★ 新世纪计算机类本科规划教材

# 数 据 结 构

## ——C 语言描述

（第二版）

耿国华　等编著

西安电子科技大学出版社
2 0 0 8

## 内 容 简 介

本书主要包括数据结构的基本概念、基本的数据结构(线性表、栈和队列、串、数组与广义表、树、图)和基本技术(查找方法与排序方法)三个部分。本书除重点介绍了数据的组织技术外，还贯穿了程序设计中应掌握的技术，如参数传递技术、动态处理的指针技术、数组技术、递归技术与队列技术等。另外，本书给出了许多经典的查找与排序算法，为读者继续拓展思路提供线索。

本书是在第一版的基础上修订而成的，内容丰富，概念清晰，技术实用，同时还配有大量的例题、习题和实习题。本书将读者熟悉的标准 C 语言作为算法描述的语言，采用了面向对象的方法来讲述数据结构中的技术，这种描述体系也是本书特色之一。

本书既可作为大专院校计算机等专业数据结构课程的教材，也可供从事计算机开发和应用的工程技术人员学习和参考。

需要本书所列结构定义、函数原型定义及每章演示示例的读者，可通过西北大学校园网下载获取。本书同时配有多媒体教学课件，可供教师助教使用，需要者可与作者联系：ghgeng@nwu.edu.cn。

★本书配有电子教案，需要的教师可与出版社联系，免费提供。

**图书在版编目(CIP)数据**

数据结构：C 语言描述/耿国华等编著. —2 版.
—西安：西安电子科技大学出版社，2008.7
新世纪计算机类本科规划教材
ISBN 978-7-5606-1114-3

Ⅰ. 数… Ⅱ. 耿… Ⅲ. ① 数据结构—高等学校—教材 ② C 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 054763 号**

策　　划　臧延新
责任编辑　臧延新
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话　(029)88242885　88201467　　邮编　710071
http://www.xduph.com　　E-mail：xdupfxb001@163.com
经　　销　新华书店
印　　刷　西安文化彩印厂
版　　次　2008 年 7 月第 2 版　2008 年 7 月第 11 次印刷
开　　本　787 毫米×1092 毫米　1/16　印张 19
字　　数　446 千字
印　　数　42 001～46 000 册
定　　价　27.00 元
ISBN 978-7-5606-1114-3/TP・0561

**XDUP 1385022-11**

＊＊＊如有印制问题可调换＊＊＊
本社图书封面为激光防伪覆膜，谨防盗版。

# 第1章 绪 论

在陈火旺院士为《计算机学科教学计划2000》所做的序言中，把计算机50多年的成就概括为五个“一”：开辟一个新时代——信息时代；形成一个新产业——信息产业；产生一个新学科——计算机科学与技术；开创一种新的科研方法——计算方法；开辟一种新文化——计算机文化。这一概括深刻阐明了计算机对社会发展广泛而深远的影响。

数据结构被称为是计算机科学的两大支柱之一。著名的计算机科学家P. Wegner指出：“在工业革命中起核心作用的是能量，而在计算机革命中起核心作用的是信息。”计算机科学就是“一种关于信息结构转换的科学”。

关于数据结构理论的研究，可以追溯到1972年C. A. R. Hoare奠基性的论文《数据结构笔记》；而现代计算机所大量采用的各种数据结构，最早的系统论述应归于D. E. Knuth的名著《计算机程序设计技巧》。三十多年来，随着计算机科学的飞速发展，数据结构的基础研究也逐渐走向成熟。

计算机科学是关于信息结构转换的科学，信息结构(数据结构)应当是计算机科学研究的基本课题。计算机科学的重要基石是关于算法的学问，数据结构又是算法研究的基础。

在开始数据结构课程之前，我们需要在绪论中回答以下问题：

(1) 定义(什么是数据结构)；

(2) 内容(数据结构的研究范围)；

(3) 方法(研究采用的方法)；

(4) 描述(算法规则描述的工具)；

(5) 评价(对算法作性能评价)；

(6) 关于数据结构的学习。

本章将通过对这些问题与概念的简要介绍，描述数据结构基本内容与主要概念，作为对本门课程内容的梗概之序。

## 1.1 什么是数据结构(定义)

首先介绍数据结构的相关名词。

此为试读，需要完整PDF请访问：www.ertongbook.com

**1. 数据(Data)**

数据是描述客观事物的数值、字符以及能输入机器且能被处理的各种符号集合。换句话说，数据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。简而言之，数据就是计算机化的信息。

数据概念经历了与计算机发展相类似的发展过程。计算机一问世，数据作为程序的处理对象随之产生。早期计算机主要应用于数值计算，数据量小且结构简单，数据仅有进行算术运算与逻辑运算的需求，数据只包括整型、实型、布尔型，那时程序工作者把主要精力放在程序设计的技巧上，而并不重视如何在计算机上组织数据。

随着计算机软件硬件的发展与应用领域的不断扩大，计算机应用领域发生了战略性转移，非数值运算处理所占的比例越来越大，现在几乎达到 90％以上，数据的概念被大大推广了。数据包含数值、字符、声音、图像等一切可以输入到计算机中的符号集合，多种信息通过编码而被归于数据的范畴，大量复杂的非数值数据要处理，数据的组织显得越来越重要。20 世纪 70 年代后，微型机的普及，数据库、人工智能的研究推动了计算机技术的发展，人们越来越重视运用科学工具来探索数据和程序的内部关系以及它们之间的关系，采用新的观点来设计计算机体系，使计算技术发展为一门科学。

数据的概念不再是狭义的，数据已由纯粹的数值概念发展到图像、字符、声音等各种符号。

例如对 C 源程序，数据概念不仅是源程序所处理的数据，相对于编译程序来说，C 编译程序相对于源程序是一个处理程序，它加工的数据是字符流的源程序(.c)，输出的结果是目标程序(.obj)；对于链接程序来说，它加工的数据是目标程序(.obj)，输出的结果是可执行程序(.exe)，如图 1.1 所示。

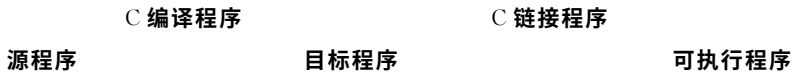


图 1.1 编译程序示意图

而对于 C 编译程序，由于它在操作系统控制下接受操作系统的调度，因此相对操作系统来说它又是数据。

**2. 数据元素(Data Element)**

数据元素是组成数据的基本单位，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。一个数据元素可由一个或多个数据项组成，数据项(Data Item)是有独立含义的最小单位，此时的数据元素通常称为记录(Record)。如表 1－1 所示，学生登记表是数据，每一个学生的记录就是一个数据元素。

表 1－1 学 籍 表

数据项

| 学 号 | 姓 名 | 性 别 | 籍 贯 | 出生年月 | 住 址 |
|---|---|---|---|---|---|
| 101 | 赵虹玲 | 女 | 河北 | 1983.11 | 北京 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

记录

**3. 数据对象(Data Object)**

数据对象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，±1，±2，…}，字母字符数据对象是集合 C={'A'，'B'，…，'Z'}，表 1-1 所示的学籍表也可看做一个数据对象。由此可看出，不论数据元素集合是无限集(如整数集)、有限集(如字符集)，还是由多个数据项组成的复合数据元素(如学籍表)，只要性质相同，都是同一个数据对象。

综上 1～3 所述，再分析数据概念：

其一：数据特点
- 可放入机器(与机器的关联性)
- 可被加工(能被处理)

其二：数据构成
- 数据元素——组成数据的基本单位(与数据的关系是集合的个体)
- 数据对象——性质相同的数据元素的集合(与数据的关系是集合的子集)

**4. 数据结构(Data Structure)**

数据结构是指相互之间存在一种或多种特定关系的数据元素集合，是带有结构的数据元素的集合，它指的是数据元素之间的相互关系，即数据的组织形式。由此可见，计算机所处理的数据并不是数据的杂乱堆积，而是具有内在联系的数据集合，如表结构(如表 1-1 所示的学籍表)、树形结构(如图 1.2 所示的学校组织结构图)、图结构(如图 1.3 所示的交通流量图)。我们关心的是数据元素之间的相互关系与组织方式，以及对其施加运算及运算规则，并不涉及数据元素的内容具体是什么值。

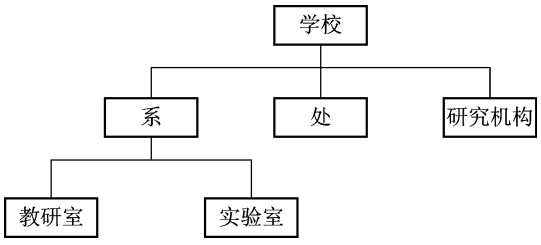


图 1.2 学校组织层次结构图

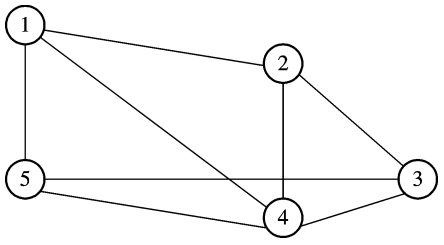


图 1.3 交通流量图

例如：一维数组是向量 $A=(a_1, \cdots, a_n)$ 的存储映像，使用时采用下标变量 A[i]的方式，关注其按序排列、按行存储的特性，并不关心 A[i]中存放的具体值。同理，二维数组 A[i，j]是矩阵 $A_{m\times n}$ 的存储映像，我们关心结构关系的特性而不涉及其数组元素本身的内容。

**5. 数据类型(Data Type)**

数据类型是一组性质相同的值集合以及定义在这个值集合上的一组操作的总称。数据类型中定义了两个集合，即该类型的取值范围，以及该类型中可允许使用的一组运算。例如高级语言中的数据类型就是已经实现的数据结构的实例。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)。在高级语言中，整型类型可能的取值范围是－32 768～＋32 767，可用的运算符

集合为加、减、乘、除、乘方、取模(如 C 语言中+、-、*、/、%等)。

从硬件的角度来看，它们的实现涉及到“字”、“字节”、“位”、“位运算”等等；从用户的观点来看，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，用户只需要了解整数运算的外部运算特性，而不必了解机器内部位运算的细节，就可运用高级语言进行程序设计。引入数据类型的目的，从硬件的角度是将其作为解释计算机内存中信息含义的一种手段，对使用数据类型的用户来说则实现了信息隐蔽，将一切用户不必关心的细节封装在类型中。如两整数求和问题，用户只仅仅注重其数学求和的抽象特性，而不必关心加法运算涉及的内部位运算实现。

按“值”的不同特性，高级程序语言中的数据类型可分为两大类：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 语言中的标准类型(整型、实型和字符型)及指针。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。数据类型指由系统定义的、用户可直接使用且可构造的数据类型。

思考　C 语言中的指针类型属于原子类型还是结构类型？

**6. 数据抽象与抽象数据类型**

抽象的本质是抽取反映问题的本质点，忽视非本质的细节，这正是从事计算机研究的本质。

1) 数据的抽象

计算机中使用的是二进制数，汇编语言中则可给出各种数据的十进制表示，如 98.65、9.6E3 等，它们是二进制数据的抽象；使用者在编程时可以直接使用，不必考虑实现细节。在高级语言中，则给出更高一级的数据抽象，出现了数据类型，如整型、实型、字符型等。到抽象数据类型出现，可以进一步定义更高级的数据抽象，如各种表、队、栈、树、图、窗口、管理器等，这种数据抽象的层次为设计者提供了更有利的手段，使得设计者可以从抽象的概念出发，从整体考虑，然后自顶向下、逐步展开，最后得到所需结果。可以这样看，高级语言中提供整型、实型、字符、记录、文件、指针等多种数据类型，可以利用这些类型构造出像栈、队列、树、图等复杂的抽象数据类型。

2) 抽象数据类型

抽象数据类型(Abstract Data Type，ADT)是指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 ADT 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。ADT 通常是指由用户定义且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，并包括一组相关服务操作。

ADT 包括定义和实现两方面，其中定义是独立于实现的。定义仅给出一个 ADT 的逻辑特性，不必考虑如何在计算机中实现。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，不局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复用程度就越高。ADT 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，使用者可以使用公共界面中的服务对该抽象数据类型进行操作。从使用者的角度来看，只要了解该抽象数据类型的规格说明，就可以利用其公用界面中的服务来使用这个类型，不必关心其物理实现，从而集中考虑如何解决实际问题。

ADT 物理实现作为私有部分封装在其实现模块内，使用者不能看到，也不能直接操作该类型所存储的数据，只有通过界面中的服务来访问这些数据。从实现者的角度来看，把抽象数据类型的物理实现封装起来，有利于编码、测试，也有利于修改。当需要改进数据结构时，只要界面服务的使用方式不变，只需要改变抽象数据类型的物理实现，所有使用该抽象数据类型的程序不需要改变，这样就会提高系统的稳定性。

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。严格地可以用一代数系统形式定义一个抽象数据类型，直觉地可以把抽象数据类型看成是定义了一组运算的数学模型。

抽象数据类型的概念不仅包含数学模型，同时还包含这个模型上的运算。过程反映程序设计的两级抽象，过程调用完成做什么，过程定义去规范如何做。抽象数据类型不仅发展了数据抽象的概念，而且将数据抽象和过程抽象结合起来。一个抽象数据类型确定了一个模型，但将模型的实现细节隐藏起来；它定义了一组运算，但将运算的实现过程隐藏起来。

无论是从计算机理论还是从计算机工程的角度来看，抽象数据类型的概念都是十分重要的。一方面，它抽象和推广了高级程序设计语言(例如 C 语言)中的类型概念；另一方面，也为软件工程的实现提供了一个自顶向下方式。

用抽象数据类型的概念来指导问题求解的过程，可以用图 1.4 来表示。其中，第一步是选用适当的数学模型来描述要处理的问题，与此同时确定解决问题的算法的基本思想。第二步是用一种比较形式的方法将解决问题的算法表达出来。描述算法的工具可以采用一种伪语言(比如说类似 C 的语言)。与这一工作并行的是为算法中用到的每个非基本的数据类型建立一个抽象数据类型，用过程名给这个类型上的每个操作命名，同时用这些过程的调用来取代算法中的每个操作。第三步是对每个抽象数据类型选择一种实现的方法，同时编写出这些抽象数据类型上定义的所有操作的过程。伪语言程序中非标准语句也要用程序语言中的标准语句加以改写，以得到一个可执行的程序。

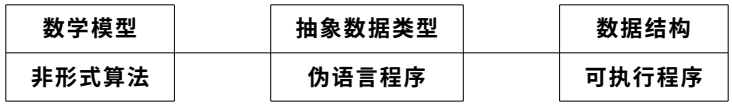


图 1.4 用抽象数据类型指导问题求解

根据上述，可以看出数据结构与抽象数据类型的关系有些类似于程序语言中的类型与值的关系。在高级语言中，整数类型代表语言(实际上是机器)中所能使用的全体整数的集合。与整数类型相关的一组运算是加、减、乘、除、存、取和比较等。这些运算都是由系统内部实现的，程序员只要会使用这些运算即可。程序中定义一个整型量(变量或常量)后，

此为试读，需要完整PDF请访问：www.ertongbook.com

就可以施加上述运算进行处理，至于这个整型量在机器内部怎样存放，是十进制还是二进制，是占 16 位还是占 32 位，整数的处理究竟如何实现，采用了哪一种除法的算法等等，对于用户来说都是隐藏的。与初等类型不同的是，系统并没有预先提供抽象数据类型的运算，系统也没有约定抽象数据类型的量(或称实例、实体)在机器中怎样表示(或称存储)，这些都要由程序员来安排。因而要实现一个抽象数据类型，首先要用适当的方式来说明数学模型在机器内的表示方法，这常常是借助于语言中已有的初等类型和构造组合类型的手段(例如记录，数组等)来完成的；其次还要根据选择的表示方法，建立一组过程来实现这个模型上的一组运算。选用的表示方法不同，实现运算的过程也就不同。有了这些基础，抽象数据类型就可以和初等类型处于相同的地位了。这种抽象数据类型的一个实体就是我们所说的一个数据结构。由于我们的研究都是基于目前的计算机系统和现有的高级语言，因此在我们研究数据结构时，不仅要研究体现抽象数据类型的内在模型(逻辑结构)和这个模型上定义的操作(运算)的实现，更要仔细研究这个实体在目前系统中的表示(存储结构)，因为不同的表示方法决定了不同的实现算法和不同的算法开销。

事实上，图 1.4 所说明的不仅是问题求解的一般过程，也是目前软件自动生成常用的模式。数学模型→抽象数据类型→数据结构，恰好反映了信息结构转换的三个重要阶段，而在这个转换过程中，数据结构是基础，抽象数据类型是中枢。

一个线性表的抽象数据类型的描述如下：

ADT Linear_list

数据元素　所有 $a_i$ 属于同一数据对象，i=1，2，…，n，n≥0。

逻辑结构　所有数据元素 $a_i$(i=1，2，…，n−1)存在次序关系<$a_i$，$a_{i+1}$>，$a_1$ 无前趋，$a_n$ 无后继。

操作　设 L 为 Linear_list：

InitList(L)：初始化空线性表；

ListLength(L)：求线性表的表长；

GetData(L，i)：取线性表的第 i 个元素；

InsList(L，i，b)：在线性表的第 i 个位置插入元素 b；

DelList(L，i)：删除线性表的第 i 个元素。

上述 ADT 很明显是抽象的。数据元素所属的数据对象没有局限于一个具体的整型、实型或其它类型，所具有的操作也是抽象的数学特性，并没有具体到何种计算机语言指令与程序编码，而数据结构就可讨论对 ADT 的具体实现。

3) 抽象数据类型实现

实现抽象数据类型需要借助于高级语言，对于 ADT 的具体实现依赖于所选择的高级语言的功能。从程序设计的历史发展来看，有传统的面向过程的程序设计，“包”、“模型”的设计，面向对象的程序设计等几种不同的实现方法。考虑到前续基础，本书仍然以第一种方式即面向过程的程序设计为主进行讨论。

下面分三种情况予以介绍。

第一种情况：传统的面向过程的程序设计。它也就是我们现在常用的方法，根据逻辑结构选定合适的存储结构，根据所要求操作设计出相应的子程序或子函数。

在标准 PASCAL 和 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。由于标准 PASCAL 语言的程序结构框架是由严格规定次序的“段”(包括程序首部、标号说明、常量定义、类型定义、变量说明、过程或函数说明、语句部分)组成，因此所有用户使用已定义的抽象数据类型的外部用户，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的基本方法时，限制不拥有某个数据结构的模块不能访问该数据结构，称之为数据结构受限访问。

第二种情况：“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，它不仅可以单独编译，而且为外部使用抽象数据类型提供了方便。使用这类结构实现 ADT 比使用第一种方法有了一定的进步。

第三种情况：面向对象的程序设计(Object Oriented Programming，OOP)。在面向对象的程序设计语言中，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。在前面对数据类型的讨论中可以看到，在面向对象的程序设计语言中，“类型”的概念与“操作”密切相关，同一种数据类型和不同操作组将组成不同的数据类型，结构说明和过程说明被统一在一个整体对象之中，其中，数据结构的定义为对象的属性域，过程或函数定义在对象中，称为方法(Method)，它是对对象的性能描述。

4) ADT 的表示与实现

■ ADT 的定义

ADT 的定义格式不唯一，我们采用下述格式定义一个 ADT：

ADT ＜ADT 名＞

{数据对象：＜数据对象的定义＞
结构关系：＜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：

＜操作名称＞(参数表)
操作前提：＜操作前提描述＞
操作结果：＜操作结果描述＞

■ 关于参数传递

参数表中的参数有两种：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，操作结果描述操作执行之后，数据结构的变化状况和应返回结果。ADT 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。不同语言的表示和实现方法不尽相同，如 ADT 中“返回结果的参数”，PASCAL 语言用“变参” 实现，C＋＋语言通过“引用型参数”实现，而 C 语言用“指针参数”实现。

用标准 C 语言表示和实现 ADT 描述时，主要包括以下两个方面：

(1) 通过结构体将 int、float 等固有类型组合到一起，构成一个结构类型，再用 typedef 为该类型或该类型指针重新起一个名字。

(2) 用 C 语言函数实现各操作。

5) 面向对象的概念

Coad 和 Yourdon 给出面向对象的概念：

面向对象＝对象＋类＋继承＋通信

对象：是指在应用问题中出现的各种实体、事件和规格说明等，它是由一组属性和在这组值上的一组服务构成的，其中属性值确定了对象的状态。

类：把具有相同属性和服务的对象归到同一类，而把一个类中的每一个对象称为该类的一个实例，它们具有相同的服务。

继承：面向对象方法的最有特色的方面。

各个类的对象间通过消息进行通信。

面向对象程序设计语言的特点是不仅具有封装性(encapsulation)，还有继承性(inheritance)和多态性(polymorphism)。从以上的讨论中可以看出，与数据结构密切相关的是定义在数据结构上的一组操作。操作的种类和数目不同，即使逻辑结构相同，这个数据结构的用途也会大不相同。定义在数据结构上的操作的种类是没有限制的，可以根据具体需要而定义。

基本操作主要有以下几种：

(1) 插入：在数据结构中的指定位置上增添新的数据元素；

(2) 删除：删去数据结构中某个指定数据元素；

(3) 更新：改变数据结构中某个元素的值，在概念上等价于删除和插入操作的组合；

(4) 查找：在数据结构中寻找满足某个特定要求的数据元素(的位置和值)；

(5) 排序：(在线性结构中)重新安排数据元素之间的逻辑顺序关系，使数据元素按值由小到大或由大到小的次序排列。

根据增、删、改、查找、排序等基本操作的特性，所有的操作可以分为两大类：一类是加工型操作，其操作的结果改变了结构的值；另一类是引用型操作，其操作的结果不改变结构的值。

6) 结构化的开发方法与面向对象的开发方法的不同点

结构化的开发方法是面向过程的开发方法，首先着眼于系统要实现的功能。从系统的输入和输出出发，分析系统要实现的功能，用自顶向下、逐步细化的方式建立系统的功能结构和相应的程序模块结构。一旦程序功能需要修改，就会涉及多个模块，修改量大，易于出错，并会引起程序的退化。

面向对象的方法首先着眼于应用问题所涉及的对象，包括对象、对象的属性、要求的操作，从而建立对象结构和为解决问题所需要执行的时间序列，据此建立类的继承层次结构，通过各个类的实例之间的消息连接实现所需的功能。类的定义充分体现了抽象数据类型的思想，基于类的体系结构可以把对程序的修改局部化，如果系统功能的需求发生变化，只需修改类中间的服务即可，此时类所代表的对象基本不变，从而确保系统不致因修改而退化。

由于用面向对象开发方法建立起来的软件易于修改，与传统方法相比，程序具有更高

的可靠性、可修改性、可维护性、可复用性、可适用性和可理解性。

## 1.2 数据结构的内容

在1.1节中已给出数据结构的一个概念，数据结构是指相互之间存在一种或多种特定关系的数据元素集合。这个描述是一种非常简单的解释。数据元素间的相互关系具体应包括三个方面：数据的逻辑结构、数据的物理结构和数据的运算集合。

**1. 逻辑结构**

数据的逻辑结构是指数据元素之间的逻辑关系描述。

数据结构的形式定义：数据结构是一个二元组 Data_Structure=(D，R)，其中D是数据元素的有限集，R是D上关系的有限集。

根据数据元素之间关系的不同特性，通常有下列四类基本的结构(如图1.5所示)：

(1) 集合结构：结构中的数据元素之间除了同属于一个集合的关系外，无任何其它关系。

(2) 线性结构：结构中的数据元素之间存在着一对一的线性关系。

(3) 树形结构：结构中的数据元素之间存在着一对多的层次关系。

(4) 图状结构或网状结构：结构中的数据元素之间存在着多对多的任意关系。

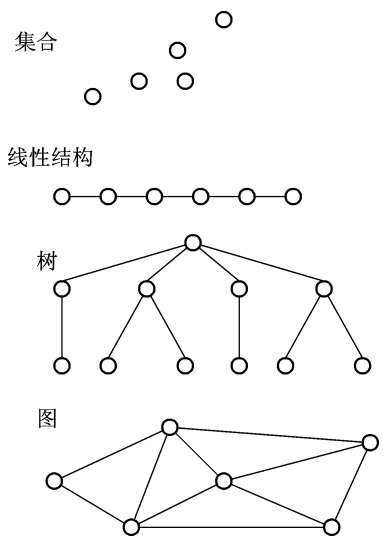


图1.5 四类基本数据结构示意

由于集合的关系非常松散，因此可以用其它的结构代替它，故数据的逻辑结构可概括如下：

逻辑结构 { 线性结构——线性表、栈、队、字符串、数组、广义表
非线性结构——树、图

**2. 存储结构**

存储结构(又称物理结构)是逻辑结构在计算机中的存储映像，是逻辑结构在计算机中的实现，它包括数据元素的表示和关系的表示。

形式化描述：D要存入机器中，建立一从D的数据元素到存储空间M单元的映像S，D→M，即对于每一个d，d∈D，都有唯一的z∈M，使S(D)=Z，同时这个映像必须明显或隐含地体现关系R。

逻辑结构与存储结构的关系：存储结构是逻辑关系的映像与元素本身的映像；逻辑结构是数据结构的抽象，存储结构是数据结构的实现，两者综合起来建立了数据元素之间的结构关系。

数据元素之间关系在计算机中有两种不同的表示方法：

- 顺序映像(顺序存储结构)；

此为试读，需要完整PDF请访问：www.ertongbook.com

· 非顺序映像(非顺序存储结构)。

数据结构在计算机中的映像包括数据元素映像和关系映像。关系映像在计算机中可用顺序存储结构或非顺序存储结构这两种不同的表示方式来存放。逻辑结构在计算机存储器中实现时，可采用不同的存储器来表示，不论是内存表示还是外存表示，都要以反映逻辑关系为原则。

3. 运算集合

讨论数据结构的目的是为了在计算机中实现操作，因此在结构上的运算集合是很重要的部分。数据结构就是研究一类数据的表示及其相关的运算操作。

通过下面的工资表实例对数据结构的内容作一概括：

表 1－2 所示的工资表采用线性表的逻辑结构，因为结点与结点之间是一种简单的线性关系；存储结构：工资表可包括几千名职工信息，可采用顺序方式存放，也可采用非顺序方式存放。怎么存这就是具体存储结构问题。对于工资表，当职工调离时要删除数据元素，调进时要增加数据元素，调整工资时要修改数据元素。这里的增删改就是数据的运算集合。

表 1－2 工 资 表

| 编 号 | 姓 名 | 性别 | 基本工资 | 工龄工资 | 应扣工资 | 实发工资 |
|---|---|---|---|---|---|---|
| 100001 | 张爱芬 | 女 | 345.67 | 145.45 | 30.00 | 451.12 |
| 100002 | 李林 | 男 | 445.90 | 185.60 | 45.00 | 586.50 |
| 100003 | 刘晓峰 | 男 | 345.00 | 130.00 | 25.00 | 450.00 |
| 100004 | 赵俊 | 女 | 560.90 | 225.90 | 65.00 | 721.80 |
| 100005 | 孙涛 | 男 | 450.60 | 190.80 | 50.00 | 591.80 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 1000121 | 张兴强 | 男 | 1025.98 | 365.53 | 100.00 | 1291.51 |

综上所述，数据结构的内容可归纳为三个部分：逻辑结构、存储结构和运算集合。按某种逻辑关系组织起来的一批数据，按一定的映像方式把它存放在计算机的存储器中，并在这些数据上定义了一个运算的集合，就叫做数据结构。

数据结构是一门主要研究怎样合理地组织数据，建立合适的数据结构，提高计算机执行程序所用的时空效率的学科。数据结构课程不仅讲授数据信息在计算机中的组织和表示方法，同时也训练高效地解决复杂问题的能力。

## 1.3 算 法

数据结构与算法之间存在着本质联系，在某一类型数据结构上，总要涉及其上施加的运算，而只有通过对定义运算的研究，才能清楚地理解数据结构的定义和作用；在涉及运算时，总要联系到该算法处理的对象和结果的数据。

在本门课程中，我们将大量地遇到算法问题，因为算法联系着数据在计算过程中的组织方式，为了描述实现某种操作，常常需要设计算法，因而算法是研究数据结构的重要途径。

**1. 算法(Algorithm)的定义**

Algorithm is a finite set of rules which gives a sequence of operation for solving a specific type of problem.（**算法是规则的有限集合，是为解决特定问题而规定的一系列操作。**）

**2. 算法的特性**

（1）**有限性：有限步骤之内正常结束，不能形成无穷循环。**

（2）**确定性：算法中的每一个步骤必须有确定含义，无二义性。**

（3）**输入：有多个或零个输入。**

（4）**输出：至少有一个或多个输出。**

（5）**可行性：原则上能精确进行，操作可通过已实现的基本运算执行有限次而完成。**

**在算法的五大特性中，最基本的是有限性、确定性和可行性。**

**3. 算法设计的要求**

**当我们用算法来解决某问题时，算法设计要达到的目标是正确、可读、健壮、高效低耗。通常作为一个好的算法，一般应该具有以下几个基本特征。**

1）**算法的正确性**

**算法的正确性是指算法应该满足具体问题的需求。其中“正确”的含义大体上可以分为四个层次：**

（1）**所设计的程序没有语法错误；**

（2）**所设计的程序对于几组输入数据能够得出满足要求的结果；**

（3）**所设计的程序对于精心选择的典型、苛刻而带有刁难性的几组输入数据能够得到满足要求的结果。**

（4）**程序对于一切合法的输入数据都能产生满足要求的结果。**

**对于这四层含义，其中达到第(4)层含义下的正确是极为困难的。一般情况下，以第(3)层含义的正确作为衡量一个程序是否正确的标准。**

**例如：要求 n 个数的最大值问题，给出示意算法如下：**

```
max:=0;
for(i=1; i<=n; i++)
   { scanf("%f", &x);
     if (x>max) max=x;
   }
```

**求最大值的算法无语法错误。虽当输入 n 个数全为正数时，结果也对，但对输入 n 个数全为负数时，求得的最大值为 0，显然这个结果不对，由这个简单的例子可以说明算法正确性的内涵。**

**问题 上面求最大值算法到底应当算第几层次？是否能算是正确算法？**

2）**可读性**

**一个好的算法首先应该便于人们理解和相互交流，其次才是机器可执行。可读性好的算法有助于人对算法的理解，并且难懂的算法易于隐藏错误且难于调试和修改。**

3）**健壮性**

**作为一个好的算法，当输入的数据非法时，也能适当地做出正确反应或进行相应的处**

理，而不会产生一些莫名其妙的输出结果。

4）高效率和低存储量

算法的效率通常是指算法的执行时间。对于一个具体问题的解决通常可以有多个算法，对于执行时间短的算法其效率就高。所谓的存储量需求，是指算法在执行过程中所需要的最大存储空间，这两者都与问题的规模有关。

## 1.4 算法描述的工具

著名的计算机科学家 N. 沃思给出了一个著名的公式：算法＋数据结构＝程序，说明数据结构和算法是程序的两大要素，二者相辅相成，缺一不可。

**1. 算法、语言和程序的关系**

首先分析数据结构中算法、语言和程序的关系：

（1）算法：描述了数据对象的元素之间的关系（包括数据逻辑关系、存储关系描述）。

（2）描述算法的工具：算法可用自然语言、框图或高级程序设计语言进行描述。自然语言简单但易于产生二义，框图直观但不擅长表达数据的组织结构，而高级程序语言则较为准确但又比较严谨。

（3）程序是算法在计算机中的实现（与所用计算机及所用语言有关）。

**2. 设计实现算法过程的步骤**

- 找出与求解有关的数据元素之间的关系（建立结构关系）。
- 确定在某一数据对象上所施加的运算。
- 考虑数据元素的存储表示。
- 选择描述算法的语言。
- 设计实现求解的算法，并用程序语言加以描述。

**3. 类描述算法的语言选择**

高级语言描述算法具有严格准确的优点，但用于描述算法，也有语言细节过多的弱点，为此采用类语言形式。所谓类语言，是指接近于高级语言而又不是严格的高级语言，它具有高级语言的一般语句设施，撇掉语言中的细节，以便把注意力主要集中在算法处理步骤本身的描述上。传统的方法是采用 PASCAL 语言，由于该语言语法规范严谨，非常适合于数据结构课程教学。在 Windows 环境下出现了一系列的功能强大且面向对象的程序开发工具，如 Visual C＋＋、Boland C＋＋、Visual Basic 等。近年来在计算机科学研究、系统开发、教学以及应用开发中，C 语言的使用范围越来越广泛，C 语言成为计算机专业与非计算机专业必修的高级程序设计语言。C 语言类型丰富，执行效率高，很多学生在学习数据结构课程之前，都已具备了熟悉 C 语言的基础条件，因此本教材采用了标准 C 语言作为算法描述的工具。为了便于学习者掌握算法的本质，尽量压缩语言描述的细节，在每一部分所使用的结构类型都统一在相应部分的首部统一定义，类型定义不重复，目的是能够简明扼要地描述算法，突出算法的思路，而不拘泥于语言语法的细节。

本书中所采用的是 C 语言，个别处使用了对标准 C 语言的一种简单化表示。在此我们对所用 C 语言作如下说明：

(1) 预定义常量和类型。

本书中用到以下常量符号，如 True、False、MAXSIZE 等，约定用如下宏定义预先定义：

```
# define TRUE 1
# define FALSE 0
# define MAXSIZE 100
# define OK 1
# define ERROR 0
```

(2) 本书中所有的算法都以如下的函数形式加以表示，其中的结构类型使用前面已有的定义。

[数据类型] 函数名([形式参数及说明])

```
{   内部数据说明;
    执行语句组;
} /* 函数名 */
```

函数的定义主要由函数名和函数体组成，函数体用花括号“{”和“}”括起来。函数中用方括号括起来的部分如“[形式参数]”为可选项，函数名之后的圆括号不可省略。函数的结果可由指针或其它方式传递到函数之外。执行语句可由各种类型的语句组成，两个语句之间用“;”号分隔。可将函数中的表达式的值通过 return 语句返回给调用它的函数。最后的花括号“}”之后的/* 函数名 */为注释部分，这是一种习惯写法，可按实际情况取舍。

(3) 赋值语句。

■ 简单赋值

① <变量名>=<表达式>，表示将表达式的值赋给左边的变量；

② <变量>++，表示变量加 1 后赋值给变量；

③ <变量>--，表示变量减 1 后赋值给变量。

■ 串联赋值

<变量 1>=<变量 2>=<变量 3>=…=<变量 k>=<表达式>；

■ 条件赋值

<变量名>=<条件表达式>?<表达式 1>:<表达式 2>；

(4) 条件选择语句。

■ if (〈表达式〉) 语句；

■ if (〈表达式〉) 语句 1；
    else 语句 2；

■ 情况语句

```
switch (<表达式>)
  {case 判断值 1:
        语句组 1;
        break;
   case 判断值 2:
        语句组 2;
        break;
```

```
        ...
  case  判断值 n:
        语句组 n;
        break;
  [default:
        语句组;
        break; ]
  }
```

switch 语句是先计算表达式的值，然后用其值与判断值相比较，若它们相一致时，就执行相应 case 下的语句组；若不一致，则执行 default 下的语句组；其中的方括号代表可选项部分。

使用情况语句时，重要的是要善于使用 switch 来简化多重条件和嵌套条件，使多分支结构清晰。

(5) 循环语句。

■ for 语句

for (<表达式 1>；<表达式 2>；<表达式 3>)

{循环体语句；}

首先计算表达式 1 的值，然后求表达式 2 的值，若结果非零(即为真)，则执行循环体语句，最后对表达式 3 运算，如此循环，直到表达式 2 的值为零(即不成立为假)时为止。

■ while 语句

while (<条件表达式>)

{循环体语句；}

while 循环首先计算条件表达式的值，若条件表达式的值非零(即条件成立)，则执行循环体语句，然后再次计算条件表达式的值，重复执行，直到条件表达式的值为零(即为假)时退出循环，执行该循环之后的语句。

■ do - while 语句

do { 循环体语句

}while (<条件表达式>)

该循环语句首先执行循环体语句，然后计算条件表达式的值，若条件表达式成立，则再次执行循环体并计算条件表达式的值，直到条件表达式的值为零，即条件不成立时结束循环。

(6) 输入、输出函数。

输入：用函数 scanf 实现；特别当数据为单个字符时，用 getchar 函数实现；当数据为字符串时，用 gets 函数实现。

输出：用 printf 函数实现；当要输出单个字符时，用 putchar 函数实现；当数据为字符串时，用 puts 函数实现。

其中输入输出函数中的类型部分不做严格要求，淡化表述。

(7) 其它一些语句。

① return <表达式>或 return：用于函数结束。

② break 语句：可用在循环语句或 switch 语句中结束循环过程或跳出情况语句。

③ continue 语句：可用在循环语句中结束本次循环过程，进入下一次循环过程。

④ exit 语句：表示出现异常情况时，控制退出函数。

(8) 注释形式。

/* 字符串 */

注释句的作用是增强算法的可阅读性，在算法描述中要求在函数首部加上对算法功能的必要注释描述。加注释说明时如果没有涉及到的参量一定是多余的，而涉及到的内容应当作为参量，这实际上是程序设计中的一个素质要求，希望多加注意。

(9) 一些基本的函数，例如：

max 函数：用于求一个或几个表达式中的最大值；

min 函数：用于求一个或几个表达式中的最小值；

abs 函数：用于求表达式的绝对值；

eof 函数：用于判定文件是否结束；

eoln 函数：用于判断文本行是否结束。

**4. 指针、结构体与类型定义**

在数据结构的具体实现中，频繁地使用指针、结构体以及类型定义语句 typedef。下面通过两个简单的例子来说明指针、结构体以及 typedef 的用法。

**例 1-1 键盘输入示例。**

```
#include <stdio.h>
typedef int integer;
typedef int *ipointer;
main()
{ integer m;
   ipointer ip1, ip2;
   m = 10;
   ip1 = &m;
   *ip1 = 100;
   printf("\n m == %d", m);
   ip2 = (ipointer)malloc(sizeof(int));
   *ip2 = m + *ip1;
   printf("\n\n      *ip2 == %d", *ip2);
  getch();      /* 利用读字符函数实现暂停，敲任意键可以继续 */
}
```

**例 1-2 简单的复数运算。**

```
#include <stdio.h>
typedef struct {
  float realpart;            /* 实部 */
  float imagpart;            /* 虚部 */
} Complex;                   /* 复数类型 */

main()
{ float a,b;
```

此为试读，需要完整PDF请访问：www.ertongbook.com