铁路中专计算机统编教材

数据结构

（C语言版）

于训全　主编

电子科技大学出版社

铁路中专计算机统编教材

数据结构

（C语言版）

于训全　主编

电子科技大学出版社

内 容 提 要

本书主要介绍数据结构和算法的概念和常用术语、线性表、栈、队列、数组、串、树以及图等基本类型的数据结构及其应用、查找、排序以及常用的文件结构。全书采用C语言作为数据结构和算法的描述语言。

本书可供中等专业学校计算机类各专业使用，也可作为高等职业技术院校计算机类专业教材，以及供从事计算机专业工作人员参考。

图书在版编目（CIP）数据

数据结构：C语言版/于训全主编. 一成都：电子科技大学出版社，2001.1
ISBN 7-81065-628-7

Ⅰ.数… Ⅱ.于… Ⅲ.数据结构 Ⅳ.TP311.12

中国版本图书馆CIP数据核字（2001）第03154号

铁路中专计算机统编教材

数据结构（C语言版）

于训全 主编

出　　版：电子科技大学出版社(成都建设北路二段四号　邮编：610054)
责任编辑：文　利
发　　行：新华书店
印　　刷：四川导向印务有限公司
开　　本：787×1092　1/16　印张　11.75　字数　286千字
版　　次：2001年1月第一版
印　　次：2001年1月第一次印刷
书　　号：ISBN 7—81065—628—7/TP・421
印　　数：1—4000册
定　　价：16.00元

前　言

《数据结构》是计算机专业的一门专业技术基础课程。它所研究并要解决的问题是：如何根据实际应用的要求，对数据进行有效地组织、存储和处理，进而编制出相应的算法。

通过数据结构这门课程的学习，应使学生能运用数据结构的知识和技巧更好地进行算法和程序的设计，也为学习操作系统等后续课程打好良好的基础。

本书的第一章介绍数据结构和算法的概念和常用术语；第二章到第六章分别讨论线性表、栈、队列、数组、串、树以及图等基本类型的数据结构及其应用；第七章和第八章介绍常用的几种查找方法和排序方法，第九章介绍了常用的文件结构。全书采用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，尽量考虑了C语言的特色。

本书是铁道部中专计算机及应用教学指导委员会统编教材之一，由株洲铁路机械学校于训全主编，株洲铁路机械学校徐建军担任主审，郑州铁路职业技术学院牛红霞编写第一章、第九章，沈阳铁路机械学校叶慧婷编写第二章，天津铁路工程学校孟庆菊编写第三章，哈尔滨铁路成人中专学校刘彦编写第四章，齐齐哈尔铁路工程学校商丹编写第五章，株洲铁路机械学校于训全编写第六章，苏州铁路机械学校王君秋编写第七章，湖南铁道职业技术学院张杰编写第八章，铁道部中专计算机及应用教学指导委员会主任徐维祥认真审阅了本书的初稿，并提出了很多宝贵建议和意见，本书从编写到发行过程中，铁道部中专计算机及应用教学指导委员会给予了高度重示和关怀，在此表示感谢。

由于本书作者水平有限，书中难免有错误和不足之处，恳请读者批评指教。

2000年8月

目　录

第1章 绪 论

计算机科学是一门研究信息表示、组织和处理的科学，而信息的表示和组织，直接关系到处理信息时的效率。随着计算机应用领域的扩大，作为信息载体的数据量也随之增多，数据结构也随之复杂化。导致了许多系统程序和应用程序规模很大，其结构也相当复杂。因此，单纯依靠程序设计人员的经验和技巧已不能编制出高效率的应用程序。

瑞士著名计算机科学家沃斯（N. Wirth）教授曾经提出：算法+数据结构=程序，这个等式明确指出了程序的两大部分。这里的数据结构是指数据的逻辑结构和存储结构，而算法则是对数据进行处理的描述。由此可见，程序设计的实质是对实际问题选择一种恰当的数据结构、并加之设计一种好的算法，而这个好的算法又在很大程度上取决于如何表示被处理的数据——数据结构。

本章我们将介绍数据结构的发展，数据、数据结构和算法的基本概念和术语，算法描述和算法分析等内容。

§1.1 数据结构的发展

世间万物，需求是产生与发展的动力。数据结构这门课的形成与发展也不例外。

电子计算机自本世纪40年代问世以来，发展神速，无论是软件方面，还是硬件方面，都已远远超出人们的预想。随着计算机的高速发展以及微型计算机的日益普及，计算机已广泛深入到人类社会的各个领域，因此，计算机已不再局限于解决那些纯数值计算的问题，更多更广泛的是应用于非数值计算的各个领域，如企业管理，工业过程控制，经济系统工程，管理信息系统等等。与此相应，计算机处理的对象也由纯粹的数值数据发展到诸如字符、图像、声音、信号等非数值数据，对这些非数值数据进行处理、编程，就必须分析这些数据间的关系，即数据结构，从而建立描述非数值计算的模型，这就是数据结构这门学科形成和发展的背景。

例如，图书馆书目管理系统。图书馆有众多图书，我们可把图书杂乱的信息分为：图书编号、书名、作者、出版社、数量、馆存量几项，如表1.1所示。

其中，我们可以分析一下数据间的关系：同一个书名可能有多个作者或出版社，而同一个作者可能写了多部著作，拥有多个书名和出版社。通过分析数据之间的逻辑关系，编制不同应用目的的程序，这样图书馆的管理系统就归结为计算机对一个表的有关操作。如表1.1是按图书到馆的先后顺序录入的，若在该表中查找《数据结构》的书，我们只有一一排查；若改变表1.1数据之间的结构，使之按“书名”排序，相应的查找程序则不同，查找会更快捷。除此之外，对报废的图书，可编制删除程序；对新进的图书，可编制追加

程序等。所有这些都是对表 1.1 的操作。

表 1.1　图书管理系统

图书编号	书　名	作　者	出 版 社	数量	馆存量
100045	数据结构	严蔚敏	清华大学出版社	5	3
500781	C 语言程序设计	谭浩强	清华大学出版社	8	5
500003	计算机软件基础	杨德元	高等教育出版社	3	3
100067	数据结构	李　平	电子工业出版社	4	0
…	…	…	…	…	…

从上述例题来看，许多实际问题不能使用数值计算的方法解决，而且它们的数据有时是大量的，所以，要使计算机高效正确地解决问题，对数据如何表示、组织、存储以及如何操作等问题的研究就显得越来越迫切，越来越重要了。

在国外，数据结构成为一门独立的课程始于 1968 年，在我国还要稍晚一些。此前，现在的“数据结构”课程的某些内容出现在诸如“编译方法”、“操作系统”的课程中。尽管后来美国的一些大学的计算机系在教学计划中将“数据结构”列为一门独立的课程，但对课程的范围仍然没有作出明确的规定。当时的“数据结构”几乎同图论，特别是表处理、树的理论是一回事。随后，数据结构的概念被扩充到包括网络、集合代数论等方面，从而变成了称为“离散结构”的内容。60 年代中期出现了类似于现在的“数据结构”课程，叫做“表处理语言”，主要介绍处理表结构和树结构的语言。1968 年，美国出版了一本大型著作《计算机程序设计技巧》第一卷《基本算法》（作者 D.E.克努特），较为系统地阐述了数据的逻辑结构与物理结构以及运算的理论方法与技巧。60 年代末与 70 年代初，出现了大型程序，软件也相对独立，结构化程序设计逐步成为程序设计方法学的主要内容。人们越来越重视数据结构，已经认识到程序设计的实质就是对所确定的问题选择一种好的结构，从而设计一种好的算法。从那以后，各种版本的《数据结构》著作相继问世。

《数据结构》在计算机科学中有着十分重要的地位，它不仅作为计算机专业教学计划中的重点核心课程之一，而且也开始成为其它许多非计算机专业的主要选修课。作为计算机专业在程序设计课程之后、各专业课程之前的一门最重要的专业基础课，“数据结构”不仅为学习“编译原理”、“操作系统”、“数据库原理”等后续课程提供必要的基础，也直接为从事各种系统软件和应用软件的设计人员提供了必备的知识与方法。

《数据结构》有着自己的理论、研究对象和应用范围，而且其研究内容还在不断地扩充和深化。它作为一门方兴未艾的新兴科学，目前仍然处在一个蓬勃发展的阶段。

§1.2　数据结构和算法

众所周知，计算机是一种信息处理装置。信息中各个数据元素并不是孤立存在的，它们之间有着一定的结构关系。如何表示这些结构关系，如何在计算机中存储数据和信息，

采用什么样的方法技巧去加工处理这些数据等等，都是数据结构这门课程要努力解决的问题。

§1.2.1 数据结构的概念

什么是数据结构呢？在回答这个问题之前，我们先熟知下列名词：

1. 数据（data）：数据是描述客观事物的数字、字符以及一切能输入到计算机中，并能被计算机处理的符号集合。简而言之，数据就是计算机加工处理的“原料”。数据的含义十分广泛，在不同的场合有不同的含义。例如，在数值计算的问题中，计算机处理的对象大多数是整数和实数；在文字处理程序中，计算机处理的对象是一些符号串；而在某些控制过程中，可能是某种信号。随着计算机硬、软件的发展，计算机应用领域的扩大，数据的含义也随之拓宽。例如，图像、声音等都可以通过编码而归于数据的范畴。

2. 数据元素（data element）：数据元素是数据的基本单位，即数据这个集合的一个个客体。数据元素也称为结点、顶点、记录。有时，一个数据元素可由若干个数据项（也称为字段、域）组成，数据项是具有独立含义的最小单位。惟一能标识一个数据元素的数据项称作该元素的关键字。例如，在表 1.1 图书管理系统这个数据文件中，每一个数据元素就是一个图书的记录，而每个数据记录又是由若干个描述图书某一方面特征的数据项（字段）组成，如图书编号、书名、作者、出版社、数量、馆存量，而数据项中的“图书编号”则惟一能标识一个元素，称为关键字。

3. 数据结构（data structure）：数据结构指的是数据与数据相互之间存在着一种或多种关系的数据元素的集合。也就是说，数据结构是数据的集合，数据之间存在的这样那样的关系称为结构。

例如，计算机专业选修课有：计算机控制、A/D（D/A）转换原理、图像处理、语言识别、人工智能等六门课，分别以 A，B，C，D，E，F 表示，每个学生可选修 1~3 门课，一个班部分学生的基本情况及选修课列表如表 1.2 所示。

表 1.2 学生班选修课程及基本情况表

学　号	姓　名	选修课程	爱　好	职　务
990401	王临轩	B，F	无	班长
990402	李大民	A，C	体育	
990403	姚立志	A，D，F	体育	体育委员
990404	杜　鹃	D，E	文娱	
990405	于海涛	C，E	文娱	
990506	金鹏程	D，F	体育	
990407	郭书楷	B，E	体育	
990408	史向东	A，E	体育	
990409	范思理	B，F	文娱	文娱委员
990410	周厚源	B，D	文娱	

由表 1.2 可见，在我们研究的数据对象中含有 10 个数据元素，每个数据元素之间存在着一定的关系，即结构。下面，从不同的角度分析数据间的关系。

[例 1.1] 按照学号的最后二位顺序建立并表示该数据集合的关系时，则得到如图 1.1 所示结构图。

图 1.1 线性结构

这里，称一个元素的前一个元素为该元素的直接前驱元素，一个元素的后一个元素为该元素的直接后继元素。本例中除了第一个元素与最后一个元素外，其余每个元素有且仅有一个直接前驱元素，有且仅有一个直接后继元素，数据元素之间存在一对一的关系，我们将这种逻辑关系称为线性关系，即该文件的逻辑结构为线性结构，也称该文件为一个线性表。

[例 1.2] 按其职务和爱好建立关系时，则为图 1.2。

依照例 1.1 的分析，本例结构中的元素存在一对多的关系，按此形式建立起来的关系称为树形结构。

例 1.3 按班干部及其选修课程建立关系时，则为图 1.3。

该结构中的元素存在多对多的关系，按此形式建立起来的关系称为图结构。

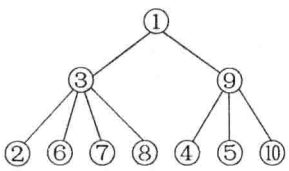

图 1.2 树形结构

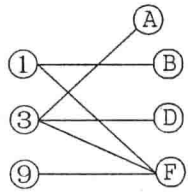

图 1.3 图结构

数据结构通常采用二元组表示：

Data Stucture =(D，R)

其中：D 是数据元素的有限集，R 是 D 上关系的有限集。以上三例分别表示为：

Linear—List =(D，R)

其中：D={01，02，…，09，10} （关键字“学号”后二位表示）

R={<01，02>，<02，03>，… ，<08，09>，<09，10>}

Tree =(D，R)

其中：D={01，02，…，09，10}

R={<01，03>，<01，09>，<03，02>，<03，06>，<03，07>，<03，08>，<09，04>，<09，05>，<09，10>}

Graph=(D，R)

其中：D={01，02，…，09，10，A，B，C，D，E，F}

R={<01，B>，<01，F>，<03，A>，<03，D>，<03，F>，<09，B>，

<09，F>}

上述数据结构是从逻辑关系上描述数据，它与数据的存储无关，称为数据的逻辑结构。研究数据结构的目的是为了在计算机中实现对它的操作，为此还需研究数据在计算机中的存储形式，即数据的物理结构或存储结构。

数据的物理结构主要有顺序存储结构、链式存储结构和散列结构。顺序存储结构是指在存储器中用一块地址连续的存储单元依次存放各个数据元素，数据元素物理上的先后关系映射了逻辑上的先后关系，这种存储结构的特点是通过数据元素的存储地址来直接反映数据元素之间的逻辑关系。如例 1.1 中的线性逻辑结构用顺序存储如图 1.4 所示，图中每个方框表示一个存储地址。

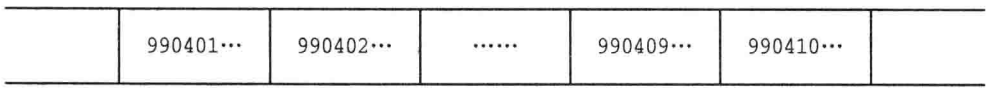

图 1.4　顺序存储结构

链式存储结构则是在计算机存储器中采用地址任意的（连续的或者不连续的）存储单元，依次存放各个元素，我们称每个元素占用的若干存储单元的组合为一个链结点。每个链结点不仅用来存放一个数据元素的数据信息，还要存放一个地址，这个地址通常用指针表示，它指向该元素逻辑关系中的直接后继元素所在链结点的位置。因此，链式存储结构的特点是通过指针这样的地址间接地反映数据元素的逻辑关系。链式存储结构简称为链表。对例 1.1 中的线性逻辑结构，若采用链式存储结构存储，如图 1.5 所示。图中的各个箭头表示一个指针。

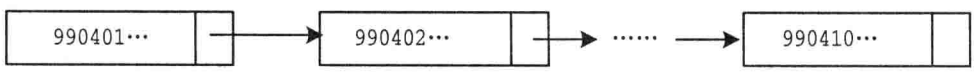

图 1.5　链式存储结构

除顺序存储结构与链式存储结构外，还有一种称为散列结构的存储结构，在这种存储结构中，元素的存储位置不是任意的，而是由一个称为散列函数的关系来确定，对数据元素的操作也主要基于这个函数关系来进行。有关这种存储结构的具体细节将在以后查找一章中介绍。

另外，可以根据数据的结构（逻辑结构和存储结构）特性在数据的生存期间的变动情况，将数据结构分成静态结构与动态结构。所谓静态结构就是指数据结构存在期间不发生变动，如数组是一种线性结构，但也是一种静态结构，它不随操作的进行而改变数组的大小。而动态结构是指在一定范围内结构的大小要发生变动，如堆栈、队列以及树型结构等都属于动态结构。

由上述讨论可以看到，从不同角度对数据结构进行分类是为了更好地认识它们，更深入地了解数据结构的特性及关系。在设计某个操作时，首先要清楚数据元素之间具有什么样的逻辑结构，然后采用合适的存储结构来具体实现这种操作。同一种操作在不同存储结构中的实现方法可以不同，有的则完全依赖于所采用的存储结构。

综上所述，数据结构所要研究的主要内容可以简要归纳为以下三个方面：

（1）研究数据元素之间固有的客观联系（逻辑结构）；

（2）研究数据在计算机内部的存储方法（存储结构）；

（3）研究如何在数据的各种结构（逻辑的和物理的）上实施有效的操作或处理（即算法）。

§1.2.2 算法及其特性

算法就是解决某个具体问题的办法，是用来解决某个特定课题的一些指令的集合。

[例 1.4] 求 1+2+3+…+n 的和，用 C 语言写出的算法如下：

```
void s( )
{
  int  sum，i，n;
  for (i=1;i<=n;i++)
    sum=sum+i;
  printf("%d"，sum);
}
```

算法具有以下基本特性：

1. 输入：由外部 n≥0 个有限量作为输入。这里的 0 个是指在算法内部确定初始条件。

2. 输出：至少有一个量作为输出，没有输出的算法毫无含义。

3. 有穷性：算法必须在有限的步骤内结束。

4. 确定性：组成算法的指令必须清晰、无二义性。就是说算法的每一步都必须准确定义。

5. 有效性：算法的指令必须具有可执行性。也就是说，算法所实现的操作都应该是基本的、可实施的。例如，一个数除以 0，这个动作是不能执行的。

算法的含义与程序十分相似，但又有所区别。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，它将永远等待、不会停止，即使没有作业需要处理，它仍处于一个等待循环中，以待新的作业进入。因此操作系统程序不是一个算法。另外，程序中的指令必须是机器可执行的，而算法中的指令则无此限制。但一个算法若用计算机语言来书写，它就是一个程序。

数据结构和算法有着密切的关系，它们是一个不可分割的主体。对于一个给定问题的初始数据，如何组织，在计算机内如何存储，即采用什么样的数据结构（逻辑的和物理的），以便于计算机处理，同时，如何选择合适的算法以提高求解问题的可靠性都是至关重要的，我们不能把两者分开来孤立讨论。如前面所说的图书管理系统，若图书在数据文件中无规律地存放，查找图书的算法只能顺序地逐个查找，若按某一关键字建立索引，同时选择更有效的算法（如折半查找），就可大大提高查询速度。由此可知，不同的算法必须选用相适应的数据结构才能发挥作用，即数据结构的不同直接影响算法的选择和效率。显然，根据数据处理问题的需要，为待处理的数据选择合适的逻辑结构和物理结构，进而设计出比较

满意的算法，是学习数据结构这门课程的主要目的。

§1.3 算法的描述与算法分析

§1.3.1 算法的描述

算法独立于具体的计算机，与具体的程序设计语言无关。在设计一个算法时应选择一种合适的方式来描述算法的各个步骤。在计算机发展的初期，人们用自然语言描述算法，由于自然语言描述算法很不方便，而且可能会出现二义性，后来采用流程图的形式来描述算法，比采用自然语言表达直观了一些，但依然没有解决复杂算法的表达，而且移植性也不好。实际上算法最终要通过程序设计语言变成程序，因此人们就直接采用某种具体的程序设计语言来描述，如 PASCAL 或 C 语言。考虑到 C 语言是当前国际上广泛流行和很有发展前途的计算机高级语言，它能较好地体现结构化程序设计的原则，并且简洁明了，便于教学，所以，本书采用 C 语言来描述一个具体的算法。在本书中我们有如下约定：

对于所有的算法都以函数的形式给出，形式如下：

函数类型标识符　函数名（参数表）

{语句组}

其中，参数表可含有若干个形式参数，每个参数间用逗号分隔，语句组由一个或一个以上的语句组成。

对于数据结构都以自定义结构类型给出，形式如下：

typedef　structure

{结构成员列表}；结构名

其中，花括号内是该结构体中的各个成员，对每个成员都应进行类型说明。

算法中出现的其它语句，如赋值语句、条件语句、循环语句、开关语句等，都与 C 语言相同。

§1.3.2 算法分析

同一个问题，可以用不同的算法解决，而一个算法的质量优劣将影响到算法甚至程序的效率，算法分析就是对一个算法的质量进行评价，其目的在于改进算法。那么，如何对一个算法进行评价呢？主要有以下几个方面：

(1) 正确性。这是一个前提。所谓正确的算法是指：当输入一组合理的数据时，能够在有限的时间内得出正确的结果；对于不合理的数据输入，能够给出警告信息。算法正确性的验证，可通过对数据输入的所有可能情况的分析以及上机调试进行。

(2) 时间复杂度。即依据该算法编写的程序在计算机中运行时间的度量。

(3) 空间复杂度。即依据该算法编写的程序占用计算机存储空间的度量。

(4) 其它方面。如可读性、可移植性以及测试性的好坏等。

从主观上讲，人们都希望选用一个既不占用很多存储空间，运行时间又省，而且其它性能又好的算法，然而时间与空间的开销往往是一对矛盾，很难十全十美。原因是上述要求在许多情况下是相互制约的：要缩短算法的执行时间，往往以牺牲更多的存储空间为代价，而节省了存储空间又要以牺牲更多的执行时间为代价。因此，我们只能根据具体情况有所侧重。若该程序使用次数较少，则力求算法简明易懂，易于编码与调试；对于反复多次使用的程序，应力求算法快速；若待解决的问题数据量很大，提供的存储空间较小，则相应的算法主要考虑如何节省空间。下面分别分析算法的时间复杂度、空间复杂度和其它有关方面。

1. 时间复杂度

一个程序在计算机中运行时间的多少与许多因素有关，其中主要有：

(1) 问题的规模，例如，是求 100 以内还是 1 000 以内的所有整数之和。

(2) 源程序的编译功能的强弱以及所产生的机器代码质量的优劣。

(3) 机器执行一条目标指令的时间长短。

(4) 程序中语句的执行次数。

一般情况下，前三个因素与计算机系统的硬件、软件以及要解决的问题有关，算法中一条语句的执行时间取决于机器的硬件速度、指令类型以及编译所产生的代码质量，只有第四个因素直接与算法有关，算法中语句执行次数越多，所耗时间越大。一个算法的时间复杂度为算法中所有语句执行时间之和，它等于该语句的总执行次数（也称频度）与语句执行一次所需时间的乘积。由于精确地计算出算法的总执行时间是相当复杂的，也是相当困难的，因此，只能根据给定的计算模式，粗略地估算该算法在执行时间上的量级。这里，我们用大写的 O 表示量级的概念。算法的时间复杂度记作 T(n)，它通常是问题规模的某个函数 f(n)，即 T(n)=O(f(n))，可理解为时间复杂度 T(n)与问题规模的函数 f(n)是同一数量级。

[例 1.5] 下面的三个程序段中：

```
                                        执行频度
①    x=x+1                                1
②    for （i=1；i<=n；i++)
         x = x+1;                         n
③    for (i = 1；i<= n；i++ )
       for（j=1；j<=n；j++）
       {
         x=x+1;                           n*n
       }
```

基本操作语句 x=x+1 的频度分别为 1，n 和 n×n，即这三个程序段的时间复杂度分别为 T(n)=O(1)，T(n)=O(n)和 T(n)=O(n*n)，分别称作常量，线性阶和平方阶。即 O(1）与问题规模 1 是同一数量级，O(n)与问题规模 n 是同一数量级，$O(n^2)$与 n^2 为同一数量级。算法还可以呈现的时间复杂度有：对数阶 O(log(n))，指数阶 $O(2^n)$等。不同数量级的时间复杂度