# 电子计算机档案检索

（下　　册）

邱　晓　威　编著

兵器部档案馆编印

一九八五年七月

# 第六章　档案检索系统程序设计

## §6—1　程序设计基本知识

### 一、程序和程序设计

给计算机下达的完成一项专门任务的指令表叫做程序。电子计算机之所以能自动的完成某一任务，就是由于人们为它编制了程序。根据“图灵自动机”原理，计算机（目前的数字电子计算机）的自动处理必须满足两个条件：(1)将所要解决的问题和结果符号化；(2)设计一套指令，使得计算机在执行过程中无需任何想象力。这两条，正是人们编制程序的原因和宗旨。从问题及结果的符号化直至获得计算机可以执行的一系列指令，就是程序设计的过程。

### 二、程序设计基本知识

1.程序设计符号。见图6—1。

2.程序设计步骤

一般来说，程序设计的步骤是(1)根据解决问题的算法绘制流程图；(2)使用程序设计语言描述流程图，成为计算机程序。假设某同学想建立一个全班同学的通讯地址录。他的做法可能如下：(1)打开小本，拔开笔帽；(2)询问学号为1的同学，并登录其姓名、地址、电话号码等信息；(3)学号＋1；(4)判断得数是否已超过全班总人数；(5)如超过，则可以合上小本，插上笔帽，通讯录建立完毕，如果尚未超过，进行

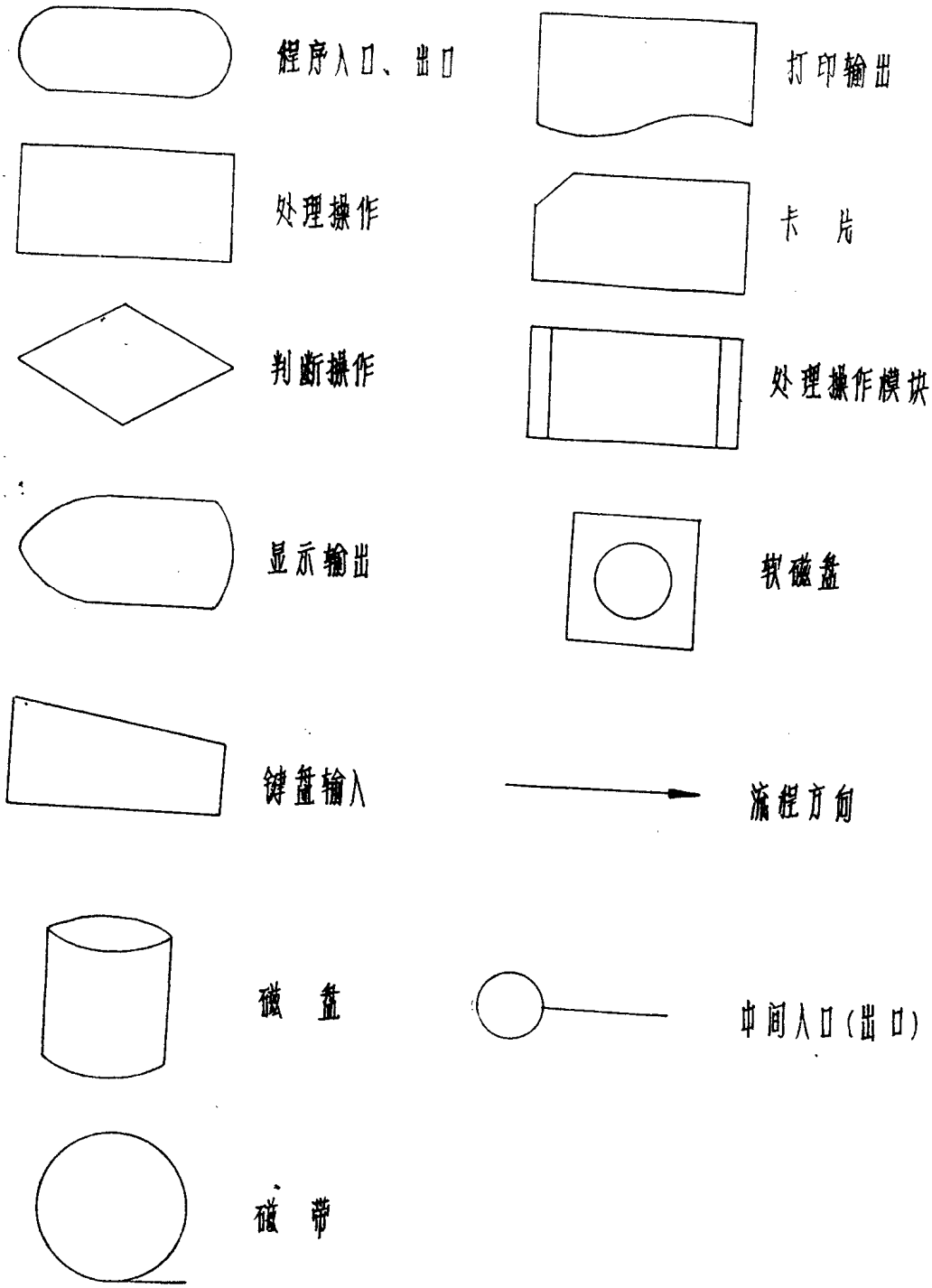

程序入口、出口
打印输出
处理操作
卡 片
判断操作
处理操作模块
显示输出
软磁盘
键盘输入
流程方向
磁 盘
中间入口(出口)
磁 带

图 6 — 1

下一步；(6)按所得学号数询问另一位同学，并登录信息；(7)重复第(3)步，直至通讯录建立完毕。

以上过程用程序符号表示出来，就构成了一个逻辑框图，或叫做流程框图。见图 6 — 2。

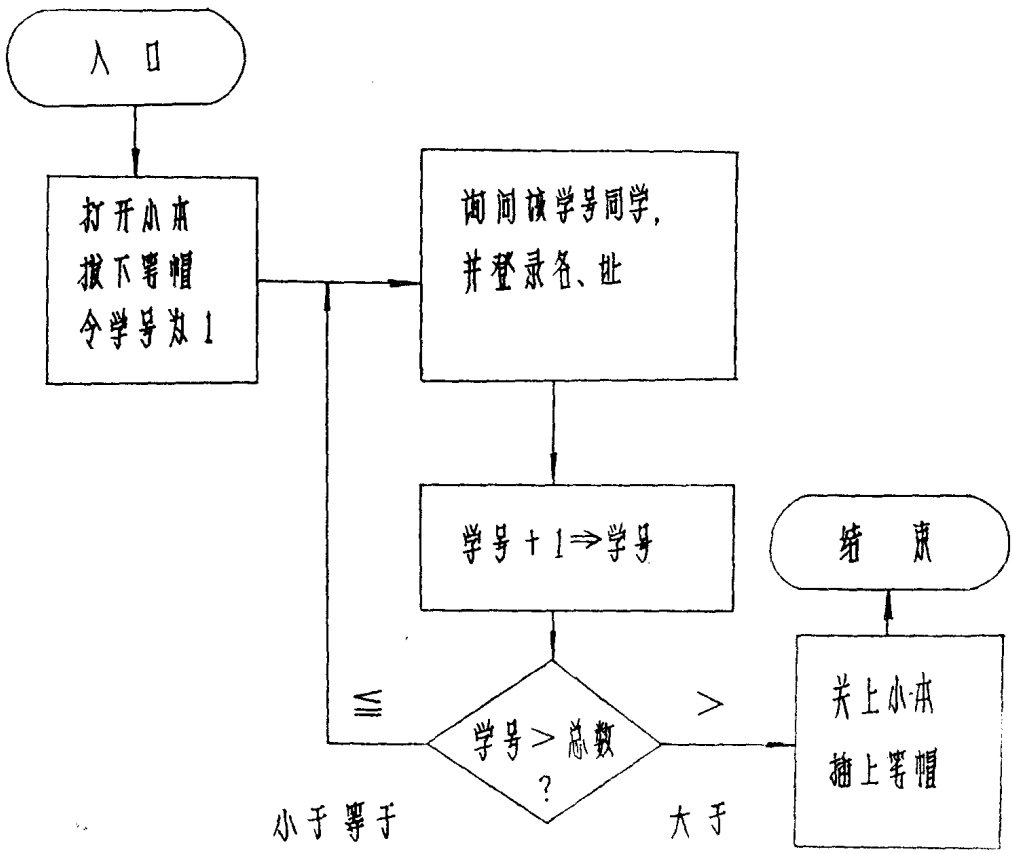



图 6 — 2

使用某一种计算机语言将上述过程描述出来，就可以得到一个计算机程序。下面给出一个使用 B A S I C 语言编写的建立通讯录的程

序：

```
10  OPEN"NAME, ADDRESS"
                              (打开名址录文档)
    FOR OUTPUT AS#1
20  XH=1                      (学号初值为1)
30  INPUT"NAME"; N
                              (询问并输入姓名)
40  INPUT"ADDRESS"; A
                              (询问并输入地址)
50  PRINT#1, N; ","; A
                              (名址写入文档)
60  XH=XH+1                   (学号+1)
70  XH>40 THEN80 ELSE30
                              (学号是否超过40)
80  CLOSE#1                   (关闭文档)
99  END                       (结束)
```

第10句是打开名址录文档；第30，40句为询问姓名地址；第50句是将名址写入文档；80句是关闭文档。

三、程序设计语言

前面给出的BASAC语言编写的程序是使用计算机语言的一个

具体例子。实际上可以编写程序的语言有很多种。一般来讲较低级的语言与自然语言的差别大一些，但是计算机执行低级语言编的程序效率高。而较高级的语言比较接近自然语言，便于人们掌握，但是计算机执行起来效率要低一些。所谓汇编语言，就是比较低级的语言，它是用一些符号来代表机器指令。使用汇编语言编写程序很容易出错，而且对于已编好的程序又不容易看懂，所以不利于交流和推广。一般情况下，在编写应用程序时，程序员多采用高级语言。采用高级语言编写的程序稍加改动可以方便地在不同的计算机系统上运行。

高级语言有很多种。下面列出用最通用的高级语言增加B值语句的例子：

| | |
|---|---|
| ALGOL 60 | 算法语言60 |
| | B：＝B＋1； |
| ALGOL 68 | 算法语言68 |
| | B *plus*ab 1； |
| BASIC | 初学者通用符号指令代码， |
| | LET B＝B＋1 |
| COBOL | 面向商业的通用语言 |
| | ADD I TO B |
| FORTRAN | 公式翻译程序语言 |
| | B＝B＋1 |

PL／1　　程序设计语言1

B＝B＋1

以上列出的语言中BASIC最易掌握，而且易于实现人机对话，广泛使用在微型机上。ALGOL 60和FORTRAN两种语言最初是为科学工作设计的。COBOL是为商业应用设计的，它在事务和文献信息处理系统中最常用。PL／1是一种通用的语言，它在文献信息处理系统中的应用也很广泛。

要想得到一个高质量的应用程序，选择合适的语言非常重要。对于一个称职的程序设计人员来说，应该能做到需要采用什么语言，就选择什么语言，而不应该熟悉什么语言选择什么语言。

四、程序（算法框图）的验证

程序设计完毕后必须经过验证，判定其是否正确，并经过修改和调试才能保证程序在使用过程中不出问题，或少出问题。

(1)程序（算法框图）的验证方式：

a、可以直接使用电子计算机进行验证。步骤是，先将用某种“语言”编写的源程序输入电子计算机，然后由计算机进行语法检查，语法检查完成后，就得到了目标程序，最后试运行目标程序并输入数据来验证逻辑或算法本身是否正确。

b、采用手工模拟计算机运行程序。这种方式主要是验证算法的

正确性。算法框图或源程序都可以采用手工模拟进行验证。

(2)验证程序时，选择实例的要求：

$a$、实例要全，要能包括各种可能出现的输入信息及其组合。

$b$、被选择的例子，在程序中运行时，要能走遍各个分支。

这两条选择实例的要求一般很难完全做到，尤其是对于一些大型程序，只能尽量保证。所以既使是经过严格验证的程序，在使用中还需针对出现的问题进行修改和维护。

五、算法的优化

一个可运行使用的程序，并不一定是一个很好的程序。所以设计完一个程序后，需要经过优化的工作，才能得到较完美的程序。

(1)优化的目的是，提高程序运行效率，减少资源消耗、扩充原有功能、提高可靠性。

(2)优化途径是，算法的改进、程序的简化等等。

例如，下一节中介绍的排队算法，是一种循环排队的算法，它就是从采用线性排队算法优化而来的，而运行效率却比后者大大提高。

§ 6—2 常用数据处理技术介绍

前面我们曾介绍了数据结构的基本知识。重点介绍了信息（或数据）的存贮结构，但是计算机信息处理系统的关键问题不是存储信息，而是能够对这些信息加工处理，并以适当的方式传递和使用。比如档

案工作者最关心的问题是，我们费了很大力量制作和输入给计算机大量的档案目录信息之后，能得到些什么。这节中将对信息的基本处理技术做一些介绍。这些知识可以使我们了解计算机是怎样对信息加工处理的。

一、堆栈和排队

日常生活中经常遇到的这样两种“纪律规则”与数据处理技术中的含义是非常一致的。

买东西时，先到的先买，后到的排在后面，并且后买。这是“排队”的规则，在排队中“加塞儿”是违反规则的。

堆栈的例子也很常见。比如一叠书在桌子上，我们常常最先取用最上面的那本书，而最上面的那本书恰恰是最后放上去的。冲锋枪的弹夹也是“堆栈”，最后压入的子弹最先发射出去。

图6--3给出了“堆栈”的图示。图6—4给出了排队的图示。

图6—3

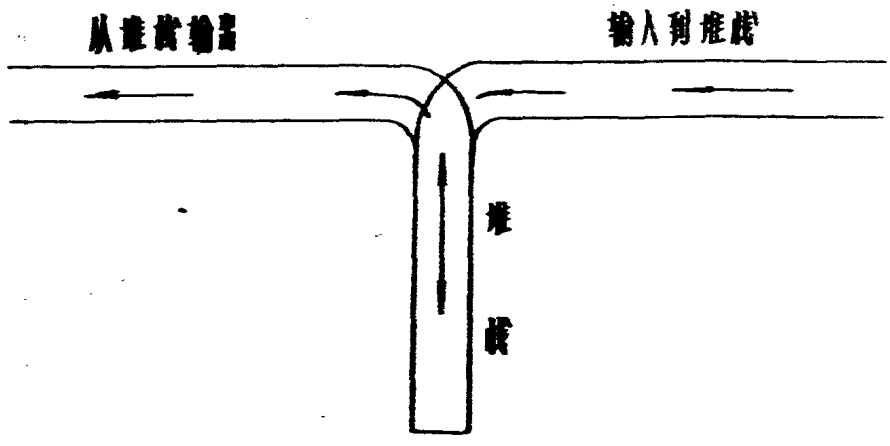


图 6－4

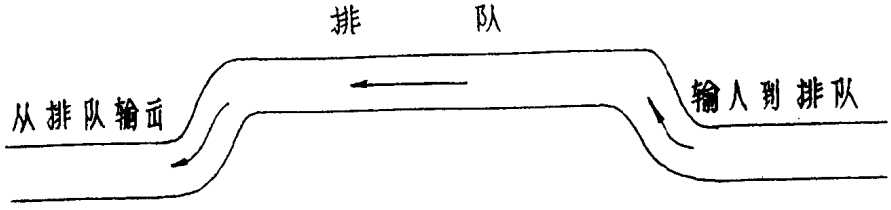



图 6—5 为信息进栈和出栈的算法框图。图中顺序结构的栈是它的各信息单元的结点由一组连续的内存单元N（ P ）组成。 P为栈指示器，其最大值为栈的总容量$n$。当栈内信息为“空”时， P为O，随着信息单元的装入， P值增加，并且始终指示最后装入的信息单元。

因为栈是动态数据结构，所以在进行进栈和出栈操作时，要注意栈的“满”与“空”两种情况的发生。栈“满”时不能执行进栈操作，栈“空”时不能进行出栈操作。否则将发生“上溢”或下溢”，这有时也称为栈的边界条件。

图 6—6 为信息在排队中的进排和出排的算法框图。图中所示的顺序存储结构的排队是由一组连续的内存单元N（ P ）组成的。P=?, $n$为最大排队空间。因为排队是在结构的两端分别进行进排和出排操作，所以，与栈不同，它需要两个指示器。一个是排首指针F，用来指当前排首（排队中最前面的信息结点）的位置。一个是排尾指针R，　用来指示当前排尾（排队中最后一个信息结点的下一结点）的位置。

当排队内没有任何信息时，排队为空。此时（P）＝*1*，（R）＝0，即排队的初始状态。每进入排队的队尾一个信息，排尾向后延伸一个，指针R的内容加1。从排队的队首每出去一个信息，排首向后移动一个结点位置，也就是指针F的内容加*1*。

这样，排队的排首和排尾处于不断变化之中。当排尾到n时，如果再进队一个信息；排尾指针的内容不能再加*1*，此时如果队首尚有空余结点，则允许指针（R）＝*1*，这种现象叫做排队的翻转。排首指针也有类似情况。当（F）＝n时，如果这时排首的节点要出排的话，排首指针F的内容由n变为*1*。

当（F）＝（R）时，排队的存储空间已经占满。

排队空时（（R）＝0），不能进行出排操作；排队满时（R）＝（F），不能进行进排操作。

由框图可以看出，进排时，首先检查排队是否已满。如果R＝F，表示排队已满，信息不能进排。这时，不执行进排操作而返回，如果排队未满，就进一步检查排队是否空。如果R＝O，则说明排队空。这时，将F值赋予R。然后将A中的信息放入排队的N（R）单元。信息进排以后，要检查一下排尾是否到达排队的最后存储结点（即表尾）。如果R＝*n*，表明排尾已到达排队的最后结点空间，于是排队执行翻转，即令R＝1。否则，只需将R值加1。最后返回。

信息出排时，首先要查看排队是否空，因为不能从空的排队取走

信息。排队空的标志是R＝0。如果排队空，则将返回。否则，按F指示的位置取出信息。信息出排后，接着要做的工作是修改指针F D的内容。修改F要根据两种情况分别进行。一种情况是队首就是排队的最后一个结点，即F＝n。此时要进行排队的翻转，也就是令F＝*1*。一般情况下，只需把F内容加*1*。最后检查一下信息出队后，排队是否变成空队。如果不是空队，就返回主程序。如果是空队，则要把指针R置0，F置*1*，然后返回。

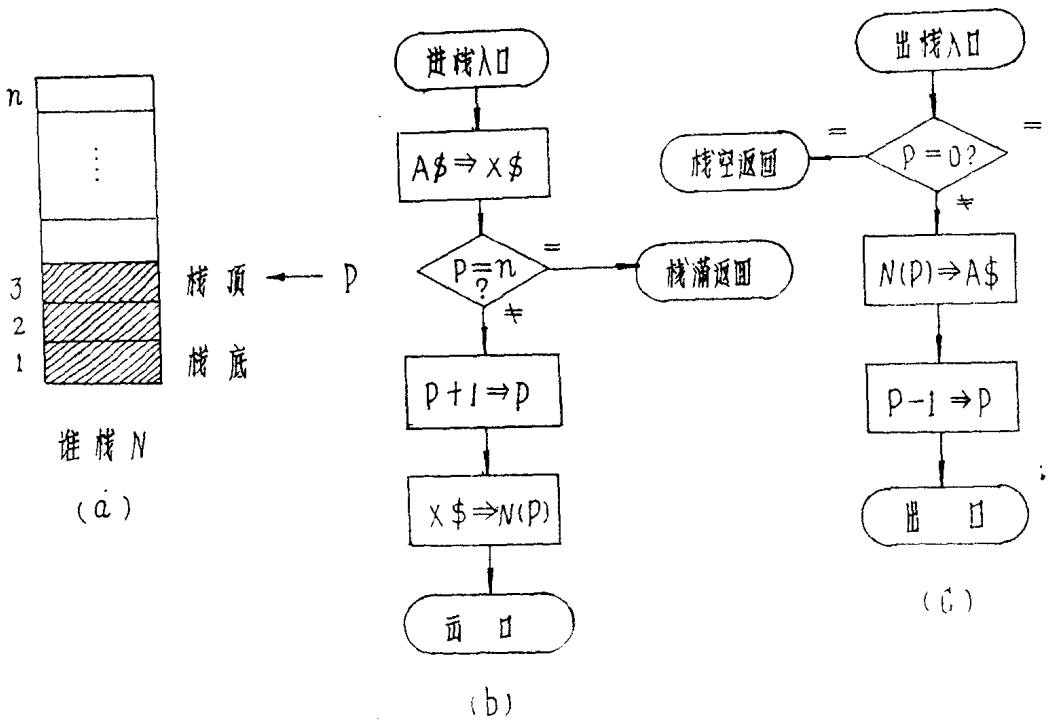
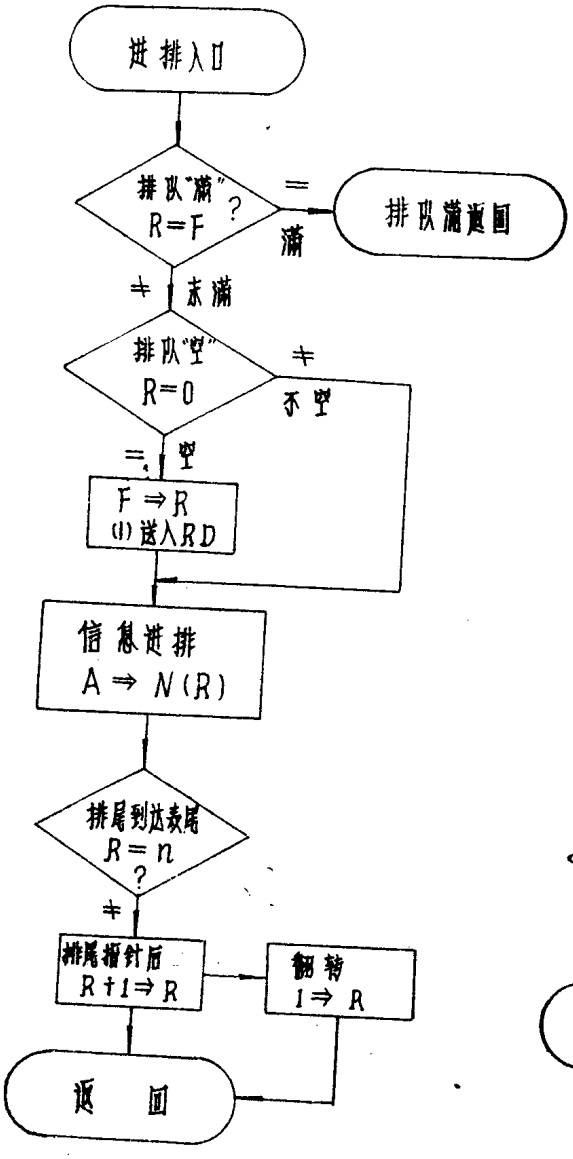
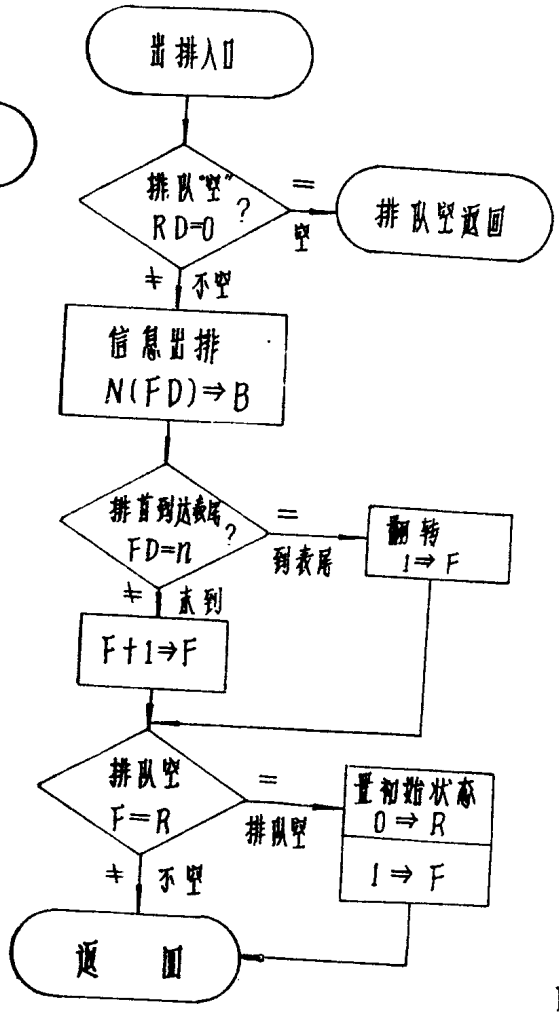


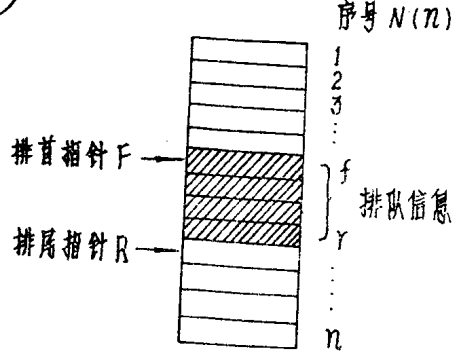

图 6 － 5


进 排入口
排队"满"?
R=F
=
满
排队满返回
≠
未满
排队"空"
R=0
≠
不空
=
空
F⇒R
(1)送入RD
信息进排
A⇒N(R)
排尾到达表尾
R=n
?
≠
排尾指针后
R+1⇒R
翻转
1⇒R
返 回

(a)

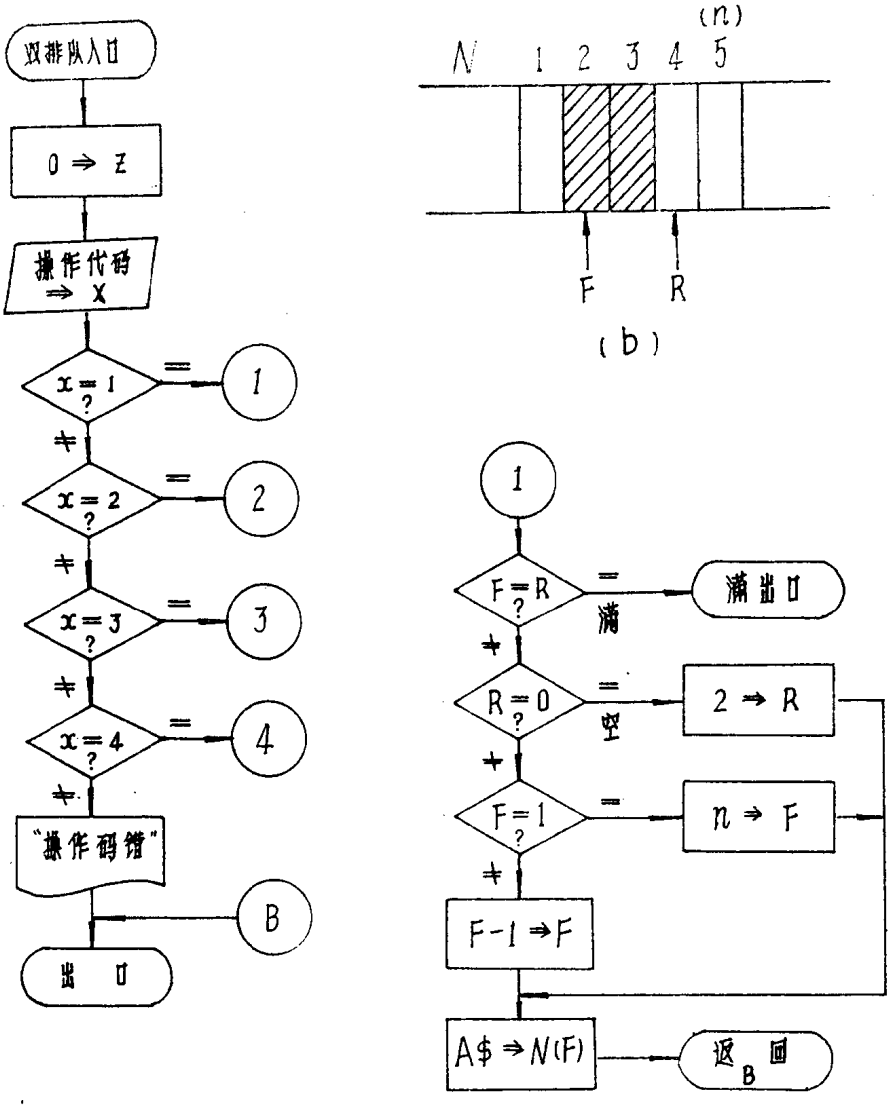

出排入口
排队"空"
RD=0
?
=
空
排队空返回
≠
不空
信息出排
N(FD)⇒B
排首到达表尾
FD=n
?
=
到表尾
翻转
1⇒F
≠
未到
F+1⇒F
排队空
F=R
=
排队空
置初始状态
0⇒R
1⇒F
≠
不空
返 回

(b)

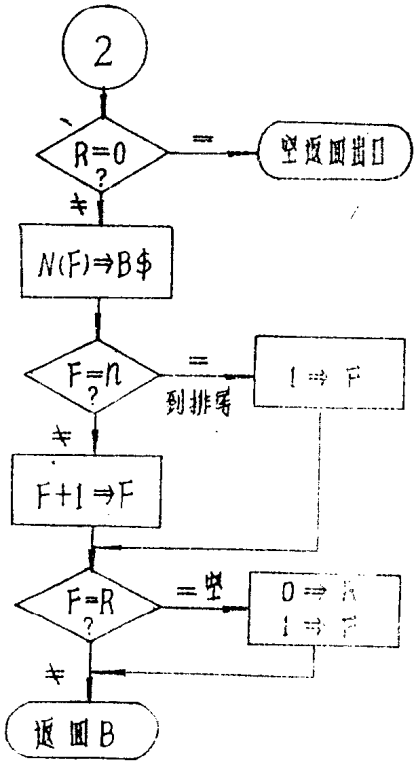

序号 N(n)
1
2
3
排首指针 F
f
排队信息
r
排尾指针 R
n

(C)

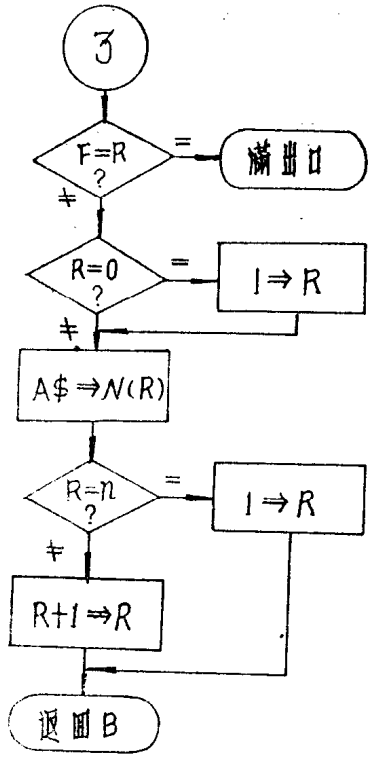
图 6 — 6


双排队入口
0 ⇒ Z
操作代码 ⇒ X
X=1 ?
=
1
≠
X=2 ?
=
2
≠
X=3 ?
=
3
≠
X=4 ?
=
4
≠
"操作码错"
B
出 口
(n)
N 1 2 3 4 5
F
R
(b)
1
F=R ?
=
满
满出口
≠
R=0 ?
=
空
2 ⇒ R
≠
F=1 ?
=
n ⇒ F
≠
F-1 ⇒F
A$ ⇒N(F)
返回
B

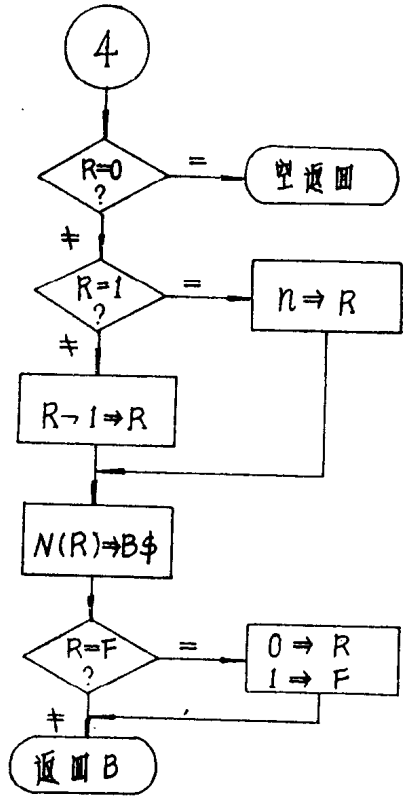
(a)　　图 6－8 a b c　　(C)


2
R=0
?
=
空返回出口
≠
N(F)⇒B$
F=n
?
=
到排号
1⇒F
≠
F+1⇒F
F=R
?
=空
0⇒R
1⇒F
≠
返回B

(d)


3
F=R
?
=
满出口
≠
R=0
?
=
1⇒R
≠
A$⇒N(R)
R=n
?
=
1⇒R
≠
R+1⇒R
返回B

(e)

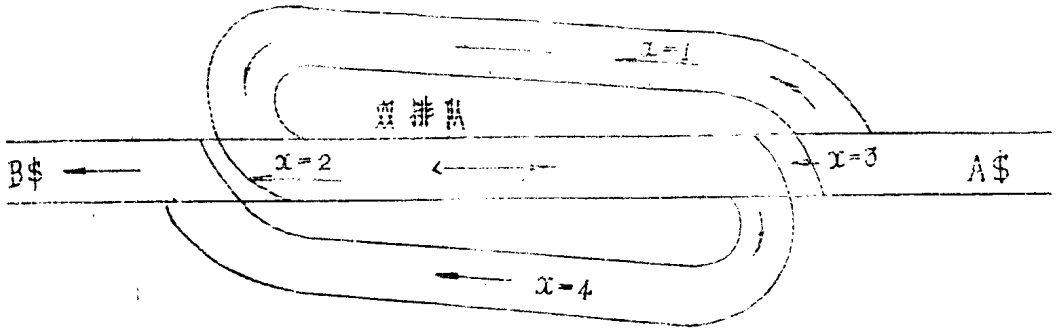

4
R=0
?
=
空返回
≠
R=1
?
=
n⇒R
≠
R-1⇒R
N(R)⇒B$
R=F
?
=
0⇒R
1⇒F
≠
返回B

(f)

图 6 — 8 d e f

还有一种双排队。它将堆栈和排队结合在一起，使用起来更加灵活。图6－7给出了双排队的图示。

图6－7




在程序运行过程中，按照语言的先后顺序执行各语句的功能，就是一种排队。当程序运行到某句需要转子程序时，必须记下该句行号，以便返回时可继续运行主程序，如果在运行子程序时就又一次转子程序，就应当以堆栈的形式依次记录每个转子程序语句行号，才可以逐步返回主程序。

图6－8所示是双排队算法框图。算法要点是：①设立一个首指针F，指向排首信息存贮单元；②设立一个尾指针R，指向排尾的第一个单元；③所有信息存贮单元顺序排列，并首尾相接，构成一个循环链表。④初始（空）时的指针值为：R＝O，F＝1。⑤当R＝F时双排队满；⑥当R－1＝O时，或者F＋1＝$n$时指针翻转。⑦出排后若R＝F，说明双排队空，则重新赋予初值：R＝O,F＝1。

## 二、紧排顺序存贮结构

第五章中图5－23所示的磁带上的顺序文档就是所谓的紧排顺

序数据存贮结构。

在紧排顺序存贮结构中，数据信息所构成的最小逻辑单元称为一个记录。在这种结构中记录的长短随着所含信息数据的多少而定，是一种可变长记录。图6—9所示为一种存贮于磁带上的紧排顺序文档的结构图。图中所示每个记录的第0个字符为记录标识符“*”；第1—3位字符合起来表示记录号；第4—7位字符合起来表示本记录长，第8位字符表示记录状态（为“N”表示有效记录，为“C”表示本记录已被删除）；“$”为文档结束符。

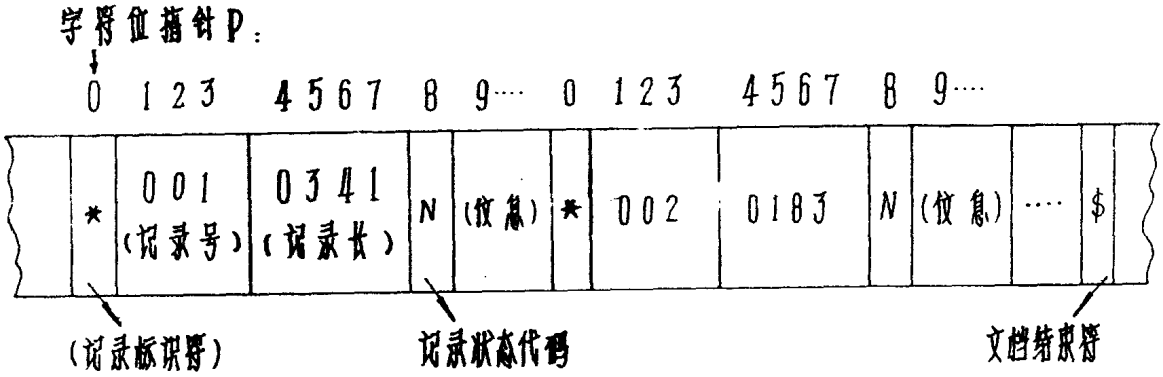



图6—9

1. 紧排顺序文档的建立

建立紧排顺序文档的算法思想是：

①先在内存中开辟一个记录存贮区M（P）。

②M（0～8）的字符位为固定长的记录说明项：第0位为记录标识符“*”；第1～3位为记录号；第4～7为记录长；第8位为记录状