电脑速成系列丛书

数据结构入门

徐士良

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书是为已经初步学会电脑操作而开始应用电脑的读者编写的。书中介绍了数据处理领域中常用的基本数据结构及其操作,内容包括:数据结构的基本概念、线性表、栈、队列、数组、线性链表、查找与排序技术。

本书通过对实际例子的讲解介绍各种基本数据结构及其操作的方法,完全不涉及抽象的理论及深奥的计算机专业知识。书中对所有的算法都作了详细的说明。在与本书配套的软件中,还为读者提供了数据结构操作的模拟环境,给出了动态的操作过程。

本书语言通俗易懂,实例丰富,适用于初涉计算机应用领域的各类人员,可作为学习数据结构的自学教材或各类培训班的教材。

图书在版编目(CIP)数据

数据结构入门/ 徐士良编 .- 北京:清华大学出版社,1995.2
(电脑速成系列丛书)
ISBN 7-302-01705-0

Ⅰ.数… Ⅱ.徐… Ⅲ.数据结构-基本知识 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字 (94) 第 14493 号

出版者: 清华大学出版社(北京清华大学校内,邮编 100084)
印刷者: 北京密云胶印厂
发行者: 新华书店总店科技发行所
开 本: 850×1168 1/ 32 印张:5 字数:94 千字
版 次: 1995 年 2 月第 1 版 1995 年 12 月第 2 次印刷
书 号: ISBN 7-302-01705-0/ TP·744
印 数: 3001 - 8000
定 价: 45.00 元(含软盘) 4.80 元(不含软盘)

目　录

第三章 栈与队列

第四章 数 组

第五章 线性表的链式存储

第六章　树与二叉树

第七章　查　　找

第八章　排　　序

前　言

计算机的应用已涉及到各个领域,而用计算机对数据进行处理已经成为计算机应用的一个基本课题。家庭电脑的使用也离不开数据处理。

如果说本套系列丛书中的其它部分是为初学电脑者而写,那么本书是为已经初步学会电脑操作而开始进入电脑应用领域的读者而编写。

关于数据结构的书有很多,但适用于初涉电脑应用人员的书不多(甚至很少),而能够通过相应的学习软件进入“数据结构”大门的书就更少!

本书的起点很低,适用的读者面很广。书中对每一种数据结构都没有作抽象的定义,也没有涉及基本理论与计算机专业知识。在介绍一种数据结构时,总是从实例出发,通过对实例的分析、讲解来掌握基本概念,总结出每一种数据结构的特点及其应用。本书好象是在谈“家常”,但不知不觉地将你带入“数据结构”这个大门。当然,为了更方便地学习本书,最好是你已经初步学会了一种程序设计语言,并且对电脑已经有了初步的了解。在本书中,描述数据结构操作的算法语言与程序设计语

言是不同的,但读者是很容易认识它们的。对于初学者,也可以不必阅读书中的算法描述。只要你有一台装有 DOS 系统的 IBM PC 或兼容机(至少配置有单色图形显示卡或 EGA 卡),你就可以通过阅读本书及利用配套软件(运行 CAI 文件)很轻松地进入“数据结构”的大门,掌握各种基本的数据结构。

在与本书配套的软件中,为读者提供了对数据结构进行操作的模拟环境。在介绍每一种基本数据结构时,你都能在屏幕上很直观地观察到各数据元素之间的关系以及在计算机中存储的过程。在对数据结构进行操作时,你会感到身临其境,好象亲自在搬动这些数据元素,完全避免了只在书本上进行抽象解释的烦恼感。可以说,软件提供了学习、理解基本概念的活动场所,读者可以从动态过程中理解基本概念的真正含义。配套软件中每一单元的后面都配有“自我练习”,通过它可以检验你的学习效果以及对基本概念的理解程度。

由于作者水平所限,本书及配套软件中难免有不当及错误之处,恳请读者批评指正。

作 者

1994 年 5 月

第一章
对数据结构的认识

数据处理是计算机应用的一个重要领域。通常,实际需要处理的数据元素可能有很多,而这些大量的数据元素是存放在计算机中的。因此,大量的数据元素按什么结构存放在计算机中,以便提高数据处理的效率,这是进行数据处理的关键问题。

显然,杂乱无章的数据是不便于处理的。而将大量数据随意地存放在计算机中,实际上是“自找苦吃”,对处理更是不利。

“数据结构”作为计算机的一门学科,主要研究和讨论以下三方面的问题:

·数据的逻辑结构(即数据元素之间的逻辑关系)

·数据的存储结构(即数据的逻辑结构在计算机中的表示)

·施加在各种数据结构上的运算

讨论的主要目的是为了提高数据处理的效率。所谓提高效率包含两个方面:一是提高处理速度,二是节省计算机的存储空间。

本章首先通过两个实例说明不同的数据结构对处理效率的影响,然后介绍数据结构的基本概念、表示及其类型。

1.1 看两个例子

本节通过两个实例说明不同的数据结构对处理效率的影响。

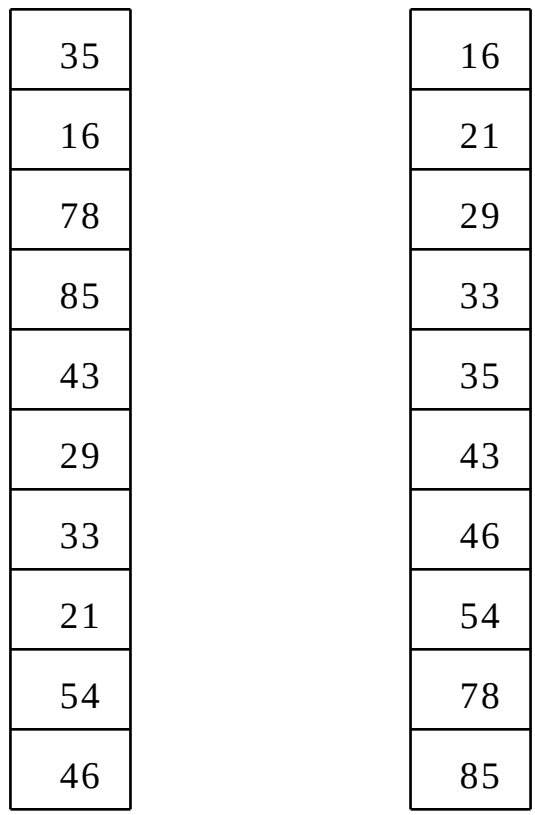
例 1.1 无序表的顺序查找与有序表的对分查找。

图 1.1 为两个表。从图中可以看出,在这两个表中存放的是同一批数,只不过是存放的顺序有所不同。其中图 1.1(a)所示的表中,数的存放顺序没有一定的规则;而图 1.1(b)所示的表中,数是按照从小到大的顺序存放的。我们称前者为无序表,后者为有序表。

现在考虑在这两种表中进行查找的问题。

首先考虑在图 1.1(a)所示的无序表中进行查找。

由于图 1.1(a)所示的表中,数的存放顺序没有一定的规则,因此,要在这个表中查找一个数时,只能从第一个元素开始,逐个将表中的元素与被查数进行比较,直到表中的某个元素与被查数相等(查找成功)或者表中所有元素与被查数都进行了比较且都不相等(查找失败)为止。这种查找方法称为顺序查找。(在第七章中我们将给出顺序查找的算法)。



(a) 无序表　　(b) 有序表

图 1.1　存放同一批数的两个表

显然,在顺序查找中,如果需要查找的数在表的前部,则比较的次数就少;但如果需要查找的数在表的后部,则比较的次数就多。特别是,如果需要查找的数正好是表中的最后一个或者根本不在表中,则需要与表中所有的元素进行比较。当表很大时,这种查找是很费时间的。但对于无序表来说,没有更好的查找方法,只能用顺序查找。

现在考虑在图 1.1(b)所示的有序表中进行查找。

由于有序表中的数是从小到大进行排列的,因此,在查找时可以利用这个特点,使查找过程中比较次数大大减少。在有序表中查找一个数可以这样进行:

将被查数与表中的中间这个元素进行比较:

若相等,则表示查找成功,过程结束;

若被查数大于表中的中间这个元素,则抛弃表的前部;

若被查数小于表中的中间这个元素,则抛弃表的后部;

然后对剩下的部分再用上述方法进行查找。这个过程直到在某一次的比较中相等(查找成功)或者表的剩下部分已空(查找失败)为止。这种查找方法称为有序表的对分查找。(在第七章中我们将给出有序表对分查找的算法)

显然,在有序表的对分查找中,不论你查找的是什么数,也不论你要查找的数在表中有没有,都不需要与表中的所有元素进行比较,并且只与表中很少的元素进行比较。但对分查找只适用于有序表。

在与本书配套的软件中,对本例中的两种查找方法进行了比较,通过观察动态的查找过程,你会有更深的体会。

实际上,在我们的日常工作与学习中,经常需要查找词典,采用的方法也类似于对分查找,这是因为词典中的各单词是以英文字母为顺序排列的。如果词典中的各单

词不是以英文字母为顺序排列,而是随意排列,那就也只能用顺序查找了,但这显然是糟糕得不能忍受。因此,数据元素在表中的排列顺序对查找效率是有很大影响的!

例 1.2　设有一学生登记表如表 **1.1** 所示。在表 1.1 中,有关每个学生的情况是以学号为顺序排列的。

显然,如果要查找给定学号的某学生的情况是很方便的。但是,如果要查找成绩在 90 分以上的所有学生的情况,则需要从头到尾地扫描全表,才能不会漏掉某一个成绩在 90 分以上的学生情况。在这种情况下,成绩在 90 分以下的所有学生也都要被扫描到。由此可以看出,要在表 1.1 中查找成绩在某个分数段中的学生情况,其效率很不高,尤其是当表很大时更为突出。

为了便于查找成绩在某个分数段中的学生情况,可以将表 1.1 中所登记的学生情况进行重新组织。例如,将成绩在 90 分以上、80～89 分、70～79 分、60～69 分之间的学生情况分别登记在四个独立的子表中,如表 1.2 所示。现在,如果要查找成绩在 90 分以上的所有学生的情况,就可以直接在子表 L_1 中进行查找,避免了对成绩在 90 分以下的学生情况进行扫描,从而提高了查找效率。

表 1.1 学生登记表

学 号	姓 名	性别	年龄	成绩
80156	张小明	男	20	86
80157	李小青	女	19	83
80158	赵 凯	男	19	70
80159	李启明	男	21	91
80160	刘 华	女	18	78
80161	曾小波	女	19	90
80162	张 军	男	18	80
80163	王 伟	男	20	65
80164	胡 涛	男	19	95
80165	周 敏	女	20	87
80166	杨雪辉	男	22	89
80167	吕永华	男	18	61
80168	梅 玲	女	17	93
80169	刘 健	男	20	75

表 1.2　子表

子表 L_1

学号	姓名	性别	年龄	成绩
80159	李启明	男	21	91
80161	曾小波	女	19	90
80164	胡　涛	男	19	95
80168	梅　玲	女	17	93

子表 L_2

学号	姓名	性别	年龄	成绩
80156	张小明	男	20	86
80157	李小青	女	19	83
80162	张　军	男	18	80
80165	周　敏	女	20	87
80166	杨雪辉	男	22	89

子表 L_3

学号	姓名	性别	年龄	成绩
80158	赵　凯	男	19	70
80160	刘　华	女	18	78
80169	刘　健	男	20	75

子表 L_4

学号	姓名	性别	年龄	成绩
80163	王　伟	男	20	65
80167	吕永华	男	18	61

由这个例子可以看出,在对数据进行处理时,可以根据所需要做的运算不同,而将数据组织成不同的形式,以便提高数据处理的效率。

1.2　什么叫数据结构

概略地说,所谓数据结构是指反映数据元素之间关系的数据元素集合的表示。

数据元素具有广泛的含义。一般来说,现实世界中客观存在的一切个体都可以是数据元素。例如:

描述一年四季的季节名

春,夏,秋,冬

可以作为数据元素;

表示数值的各个数

18,11,35,23,16,…

可以作为数据元素;

表示家庭成员的各成员名

父亲,儿子,女儿

也可以作为数据元素。

甚至每一个客观存在的事件,如

一次演出,一次借书,一次比赛,……等也可以看作为数据元素。

总之,在数据处理领域中,每一个需要处理的对象都可以抽象成数据元素。数据元素一般简称为元素。

在实际应用中,被处理的数据元素一般很多,而且,作为某种处理,这些数据元素一般具有某共同特征。例如,{春,夏,秋,冬}这四个数据元素有一个共同的特征,即它们都是季节名,分别表示了一年中的四个季节,从而这四个数据元素构成了季节名的集合。又如,{父亲,儿子,女儿}这三个数据元素有一个共同特征,即它们都是家庭的成员名,从而构成了家庭成员名的集合。一般来说,人们不会同时处理特征完全不同的各类数据元素,对于具有不同特征的数据元素总是分别进行处理。

一般情况下,在具有相同特征的数据元素集合中,各数据元素之间存在有一定的关系,这些关系反映了该集

合中的数据元素所固有的结构。在数据处理领域中,通常把数据元素之间这种固有的关系简单地用前后件关系(或前驱后继关系)来描述。例如:

在考虑一年四个季节的顺序关系时,则“春”是“夏”的前件(即前驱),“夏”是“春”的后件(即后继)。同样,“夏”是“秋”的前件,“秋”是“夏”的后件;“秋”是“冬”的前件,“冬”是“秋”的后件。

在考虑家庭成员间的辈份关系时,则“父亲”是“儿子”与“女儿”的前件,而“儿子”与“女儿”均是“父亲”的后件。

前后件是数据元素之间的一个基本关系,但前后件关系所表示的实际意义是随具体对象的不同而不同。数据元素之间任何其它的复杂关系都可以用前后件关系来表示。

在本节的开头已经提到,数据结构是指反映数据元素之间关系的数据元素集合的表示。更通俗地说,数据结构是指带有结构的数据元素的集合。在此,所谓结构实际上是指数据元素之间的前后件关系。

因此,一个数据结构包含两个方面的信息:

·表示了所有数据元素的信息;

·表示了各数据元素之间的前后件关系。

上面所述的数据结构中,其中数据元素之间的前后件关系是指它们的逻辑关系,而与它们在计算机中的存储位置无关。因此,上面讲到的数据结构实际上是数据

的逻辑结构。所谓数据的逻辑结构,是指反映数据元素之间逻辑关系的数据结构。

数据处理是计算机应用的一个领域,在实际进行数据处理时,被处理的各数据元素总是被存放在计算机的存储空间中,并且,各数据元素在存储空间中的位置关系与它们的逻辑关系不一定是相同的,而且一般也不可能相同。例如,在前面讲到的一年四季的数据结构中,“春”是“夏”的前件,“夏”是“春”的后件,但在计算机存储空间中,“春”这个数据元素的信息不一定存储在“夏”这个数据元素信息的前面,可能在后面,也可能不是紧邻在前面,而是中间被其它的信息所隔开。又如,在家庭成员的数据结构中,“儿子”与“女儿”都是“父亲”的后件,但在计算机存储空间中根本不可能将“儿子”与“女儿”这两个数据元素的信息都紧邻在“父亲”这个数据元素信息的后面,即在存储空间中与“父亲”紧邻的只可能是其中的一个。由此可以看出,一个数据结构中的各数据元素在计算机存储空间中的位置关系可能与逻辑关系不同。

数据的逻辑结构在计算机存储空间中的存放形式称为数据的存储结构。

由于数据元素在计算机存储空间中的位置关系可能与逻辑关系不同,因此,为了表示存放在计算机存储空间中的各数据元素之间的逻辑关系(即前后件关系),在数据的存储结构中,不仅要存放各数据元素的信息,还需要存放各数据元素之间的前后件关系的信息。