳实践方案。

表1-1 数据结构课程内容体系

层次 \ 方面	数据表示	数据处理
抽象	逻辑结构	基本操作
实现	存储结构	算法
评价	不同数据结构的比较及算法分析	

针对要处理的问题，设计最有利于操作系统处理的数据结构时需考虑下列因素。

(1) 数据的数量。

(2) 数据的使用次数和方式。

(3) 数据的性质是动态的还是静态的。

(4) 数据结构化后需要多大的存储空间。

(5) 存取结构化后的数据所需花费的时间。

(6) 是否容易程序化。

目前，“数据结构”已经是计算机科学及相关专业的一门专业基础课，它与数学、计算机硬件和计算机软件之间的关系如图1-1所示。在计算机科学中，数据结构不仅是一般程序设计的基础，而且是设计和实现编译程序、操作系统以及数据库系统等大型程序的基础。无论是系统软件还是应用软件都要用到各种类型的数据结构。因此，学好“数据结构”这门课程，对学习计算机专业的其他课程是十分有益的。

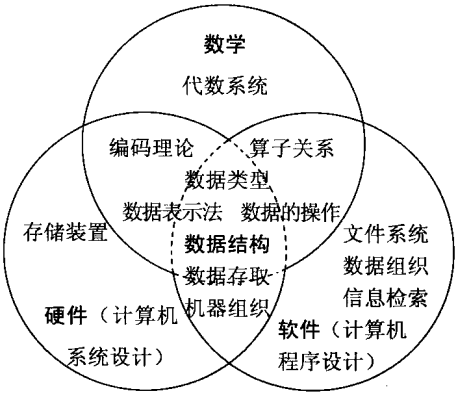

图1-1 数据结构的地位

1.1.1 为什么要学习数据结构

在计算机发展的初期，人们使用计算机的主要目的是处理数值计算问题。使用计算机解决具体问题一般需要经过以下几个步骤：首先从具体问题抽象出适当的数学模型，然后设计或选择解此数学模型的算法，接着编写程序并进行调试、测试，直至得到最终的解。

由于最初涉及的运算对象是简单的整型、实型或布尔型数据，所以程序设计者的主要精力集中于程序设计的技巧上，而无需重视数据结构。随着计算机应用领域的扩大和软硬件的发展，非数值计算问题显得越来越重要。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加以描述。因此，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。

著名的瑞士计算机科学家沃思（N. Wirth）教授曾提出：

$$\text{算法} + \text{数据结构} = \text{程序}$$

程序设计的实质是对实际问题设计、选择好的数据结构和好的算法，而好的算法在很大程度上取决于描述实际问题的数据结构。

例1.1 电话号码查询问题。编写一个查询某个私人电话号码的程序。要求任意给出的一个姓名，如果找到这个人，则迅速找到其对应的电话号码；否则指出没有这个人的电话号码。

要解决此问题首先需构造一张电话号码登记表。表中每个结点存放两个数据项：姓名和电话号码。

能否写出好的查找算法，取决于这张表的结构及存储方式。最简单的方式是将表中结点顺序地存储在计算机中。查找时从头开始依次查对姓名，直到找出正确的姓名或是遍历整个表都没有找到为止。这种查找算法对一个有少量私人电话的区域或许是可行的，但对一个有成千上万私人电话的城市就不适用了。若这张表是按姓氏排列的，则可另建一张姓氏索引表，采用如图1-2所示的存储结构。查找时先在左边的索引表中查对姓氏，然后根据索引表中的地址到右边的电话号码登记表中核查姓名，这样查找登记表时就无需查找其他姓氏的名字了。因此，在这种新的结构上产生的查找算法就更加有效。

诸如此类的还有电话自动查号系统、考试查分系统、仓库库存管理系统等。在这类文档管理的数学模型中，计算机处理的对象之间通常存在着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。

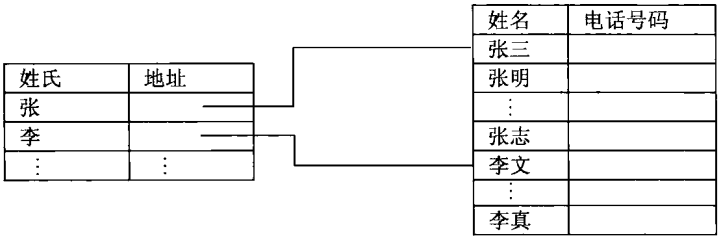

图1-2 电话号码的索引存储

例1.2 四皇后问题，即在一个4×4的棋盘上放置4个皇后，使得任意两个皇后在行、列和斜方向上都不在一条线上。

在四皇后问题中，处理过程不是根据某种确定的计算法则求解的，而是利用试探和回溯的探索技术求解。为了求得合理布局，在计算机中要存储布局的当前状态。从最初的布局状态开始，一步步地进行试探，每试探一步形成一种新的状态，整个试探过程形成了一棵隐含的状态树，如图1-3所示。回溯法求解过程实质上就是一个遍历状态树的过程。在这个问题中所出现的树也是一种数据结构，它可以应用在许多非数值计算的问题中。

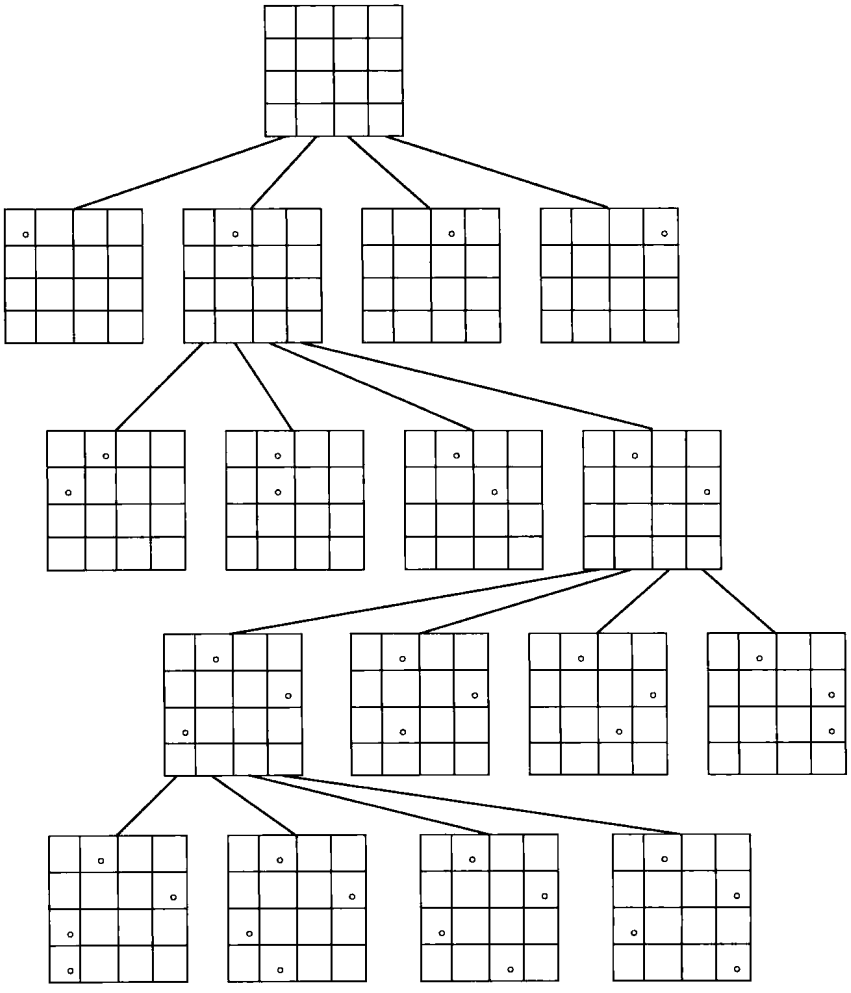

图1-3 四皇后问题中隐含的状态树

例1.3 田径赛的时间安排问题。假设某校的田径选拔赛共设6个项目，即跳高（*A*）、跳远（*B*）、标枪（*C*）、铅球（*D*）、100米短跑（*E*）和200米短跑（*F*），规定每个选手至多参加3个项目的比赛。现有5名选手报名参赛，选手所选择的项目如表1-2所示。要求设计一个竞赛日程安排表，使得在尽可能短的时间内完成比赛。

(1) 首先选择一种合适的数据结构，如图1-4所示。图中顶点代表竞赛项目，边代表所连接的两个竞赛项目的比赛时间不发生冲突，因此一个选手选择的项目间应该两两有边相连。

(2) 竞赛项目的时间安排问题可以抽象为对无向图进行“着色”操作，即用尽可能少的颜色给图中每个顶点着色，使得任意两个有边连接的相邻顶点具有不同的颜色。每一种颜色表示一个比赛时间，同一种颜色表示可以安排在同一时间进行比赛。由此可得，只要安排4个不同的比赛时间即可。比如，在时间1比赛跳高（*A*）和标枪（*C*），在时间2比赛跳远（*B*）和铅球（*D*），在时间3和时间4分别比赛100米（*E*）和200米（*F*）。

表1-2 参赛选手比赛项目表

姓名	项目1	项目2	项目3
丁一	跳高	跳远	100米
马刚	标枪	铅球	—
张明	标枪	100米	200米
李远强	铅球	200米	跳高
王立	跳远	200米	—

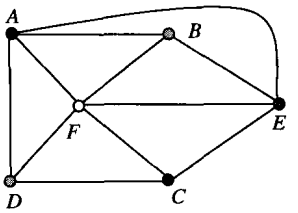

图1-4 安排竞赛项目的数据结构模型

由以上3个例子可见，描述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。相应地，解决问题的关键步骤是设计合适的数据结构来表示问题，然后才能写出有效的算法。

1.1.2 相关概念和术语

在系统地学习数据结构知识之前，你应该先了解一些基本概念和术语。

数据（data）是信息的载体，能够被计算机识别、存储和处理。它可以是数值数据，也可以是非数值数据，如字符、文字、图形、图像和语音等。

数据元素（data element）是数据的基本单位。在不同的条件下，数据元素又可称为*元素*、*结点*、*顶点*和*记录*等。

一个数据元素可由若干个*数据项*（data item）组成。例如，学生信息表中的一条记录可以包括学号、姓名、性别、籍贯、出生年月和成绩等数据项。数据项可以分为两种：一种叫*原子项*（如学生的性别、籍贯等），是数据处理时不能再分割的最小单位；另一种叫组合项（如学生的成绩，可以再划分为数学成绩、物理成绩和化学成绩等更小的项）。

数据对象（data object）或*数据元素类*（data element class）是具有相同性质的数据元素的集合。在某个具体问题中，数据元素都具有相同的性质（元素值不一定相等），是数据元素类的一个实例。例如，例1.3中，所有顶点的集合是一个数据元素类，顶点*A*和顶点*B*各自代表一个运动项目，是该数据元素类中的两个实例。

数据结构（data structure）是指互相之间存在着一种或多种关系的数据元素的集合。根据数据元素之间关系的不同特性，通常有下列4类基本结构（示意图见图1-5）。

- *集合结构*。其中所有的数据元素都属于同一个集合。集合是元素关系极为松散的一种结构，因此也可用其他结构来表示。
- *线性结构*。数据元素之间存在一对一的关系。
- *树形结构*。数据元素之间存在一对多的关系。

- 图形结构。数据元素之间存在多对多的关系。图形结构也称作网状结构。

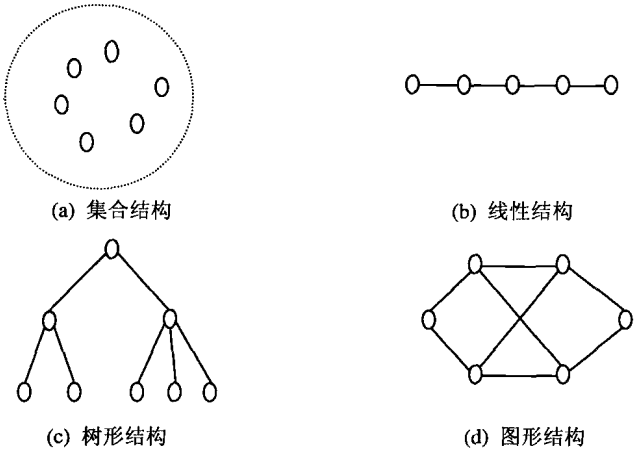

图1-5 4类基本结构

由此可知，一个数据结构具有两个要素：一个是数据元素的集合，另一个是关系的集合。因此在形式上数据结构可以采用一个二元组来表示。

数据结构的形式定义为

$$\text{数据结构} = (D, R)$$

其中，D是数据元素的有限集合，R是D上关系的有限集合。例如，复数可被定义为一种数据结构Complex $= (D, R)$，其中，$D = \{x \mid x \text{是实数}\}$，$R = \{<x, y> \mid x, y \in D, x\text{称为实部}, y\text{称为虚部}\}$。

研究数据结构主要是研究数据的逻辑结构、存储结构及相关操作。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，与数据的存储无关。数据的*存储结构*（物理结构）表示数据结构在计算机中的实现方法，即数据结构中元素及元素间关系的表示。操作是数据结构对外表现的行为或方法。

1. 数据的逻辑结构

数据的逻辑结构描述数据元素之间的逻辑关系，分为*线性结构*和*非线性结构*。线性结构的逻辑特征是，若结构是非空集，则有且仅有一个首元素结点和一个尾元素结点，并且所有结点都最多只有一个直接前驱和一个直接后继。例如，线性表、栈、队列和串等都是线性结构。非线性结构的逻辑特征是，一个结点可能有多个直接前驱和直接后继。例如，数组、广义表、树和图等都是非线性结构。

2. 数据的存储结构

一般来说，一个存储结构包括以下3个主要部分。

- 存储结点，在不致混淆时简称为结点，每个存储结点存放一个数据元素。
- 数据元素之间的关联方式表示。
- 附加设施，如为便于运算实现而设置的“哑结点”等。

同一逻辑结构采用不同的存储方法，可以得到不同的存储结构。下面是常用的4种基本存储方法。

(1) *顺序存储方法*。该方法把逻辑上相邻的结点存储在物理位置上相邻的存储单元里，结点间的逻辑关系由存储单元的邻接关系来体现。由此得到的存储表示称为*顺序存储结构*。顺序存储方法为使用整数编码访问数据结点提供了便利。顺序存储结构中没有存储其他附加的信息，所以也称为*紧凑存储结构*。这种存储方法主要应用于线性数据结构。非线性数据结构也可通过某种线性化的方法实现顺序存储。

(2) *链接存储方法*。这种存储方法不要求逻辑上相邻的结点在物理位置上亦相邻，结点间的逻辑关系由附加的指针字段表示。由此得到的存储表示称为*链式存储结构*，通常借助于程序语言的指针类型描述。链接存储方法适用于需要经常进行增删结点的复杂数据结构。

(3) *索引存储方法*。这种存储方法是顺序存储方法的一种推广，用于大小不等的数据结点的顺序存储。通过建造一个由整数域映射到存储地址域的函数，把整数索引值映射到结点的存储地址，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结点。

索引表由若干索引项组成。若每个结点在索引表中都有一个索引项，则称该索引表为*稠密索引*（dense index）。若一组结点在索引表中只对应一个索引项，则称该索引表为*稀疏索引*（spare index）。索引项的一般形式如下：

键	地址

键是能唯一标识一个结点的数据项。稠密索引中索引项的地址指示结点所在的存储位置，稀疏索引中索引项的地址指示一组结点的起始存储位置。

(4) *散列存储方法*。作为索引存储方法的一种延伸和扩展，散列存储方法利用散列函数进行索引值的计算，然后通过索引表求出结点的指针地址。

以上4种基本存储方法既可单独使用，也可组合使用。

值得注意的是，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讨论数据结构的操作，因此并不直接以内存地址来描述存储结构，而是借用高级程序语言中提供的“数据类型”来描述它们。

3. 数据的操作

数据的操作定义在数据的逻辑结构上，每种逻辑结构都有一个操作的集合。最常用的操作包括查找、插入、删除、更新、排序等。上述几种操作可以分成两类：引用型操作和加工型操作。前者不会改变结构，操作只是查询或获得结点的值（如查找等），而后者往往引起结构的改变（如插入、删除、更新等）。

1.2 抽象数据类型

首先回顾一下在程序设计语言中出现的各种数据类型。

1.2.1 数据类型

数据类型是与数据结构密切相关的一个概念，最早出现在高级程序设计语言中是为了刻划程序中操作对象的特性。在用高级语言编写的程序中，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。类型显式地或隐含地规定了变量或表达式在程序执行期间所有可能取值的范围，以及在这些值上允许进行的操作。例如，在C/C++语言的“整数类型”中，就定义了整数的取值范围（其最大值依赖于具体机器）以及对整数可施加的加、减、乘、除和取模等操作。因此，*数据类型*（data type）是一个值的集合和定义在这个值集上的一组操作的总称。通常数据类型可以看作是程序设计语言中已实现的数据结构。

数据类型的描述与它在程序中的实现有很大的区别。例如，实现线性表数据类型有两种传统的数据结构，即链表和顺序表（基于顺序存储的线性表），因此，可以在链表或者顺序表中选择一种方式来实现线性表数据类型。

按“值”是否可分解，可将数据类型划分为如下两类。

- *原子类型*：其值不可分解，通常是由语言直接提供。例如，C/C++语言的整型、字符型等标准类型及指针等简单的导出类型。
- *结构类型*：其值可分解为若干个成分（分量），是用户借助语言提供的描述机制自己定义的，通常是由标准类型派生的。例如，C/C++的数组、结构等类型。

1.2.2 抽象数据类型

抽象数据类型（abstract data type，ADT）是指一个数学模型以及定义在该模型上的一组操作。它通常是指对数据的某种抽象，定义了数据的取值范围及其结构形式，以及对数据的操作的集合。抽象数据类型的定义取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关。

抽象数据类型和数据类型实质上是一个概念。例如，整数类型就是一个抽象数据类型，尽管它们在不同处理器上的实现方法可以不同，但由于其定义的数学特性相同，在用户看来都是相同的。因此，“抽象”的意义在于数据类型的数学抽象特性。另一方面抽象数据类型的范畴更广，它不再局限于处理器中已定义并实现的数据类型，还包括用户在设计软件系统时自己定义的数据类型。

抽象数据类型是描述数据结构的一种理论工具，其目的是使人们能够独立于程序的实现细节来理解数据结构的特性。抽象数据类型的特征是将使用与实现相分离，从而实行封装和信息隐蔽。抽象数据类型通过一种特定的数据结构在程序的某个部分得以实现，而在设计使用抽象数据类型的程序时，只关心这个数据类型上的操作，而不必关心数据结构的具体实现。

例如，圆是平面上与圆心等距离的所有点的集合。从图形显示角度看，圆的抽象数据类型包括圆心和半径，而从计量角度看，它的抽象数据类型只包括半径。如果从计量角度给出圆的抽象数据类型，那么这个抽象数据类型的数据取值范围应为半径的取值范围，即为非负实数，而它的操作形式为确定圆的半径（赋值），求圆的面积，求圆的周长。

1. 抽象数据类型的描述方法

抽象数据类型可用(D, S, P)三元组表示。其中，D是数据对象，S是D上的关系集，P是对D的基本操作集。

由于抽象数据类型是由一种数据结构以及定义在其上的一组操作组成的，而数据结构又包括数据对象及对象间的关系，因此抽象数据类型一般可以由数据对象、数据关系及基本操作来定义。

```
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
```

其中，数据对象和数据关系的定义用集合描述，基本操作的定义格式为

```
返回类型 基本操作名(参数表)
```

对于每个操作，包含下列5个基本要素。

- 输入：对输入数据的说明。
- 前置条件：描述了操作执行之前数据结构和参数应满足的条件。若不满足，则操作失败，并返回相应出错信息。
- 过程：对数据执行的操作。
- 输出：对返回数据的说明。
- 后置条件：执行本操作后系统的状态，“系统”可看作是某个数据结构。

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，除可提供输入值外，还将返回操作结果。

例如，抽象数据类型Complex可以定义为如下形式：

```
ADT Complex {
    数据对象  D={e1, e2 | e1, e2∈RealSet}
    数据关系  R1={<e1, e2> | e1是复数的实数部分，e2是复数的虚数部分}
    基本操作
        InitComplex(&Z, v1, v2)
            操作结果：构造复数Z,其实部和虚部分别被赋以参数v1和v2的值
```

```
        DestroyComplex(&Z)
            操作结果：复数Z被销毁
        GetReal(Z, &realPart)
            初始条件：复数已存在
            操作结果：用realPart返回复数Z的实部值
        GetImag(Z, &ImagPart)
            初始条件：复数已存在
            操作结果：用ImagPart返回复数Z的虚部值
        Add(z1, z2, &sum)
            初始条件：z1和z2是复数
            操作结果：用sum返回两个复数z1与z2的和
} ADT Complex
```

2. 抽象数据类型的特征

(1) 抽象性。对程序处理的实体的描述强调的是其本质的特征、其所能完成的功能以及它与外部的接口（即外界使用它的方法）。

(2) 封装性。封装性就是把对象的属性和服务结合成一个独立的具有一致性的单位，并尽可能隐蔽对象的内部细节，它包含以下两层含义。

- 把对象的全部属性和全部服务结合在一起，形成一个不可分割的独立单位。
- 信息隐蔽，即尽可能隐蔽对象的内部细节，对外形成一个边界（或者说形成一道屏障），只保留有限的对外接口，使之与外部发生联系。

封装的原则在软件上的反映是，要求对象以外的部分不能随意访问对象的内部数据（属性），从而有效地避免了外部错误对它的“交叉感染”，使软件错误能够局部化，大大降低查错和排错的难度。

(3) 继承性。特殊类型的对象拥有其一般类型的全部属性与服务，称作特殊类型对一般类型的继承。例如，工程师拥有人的全部属性和服务。一个类型可以是多个一般类型的特殊类型，它从多个一般类型中继承了属性与服务，这称为多继承。例如，客轮是轮船和客运工具的特殊类型。

(4) 多态性。对象的多态性是指在一般类型中定义的属性或服务被特殊类型继承之后，可以具有不同的数据类型或表现出不同的行为。这使得同一个属性或服务在一般类型及其各个特殊类型中具有不同的语义。以“水果”的“吃”方法为例来说明，“苹果”和“香蕉”都是“水果”的子类，其“吃”方法功能不同。另外，“苹果”本身也有不同的“吃”方法。

3. 抽象数据类型的实现

实现抽象数据类型的具体实现依赖于所选择的高级语言功能。实现抽象数据类型的常见方法有下列3种。

(1) 传统的面向过程的程序设计方法。这也是现在常用的方法，根据逻辑结构选定合适的存储结构，根据所要求的功能操作设计出相应的子程序或子函数。

在标准Pascal、C等面向过程的语言中，用户可以自己定义数据类型。由此可以借助过程和函数，利用固有的数据类型来表示和实现抽象数据类型。对使用已定义的抽象数据类型的外部用户来说，必须将已定义的抽象数据类型说明和过程说明嵌入到自己程序的适当位置。可见这类语言利用抽象数据类型进行程序设计时，限制不拥有某个数据结构的模块访问和修改该数据结构，因此可以称之为数据结构的受限访问。

(2) “包”、“模型”的设计方法。Ada语言提供了“包”（package），Module-2语言提供了“模块”（module）结构，Turbo Pascal语言提供了“单元”（unit）结构，每个模块可含有一个或多个抽象数据类型，不仅可以单独编译，而且为外部使用抽象数据类型提供了方便。用这类结构实现ADT比起传统的面向过程方法有一定的进步。

(3) 面向对象的程序设计（OOP）方法。在面向对象程序设计语言中，借助对象描述抽象数据类型，存储结构的说明和操作函数的说明被封装在一个整体结构中，这个整体结构称为类（class），属于某个类的具体变量称为对象（object）。OOP与ADT的实现更加接近和一致。在面向对象程序设计语