高等学校试用教材

数据组织与管理

美国奥本大学(蒙) 何耀钦
复 旦 大 学 薛华成 主编

机械工业出版社

高等学校试用教材

数据组织与管理

美国奥本大学(蒙) 何耀钦
复旦大学 薛华成 主编

机械工业出版社

(京)新登字054号

本书共三篇。第一篇数据结构，包括：堆栈、队列与链表，树与图，排序和查找；第二篇文件组织，包括：文件组织概述，顺序文件、索引文件和倒排文件，文件的排序与合并，文件管理；第三篇数据库系统，包括：数据库系统的概念，数据库数据模型，数据库实例ORACLE，数据库设计，数据库的维护与管理。这三部分并非简单拼凑，而是与之相关内容的有机结合，以适应于进行企业或组织的数据组织与管理的需要。

本书系统性强，理论结合实例，论述深刻，文字通俗易懂。每章附有习题，便于读者学习。本书为高等学校管理信息系统专业试用教材。也可作为其他管理类及计算机应用专业的本科生、研究生、教师及工程技术人员的教材或参考书。

数据组织与管理

美国奥本大学(蒙)　何耀钦
复　旦　大　学　薛华成　主编

*

责任编辑：刘同桥　责任校对：韩　晶
封面设计：郭景云　版式设计：乔　玲
责任印制：王国光

*

机械工业出版社出版(北京阜成门外百万庄南街一号)
(北京市书刊出版业营业许可证出字第117号)
机械工业出版社京丰印刷厂印刷
新华书店北京发行所发行·新华书店经售

*

开本 787×1092 1/16·印张19 1/4·字数474千字
1992年5月北京第1版·1992年5月北京第1次印刷
印数 0,001—1,600　定价：5.40元

*

ISBN7-111-03052-4/F·412(课)

前　言

本书是根据高等学校管理信息系统专业教学指导小组制定的教材编写规划和审定的教学大纲编写的。美国奥本大学（蒙）（Auburn University MontgoMery）美籍华裔教授何耀钦（Yaw-Chin Ho）和复旦大学薛华成教授担任主编，北方交通大学陈景艳教授担任主审。薛华成拟定全书结构和编写大纲，并撰写绪论和统稿；何耀钦对全书构思作了指导并亲自编写了第一篇数据结构；复旦大学杨缨琳副教授编写了第二章图的部分并为第一篇编写了习题；中国纺织大学归瑶琼副教授、张礼平副教授编写了第二篇；清华大学齐英杰副教授编写了第三篇。

本书主要的特点是从实用的观点，着重于物理概念的论述，因而虽然内容不浅但通俗易懂。在内容上包括了一般计算机专业的《数据结构》《文件系统》及《数据库》三门课的内容，这三部分并非简单拼凑，而是与之相关内容的有机结合，有利于读者从全局的观点考虑问题及综合应用。在编写中，删除了许多过细的公式推导。

本书是管理信息系统专业的试用教材。也可作为管理各类[illegible]计算机应用专业的教材。还可作为各种干部培训班及企业信息系统开发设计人员的教材或参考书。

由于编者水平所限且时间仓促，错误和不当之处在所难免，敬请读者指正。

编　者

1990年5月

目　录

绪　论

一、数据与信息

随着社会生产和管理现代化的发展，信息已成为企业和事业单位日益重要的资源，甚至成为它们的生命线。如果说过去一个企业可能会因没有资金、没有原材料或没有能源而濒于倒闭的话，那末今天已有许多企业因为没有信息而宣告破产。在国际市场竞争日益激烈的情况下，这种情况可以常常听到。例如，美国的汽车制造业由于没有估计到会发生石油危机，没有尽早地生产低油耗汽车而竞争不过日本，使得亏损达到无法承受的地步，不得不依靠政府的帮助。又如，美国的王安计算机公司由于没有把握住微型机的市场需求发展趋势，生产的计算机不适合用户的要求而推销不出去，每年亏损达到上百亿美元。国内也有类似的情况。一方面产品供不应求，另一方面又有许多同类产品因为不适销对路而卖不出去，结果造成产品大量积压，流动资金被大量占用，使企业陷入了无资金、无材料、被迫停产的境地。

信息被人们利用于管理，大致分为三个阶段。第一个阶段为提高效率阶段，即用信息技术来代替人的手工劳动进行信息处理，加快了处理时间，提高了工作效率。这个阶段的着重点在于处理。第二个阶段为及时转化价值阶段。例如得到了某个车间可能有窝工的消息，就及时安排工作任务，信息就转化成为价值。这个阶段的着重点在于建立完善的数据库，收集，存贮内部信息，利用信息进行控制。产生异常报告是这个阶段的重要特征。第三个阶段为寻找机会阶段。例如在国际市场上发现石油危机可能出现，生产低油耗的汽车技术已经成熟，领导和环境有可能接受新的生产方案，因而提出利用新机会的报告。这个阶段的特点在于不确定性或风险性，信息收集的外向性以及决策研究的重要性。

由于信息的日益重要和信息同管理现代化有着密切的关系，在80年代产生了信息管理学派。他们认为信息是企业管理最重要的资源，掌握了信息就等于掌握了一切，没有材料就可以搞到材料；没有资金就可以搞到资金；没有能源就可以搞到能源……。因此，现代管理离不开信息。我国现在也认识到了信息的重要性，十分重视企业的信息利用和管理信息系统的建设。国家体制改革委员会在企业等级规范中就明确地规定了对管理信息系统的要求。例如，如果企业要想升为二级，必须有管理信息系统规划；要想升为一级，必须实现一些管理信息系统规划。这说明我国对信息已有了高度重视，对信息的利用出现了高潮。

信息和数据在形式上有许多共同之处，但本质上却不相同，两者的概念不应混淆。数据是描述现实世界的符号，相关性与正确性是数据的重要特性，也是决定数据对使用者是否可用的关键因素。由于描述现实世界有不同的方法，故描述的表达方式也多种多样。例如，有些情况宜用照片和草图表达，有些情况宜用文字描述表达，有些情况则宜用数字表达。所以，这里所说的数据（*Data*）是广义的。因为它不仅限于数字，故有人也把它翻译成"资料"。但是，不管数据的形式如何，它都只是对现实世界的一种描述。信息则不同，信息是经过加工后，对使用者的行为能产生影响的数据。它在形式上虽然也是一种数据，但是只有它具有影响使用者的作用时才能算得上是信息。例如，报纸上刊登的卖录音机的广告，只有对想买录音机的人来说它才是信息。信息的另一种概念是对不确定性的度量，也就是说通过信息的传播和接收可以消除人们

对客观世界了解的不确定性或降低其程度。这实际上也会对信息使用者的行为产生影响。信息和数据是可以互相转化的，一个系统输出的信息可能成为另一系统的数据，过时的信息可成为数据，过时数据的积累也可能成为信息，一个系统中的数据经过加工以后可以变成有用的信息。数据好像是信息的原材料，只有有了好的原料才能加工出好的信息。为了要准备好数据原料除了要求个体数据具有相关性和准确性以外，对群体数据还要进行合理组织与管理。数据合理组织的目的不外乎是减少冗余以使占用的存贮空间较少，能较快和较方便地存取，能较快和较方便地组成用户需要的数据形式。数据组织一般来说只涉及数据的形式而不涉及数据内容。这是和信息组织的不同之处。信息组织要根据管理的目标决定需要存贮哪些信息，存贮的信息应当保存多长时间等等。本书主要研究数据的组织与管理，不讨论数据的内容。

二、数据与计算机

计算机已成为处理数据的最有效的工具。计算机可以完成数据处理的几乎所有工作，包括数据的收集、传输、加工、更新、维护和使用等。当然在有些方面计算机表现得非常优越，而在另一方面则表现得还不够理想。当前使用计算机存贮、加工数据和支持使用而存取检索数据是最为理想的。

为使计算机能合理地处理数据，就要求数据有适合于计算机的组织形式。适合计算机化的数据形式是结构化的数据。结构化的数据组织称为数据结构（*Data Structure*）。为了清楚了解计算机中数据结构的概念，让我们描述一下世界上的事物和数据的关系。

世界上只有哲学家能以全局的观点看待世界，其他的任何人都只能窥视世界的一个小的剖面。任何一个人要想解决某个问题，他只会对一定的目标感兴趣。每个目标可以看作是一个个体（*Individual*），个体的集合被看作是总体（*Universe*）。个体可能是各式各样的。例如：个体可能是人，如工资单上的人员；可能是物体，如工厂制造的零件；可能是结构（一种虚构的个体），如一笔会计帐。这里所说的总体，对同一个企业讲可能有不同的概念。例如，企业处理按周计算的工资就和处理按月计算的工资不同，企业的职工作为总体将被分成两个或多个总体。

每个个体又有许多品质。对于某个问题来说，我们所感兴趣的品质只是它所有品质中的一部分，这部分品质叫"属性"（*Attribute*）。例如对职工这个个体来说，其属性可能是姓名、性别、工资等。这里个体名是职工，属性名是姓名、性别、工资。如果在职工中有人名叫张三、男、工资105，那么这个张三、男、105就是属性值（*Attribute value*）。为了要描述一个职工，要在职工的每一个属性中指明其属性值，这样才能把他描述出来。一般来说属性值有个范围（*Range*）。例如职工年龄一般是18～70岁，不可能超过100岁。规定范围以后可以用范围粗略地判断属性值是否正确。如果在年龄的属性值中发现超过100，这就是个异常情况，在数据处理系统中就要作出异常报告。

属性的选择应当针对问题。与问题无关的属性一般应予以排除，这样可以简化问题，和使处理有效。因此，当解决问题时，首先要划分与问题有关的总体。

例如，形象地将一个班的孩子看作为一个总体，参见图0-1。每个孩子是一个个体。对于某一个孩子，他所有的属性值组成一个记录（*Record*）。一个记录包含许多数据项或场（*Field*），每个场存放着一个属性值。对此问题可列表表示如表0-1。这里，王二毛　男、6、1.2、30组成一个记录，而6或1.2或30是一个场。每个场又可能由n个字符组成，这些字符可能是汉字、字母、数字、标点符号以及空格等。

许多记录的集合组成一个文件（*File*），各种各样文件的集合称为文件库（*Library*）。对于场、记录、文件和文件库来说，均存在着子集（*Subset*），也就是它们中一部分的集合。这些子集可能和总体有一些不同的特点。

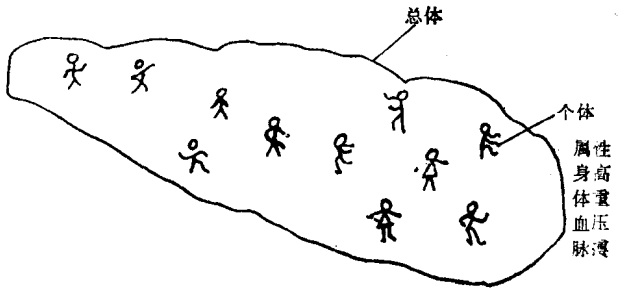

图0-1 总体、个体、属性

三、数据的组织与管理

顾名思义，数据组织与管理是把数据按适合于计算机处理的形式组织起来，然后进行管理。要使数据成为计算机容易处理的形式，数据就要具有一定的结构，因而研究数据的结构就是数据组织的第一步。数据结构又分为数据的内存结构和外存结构。内存结构是指如何把计算机内存中的数据组织起来（例如，如何把数据组成顺序表、链表或树形结构），如何在这种结构下进行操作（如何进行增加、删除、查找、转换等操作）以及实现这些操作的可行性及其效率。外存结构是指数据在磁盘上的存贮结构及其操作方法。由于存在磁盘上的数据均是以文件的形式存贮的，所以外存结构往往又称文件结构。文件的形式有顺序文件、索引文件、随机文件等。文件和管理它们的程序的总体就叫做文件系统。

表0-1 属性值

姓　名	性　别	年　龄	身　高	体　重
王二毛	男	6	1.2	30
张玲玲	女	5	1.1	28

利用文件来存贮数据是数据处理上的一大进步。它把数据和程序分开了。这样，一个文件可以被多个程序使用；一个程序也可以使用多个文件上的数据；在修改程序和文件数据时也不互相影响。当然程序存贮的概念也可以在这种情况下应用，也就是程序可按文件的形式存于存贮器中。因此，在存贮器中实际上有两种文件：一种是数据文件，一种是程序文件。它们在形式上都是数据，但是其用途是不相同的。文件系统虽然在数据组织上有了很大的进步，但是它还有许多不足之处，尤其当数据很多时，各个文件所拥有的数据相互之间有很多重复，或者叫冗余，如图0-2所示。同时，在文件之间传送数据也很不方便。

为了解决这个问题，在60年代后期进行了大量的关于数据库（*Data Base*）的研究。数据库是个通用化的综合性的数据集合。它是从全局的观点来组织数据的，各文件中重复的数据在数据库中只存贮一次，因而减少了对存贮容量的压力。用数据库来组织数据还可使各种程序、数据比较容易应用，存取维护也都比较方便。

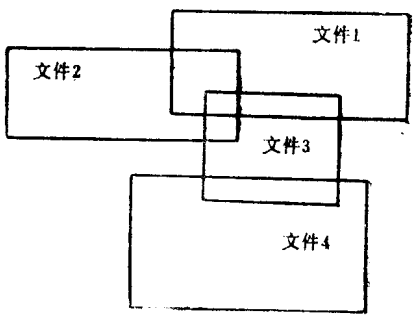

图0-2 文件数据的重复

数据的管理是指如何对数据进行维护，保证它们均能反映现时的最新状态。如何方便地建立、更新、

删除、存取数据以及在事故状态下如何恢复数据和如何保证数据的保密等问题。当一个总体中增加一个个体时叫做诞生；当一个总体中减少一个个体时叫做死亡。诞生就要求建立一个记录，死亡则要求删除一个记录。如果建立和删除的操作能使计算机的数据和总体的最新状态保持一致就叫做更新。存取是在不改动记录的情况下把记录调出调进。当然存取的过程也可能伴随着建立、删除和修改。所有这些操作可以针对一个记录，也可以针对一个文件或记录中的一个数据项。以上所述是狭义的数据管理内容。一般并不涉及数据的内涵性质。广义的数据管理应当和企业的目标联系起来决定应当存什么数据，以何种形式存，是和程序一起存还是用文件或数据库存，存多长时间，以及如何用它们进行控制和决策等。

本书针对计算机内部的数据组织与管理，重点阐述结构化的数据组织与管理的方法。包括内存结构、文件结构和数据库结构。本书在编写中，从应用的角度出发，系统地、全面地论述了数据组织与管理的理论与应用方法，以便于读者在实际工作中根据需要评价和选择合适的方法。

第一篇 数据结构

第一章 堆栈、队列与链表

堆栈和队列 (*Stack and Queue*) 是操作系统 (*Operating systems*) 的两种主要的基本结构。操作系统用堆栈来记录执行程序 (*Execute programs*) 的先后次序，用队列来控制计算机执行作业(*jobs*)的前后次序。但在管理应用中，堆栈与队列只具有间接达成某种算法的辅助作用。所以它们只是一种暂时存取的物理结构。

在解释堆栈与队列之前，有必要先了解线性表 (*Liner list*)的概念。线性表是由一组有顺序的元素 (*An ordered set of elements*) 组成的。线性表里的元素可增可减，所以，如果用 L 代表有 N 个元素的线性表时，这个线性表可用下列方式表示：

$$L=(L_1, L_2, \cdots, L_N)$$

如果 $N=0$，L 就是"空表" (*null list*)。表中任何一个元素，都可以从表中的任何一个位置删除，同时新元素也可以插入表中的任何一个位置。线性表可以用链表 (*Linked list*)实现。

§1-1 堆 栈

堆栈是线性表的一种。数据元素存入堆栈中，依照的是后存先取的规则。所以堆栈又称为后进先出表，简称LIFO (*Last in first out*) 表。这一概念可用食堂里洗盘子和取用盘子的过程来解释。通常洗盘子时，总是把先洗好的盘子放在底下，然后依次叠放在上面。由于后洗净的盘子堆放在上面，所以取用时往往先拿上面后洗净的盘子。其过程如图1-1所示，第一个盘子洗净后置于堆底，第四个盘子反而在这一堆的最上面，取用时，则先拿最后洗净的第4个盘子。

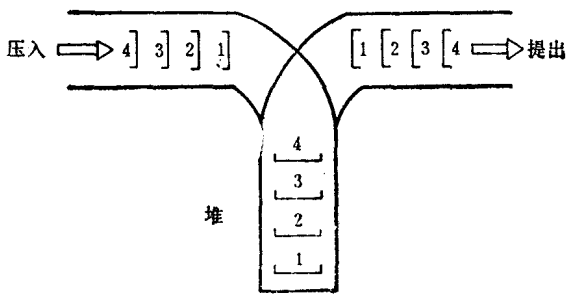

图1-1 堆栈的图例

堆栈可使用一维数组 (*One dimensional array*) 或称向量 (*Vector*) 来表示。使用向量时，先决定维的界限，堆栈存取时用指标 (*Index*) 指明存和取的位置。因为堆栈遵从后进先出的规则，存取时不但在同一位置上，而且总是在最上面的一个位置上，所以，习惯上都用TOP作为指标。图1-2是堆栈的物理表示：假设向量的界限 (*Maximum*) 为5。存第一个元素时，TOP=1，第二个时，TOP=2，第三个时，TOP=3；取时则先取TOP=3，再取TOP=2，最后取TOP=1。这种运算规则，进出只由一个方向进行，所以，只要用一个指标 (TOP) 就可以实现。

堆栈的主要运算包括压入堆栈和提出堆栈。增加一个数据元素（*Data element*）到堆栈里，称为压入堆栈（*Push the stack*），从堆栈中删除一个元素，则称为提出堆栈（*Popping the stack*）。

堆栈的操作并不复杂。压入的运算只要将数据元素从栈顶压入，同时将堆栈指针加 1（TOP＋1），该数据元素就可以加入到TOP所指的位置上。如图1-3所示，最先由向量表示的堆栈的存贮空间中，其满栈界限值为 5，TOP＝0（见图1-3a）。如果要压入元素 S 时，则：

TOP＝TOP＋1 ＝ 0 ＋ 1 ＝ 1（见图1-3b）

表示应该将 S 存入*stack*(1)的位置上。在执行压入运算之前，必须检查有没有满栈的情形。当TOP的值等于界限值时，表示栈满。如图1-3c所示。当*K*压入堆栈之后，TOP的值等于界限值 5。此时意味着满栈，就不能压入新的数据元素。

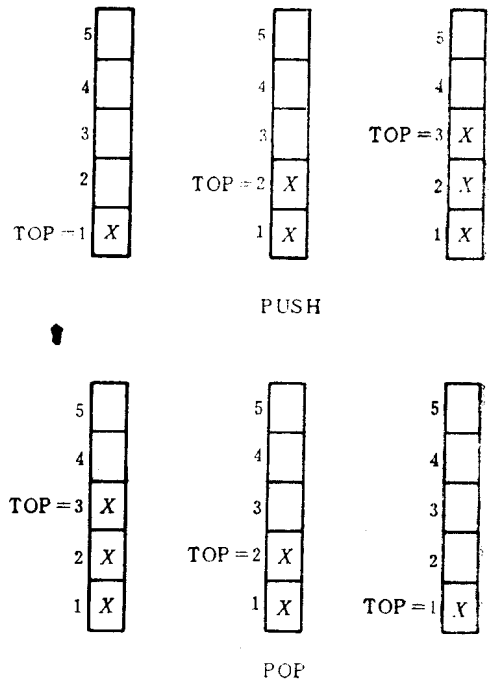

图1-2　堆栈压入和提出实例

综合以上的叙述，压入的算法可以用图1-4的程序框图表示。

因为堆栈的运算规则是单方向操作，所以提出堆栈的算法也跟压入堆栈的算法相似，也以TOP为指针，指出被提出的堆栈位置，如图1-5a所示。栈顶（TOP）在第五个位置（TOP＝5）时，要从这个堆栈中提出一个数据时，必须由第五个位置（TOP＝5）提出。当*K*被提出之后，栈顶就从第五个位置降落到第四个位置。由此类推，在连续提出四个元素之后，栈顶就落到第一个位置。在压入之前，必须先检查是否栈满。同样地，在提出之前，提出算法必须检查是否有栈空的情况。前者不能压入，后者则无从提出，如图1-5d所示。在所有数据元素都被提出之后，就成为空栈（TOP＝0）。

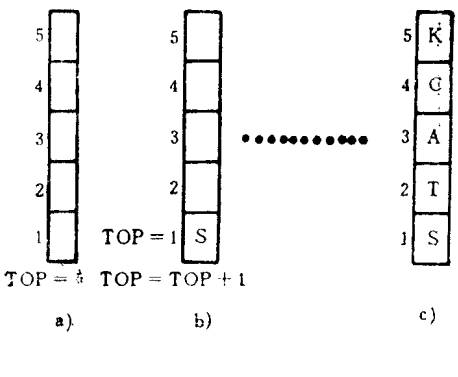

图1-3　堆栈的压入实例

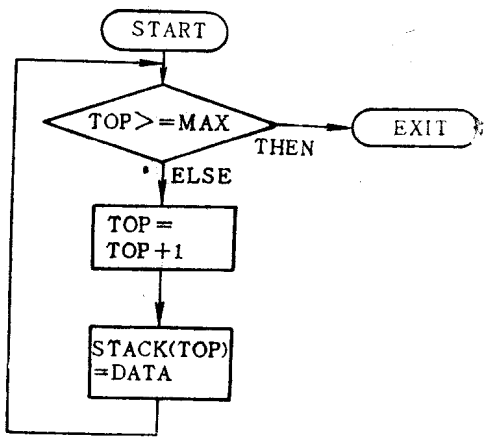

图1-4　堆栈压入的算法

综合以上的解释，可以用图1-6的框图描述提出堆栈的算法。

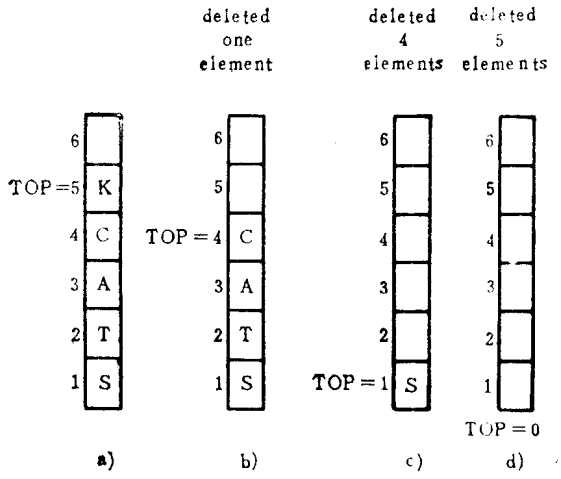

图1-5 堆栈的提出实例

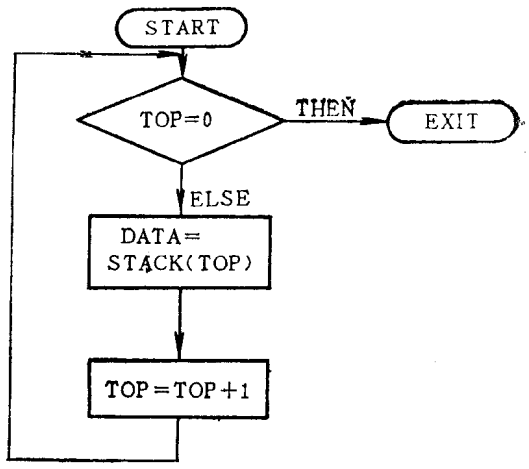

图1-6 堆栈提出的算法

§1-2 队 列

队列（*Queue*）也是线性表的一种。数据元素存入队列中，依照先存先取的规则。所以，队列又称为先进先出表，简称为FIFO（*First in first out*）表。在日常生活中，排队买票的习惯是一种先到先买的规则。也就是说，先到达售票口的先买到票，就先出列。在计算机系统中，操作系统对执行作业（*Jobs*）的次序就是依照这个先进先出的概念建立的。队列与日常排队的情形相似，进来的时候由“队尾”（TAIL）入队。出列的时候由“队首”（HEAD）出队。所以，队列系由两个方向进行。如图1-7所示。

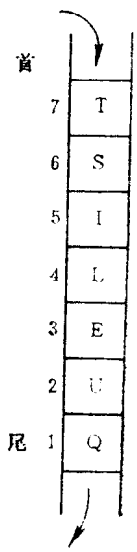

图1-7 队列存入的实例

与堆栈一样，队列也可以用一维数组（或向量）来描述，但与堆栈单一方向的操作规则不同，队列必须遵从双方向操作规则，即存取由不同方向进行。因此，在实现队列运算时，存与取必须各有其指针。习惯上在存时用“尾指针”（*Tail Index*），取时用首指针（*Head Index*）。每次存、取时其各自的指针都加1。其过程如图1-8所示。

队列的主要运算包括把数据加入队列和提出队列。从上述的例子中可以看出，如用TAIL为队尾指针，每次增加数据之前，将TAIL值加1，所得TAIL的新值指出加入队列的位置。“满队”时，TAIL值就等于队列的界限值。反之，从队列中提出数据时，是根据队首指针指出的位置提出，再将HEAD加1。新的HEAD值指出下次将被提出的队列的位置。不过，要从队列中提出数据之前，也必须检查队列是否为空的。由图1-8可以看出，HEAD=1，TAIL=0时，以及HEAD=8，TAIL=7时，队列是空的。换句话说，当HEAD的值大于TAIL的值时，队列是空的。

如果用QUEUE表示队列向量MAXIMUM表示向量界限，HEAD表示队首指针，TAIL表示队尾指针，综合以上所叙述的运算规则，队列的加入及删除算法可用图1-9的框图表示。

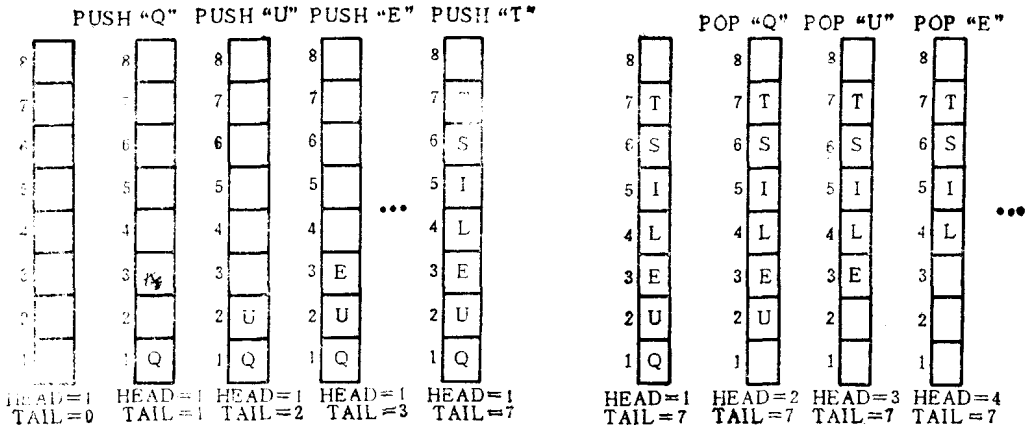

图1-8 队列存取的实例

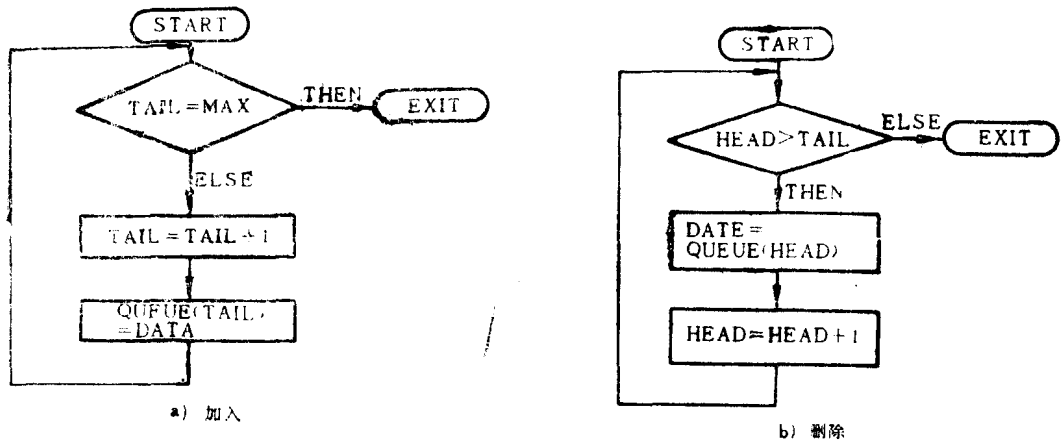

图1-9 队列的加入及删除算法

队列在具体实现时，可能面临一个困难。因为队列的运算向着队列两端进行，所以，数据元素在继续压入和提出时，可能会发生队列的物理存贮空间未满，但队尾的指针值已满的情况。这种情况可以用图1-10来解释。从图1-10a、b中可以看出，如果提出Q之后，队列就有一个存贮空间空出来，可以接受新的数据。但此时的队尾指针值却等于界限值(TAIL=MAXIMUM=7)。如果用上面所介绍的压入算法，检查时会发现队满的状态。这等于浪费了一个存贮空间。不但如此，如果再从队列中提出"U"如图1-10c所示。在物理结构上，该队列事实上有两个存贮空间可供利用，但由于队尾指标值仍然等于界限值，因此不能再加入任何数据。循环队列（*Circular Queue*）即是用来解决这一困难的。

循环队列的算法见图1-10。在图1-10a的例子中，显然地，如果在队满（即TAIL=7）时，提出Q之后，能够让TAIL值回到队列的起点（即TAIL=1），则新的数据就可以压入在*Queue*(1)的位置，如图1-11所示。

根据上面的例子，循环队列的队尾指针是随着队列的界限值循环的。即如果MAXIMU-

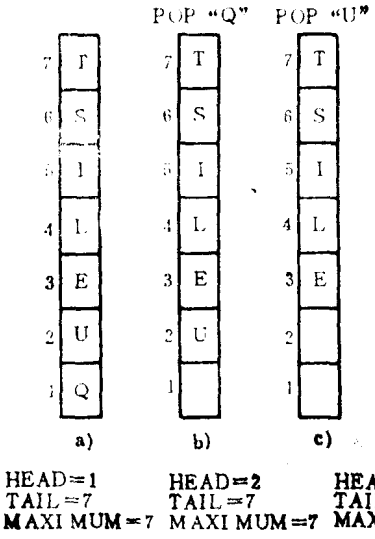

图1-10 循环队列实例

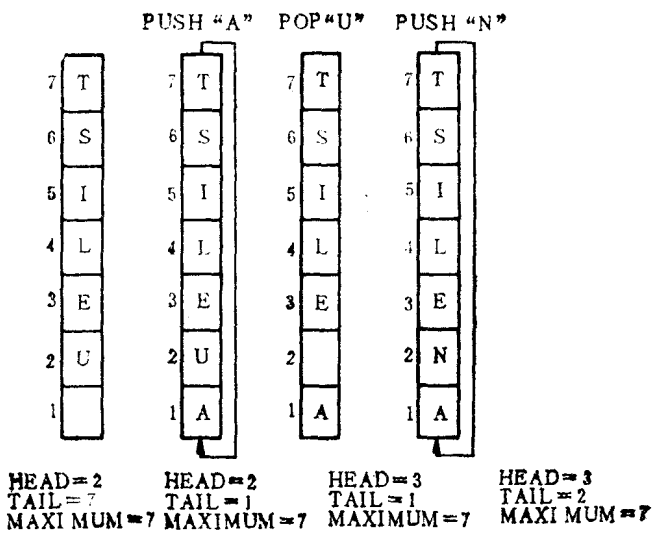

图1-11 循环队列的运算

M等于7，则队尾指针每隔7个存贮空间循环一次。如果有下列的数据单项：

Q，U，E，L，I，S，T，A，N，D，S，T，R，U，C

可以视为：

Q——1
U——2
E——3
L——4
I——5
S——6
T——7

A——1＋7
N——2＋7
D——3＋7
S——4＋7
T——5＋7
R——6＋7
U——7＋7

C——1＋7＋7

即按界限值等于7，将所有数据分为每7个一组，其加入队列的次序与位置也依照1～7排列的规则。这样，虽然当队尾的指针值增加到界限值的时候，只要有数据项被提出，队尾的指针就可以"循环"到开始的队列位置上。也就可以将要加入的数据项加在队列之内。在数学上，可以使用"模数"(*Modular*)的原则处理。例如：$8_{MOD}7=1$。与堆栈和普通队列的算法一样，

每次要加入数据项时，必须检查是否处于满与空的状态。循环队列也不能例外。不过为方便计，除了用队首、队尾、界限值(MAXIMUM QUEUE)之外，这里所要介绍的算法，还要增加使用"计数"(COUNTER)来统计存在队列中的数据项数。如果COUNTER＝0，则表示"空队列"。如果COUNTER＝MAXIMUM QUEUE，则表示"满队列"。其算法框图如图1-12所示。

插入时，根据队尾指针值，TAIL＝HEAD＋COUNTER。如以图1-11为例：COUNTER＝6，HEAD＝2，TAIL＝6＋2＝8。因为TAIL＝MAXIMUM，所以用MOD函数，TAIL＝8MOD7＝1表示新的数据可以在第一位上插入。

同样地，队列提出的算法，也使用MOD函数计算HEAD值。如图1-13所示。HEAD的起始值等于1。每次提出一个数据时就对HEAD加1，当第7个数据被提出时，HEAD＝7＋1＝8，就大于界限值，所以用MOD函数计算，HEAD＝8MOD7＝1，就将队首值循回到起始值。

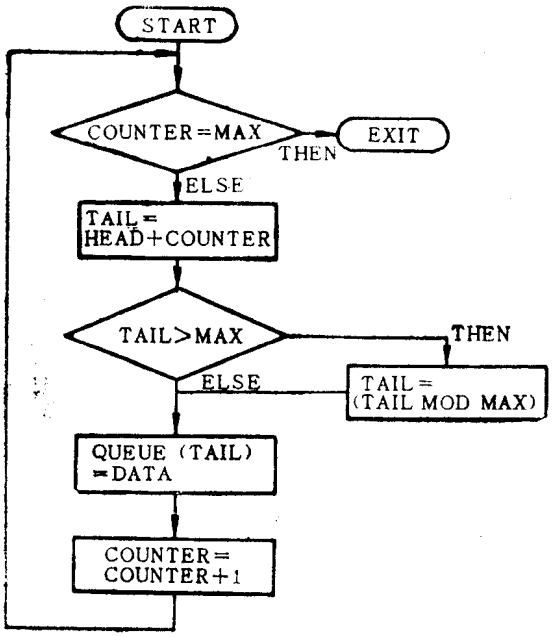

图1-12　循环队列插入算法

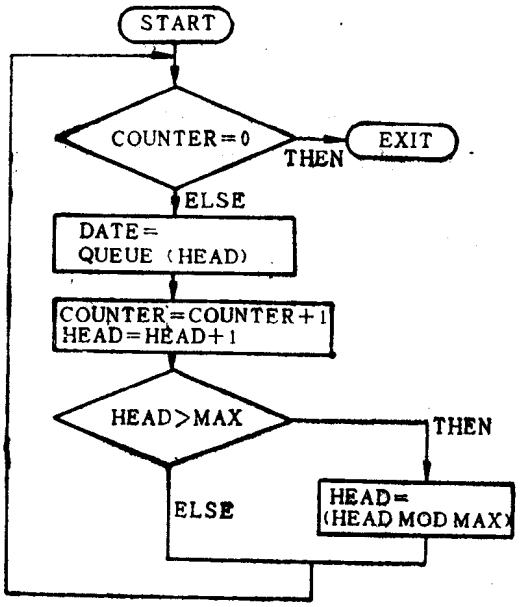

图1-13　循环队列删除算法

§1-3　链　　表

一般较高层次的计算机语言（*Higher Level Language*）所提供的文件组织方法，往往为了迁就软件的物理结构，使其存取（插入和删除）的运算过分简略，因而费时费力。以使用顺序存取法为例，在插入1个记录时，必须重写整个文件，记录不能直接插入原文件中间。下面的例子可以用来说明这种困难。如图1-14所示，原文件中有5个记录，如果想在第3和第4个记录中新加入1个记录时，使用顺序存取法及相关存取法，都必须将前面3个记录抄写到新文件的第1、2、3个记录位上。再将新记录写入第4个记录位上，最后将旧文件的第4、5个记录抄写到新文件的第5和第6个记录位上。

这种文件处理的方法，在使用有大量记录的文件时，既复杂，又费时。上面所举的例子

是从文件的物理结构上分析的。同样，如果从文件应用（*Application*）的角度来看，其困难更为显著。比方说，从图1-15所示的学生文件中，想找出各专业的学生人数的话，就必须从每一个记录中的专业域去查找各专业，再统计各专业的人数。这就是说，如果想了解信息（IS）专业有多少学生的话，就必须从第1个记录起，一直查到最后一个记录，才能统计出所有IS专业的人数。要想知道其他每一个专业的学生人数时，一定又要重新由第1个记录开始查找到最后1个记录为止。可见这种文件的处理方法是非常笨拙与繁杂的。

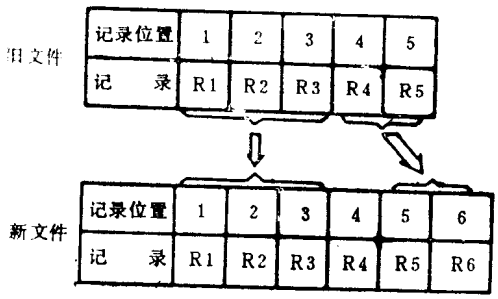

图1-14 文件的物理表示

记录位置	学生代号	专 业
1	A	IS
2	G	IS
3	B	MN
4	C	AC
5	H	MN
6	F	IS
7	D	MN
⋮	⋮	⋮

图1-15 学生信息文件

要解决上述困难，可以使用相关存取法或直接存取法组织文件，然后在每一记录中加1个指针域用来存贮作为链接各专业的指针值。如果使用相关存取法的话，则可以用记录位做指针值。如果使用直接存取法，则可使用记录的关键值。其结果如图1-16所示。

记录位置	学生代号	专 业	指 针
1	A	IS	2
2	G	IS	6
3	B	MN	5
4	C	AC	11
5	H	MN	7
6	F	IS	18
7	D	MN	10
⋮	⋮	⋮	⋮

IS第1个记录＝1

MN第1个记录＝3

AC第1个记录＝4

图1-16 学生信息文件链表

图1-16中第1、2和6记录是IS专业的学生。这3个记录的指针值分别为2、6和18表示第2记录后继第1记录，第6后继第2记录，第18记录后继第6记录。所以，要查找所有IS专业的学生时，只要找到IS专业的第1个记录，其他就可以顺藤摸瓜循着指针值直接找出，而不必查找每个记录。通常每一专业的第1个记录，存贮在链首域（*Header field*）里。

以上所用的方法称为链表。其作用就是使用指针（*Pointer*）将记录的逻辑关系"链接"起来。每一指针值指出每一记录的"后继记录"（*Next record*）。

链表虽然增加了程序设计的要求，但同时也增加了文件在应用上的效果。事实上，所有文件组织与管理的基本方法都以链表为基础。链表可以分为线性链表（*Linenly linked list*）