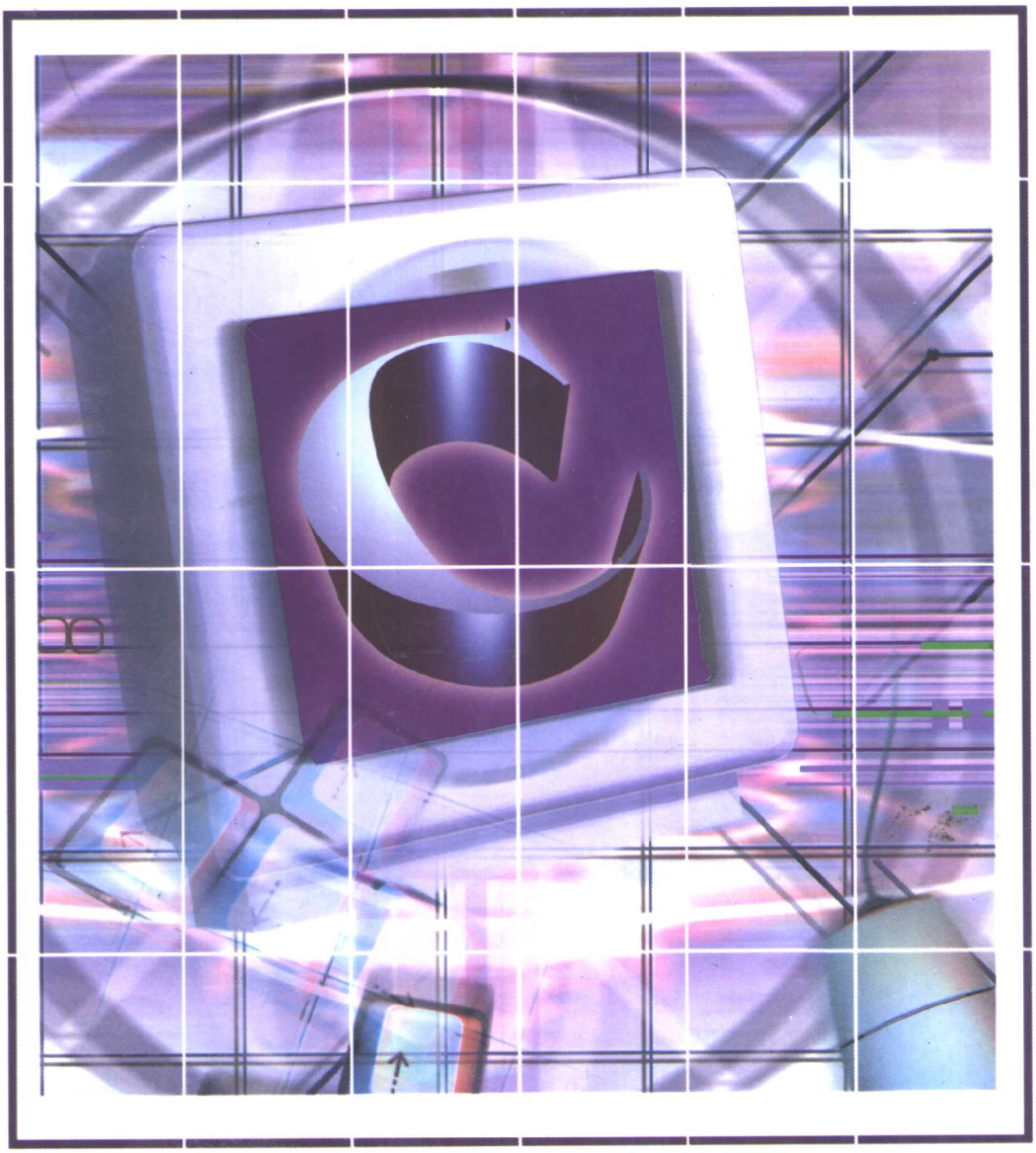
新世纪计算机类本科系列教材



数据结构——C语言描述

耿国华　等编著

西安电子科技大学出版社
http://www.xduph.com

★ 新世纪计算机类本科系列教材

数据结构

——C语言描述

耿国华　等编著

西安电子科技大学出版社
2002

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)，以及基本技术(查找方法与排序方法)等三个部分。本书除了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，为读者继续展拓思路提供线索。

本书是在作者多年教学实践的基础上编写而成，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。在本书中，使用读者熟悉的标准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科书，也可作为从事计算机开发和应用的工程技术人员的自学参考书。

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。

图书在版编目(CIP)数据

数据结构：C语言描述/耿国华等编著. —西安：西安电子科技大学出版社，2002.2
新世纪计算机类本科系列教材
ISBN 7-5606-1114-1
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

中国版本图书馆CIP数据核字(2002)第003022号

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)8227828 邮编 710071
http://www.xduph.com E-mail：xdupfxb@pub.xaonline.com
经 销 新华书店
印 刷 陕西画报社印刷厂
版 次 2002年2月第1版 2002年8月第2次印刷
开 本 787毫米×1092毫米 1/16 印张 18.25
字 数 429千字
印 数 4 001～10 000册
定 价 20.00元
ISBN 7-5606-1114-1/TP·0561

XDUP 1385001－2

＊＊＊如有印制问题可调换＊＊＊

前　言

我们生活在一个物质的世界，计算机工作者又面对着数字的世界，如果将物质世界中的事与物数字化，那么它们在计算机中的表现则为数据。这些数据来源于现实，表征着具体的意义，而且在计算机中有着统一的表示方法，因而成为被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、存储方法以及在其上的典型处理方法，就构成了数据结构课程的主要内容。

早在20世纪80年代初，数据结构课程就已成为国内计算机专业教学计划中的核心课程。目前，ACM/IEEE CC－2001教程已将算法与数据结构类课程列为核心课程之首，数据结构愈益显出其在信息学科中的重要地位。

由于数据是计算机处理的对象，使用计算机的过程就是对数据加工处理的过程，因而数据的组织与结构被确立为计算机科学中最为基本的内容。通过对数据结构的学习，使读者能够以问题求解方法、程序设计方法及一些典型的数据结构算法为研究对象，学会分析数据对象的特征，掌握数据组织的方法和在计算机中的表示方法，为数据选择适当的逻辑结构、存储结构以及相应的处理算法，初步掌握算法的时间、空间复杂度的分析技巧，培养良好的程序设计风格以及进行复杂程序设计的技能。

人类解决问题的思维方式可分为推理方式和算法方式两大类。推理方式是从抽象的公理体系出发，通过演绎、归纳、推理来求证结果，解决特定问题。这种推理方式是通过数学训练得到的。算法方式则是凭借构造性思维，从具体操作规范入手，通过操作过程的构造实施来解决特定问题。在软件系统的开发过程中，其凭借的思维方法在本质上不同于公理系统的思维方法，而是一种算法构造性思维方法。让学生理解、习惯和熟悉这一套算法构造性思维方法，是本门课程教学的重要内容和主要难点。对于软件开发人员来说，仅仅知道开发工具的语言规则和使用过程是远远不够的，只有培养学生的数据抽象能力，算法设计能力以及创造性思维的方法，才能够举一反三、触类旁通，从而达到应用知识解决复杂问题的目的。

本书采用面向对象的方法来讲述数据结构的技术，并使用C语言作为算法描述语言。由于目前C语言被广泛地使用，而且数据结构的算法本身又是底层的基本算法，采用大家熟悉的C语言去刻画算法中的主要概念，可以将读者的注意力集中在算法的理解上。本书贯穿了面向对象的观点，首先引进了抽象数据类型的概念及其基本性质，然后给出了如何用C语言表示抽象数据类型的方法。即在每章开始使用抽象数据类型(ADT)定义，其中不仅包含了数学模型，同时还包含了定义在这个模型上的数据组织和数据运算的名称，接下来在具体章节中再详细介绍相应的数据结构及运算实现。抽象数据类型的概念反映了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。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。书中大量的C函数的程序实例，正是数据抽象与过程抽象的结合。这就使数据结构的表示得以简化，突出了算法表示的实质，其中所列算法只需补充上相应的类型定义与

调用，就可成为可直接上机运行使用的C程序。

本书分为三个部分，其中第一部分(第1章)是数据结构的基本概念部分；第二部分(第2～7章)是基本的数据结构部分，包括线性结构(线性表、栈和队列、串、数组与广义表)与非线性结构(树、图)；第三部分(第8～10章)是基本技术部分，包括查找方法与排序方法。除了数据组织技术外，还包括了一些重要的程序设计技术，如参数传递技术、动态处理的指针技术、数组技术(抽象规律处理)、递归技术与队列技术。此外，书中给出了许多精彩的查找与排序的典型算法，它们是人们在数据处理中智慧的结晶，我们力求将经典算法的思路表现出来，为学习者继续拓展思路提供线索。本书每章后附有习题与实习题，在附录中给出了两套标准化考题样卷，其余四套均为硕士研究生入学考试的样卷，以便于读者练习。

数据结构作为西北大学重点课程建设项目，被列为“面向21世纪课程教材”，在编写的过程中得到西北大学教务处的大力支持；曾指导我学习数据结构的清华大学的严蔚敏教授和唐泽圣教授，他们敏锐的洞察力和对教学内容的精辟讲解，包括尔后多年中所给予的多方面指导，使我受益终生；学生们学习该课程的热情为我们注入了深入教学研究的动力；赵政文教授仔细校审全书，并提出了意见与建议。所有这些都将极大地促进我们数据结构教学质量的提高，在此表示衷心的感谢！

本书由耿国华教授任主编，张德同老师任副主编，其中的第1章、2章、3章、6章、7章、9章及附录由耿国华编写，第5章、8章由张德同编写，第4章由冯宏伟编写，第10章由卢燕宁编写。本书算法均在Turbo C 2.0环境下调试通过。在本书中，我们将多年从事数据结构课程教学的体会写了出来，恳请读者赐教指正。

编著者

2001年12月

目　录

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。

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。

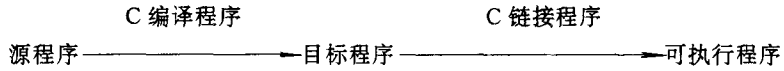


图1.1　编译程序示意图

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作一个数据对象。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(表 1－1 所示的学籍表)、树型结构(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。

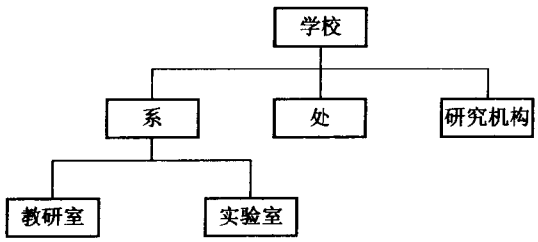


图 1.2　学校组织层次结构图

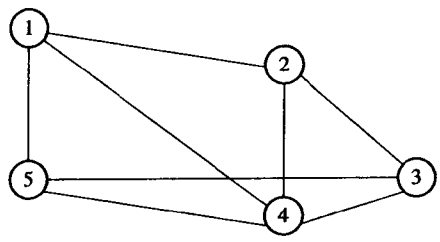


图 1.3　交通流量图

例如：一维数组是向量 A＝(a_1，…，a_n)的存储映象，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 $A_{m\times n}$的存储映象，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＋，－，＊，/，％)。

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。

思考题：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(Abstract Data Type)

抽象数据类型(简称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”的意义在于教学特性的抽象。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，也为软件工程提供了一个自顶向下的实现图式。

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。

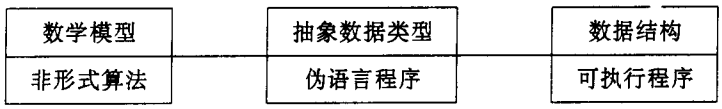


图 1.4　用抽象数据类型指导问题求解

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。

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就行了。程序中定义一个整型量(变量或常量)后，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。所以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；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，恰好反应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教学特性，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，简称 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：