（C语言版）

数据结构

严蔚敏 吴伟民 编著

清华大学出版社

数 据 结 构

(C语言版)

严蔚敏 吴伟民 编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

《数据结构》(C 语言版)是为"数据结构"课程编写的教材,也可作为学习数据结构及其算法的 C 程序设计的参考教材。

本书的前半部分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;后半部分主要讨论查找和排序的各种实现方法及其综合分析比较。其内容和章节编排与 1992 年 4 月出版的《数据结构》(第二版)基本一致,但在本书中更突出了抽象数据类型的概念。全书采用类 C 语言作为数据结构和算法的描述语言。

本书概念表述严谨,逻辑推理严密,语言精炼,用词达意。并有配套出版的《数据结构题集》(C 语言版)。既便于教学,又便于自学。

本书可作为计算机类专业或信息类相关专业的本科或专科教材,也可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:C 语言版/严蔚敏,吴伟民编著 .—北京:清华大学出版社,1996
ISBN 7-302-02368-9

Ⅰ.数… Ⅱ.① 严… ② 吴… Ⅲ.数据结构-C 语言-教材 Ⅳ.TP311.13

中国版本图书馆 CIP 数据核字(96)第 22985 号

出 版 者:清华大学出版社(北京清华大学校内,邮编 100084)
责任编辑:范素珍
印 刷 者:振华印刷厂
发 行 者:新华书店总店北京科技发行所
开 本:787×1092 1/16 印张:21.5 字数:509 千字
版 次:1997 年 4 月第 1 版 1997 年 8 月第 2 次印刷
书 号:ISBN 7-302-02368-9/TP·1185
印 数:10001～20000
定 价:22.00 元

前　言

“数据结构”是计算机程序设计的重要理论技术基础，它不仅是计算机学科的核心课程，而且已成为其他理工专业的热门选修课。本书是为“数据结构”课程编写的教材，其内容选取符合教学大纲要求，并兼顾学科的广度和深度，适用面广。

本书的第1章综述数据、数据结构和抽象数据类型等基本概念；第2章至第7章从抽象数据类型的角度，分别讨论线性表、栈、队列、串、数组、广义表、树和二叉树以及图等基本类型的数据结构及其应用；第8章综合介绍操作系统和编译程序中涉及的动态存储管理的基本技术；第9章至第11章讨论查找和排序，除了介绍各种实现方法之外，并着重从时间上进行定性或定量的分析和比较；第12章介绍常用的文件结构。用过《数据结构》(第二版)的读者容易看出，本书内容和章节编排与1992年4月出版的《数据结构》(第二版)基本一致，但在本书中更突出了抽象数据类型的概念。对每一种数据结构，都分别给出相应的抽象数据类型规范说明和实现方法。

全书中采用类C语言作为数据结构和算法的描述语言，在对数据的存储结构和算法进行描述时，尽量考虑C语言的特色，如利用数组的动态分配实现顺序存储结构等。虽然C语言不是抽象数据类型的理想描述工具，但鉴于目前和近一、两年内，“面向对象程序设计”并非数据结构的先修课程，故本书未直接采用类和对象等设施，而是从C语言中精选了一个核心子集，并增添C++语言的引用调用参数传递方式等，构成了一个类C描述语言。它使本书对各种抽象数据类型的定义和实现简明清晰，既不拘泥于C语言的细节，又容易转换成能上机执行的C或C++程序。

从课程性质上讲，“数据结构”是一门专业技术基础课。它的教学要求是：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间分析和空间分析的技术。另一方面，本课程的学习过程也是复杂程序设计的训练过程，要求学生编写的程序结构清楚和正确易读，符合软件工程的规范。如果说高级语言程序设计课程对学生进行了结构化程序设计(程序抽象)的初步训练的话，那么数据结构课程就要培养他们的数据抽象能力。本书将用规范的数学语言描述数据结构的定义，以突出其数学特性，同时，通过若干数据结构应用实例，引导学生学习数据类型的使用，为今后学习面向对象的程序设计作一些

铺垫。

本书可作为计算机类专业的本科或专科教材，也可以作为信息类相关专业的选修教材，讲授学时可为 50 至 80。教师可根据学时、专业和学生的实际情况，选讲或不讲目录页中带 * * 的章节，甚至删去第 5、8、11 和 12 章。本书文字通俗、简明易懂、便于自学，也可供从事计算机应用等工作的科技人员参考。只需掌握程序设计基本技术便可学习本书。若具有离散数学和概率论的知识，则对书中某些内容更易理解。如果将本书《数据结构》(C 语言版)和《数据结构》(第二版)作为关于数据结构及其算法的 C 和 Pascal 程序设计的对照教材，则有助于快速且深刻地掌握这两种语言。

与本书配套的还有《数据结构题集》(C 语言版)，由清华大学出版社出版。书中提供配套的习题和实习题，并可作为学习指导手册。

严蔚敏　清华大学计算机技术与科学系
吴伟民　华南师范大学计算机科学系

1996 年 7 月

目　录

第1章 绪 论

自1946年第一台计算机问世以来，计算机产业的飞速发展已远远超出人们对它的预料，在某些生产线上，甚至几秒钟就能生产出一台微型计算机，产量猛增，价格低廉，这就使得它的应用范围迅速扩展。如今，计算机已深入到人类社会的各个领域。计算机的应用已不再局限于科学计算，而更多地用于控制、管理及数据处理等非数值计算的处理工作。与此相应，计算机加工处理的对象由纯粹的数值发展到字符、表格和图象等各种具有一定结构的数据，这就给程序设计带来一些新的问题。为了编写出一个“好”的程序，必须分析待处理的对象的特性以及各处理对象之间存在的关系。这就是“数据结构”这门学科形成和发展的背景。

1.1 什么是数据结构

一般来说，用计算机解决一个具体问题时，大致需要经过下列几个步骤：首先要从具体问题抽象出一个适当的数学模型，然后设计一个解此数学模型的算法，最后编出程序、进行测试、调整直至得到最终解答。寻求数学模型的实质是分析问题，从中提取操作的对象，并找出这些操作对象之间含有的关系，然后用数学的语言加以描述。例如，求解梁架结构中应力的数学模型为线性方程组；预报人口增长情况的数学模型为微分方程。然而，更多的非数值计算问题无法用数学方程加以描述。下面请看三个例子。

例1-1 图书馆的书目检索系统自动化问题。当你想借阅一本参考书但不知道书库中是否有的时候；或者，当你想找某一方面的参考书而不知图书馆内有哪些这方面的书时，你都需要到图书馆去查阅图书目录卡片。在图书馆内有各种名目的卡片：有按书名编排的、有按作者编排的、还有按分类编排的，等等。若利用计算机实现自动检索，则计算机处理的对象便是这些目录卡片上的书目信息。列在一张卡片上的一本书的书目信息可由登录号、书名、作者名、分类号、出版单位和出版时间等若干项组成，每一本书都有唯一的一个登录号，但不同的书目之间可能有相同的书名、或者有相同的作者名或者有相同的分类号。由此，在书目自动检索系统中可以建立一张按登录号顺序排列的书目文件和三张分别按书名、作者名和分类号顺序排列的索引表，如图1.1所示。由这四张表构成的文件便是书目自动检索的数学模型，计算机的主要操作便是按照某个特定要求（如给定书名）对书目文件进行查询。诸如此类的还有查号系统自动化、仓库账目管理等。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的是一种最简单的线性关系，这类数学模型可称谓线性的数据结构。

例1-2 计算机和人对奕问题。计算机之所以能和人对奕是因为有人将对奕的策略事先已存入计算机。由于对奕的过程是在一定规则下随机进行的，所以，为使计算机能灵活对奕就必须对对奕过程中所有可能发生的情况以及相应的对策都考虑周全，并且，一个

001	高等数学	樊映川	S01	…
002	理论力学	罗远祥	L01	…
003	高等数学	华罗庚	S01	…
004	线性代数	栾汝书	S02	…
⋮	⋮	⋮	⋮	⋮

高等数学	001, 003, …
理论力学	002, …
线性代数	004, …
⋮	

樊映川	001, …
华罗庚	003, …
栾汝书	004, …
⋮	

L	002, …
S	001, 003, …
⋮	

图 1.1　图书目录文件示例

“好”的棋手在对奕时不仅要看棋盘当时的状态，还应能预测棋局发展的趋势，甚至最后结局。因此，在对奕问题中，计算机操作的对象是对奕过程中可能出现的棋盘状态——称为格局。例如图 1.2(a)所示为井字棋[①] 的一个格局，而格局之间的关系是由比赛规则决定的。通常，这个关系不是线性的，因为从一个棋盘格局可以派生出几个格局，例如从图 1.2(a)所示的格局可以派生出五个格局，如图 1.2(b)所示，而从每一个新的格局又可派生出四个可能出现的格局。因此，若将从对奕开始到结束的过程中所有可能出现的格局都画在一张图上，则可得到一棵倒长的“树”。“树根”是对奕开始之前的棋盘格局，而所有的“叶子”就是可能出现的结局，对奕的过程就是从树根沿树叉到某个叶子的过程。“树”可以是某些非数值计算问题的数学模型，它也是一种数据结构。

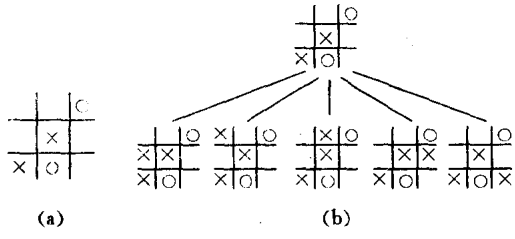

图 1.2　井字棋对奕“树”
(a) 棋盘格局示例；(b) 对奕树的局部。

例 1-3　多叉路口交通灯的管理问题。通常，在十字交叉路口只需设红、绿两色的交通灯便可保持正常的交通秩序，而在多叉路口需设几种颜色的交通灯才能既使车辆相互之间不碰撞，又能达到车辆的最大流通。假设有一个如图 **1.3(a)**所示的五叉路口，其中 **C**

① 井字棋由两人对奕。棋盘为 3×3 的方格，当一方的三个棋子占同一行、或同一列、或同一对角线时便为胜方。

和 E 为单行道。在路口有 13 条可行的通路，其中有的可以同时通行，如 A→B 和 E→C，而有的不能同时通行，如 E→B 和 A→D。那末，在路口应如何设置交通灯进行车辆的管理呢？

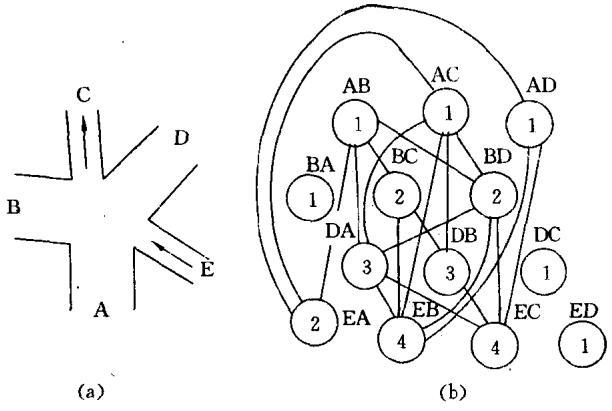

图 1.3　五叉路口交通管理示意图

(a) 五叉路口；(b) 表示通路的图。

通常，这类交通、道路问题的数学模型是一种称谓“图”的数据结构。例如在此例的问题中，可以图中一个顶点表示一条通路，而通路之间互相矛盾的关系以两个顶点之间的连线表示。如在图 1.3(b)中，每个圆圈表示图 1.3(a)所示五叉路口上的一条通路，两个圆圈之间的连线表示这两个圆圈表示的两条通路不能同时通行。设置交通灯的问题等价为对图的顶点的染色问题，要求对图上的每个顶点染一种颜色，并且要求有线相连的两个顶点不能具有相同颜色，而总的颜色种类应尽可能地少。图 1.3(b)所示为一种染色结果，圆圈中的数字表示交通灯的不同颜色，例如 3 号色灯亮时只有 D→A 和 D→B 两条路可通行。

综上三个例子可见，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、树和图之类的数据结构。因此，简单说来，数据结构是一门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等等的学科。

《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。在这之前，它的某些内容曾在其它课程，如表处理语言中有所阐述。1968 年在美国一些大学的计算机系的教学计划中，虽然把《数据结构》规定为一门课程，但对课程的范围仍没有作明确规定。当时，数据结构几乎和图论，特别是和表、树的理论为同义语。随后，数据结构这个概念被扩充到包括网络、集合代数论、格、关系等方面，从而变成了现在称之为《离散结构》的内容。然而，由于数据必须在计算机中进行处理，因此，不仅考虑数据本身的数学性质，而且还必须考虑数据的存储结构，这就进一步扩大了数据结构的内容。近年来，随着数据库系统的不断发展，在数据结构课程中又增加了文件管理(特别是大型文件的组织等)的内容。

1968 年美国唐·欧·克努特教授开创了数据结构的最初体系，他所著的《计算机程序设计技巧》第一卷《基本算法》是第一本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。从60年代末到70年代初，出现了大型程序，软件也相对独立，结构程序设计成为程序设计方法学的主要内容，人们就越来越重视数据结构，认为程序设计的实质是对确定的问题选择一种好的结构，加上设计一种好的算法。从70年代中期到80年代初，各种版本的数据结构著作就相继出现。

目前在我国，《数据结构》也已经不仅仅是计算机专业的教学计划中的核心课程之一，而且是其它非计算机专业的主要选修课程之一。

《数据结构》在计算机科学中是一门综合性的专业基础课。数据结构的研究不仅涉及到计算机硬件(特别是编码理论、存储装置和存取方法等)的研究范围，而且和计算机软件的研究有着更密切的关系，无论是编译程序还是操作系统，都涉及到数据元素在存储器中的分配问题。在研究信息检索时也必须考虑如何组织数据，以便查找和存取数据元素更为方便。因此，可以认为数据结构是介于数学、计算机硬件和计算机软件三者之间的一门核心课程(如图1.4所示)。在计算机科学中，数据结构不仅是一般程序设计(特别是非数值计算的程序设计)的基础，而且是设计和实现编译程序、操作系统、数据库系统及其它系统程序和大型应用程序的重要基础。

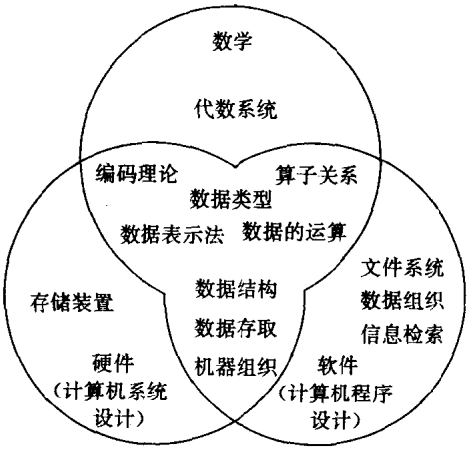

图1.4 《数据结构》所处的地位

值得注意的是，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，如多维图形数据结构等；另一方面，从抽象数据类型的观点来讨论数据结构，已成为一种新的趋势，越来越被人们所重视。

1.2 基本概念和术语

在本节中，我们将对一些概念和术语赋以确定的含义，以便与读者取得“共同的语言”。这些概念和术语将在以后的章节中多次出现。

数据(Data) 是对客观事物的符号表示，在计算机科学中是指所有能输入到计算机中并被计算机程序处理的符号的总称。它是计算机程序加工的“原料”。例如，一个利用数值分析方法解代数方程的程序，其处理对象是整数和实数；一个编译程序或文字处理程序的处理对象是字符串。因此，对计算机科学而言，数据的含义极为广泛，如图象、声音等都可以通过编码而归之于数据的范畴。

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。例如，例1-2中的“树”中的一个棋盘格局，例1-3中的“图”中的一个圆圈都被称为一个数据元素。有时，一个数据元素可由若干个**数据项**(Data Item)组成，例如，例1-1中一本书的书目信息为一个数据元素，而书目信息中的每一项(如书名、作者名

等)为一个数据项。数据项是数据的不可分割的最小单位。

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,整数数据对象是集合 $N=\{0, \pm 1, \pm 2, \cdots\}$,字母字符数据对象是集合 $C=\{'A', 'B', \cdots, 'Z'\}$。

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。这是本书对数据结构的一种简单解释①。从 1.1 节中三个例子可以看到,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**(Structure)。根据数据元素之间关系的不同特性,通常有下列四类基本结构:(1)**集合** 结构中的数据元素之间除了"同属于一个集合"的关系外,别无其它关系②;(2)**线性结构** 结构中的数据元素之间存在一个对一个的关系;(3)**树形结构** 结构中的数据元素之间存在一个对多个的关系;(4)**图状结构**或**网状结构** 结构中的数据元素之间存在多个对多个的关系。图 1.5 为上述四类基本结构的关系图。由于"集合"是数据元素之间关系极为松散的一种结构,因此也可用其它结构来表示它。

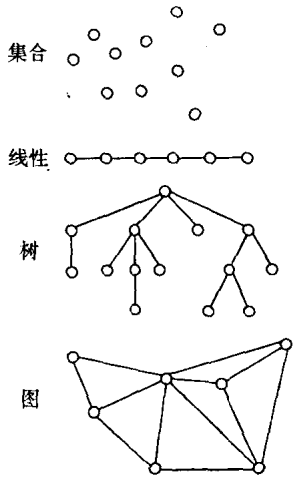

图 1.5 四类基本结构关系图

数据结构的形式定义为:数据结构是一个二元组

$$\text{Data\_Structure} = (D, S) \tag{1-1}$$

其中:D 是数据元素的有限集,S 是 D 上关系的有限集。下面举两个简单例子说明之。

例 1-4 在计算机科学中,复数可取如下定义:复数是一种数据结构

$$\text{Complex} = (C, R) \tag{1-2}$$

其中:C 是含两个实数的集合 $\{c1, c2\}$;$R=\{P\}$,而 P 是定义在集合 C 上的一种关系 $\{\langle c1, c2\rangle\}$,其中有序偶 $\langle c1, c2\rangle$ 表示 $c1$ 是复数的实部,$c2$ 是复数的虚部。

例 1-5 假设我们需要编制一个事务管理的程序,管理学校科学研究课题小组的各项事务,则首先要为程序的操作对象——课题小组设计一个数据结构。假设每个小组由一位教师、一至三名研究生及一至六名本科生组成,小组成员之间的关系是:教师指导研究生,而由每位研究生指导一至两名本科生。则可以如下定义数据结构:

$$\text{Group} = (P, R) \tag{1-3}$$

其中:$P=\{T, G_1, \cdots, G_n, S_{11} \cdots S_{nm}\}$③$_{1\leqslant n\leqslant 3, 1\leqslant m\leqslant 2}$,

$R=\{R_1, R_2\}$

$R_1=\{\langle T, G_i\rangle \mid_{1\leqslant i\leqslant n, 1\leqslant n\leqslant 3}\}$

$R_2=\{\langle G_i, S_{ij}\rangle \mid_{1\leqslant i\leqslant n, 1\leqslant j\leqslant m, 1\leqslant n\leqslant 3, 1\leqslant m\leqslant 2}\}$

---

① 对于数据结构这个概念,至今尚未有一个被一致公认的定义,不同的人在使用这个词时所表达的意思有所不同。

② 这和数学中的集合概念是一致的。

③ T 表示导师,G 表示研究生,S 表示大学生。

上述数据结构的定义仅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，因此又称为数据的**逻辑结构**。然而，讨论数据结构的目的是为了在计算机中实现对它的操作，因此还需研究如何在计算机中表示它。

数据结构在计算机中的表示（又称映象）称为数据的**物理结构**，又称**存储结构**。它包括数据元素的表示和关系的表示。在计算机中表示信息的最小单位是二进制数的一位，叫做**位**（bit）。在计算机中，我们可以用一个由若干位组合起来形成的一个位串表示一个数据元素（如用一个字长的位串表示一个整数，用八位二进制数表示一个字符等），通常称这个位串为**元素**[①]（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象。

数据元素之间的关系在计算机中有两种不同的表示方法：**顺序映象**和**非顺序映象**，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**。顺序映象的特点是借助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。例如，假设用两个字长的位串表示一个实数，则可以用地址相邻的四个字长的位串表示一个复数，如图 1.6(a)为表示复数 $z1=3.0-2.3i$ 和 $z2=-0.7+4.8i$ 的顺序存储结构；非顺序映象的特点是借助指示元素存储地址的**指针**（Pointer）表示数据元素之间的逻辑关系，如图 1.6(b)为表示复数 $z1$ 的链式存储结构，其中实部和虚部之间的关系用值为“0415”的指针来表示（0415是虚部的存储地址）[②]。数据的逻辑结构和物理结构是密切相关的两个方面，以后读者会看到，任何一个算法的设计取决于选定的数据（逻辑）结构，而算法的实现依赖于采用的存储结构。

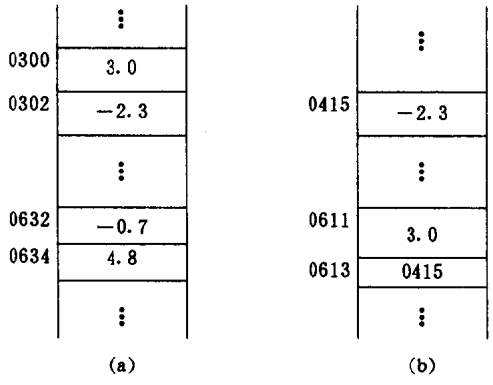

图 1.6　复数存储结构示意图

（a）顺序存储结构；（b）链式存储结构。

① 本书中有时也把数据元素简称为元素，读者应从上下文去理解分辨之。

② 在实际应用中，像复数这类极简单的结构不需要采用链式存储结构，在此仅为了简化讨论而作为假例引用之。

如何描述存储结构呢。虽则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，因此不能如上那样直接以内存地址来描述存储结构，但我们可以借用高级程序语言中提供的“数据类型”来描述它，例如可以用所有高级程序语言中都有的“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。假如我们把C语言看成是一个执行C指令和C数据类型的虚拟处理器，那末本书中讨论的存储结构是数据结构在C虚拟处理器中的表示，不妨称它为**虚拟存储结构**。

数据类型(Data Type)　是和数据结构密切相关的一个概念，它最早出现在高级程序语言中，用以刻画(程序)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”的不同特性，高级程序语言中的数据类型可分为两类：一类是非结构的**原子类型**。原子类型的值是不可分解的。如：C语言中的基本类型(整型、实型、字符型和枚举类型)、指针类型和空类型。另一类是**结构类型**。结构类型的值是由若干成分按某种结构组成的，因此是可以分解的，并且它的成分可以是非结构的，也可以是结构的。例如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。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。

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[①]（包括计算机硬件系统、操作系统、高级语言、数据库等)都提供了一组原子类型或结构类型。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即汇编语言或机器语言的数据类型来实现。引入“数据类型”的目的，从硬件的角度看，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，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“整数”类型时，既不需要了解“整数”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。如“两整数求和”，程序设计者注重的仅仅是其“数学上求和”的抽象特性，而不是其硬件的“位”操作如何进行。

抽象数据类型(Abstract Data Type 简称 ADT)　是指一个数学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个概念。例如，各个计算机都拥有的“整数”类

① 在此指广义的处理器，包括计算机的硬件系统和软件系统。