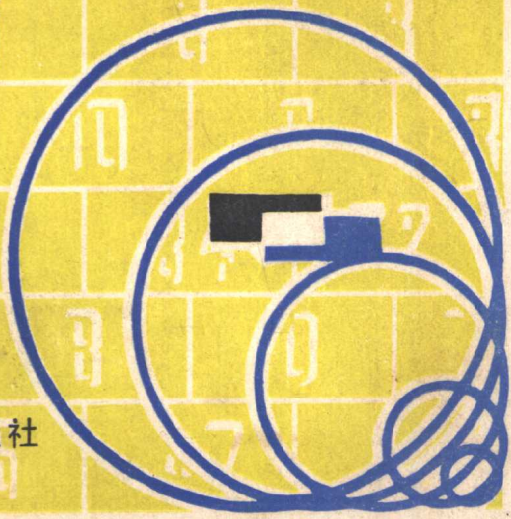
SHU JU JIE GOU CHU BU

数据结构初步

张功镀　吴炜　编著



上海科技教育出版社

数据结构初步

张功镀　吴　炜　编著

上海科技教育出版社

数据结构初步

张功箴　吴炜编著

上海科技教育出版社出版、发行

（上海冠生园路393号）

各地新华书店经销　　上海东方印刷厂印刷

开本 787×1092　1/32　印张 8.25 字数188,000

1987年3月第1版　1987年8月第1次印刷

印数 1—4,400 本

统一书号：7487·53　定价　1.70元

前　言

电子计算机是20世纪科学技术的一项卓越成就。30多年来，计算机应用的发展十分迅速，对整个社会产生了深刻的影响。越来越多的人希望掌握计算机的知识，以便利用它解决日常工作中的一些问题。

在近两年的"计算机热"中，许多非计算机专业的学生、技术人员和教师都学会了BASIC语言，有些地方还将BASIC语言课程下放到中、小学。现在，BASIC语言已成为最普及的计算机算法语言。但是，光学会BASIC语言并能够编写一些简单的程序，是远远不足以解决实际问题的，必须再学点数据结构和程序设计方法的基础知识。目前，有关介绍数据结构的书籍基本上都是为专业人员编写的，强调的是系统的内容，这给非计算机专业人员自学带来相当大困难。对于广大非计算机专业读者来说，他们的学习重点应放在提高解决问题的能力上，没有必要过份追求理论上的严密性。基于这个原因，我们拟编写本书。读者对象是学过BASIC语言，并打算进一步提高应用计算机水平的学生、技术人员、教师和计算技术爱好者。

本书第一部分介绍了线性表、堆栈、队列、链表、散列、索引、网络等数据结构和一些分类、查找的基本方法。考虑到读者的实际情况，本书这一部分的算法说明不象其它数据结构书那样采用类PASCAL或PL/1那样的过程化高级语言，而是采用直观的流程图和BASIC语言。书中对概念不追求严密的定义，而是在不会发生概念错误的前提下尽可能用通俗的语言和例子作

描述和解释。每种数据结构都有现实生活中的实际例子与之对应，以帮助读者理解。每个算法都有适用场合的说明，对算法的质量分析直接给出结果，使读者能对其优劣进行判断，不再进行复杂的推导。

本书第二部分内容是软件设计方法。介绍了软件的系统分析、系统设计、程序设计及调试这几个阶段中流行的方法。用一个具体的应用例子贯穿这一部分，使读者在读完这一部分后对软件设计过程有一个较完整的印象，有利于读者提高解决问题的信心和处理问题的能力。

书中每章的例题和算法都给出了详细的流程图，并采用流行的APPLE-Ⅱ计算机的浮点BASIC语言编写例题程序。为方便读者上机实习，书末附录提供了APPLE-Ⅱ计算机浮点BASIC语言基本语句及DOS3.3操作系统基本命名的简要说明资料。书中每章附有简单事务处理型习题，供读者练习以便掌握每章内容。

本书承张世正副教授审阅。

由于我们水平和实践经验所限，时间匆促，文中疏漏、错误之处恳望读者批评指正。

作　者

一九八六年七月

内 容 提 要

本书是为学过BASIC语言的非计算机专业学生、教师和技术人员编写的介绍数据结构的入门书。书中以浅显通俗的语言和例子介绍常用的数据结构及分类、查找、文件等内容。例题和算法均给出详细流程图并采用流行的APPLE-Ⅱ计算机浮点BASIC语言编写程序。最后一章结合具体实例向读者介绍流行的结构化分析、结构化设计和结构化程序设计的基本概念，旨在提高读者解决问题的信心和处理实际问题的能力。每章末附有习题供练习。附录内容供读者上机实习参考。

目　录

1. 基本数据结构 …… 1
§1.1 图书登记册 —— 线性表结构…… 1
§1.2 Hanoi 塔问题 —— 堆栈结构…… 6
§1.3 药品仓库管理问题 —— 队列结构……14
§1.4 动态地分配存贮空间 —— 链接结构……21
2. 分类与合并 ……34
§2.1 什么是分类……34
§2.2 造学生成绩表 —— 选择分类法……34
§2.3 整理图书卡片的方法 —— 插入分类法……39
§2.4 整理卡片的另一种方法 —— 冒泡分类法……44
§2.5 两种较快的分类方法 —— 折半插入和口袋分类法……49
§2.6 造年级学生成绩名次表 —— 合并……59
3. 查找技术 ……67
§3.1 最简单的查找方法 —— 线性查找……68
§3.2 由折半插入分类法想到的查找方法 —— 折半查找法……69
§3.3 数据的散列组织和查找……70
§3.4 仿照书本目录的数据组织 —— 索引结构……89
4. 二叉树结构 …… 105
§4.1 树结构的概念和实现…… 105
§4.2 分类二叉树的数据操作…… 111

§4.3 遍历二叉树…… 126
§4.4 分类二叉树的重组…… 138
5. 文件组织 …… 145
§5.1 文件的概念…… 145
§5.2 基本文件组织——顺序文件和随机文件…… 145
§5.3 实用文件组织…… 154
§5.4 倒排表文件…… 172
§5.5 网络结构及其实现…… 184
6. 数据结构示例 …… 199
7. 软件设计方法初步 …… 207
§7.1 软件和软件生存周期…… 207
§7.2 系统分析方法…… 209
§7.3 系统设计——结构化设计方法…… 220
§7.4 编程阶段——结构化程序设计方法…… 238
§7.5 软件测试…… 245
附录Ⅰ FP-BASIC 基本语句 …… 249
附录Ⅱ APPLE-Ⅱ DOS3.3 磁盘操作系统命令 … 252
附录Ⅲ 流程图符号说明…… …… 255

1. 基本数据结构

近十几年来，计算机的应用领域日益扩大，它已不限于解决各种数值计算问题，还深入到其它非数值计算的领域，如企业管理，情报检索，文字、图形处理以及人们日常生活的各个方面。要解决的问题越来越复杂，应用程序的规模也越来越大。在这种情况下，单靠程序员的经验和技巧是无法编出一个较好的程序来的。计算机应用的发展要求有规范的科学的程序设计方法。

程序处理的对象是数据。数据之间总是具有这样那样的联系。对数据的结构类型以及这些数据间的联系进行系统的研究，定义规范的处理方法，便是数据结构这门课程的主要内容。这一门课是程序设计的基础课程，也可以说，它标志着从技巧性程序设计到科学性程序设计的转折。

下面我们通过几个实例，来介绍几种基本的数据结构以及有关的处理方法。

§1.1 图书登记册——线性表结构

图书馆的每本图书都有两种编号：一种是分类编号，它是按内容分类编目的；另一种是书目登记号，按图书入馆的先后次序编号的。如果图书馆原有藏书的登记号从1～19583，再购进5本新书，它们的登记号就必须顺序编排下去：

书名	登记号
微处理机应用入门	19584
中学生优秀作文选	19585
科技英语阅读文选	19586
*BASIC*简明教程	19587
中学生*BASIC*教程	19588

这样一来，原来毫无关系的图书之间就存在了一种次序关系。如果按登记号的次序为图书馆中的藏书编一本登记册，那么登记册中各个项之间就有了先后的次序。如19585号书前只有19584号书，而19585号书后要么没有书(如果是最后一本书的话)，要么只有19586号书。归纳起来，如果有$n(n>1)$本书，从1到n为这n本书各自编上一个互不相同的号码，对于第i本书$(1<i<n)$来说，可以明确其前一本书号为$(i-1)$，后一本书编号为$(i+1)$。在第1本书前，没有其它编号的书；在第n本书后，也没有其它编号的书。

上述登记册有两个特点：

1．用编号可以唯一地确定一本书，不会有两本书具有相同的登记号。

2．登记号的大小规定了登记项的先后次序，而且这一次序是唯一的。当然，实际图书编号方法並非如此简单，这里仅作为说明问题的例子。

日常生活中类似的例子是很多的。比如：按准考证号填写的学生花名册；按职工号排列的职工姓名登记表等等。这些表也都具有上述图书登记册的两个特征。我们称这样的结构为“线性表结构”。接下来，可以为线性表下一个具有普遍意义的定义。

定义：假如有有限个结构类型相同的数据，给这些数据编上连续且互不相同的号码，用该号码作为下标，则数据可以表示成

$v_1, v_2, \cdots\cdots v_n$。**根据编号$i(1<i<n)$，可以确定$(v_i-1)$之后必紧跟$v_i$，$v_i$之后必紧跟$(v_i+1)$；$v_1$前没有数据，$v_n$后没有数据，这样的数据结构就叫线性表结构。$i$也称数据$v_i$的下标或地址。**

定义中有一点要特别注意：线性表中的数据必须具有相同的结构，也就是各个项要具有相同的格式，就象上述的各种表格一样。另外，数据间的顺序是根据下标决定的，这一次序不一定与数据的内容有关。

在计算机中，线性表的实现是很容易的。如果线性表中存放的是数字，只要定义一个数组就可以了。如果存放的是字符串，则要定义一个串组。数组或串组中元素的下标相当于各个数据的编号。

下面举例说明如何建立一个线性表以及如何对线性表进行数据的查、增、删、改操作。

例 1.1 假设一个班级中最多有 30 个学生，按照报到的先后次序为每一个学生编一个学号。现要求编一个程序，做下面几件工作：

1．学生前来报到时对学生名单进行补充。

2．发现名单中的错误时应能进行修改。

3．某个学生调班或留级后，从名单中删去这一学生的名字，同时修改他后面学生的学号。

4．给出一个学生的名字，查该学生的学号。

问题分析：

实现本题要求的关键是如何组织学生姓名和学生学号这些数据。可以考虑按学生学号的次序来安排学生姓名。将学生姓名放在一张线性表中，姓名的下标就是学号。这样组织的数据很直观，而且不必存学号，可以节省存贮空间。流程图见图1.1-1。程序如下：

```
90  REM……LEXP1.1……
100 DIM NA$(30)
110 PO=0
120 HOME
130 VTAB 2: HTAB 10: PRINT "1. INPUT  STUDENT'S
    DATA" (输入学生数据)
140 VTAB 4: HTAB 10: PRINT "2.SEARCH"(查询)
150 VTAB 6: HTAB 10: PRINT "3. MODIFY STUDENT'S
    DATA" (修改学生数据)
160 VTAB 8: HTAB 10: PRINT "4. DELETE STUDENT'S
    DATA"(删除学生数据)
170 VTAB 10: HTAB 4: INPUT "WHICH FUNCTION DO
    WANT? ";AX(请选择)
180 IF AX <1 OR AX> 4 THEN 170
190 ON AX GOTO 200,260,330,410
200 REM……INPUT……(输入)
210 HOME
220 IF PO> 30 THEN PRINT "STUDENTS TOO MORE": FOR
    I=1 TO 1000: NEXT I: GOTO 120(学生数太多)
230 PO=PO+1: PRINT "STUDENT'S NO.=";PO(学号)
240 INPUT "STUDENT'S NAME:"; NA$(PO)(姓名)
250 GOTO 120
260 REM……SEARCH……(查找)
270 HOME
280 INPUT "STUDENT'S NAME=";NA$(学生姓名)
290 P=1
300 IF P> PO THEN  PRINT "STUDENT";  NA$;  "DO  NOT
    FOUND": FOR I=1 TO 1000: NEXT I:GOTO 120 (没找到该
    学生)
310 IF MA$(P) <  > NA$ THEN P=P+1:GOTO 300
320 PRINT NA$; "'S NUMBER="; P: GET A$: GOTO 120 (…"的
    学号=")
330 REM……MODIFY……(修改)
340 HOME
350 INPUT "STUDENT'S NO.:";P (学号)
360 PRINT "PRESENT NAME="; NA$(P)(当前姓名)
370 IF P <1 OR P> PO THEN 350
```

```
380  INPUT "STUDENT'S NAME:";NA$(学生姓名)
390  NA$(P)=NA$
400  GOTO 120
410  REM……DELETE……(删除)
420  HOME
430  INPUT "STUDENT'S NO.:";P(学号)
440  IF P <1 OR P> PO THEN 430
450  IF P=PO THEN 470
460  FOR I =P TO PO-1:NA$(I)=NA$(I+1):NEXT I
470  PO=PO-1
480  GOTO 120
```

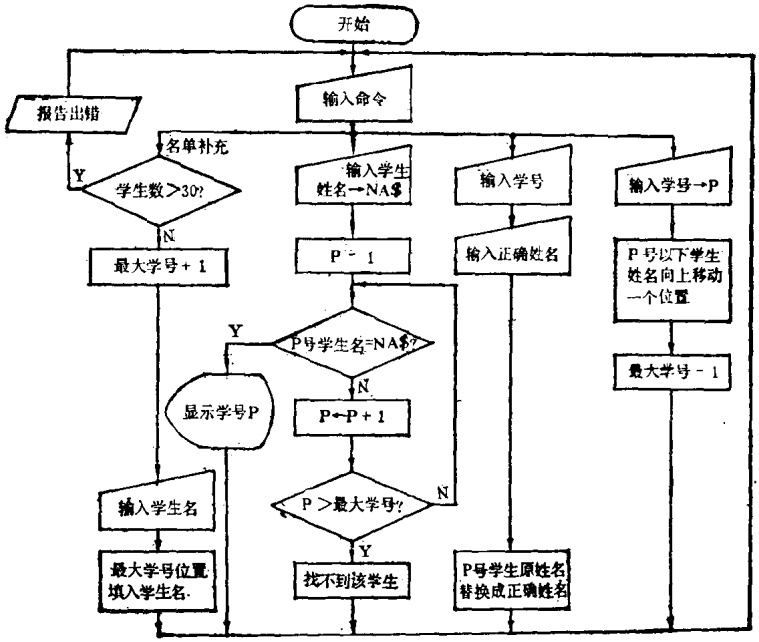


图 1.1—1 学生学号管理程序流程图

---

* 上述程序中括号内的中文字均为编者所加，并非程序所有，主要为了便于读者理解程序。以后所有程序都作了同样处理。另外，本书程序中凡出现"PRINT"语句为响铃语句。引号内容 CTRL-G，在屏幕上不能显示。

变量说明：

NA$( )：用以存放学生姓名的串组；

PO：最大学号；

P 和 NA$：辅助变量，分别用以暂时存放学号和学生姓名。

线性表数据的增、删、改操作没有很严格的规定。往线性表中增加一个数据时可以根据需要将数据插在线性表的最前或最后。可以对线性表的任一个数据直接进行修改（用新的数据替换原有的数据）。同样，也可以将线性表的任一个数据从表中删去，但要保证在该数据被删去后，其它数据依然能构成一张线性表。例 1.1 中删除一个学生姓名后要移动后面的姓名数据，其目的就是要使余下的学生姓名能构成一张线性表。

线性表结构是最简单、最自由的数据结构。其实，大家以前编程序时为存贮数据而定义了数组或串组，都是无意识地应用了线性表。以后要介绍的另外几种数据结构都是在线性表的基础上加以一定的限制后得来的。

§1.2 Hanoi塔问题——堆栈结构

Hanoi 塔问题是这样的：设有如图 1.2-1 所示几个大小不同的圆盘和三根立柱，圆盘依大小递减的顺序插在立柱 a 上，b 和 c 立柱是空的。要求把这几个圆盘以大小递减的顺序搬到 c 立

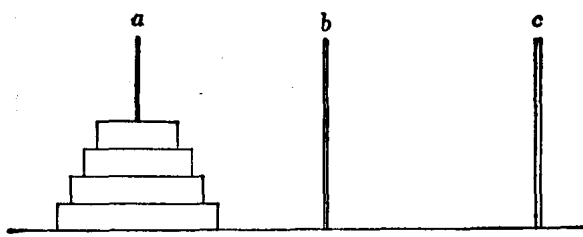


图 1.2-1 Hanoi 塔问题

柱上，仍叠成如 a 原来的塔形。搬动盘子时可借用 b 立柱暂放圆盘。搬动过程要遵守下列规则：

1．每次只能搬动一个圆盘；

2．任何时候大盘不能压在小盘上；

3．圆盘只能在 a，b，c 三根立柱中作任意搬动。

如果圆盘只有一个(即 $n=1$)，那么，只要把圆盘从 a 搬到 c。如果 $n=2$，就要先将 a 顶上第一个圆盘搬到 b；第二个圆盘搬到 c；再把 b 上的圆盘搬到 c。如果 $n=3$，先要把 a 顶上第一个圆盘搬到 c；再把 a 的第二个圆盘搬到 b；再把 c 上第一个圆盘搬到 b；第三个圆盘从 a 搬到 c；b 上第一个圆盘从 b 搬到 a；第二个圆盘从 b 搬到 c；最后把第一个圆盘从 a 搬到 c，从而完成三个圆盘的搬动。依此类推，对于 n 个圆盘，要把它们从 a 搬到 c，总是按下面的办法进行的：

1．想法按大小递减顺序把 $(n-1)$ 个圆盘从 a 搬到 b；

2．把第 n 个圆盘从 a 搬到 c；

3．再想法按大小递减顺序把 $n-1$ 个圆盘从 b 搬到 c。

如果要用计算机来解决 *Hanoi* 塔问题，很容易地就会想到定义三个数组(三张可容纳 n 个元素的线性表)，模仿三根立柱上圆盘的排列情况。圆盘从一根立柱移到另一根立柱就相当于从一个线性表顶部删除一个数据，再将该数据插入另一张线性表顶部。线性表中数据的删除和插入位置不能随意，要删除的数据只能是线性表中目前顶上的第一个数据（相当于从塔顶移走一个圆盘）· 要插入的数据也只能是在线性表的顶上(相当于将一只圆盘放在塔顶)。这样，*Hanoi* 塔问题的解法就成了按上述搬动圆盘的规则对三个线性表的顶部进行多次的数据增、删了。

对线性表的数据增、删作了上述限制后，就得到一个新的数

据结构，称为“堆栈结构”。

“堆栈”这一名称来源于日常生活。例如有一堆货箱，新到货箱只能放在货堆的顶部，而提货时只能从上到下一箱箱地将货箱取走。又如桌子上有一叠书，往这叠书上放书时，必然依次放在最上面，而取书时，则只能一本一本从上向下取。

堆栈有这样一个特点：最先进入堆栈的数据要最后才能取出，而最后进入堆栈的数据则要最先取出。如果要取堆栈中间的数据，则要将在该数据之后进入堆栈的数据都取出来，才能得到所要的数据。这一特点称为“先进后出”(*FILO*)原则。

可以给堆栈下这样的**定义：若规定线性表的数据只能在线性表的一端进行插入和删除，则称这样的数据结构为堆栈结构。**

堆栈中能够进行数据增、删操作的一端称“栈顶”，另一端称“栈底”。

在计算机中实现堆栈结构，首先要根据需要定义一个数组或串组，用以实现一张线性表。其次是编写堆栈的数据操作程序。在编写数据操作程序时还要用到一个称作“栈顶指针”的变量。这个变量的值是堆栈中最后一个进栈数据的下标，形象地说指针指向堆栈的栈顶单元。存在栈顶的数据（即最后一个进栈的数据）叫做“栈顶数据”。相应地，栈中第一个进入堆栈的数据叫“栈底数据”，其所处位置也就是栈底。堆栈中增、删数据的操作分别称为进、出栈操作。图 1.2-2 是进、出栈操作的流程图。

要注意的是，出栈操作实际上并没有真正去做清除数据的操作，而只是移动一下栈顶指针。堆栈内的数据是指在栈顶和栈底之间的数据，在此范围之外的数据都不属于堆栈。这样规定可以简化处理程序。在实际程序中，由于数(串)组定维时的限制，栈顶指针的取值范围也要受到限制，若超过了所规定的范

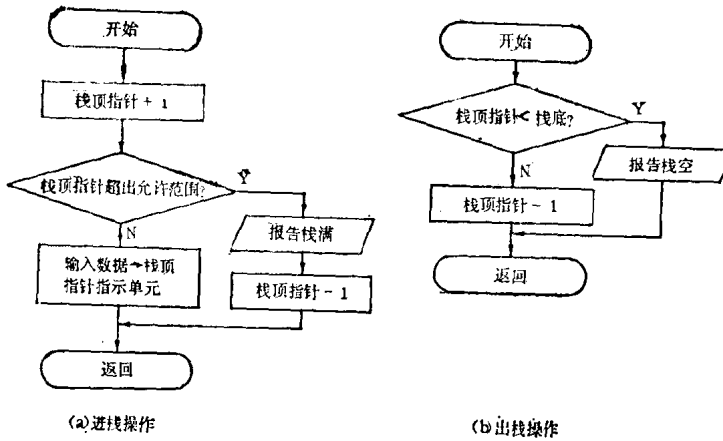


图 1.2-2 堆栈进出栈操作流程图

围就会出现越界错误。例如，为存贮数据而定义一个数组$DIM A(50)$，假设栈底为 $A(1)$，则栈顶指针不能超过50。当栈顶指针小于栈底下标时表示栈中没有数据。当栈底为$A(1)$时，栈顶指针的初值应为 0（程序开头对栈顶指针赋初值）。初学者在编写程序时经常忽略了变量的初值和取值范围限制，要特别注意。

在图1.2-2的流程图中，假定数(串)组中单元的分配是按下标从小到大进行的。当然也可以按下标从大到小的顺序分配数(串)组单元，这时栈顶指针的初值应赋为最大下标加1，指针的变化方向，判定条件都要与图 1.2-2 的流程图相反，请读者自己画。以后的章节中若无特别说明，都假定按下标从小到大分配数(串)组单元。

前面还没有提到堆栈中数据的修改和使用问题。严格地讲，数据的修改和使用只能在栈顶进行，但实际上往往不这么做。为了提高效率，在不会引起混乱的前提下，可以将堆栈看成