第6章 汉字输入技术(一)

主要内容

- 汉字输入技术的发展
- 汉字键盘输入技术
- 汉字键盘输入系统
- 汉字键盘智能输入技术

6.1 汉字输入技术的发展

- “一字一键”输入技术
- 键盘编码输入技术
- 语音输入技术
- 手写输入技术

6.1.1 大键盘方案

- 设计一个汉字大键盘，这个汉字大键盘上有几千个按键，每个按键对应一个汉字，汉字按照偏旁部首在键盘上分区排列，以便于快速检字。
- 大键盘的思想经过实践证明是失败的，它存在的问题主要是在键盘上找字困难，不能输入键盘上没有的汉字，设备本身价格太高等。

6.1.2 汉字编码输入

- 基于西文键盘的汉字编码输入是到目前为止最成功的计算机汉字输入方法；
- 汉字编码输入是最早的汉字输入方法，最早的计算机汉字编码输入始于50年代的俄汉机器翻译研究，那时采用电报码或四角号码做汉字编码；
- 1978年5月上海电工仪器研究所的工程师支秉彝创造了一种“见字识码”法，并被上海市内电话局采用，从而率先使计算机的汉字输入进入了实用阶段。

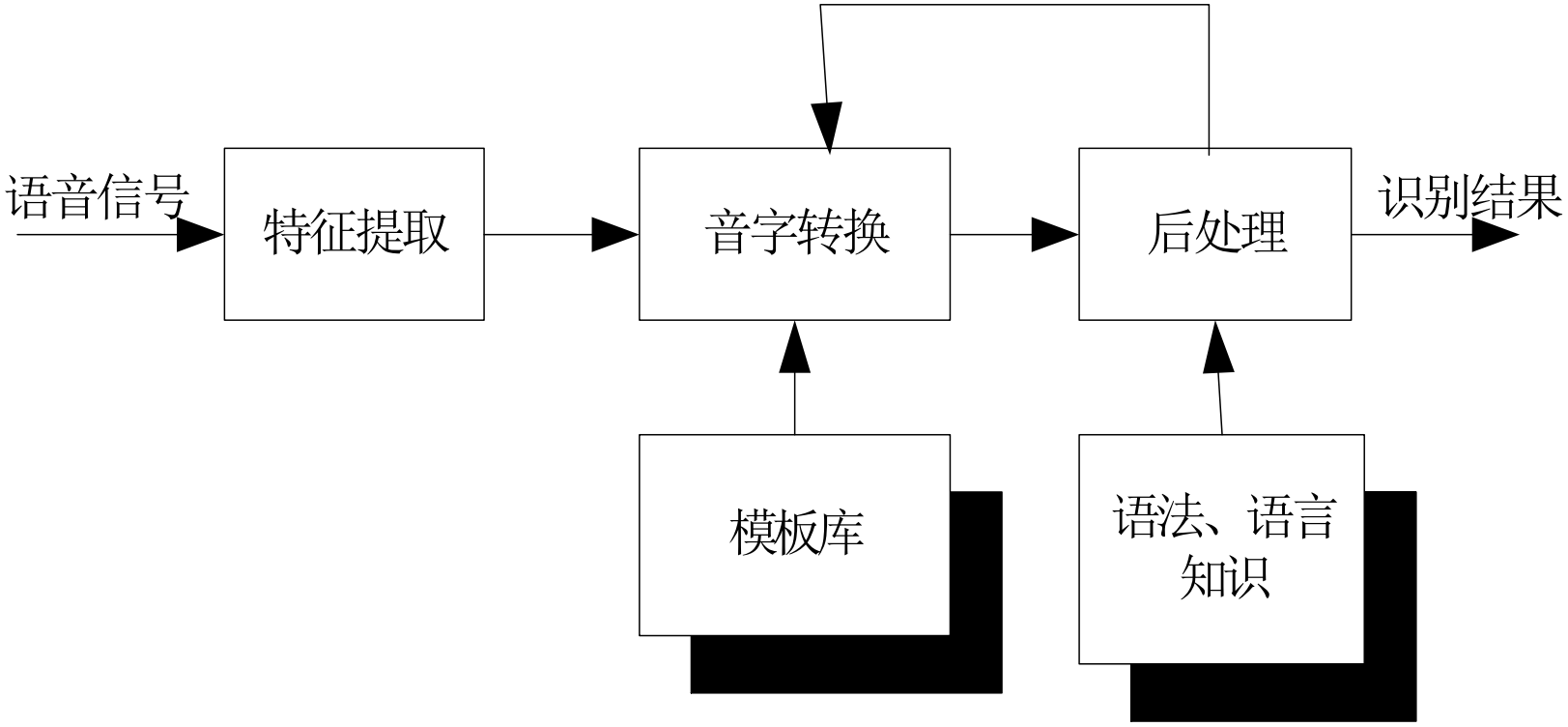
- 在香港、台湾及海外华人界，以“王安三角编码”为开端，以后陆续涌现出了“仓颉”、“呒虾米”、“行列”、“唯物”等一大批针对繁体汉字的编码输入法。
- 到了80年代，在联想汉卡、四通中文电脑打字机面市之后，中国大陆的汉字编码出现了“万码奔腾”的局面，先后涌现出了五笔字形、大众码、自然码、郑码、钱码、电拼文字、宏观码、表形码、纵横码、智能ABC、紫光拼音等上千种汉字输入法。

6.1.3 语音输入

- 语音输入是最自然的一种人机交互技术。
- 上世纪90年代，各大厂商包括IBM、微软、中自等公司纷纷推出自己的非特定人语音输入系统。
- 美国IBM公司于1997年9月在北京推出中文非特定人连续语音识别系统ViaVoice。

- Siri –语音助理

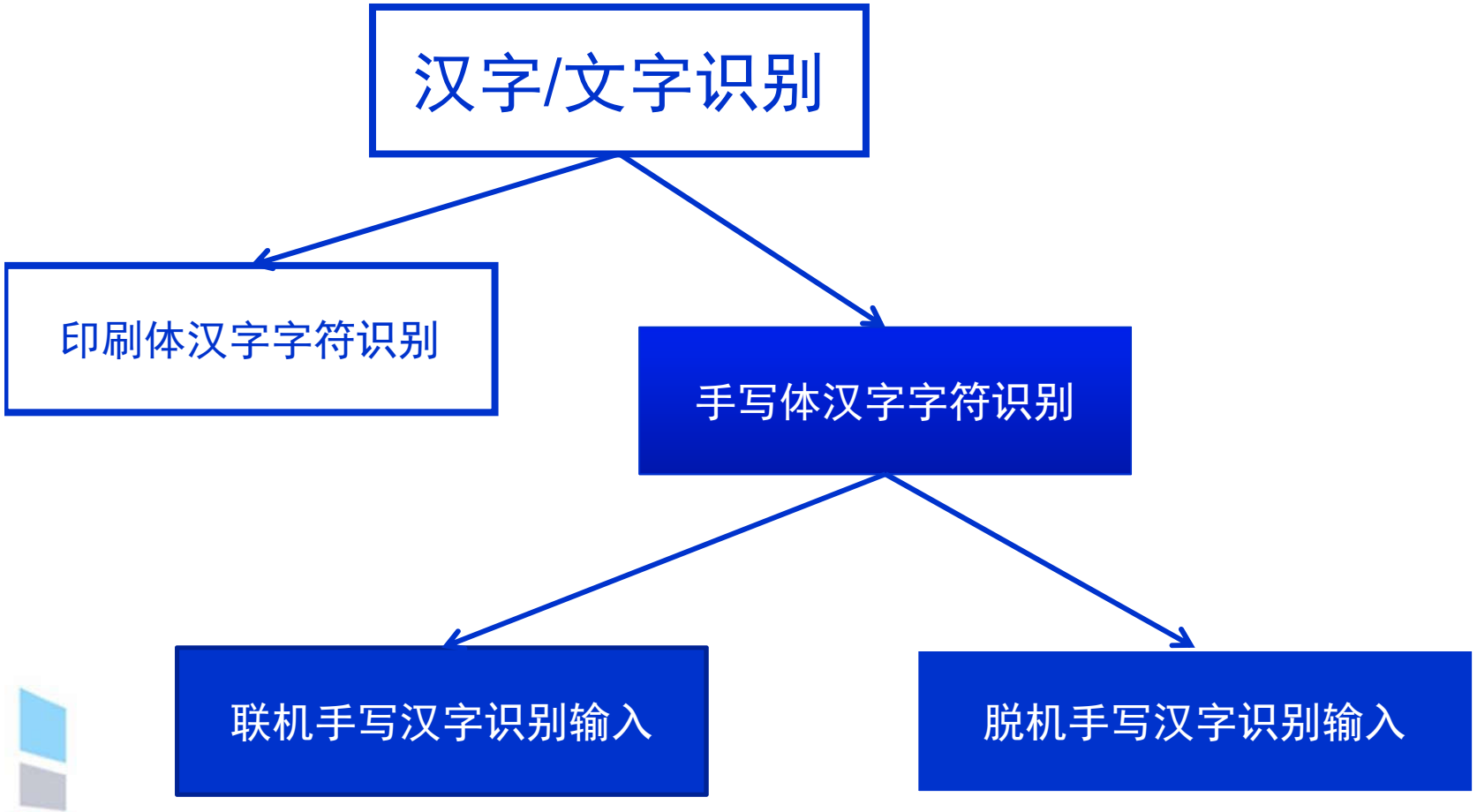
语音识别处理的过程



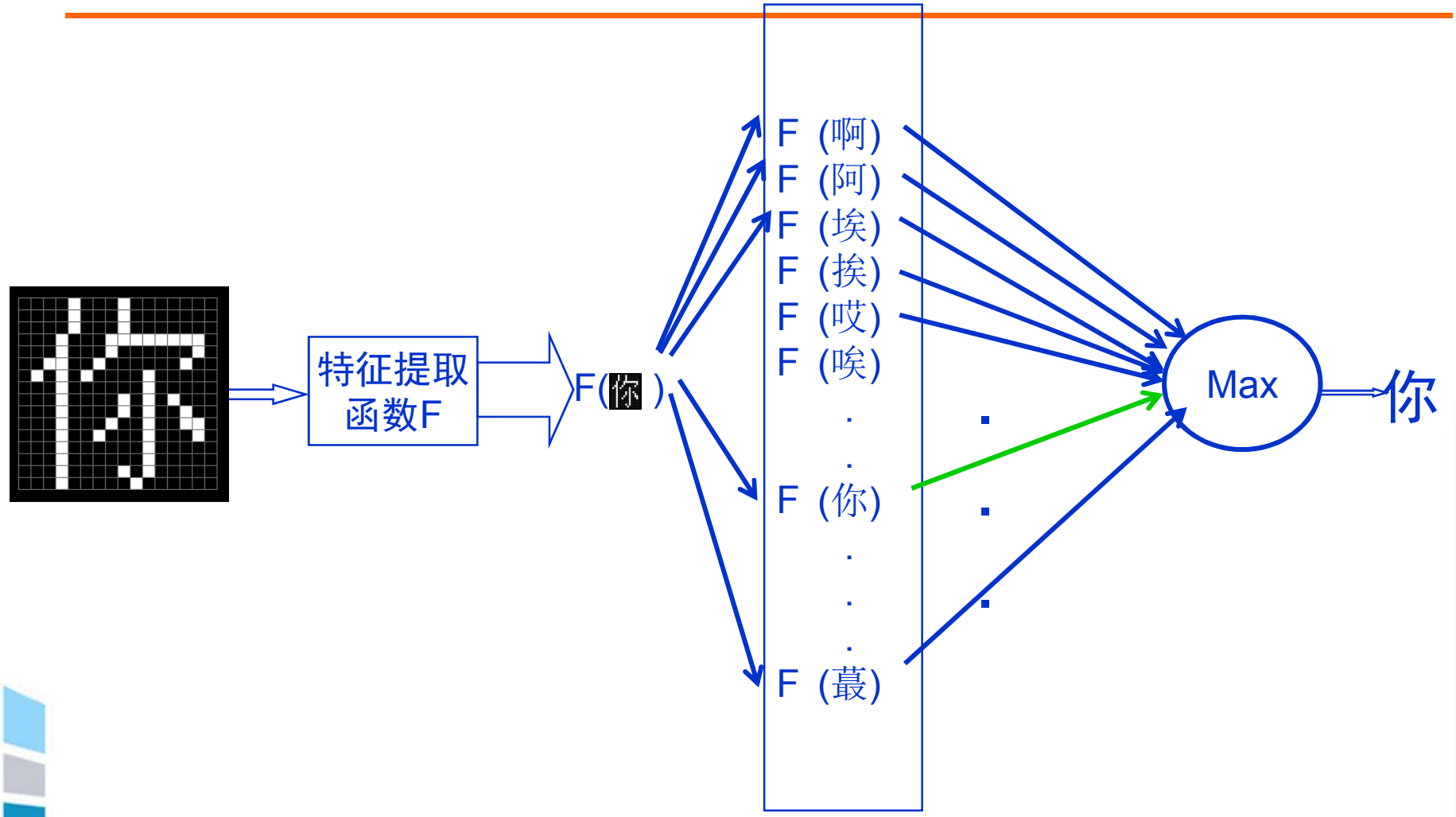
6.1.4 手写体汉字识别输入

- 通过书写汉字记录信息已经有几千年的历史，这也是使用汉字的中国人的普遍习惯，所以手写识别输入是符合中国人记录信息习惯的。
- 在1997年以后就已经有基本实用的手写汉字识别输入系统，其中佼佼者为中自公司的“汉王99”、摩托罗拉公司的“慧笔”和清华紫光公司的“紫光笔”等。

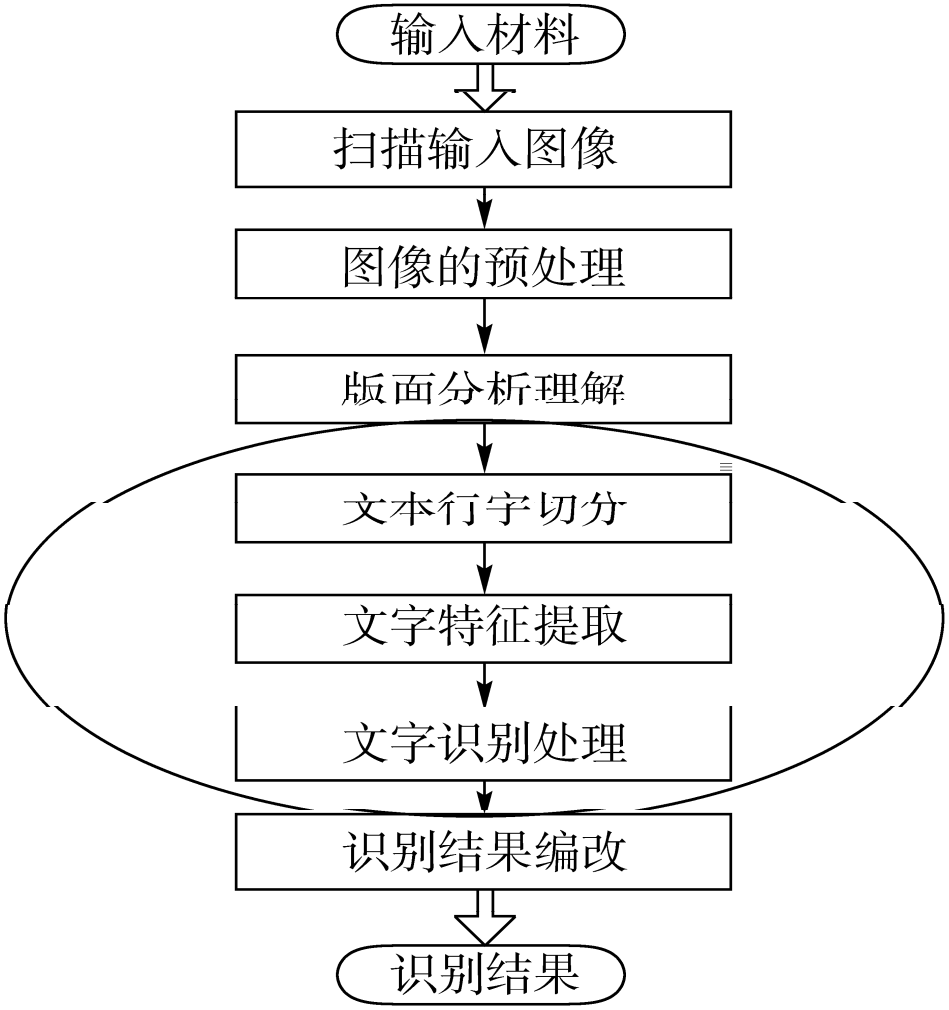
汉字/字符识别输入技术



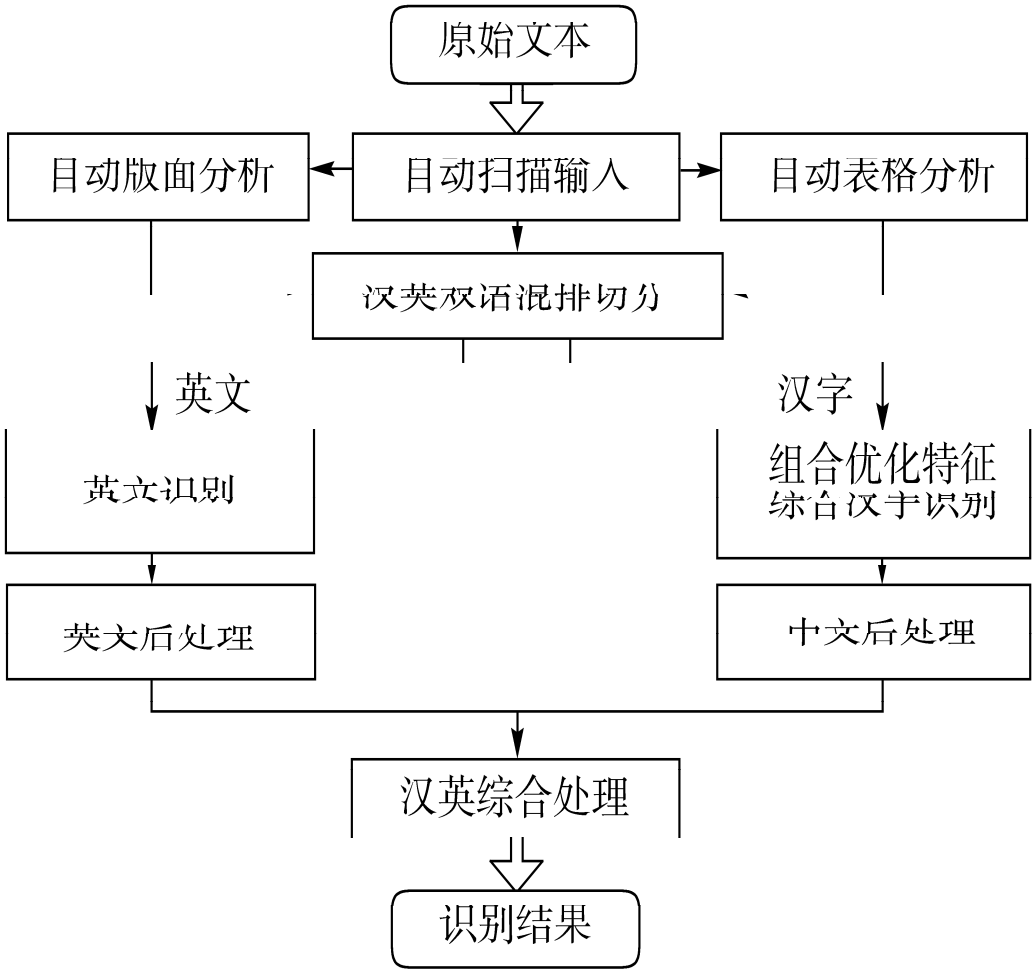
OCR基本原理



OCR技术一般流程


输入材料
扫描输入图像
图像的预处理
版面分析理解
文本行字切分
文字特征提取
文字识别处理
识别结果编改
识别结果

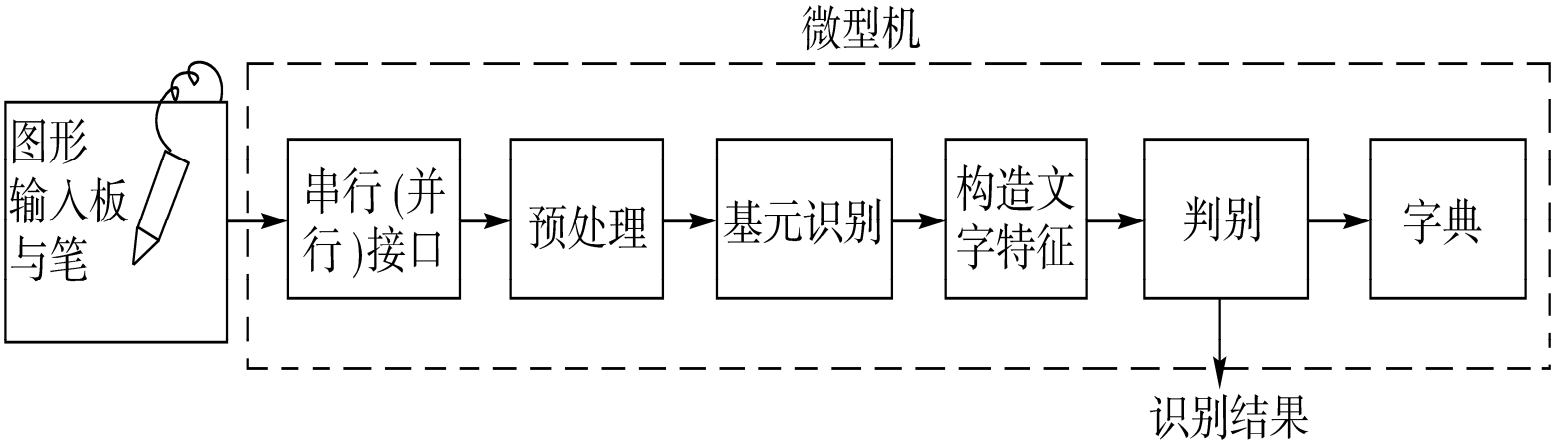
印刷汉字识别技术流程


原始文本
目动版面分析
目动扫描输入
目动表格分析
汉英双语混排切分
英文
汉字
英文识别
组合优化特征
综合汉字识别
英文后处理
中文后处理
汉英综合处理
识别结果

联机手写汉字识别

- 联机手写汉字识别是最简单的一种。联机识别时，人一面写，机器一面认，是一种实时识别方式。
- 是一种半自动输入汉字方式，和各种人工编码击键输入相比，有以下优点：①是一种很方便的输入手段，几乎不需要对书写人员进行任何训练。②可在构思文章的同时进行输入，是一种“想打”型的输入装置。③利用图形输入板可方便地进行编辑，修改和图形输入。缺点是输入速度较慢和不适应大量输入。

联机手写汉字识别流程



联机手写汉字识别的一般方法

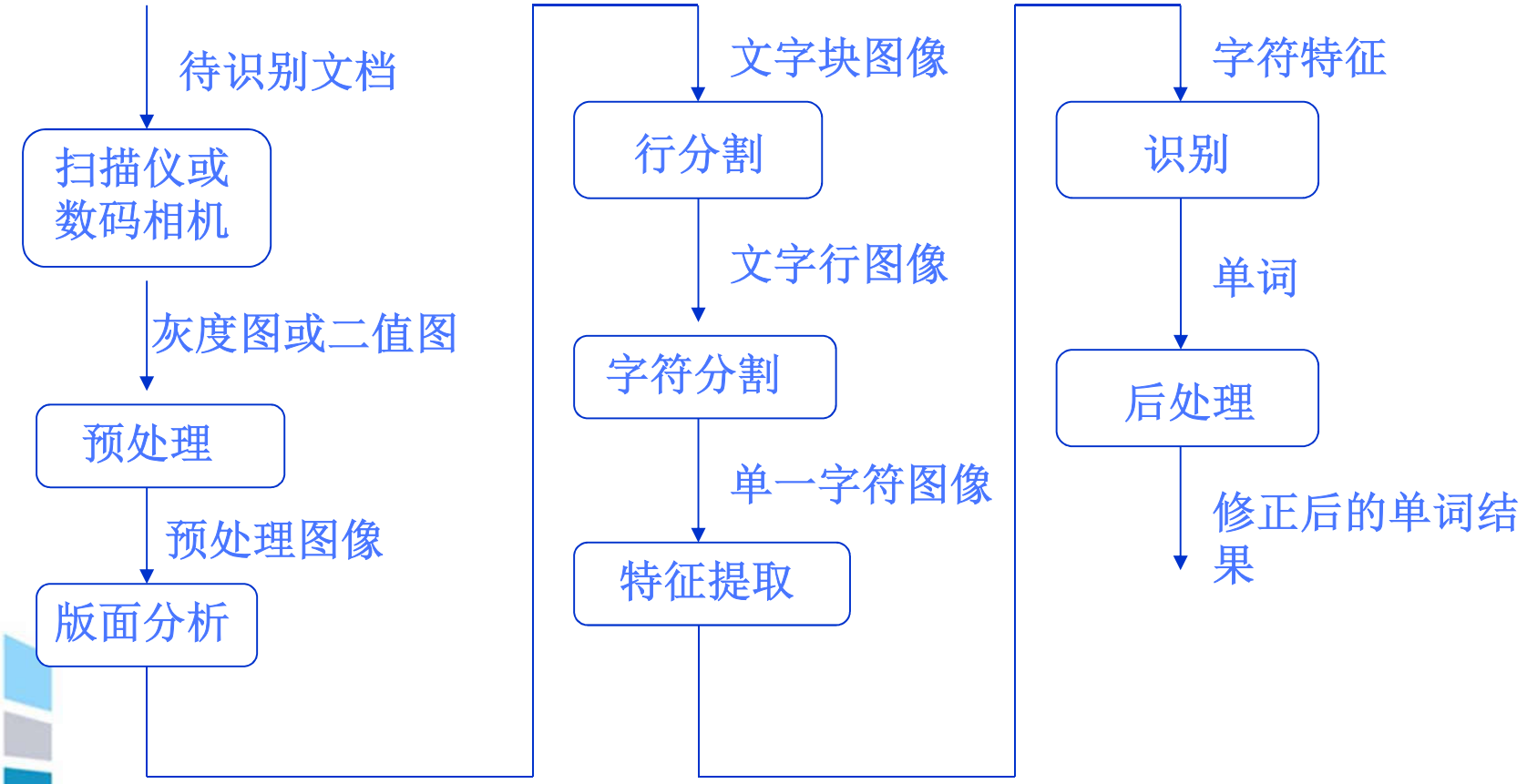
- 采用先识别笔画再判别整个汉字的**两级识别**方法。这是一种根据联机识别的特点（笔画一笔一笔写）自然产生的方法。

- 根据图形输入板输出的坐标点识别笔画

（1）由笔画书写的方向和方向的变化来识别笔画

（2）直接根据笔画上的线段方向，用动态规划法识别笔画。

脱机手写体字符识别系统结构



6.2 汉字键盘输入

- 键盘
- 键盘输入原理
- 键盘输入总体流程
- 输入码对照表的设计

6.2.1 键盘


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
43
58
30
31
32
33
34
35
36
37
38
39
40
29
42
44
45
46
47
48
49
50
51
52
53
54
57


69
53
55
74
71
72
73
78
75
76
77
79
80
81
29
82
83

一、键盘扫描码（Scan Code）

- 键盘控制器完成的主要工作是：
  - 加电或系统需要时对键盘进行检测；
  - 扫描键盘、消除重键，保存扫描代码；
  - 向主机传送按键的扫描码(Scan Code)等。

- 扫描码是指根据按键在键盘上的位置按照从左到右，从上到下的原则从1开始给按键编的代码。

二、键盘工作原理

- 键盘上按下一个键后，键盘控制器就能通过扫描矩阵得到该键的位置；
- 把这个位置信息转换为该键的扫描码，把它存储在键盘控制器中的缓冲区内；
- 键盘控制器立即产生一个键盘的硬件中断，这个硬件中断的处理程序开始运行；
- 该中断处理程序从键盘控制器的缓冲区中取得这个键的扫描码，并获得该键符号所对应的ASCII码（或者00H）；
- 把该键的扫描码和ASCII码（或者00H），存储到计算机内存中的键盘缓冲区内，供系统使用。

部分扫描码和ASCII码的对应关系

按键	扫描码	ASCII	按键	扫描码	ASCII	按键	扫描码	ASCII
A	1E	41	9	0A	39	[	1A	5B
B	30	42	`	29	60	INSERT	E0,52	
C	2E	43	-	0C	2D	HOME	E0,47	
D	20	44	=	0D	3D	PG UP	E0,49	
E	12	45	\	2B	5C	DELETE	E0,53	
F	21	46	BKSP	0E	08	END	E0,4F	
G	22	47	SPACE	39	20	PG DN	E0,51	
H	23	48	TAB	0F	09	U ARROW	E0, 48	
I	17	49	CAPS	3A		L ARROW	E0,4B	
J	24	4A	L SHFT	2A		D ARROW	E0,50	

6.2.2 汉字键盘输入原理

一、西文字符的输入

- 输入西文文字时，敲入的按键就可以直接根据当前的代码页（Code Page）转换为西文字符的内码从而显示在当前应用程序的窗口中；
- 如当前是英文键盘，那么在键盘上按下键“A”后，计算机就可以根据英文键盘的布局把这个键转换为字符“A”的ASCII码，并送给应用程序，从而在界面上显示这个字符“A”。

二、汉字的编码输入

- 汉字的输入过程，就是把汉字的输入码转换成汉字内码的过程：
  - 输入系统内把所有的汉字（内码）与其对应的输入码排列在一起，构成一张“汉字-输入码对照表”；
  - 输入汉字时，输入系统把当前输入的汉字输入码接受下来，通过查找“汉字-输入码对照表”，获得其对应的汉字内码，即完成了该汉字的输入。

汉字编码输入举例


①wo
④提示
⑤1
⑥
输出
汉字
到
应用
程序
指定档...
文件(F) 编辑(E)
格式(O) 查看(V)
帮助(H)
我
用户操作键盘
输入系统
应用程序
②查对照表
③输出汉字串
我wo
握wo
窝wo
卧wo
挝wo
沃wo
蜗wo
涡wo
斡wo
倭wo
幄wo
输入码对照表

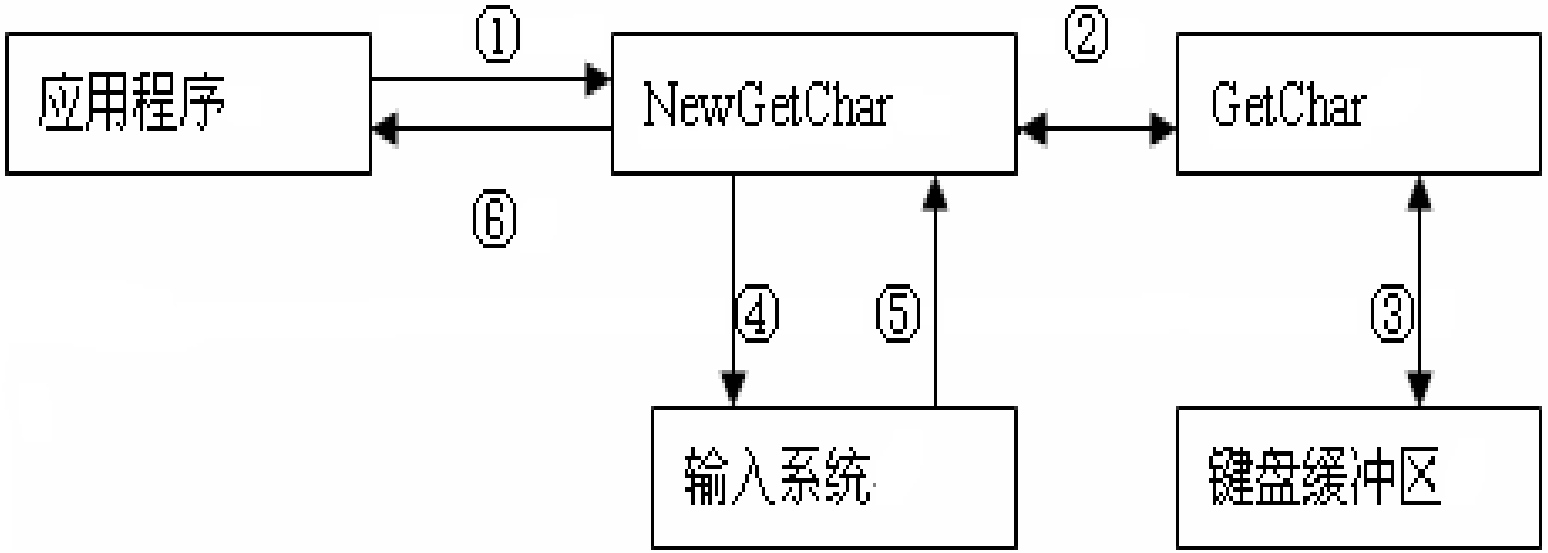
三、汉字编码输入的前提

- 要实现把汉字输入码转换成汉字内码，必须具备一个前提条件，那就是输入系统必须先于应用程序得到用户按键的扫描码或ASCII码，然后作相应的转换处理，如果应用程序先得到的话，那么输入系统就没有办法作转换处理了；
- 从键盘接收的按键消息是由操作系统的I/O模块处理的，它负责把这些扫描码转换为ASCII码，并存放到键盘缓冲区中，那么只要保证输入系统能够先于应用程序拿到键盘缓冲区中的内容就可以了。

四、键盘HOOK

- 应用程序从键盘缓冲区读入字符，是通过调用系统的读字符功能来实现的；
- 可以通过系统提供的HOOK（钩子）机制，来使输入系统先于应用程序得到键盘按键信息。

键盘HOOK原理图



- 键盘HOOK的实现原理：
  - 设系统内原来提供的读字符系统调用（函数）是GetChar(…)；
  - 在输入系统内，设计一个新的读字符系统调用（函数）NewGetChar(…)；
  - 把NewGetChar作为一个钩子钩到GetChar上，那么以后当应用程序调用函数GetChar时，实际调用的就是函数NewGetChar；

- 在NewGetChar中，它调用原来的GetChar函数从键盘缓冲区内读字符，然后把字符传递给输入系统，由输入系统进行处理，输入系统处理完以后，就把结果传给函数NewGetChar，再由它把结果返回给应用程序；
- 需要注意的是，NewGetChar和GetChar的参数和返回值应该一致。

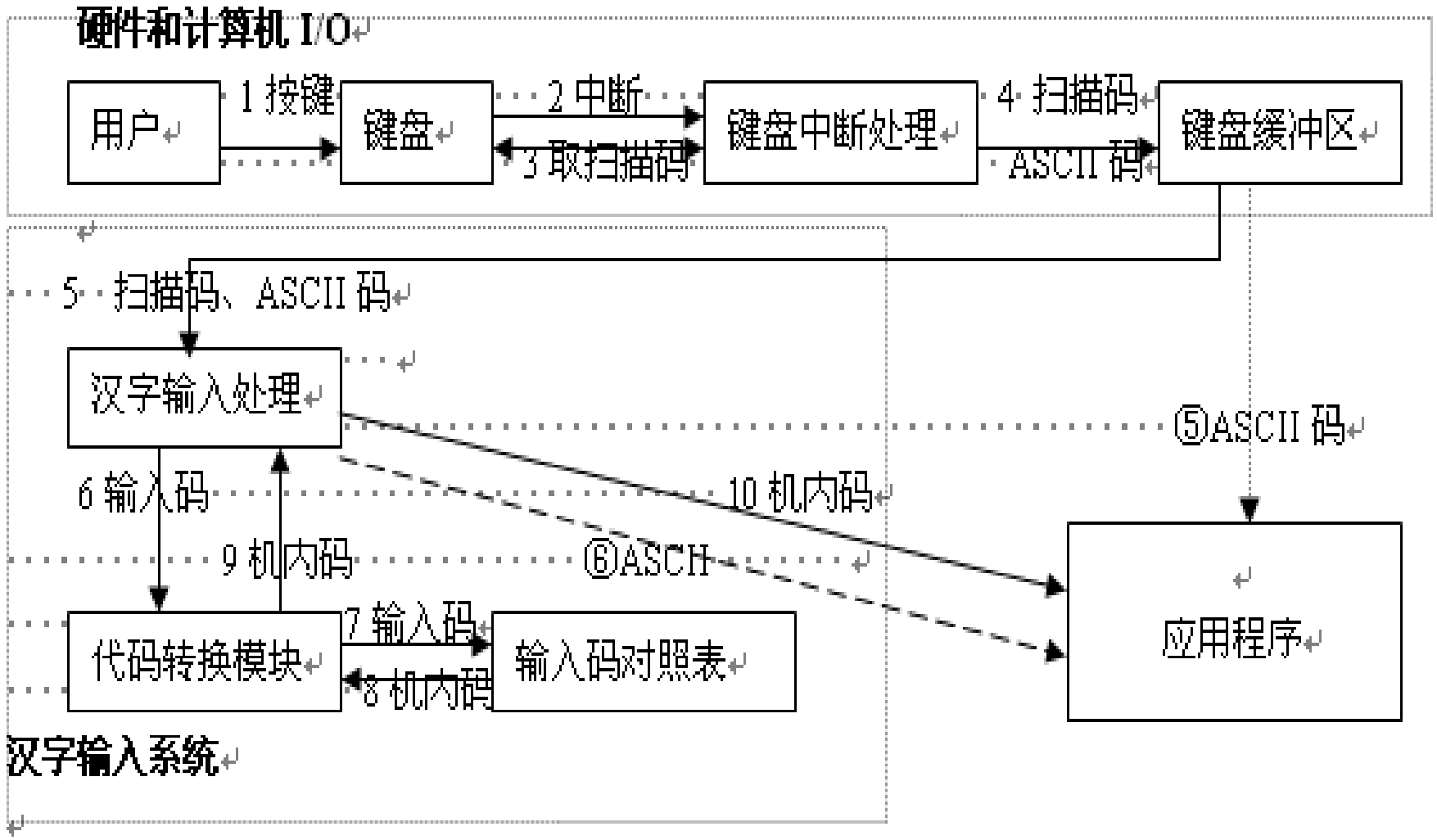
五、Windows下的HOOK

```
HHOOK SetWindowsHookEx(int idHook,
                       HOOKPROC lpfn,
                       HINSTANCE hMod,
                       DWORD dwThreadId);
```

- idHook指定钩子的类型，常用的有WH_MOUSE、WH_KEYBOARD、WH_GETMESSAGE等，键盘操作应该设定为WH_KEYBOARD；

- lpfn标识钩子函数的入口地址，当钩子钩到任何消息后便会调用这个函数，即当不管系统的哪个窗口有键盘输入时，马上就会引起lpfn所指向函数的动作；
- hMod 是钩子函数所在模块的句柄，我们可以设定其为本应用程序的实例句柄；
- dwThreadId是钩子相关函数的ID，用以指定想让钩子去钩哪个线程，其值为0时，则拦截整个系统的消息。

6.2.3 键盘汉字输入的总体流程



6.2.4 输入码对照表的设计

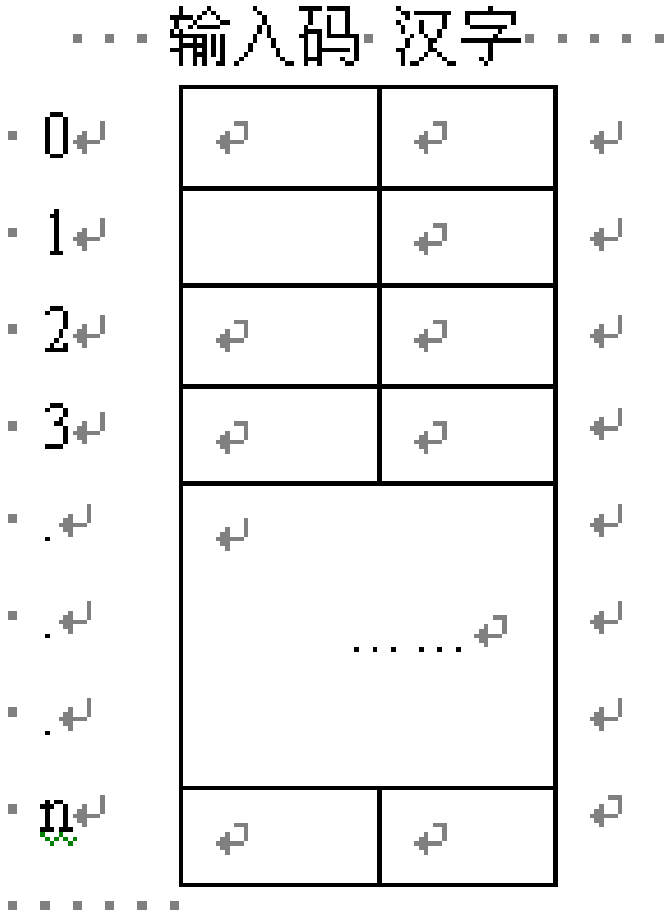
- 输入码对照表是汉字键盘输入中的一个重要数据结构，是汉字输入码到内码转换的核心，它反映了汉字输入码和内码之间的映射关系；
- 一般来说，系统中有一种输入法，就需要为它配备一张输入码对照表；
- 对照表一般有两种类型：字对照表和词对照表，在字对照表中，表示的是汉字和输入码的对应关系，而词对照表则表示词（组）和输入码的对应关系。

输入码对照表结构的分类

- 定长结构对照表
- 计算结构对照表
- 变长结构对照表
- 索引结构对照表
  - 稠密索引结构对照表
  - 稀疏索引结构对照表

定长结构对照表

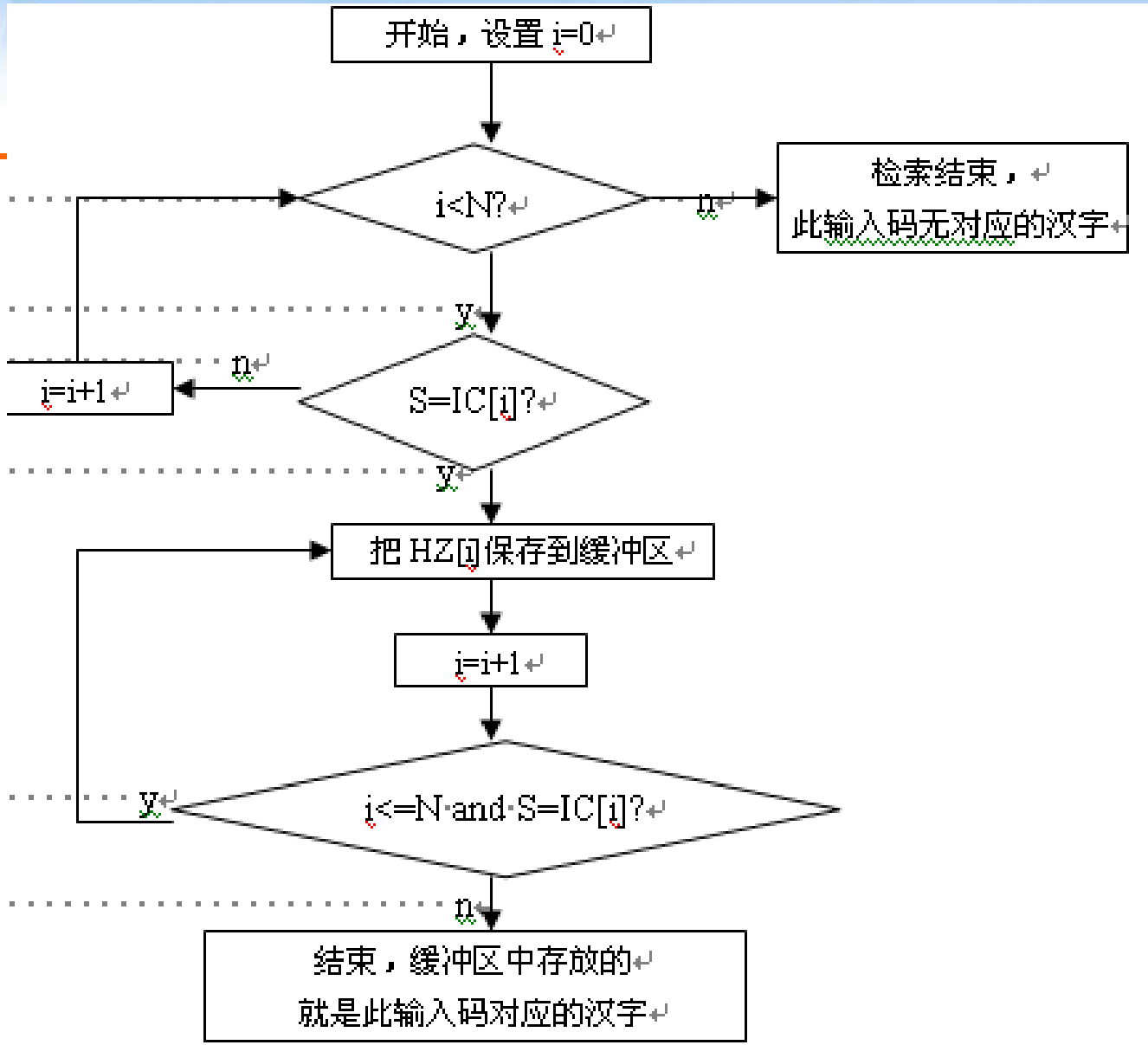
- 定长结构的对照表中，每个表项的长度相同；
- 其表项由两个域组成：一个域存放汉字或词的输入码，另一个域存放此汉字或词的内码；
- 每个域的长度是固定的，例如，表项固定长度为6个字节，其中输入码占4个字节，内码占2个字节；
- 由于一个汉字的内码为2个字节，所以字对照表可以采用定长结构；
- 词对照表一般不会采用这种结构，这是因为词的长度不一，否则容易造成存储空间的浪费。


输入码
汉字
0
1
2
3
n

- 检索方法
  - 扫描法
  - 二分法
  - 假设需要检索的输入码为S，输入码对照表的表项数为N，分别用IC[i]和HZ[i]分别表示对照表中第i个表项内的输入码域和内码域。

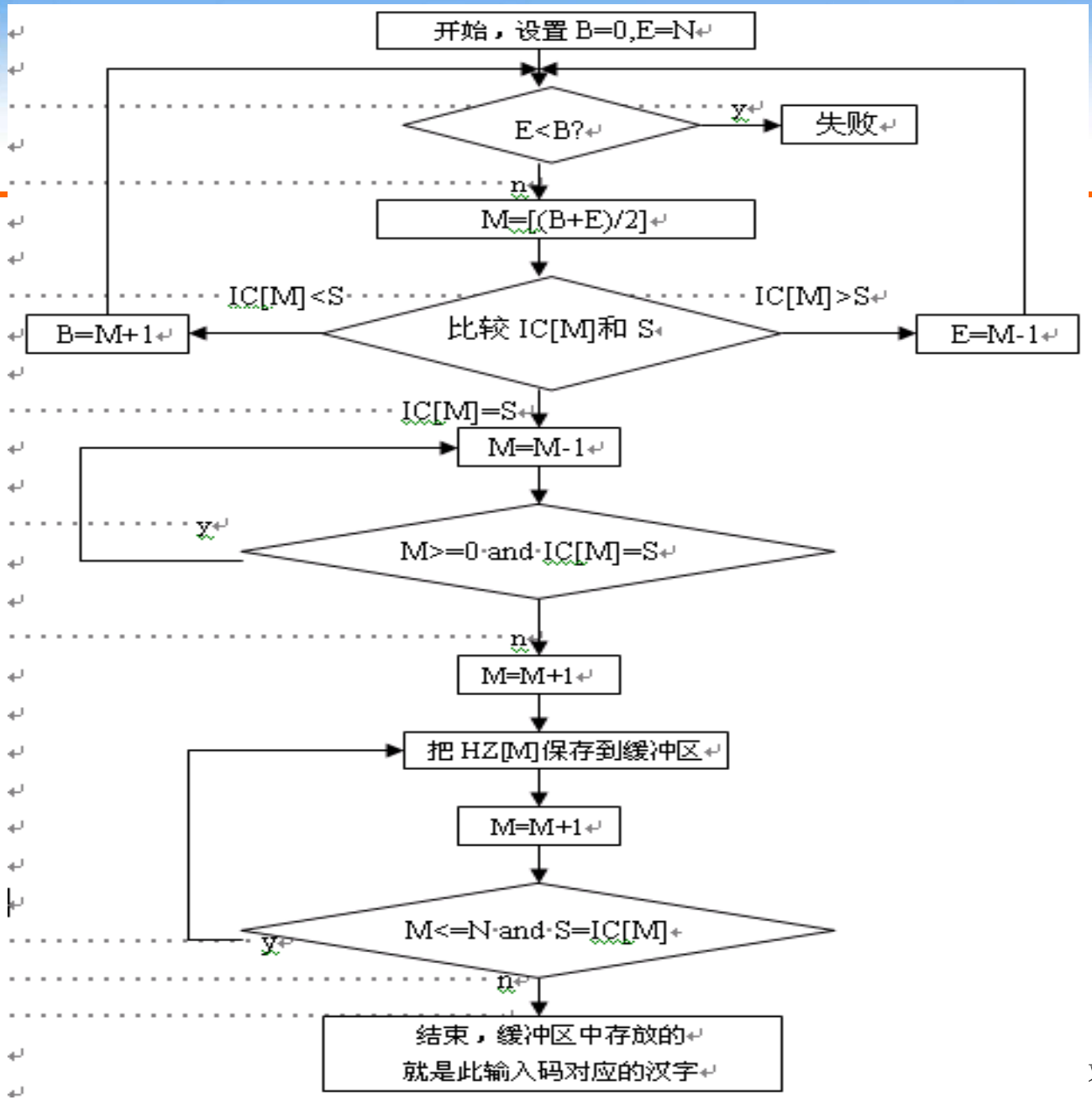
扫描法：

- 为了方便检索，需要把对照表的N个表项，按照输入码的顺序从小到大排列；
- 扫描法每次查找都从表首开始，当查找到和S相同的输入码项i后，继续再往后查找，一直找到和S不同的输入码项j时，那么第i项到第j-1项（共j-i项）内的汉字内码，就是输入码S所对应的重码汉字集；
- 扫描法最大的比较次数为N，最小的比较次数为1，其平均比较次数为(N+1)/2。


开始，设置 i=0
i<N?
n
检索结束，
此输入码无对应的汉字
y
n
i=i+1
S=IC[i]?
y
把 HZ[i]保存到缓冲区
i=i+1
y
i<=N and S=IC[i]?
n
结束，缓冲区中存放的
就是此输入码对应的汉字

二分法：

- 二分法的算法描述如下：
- ①设置一个值B=0，E=N；
- ②如果B>E，算法失败结束；否则M=[(B+E)/2]；
- ③然后把IC[M]和S比较；
- ④如果相等，则说明找到了一项，由于输入码相等的汉字是相邻存放的，所以只要进行前后的扫描，可以得到所有的输入码为S的汉字，结束；
- ⑤如果S大于IC[M]，则B=M+1，转②；
- ⑥如果S小于IC[M]，则E=M-1，转②。


开始，设置 B=0,E=N
E<B?
y
失败
n
M=[(B+E)/2]
IC[M]<S
IC[M]>S
比较 IC[M]和 S
B=M+1
E=M-1
IC[M]=S
M=M-1
y
M>=0 and IC[M]=S
n
M=M+1
把 HZ[M]保存到缓冲区
M=M+1
M<=N and S=IC[M]
y
n
结束，缓冲区中存放的
就是此输入码对应的汉字

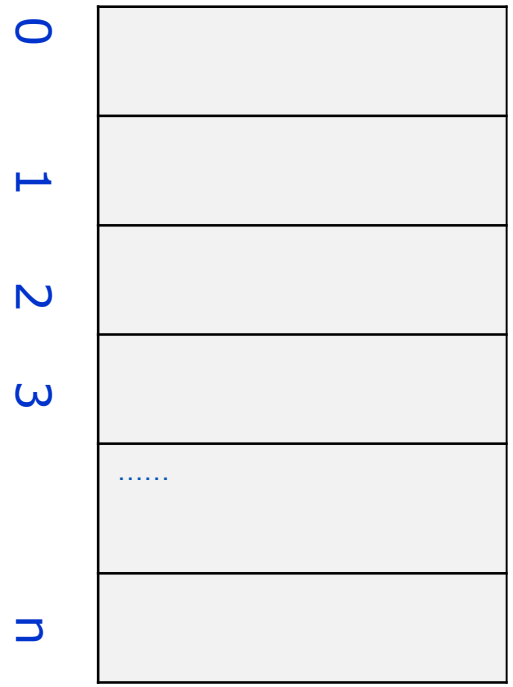
文信息处理

- 二分法的平均比较次数约为$\log_2 N-1$；
- 假设N=20902，则扫描法的平均比较次数为(20902+1)/2=10452，而二分法的平均比较次数为$\log_2 20902-1=14$，由此可见二分法的效果要明显优于扫描法；
- 定长结构的对照表具有结构简单、容易构建、检索算法简单等优点，但在存储空间上存在缺陷，它与其它结构的对照表相比，要占用较大的存储空间。

计算结构对照表

- 计算结构常用于字对照表，每个表项对应一个汉字，表项等长，其内容为该汉字对应的输入码；
- 这种结构的对照表表项必须按照汉字的内码序进行排列；
- 注意：尽管该表是有序的，但是并不是按照表项内容之序排列，所以这种结构的对照表不能用二分法来查找。

输入码



计算结构对照表实例1：

- 纵横码的码元集合为：{1,2,3,4,5,6,7,8,9,0}；
- 码长为1~4，字符内码采用Unicode（UCS-2），汉字内码值为4E00H~9FA5H，共计20902个汉字，为连续编码；
- 每个表项设计为等长，为2个字节，其内容是输入码，为1－4个数字，每个数字均用BCD码来表示，需要4位（bit），4个数字则需要4×4＝16位，用2个字节表示，如果数字不足4位，用全“1”位填充 。

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
第一位码元				第二位码元				第三位码元				第四位码元			

表项结构

每个表项的大小为2个字节。整张对照表总共需要2×20902=41804个字节，按照汉字内码的大小从小到大排列。

表项	2个字节输入码	2个字节输入码	……	2个字节输入码	2个字节输入码	……	2个字节输入码	2个字节输入码	……	2个字节输入码	2个字节输入码
序号	0	1	……	255	256	……			……	20900	20901
内码	4E00	4E01	……	4EFF	4F00		9EFF	9F00		9FA4	9FA5

相关算法

- 假设汉字的内码为hz，可以表示为：
- $hz=\{ hz \subset R | 4E00H \leq R \leq 9FA5H \}$
- 从上表我们可以得到序号i和内码hz的关系：
- $i = hz - 4E00H$
- 则有：$hz = i + 4E00H$
- 在对输入码对照表的检索过程中，如果发现第*i*个表项的内容与当前输入码为一致时，则用上面给出的公式，根据 *i* 计算出该表项所对应汉字的内码，从而实现转换过程。

计算结构对照表实例2：

- 纵横码的码元集合为：{1,2,3,4,5,6,7,8,9,0}；
- 码长为1~4，字符集采用GB 2312标准，共6763个汉字，双字节内码，每个字节的取值范围均为A1H－FEH，其编码空间为94×94，为非连续编码，但是为部分连续编码；
- 每个表项设计为等长，为2个字节，其内容是输入码，为1~4个数字，每个数字均用BCD码来表示，需要4位（bit），4个数字则需要4×4＝16位，用2个字节表示，如果数字不足4位，则用全“1”位填充 。

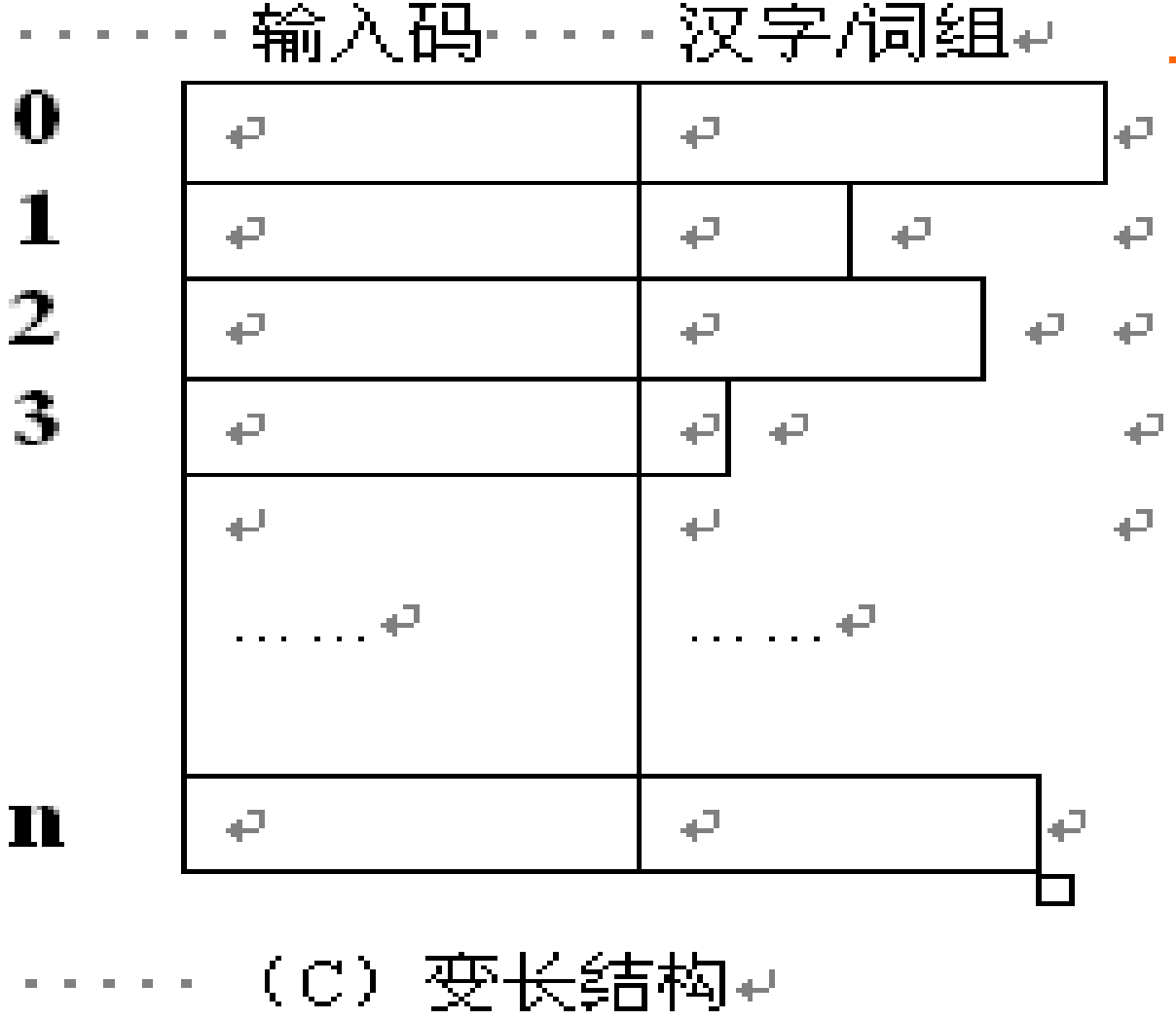
相关算法

- 假设汉字的内码**hz**可以表示为：
- $hz=\{ hz_h hz_l \mid hz_h \subset R1, hz_l \subset R2\}$
- $R1=\{R \mid A1H \le R \le FEH\}$
- $R2=\{R \mid A1H \le R \le FEH\}$
- 从上表我们可以得到序号*i*和内码**hz**的关系：
- $i=(hz_h - A1H) \times 5EH + (hz_l - A1H)$
- 则有：
- $hz_h=[i/5EH]+A1H$ // [ ]表示取整
- $hz_l=(i \bmod 5EH)+A1H$ // mod表示取模
- 在对输入码对照表的检索过程中，如果发现第*i*个表项的内容与当前输入码为一致时，则用上面的公式，根据*i* 计算出该表项所对应汉字的内码，从而实现转换过程。

- 计算结构对照表的最显著优点是节省存储空间，它与采用等长结构对照表相比较，可省下存放20902个汉字内码的空间；
- 对这种结构的输入码对照表的检索，只能采用扫描法，故比较次数较多，为(N+1)/2，因此从理论上讲，对速度有一定程度的影响；
- 另外，当一个汉字有多个输入码时，就要采取特殊措施解决，如设置附加表等方法。

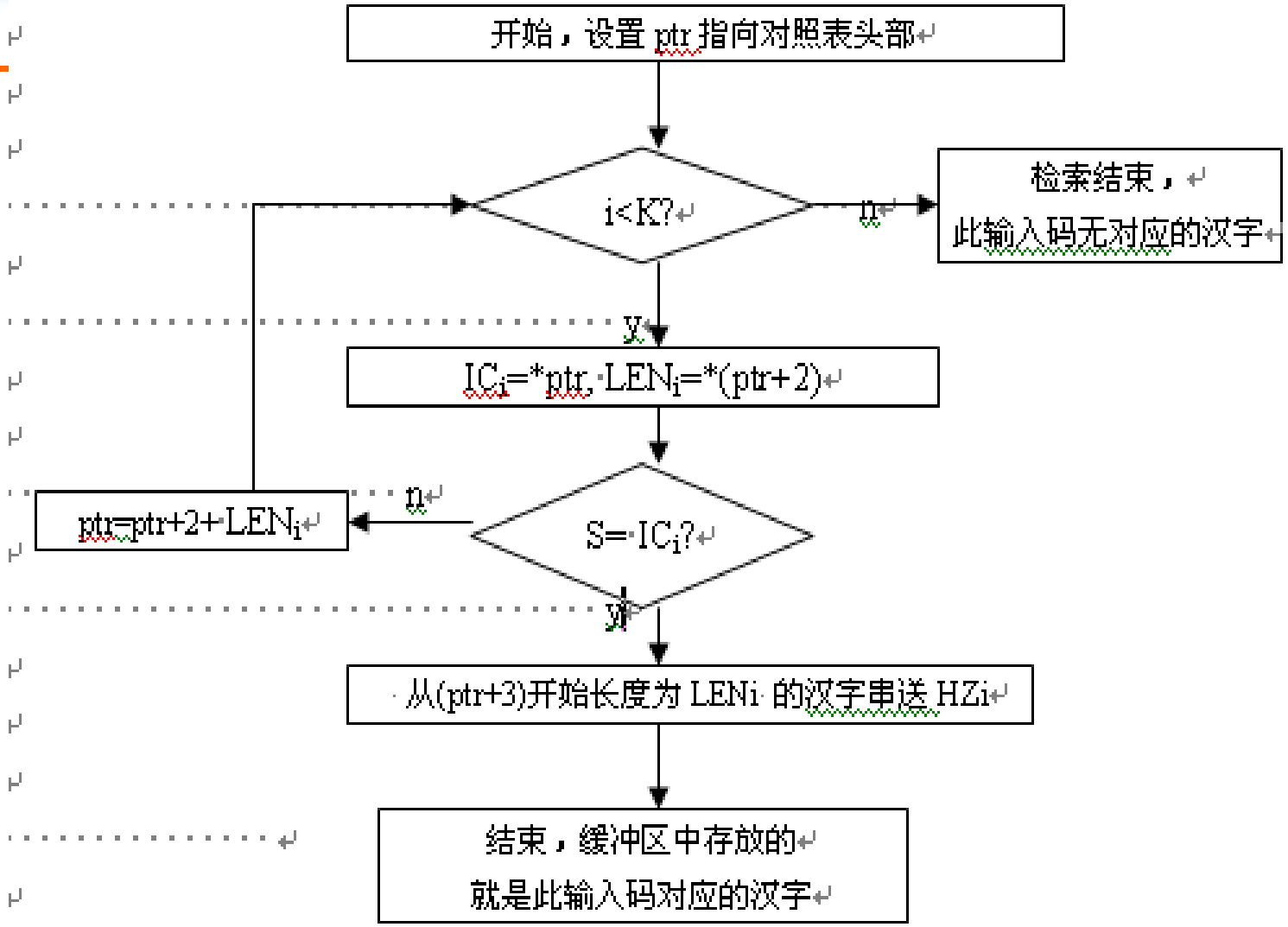
变长结构对照表

- 在变长结构的对照表中，每个表项具有两个域：一个域存放输入码，另一个域存放该输入码所对应的所有重码字或词的内码集合，由于第二个域的内容是不定长的，所以使得表项不等长；
- 这种结构的对照表中不需要为每个汉字存放输入码，故其表项较少，因此其存储空间相对等长结构的对照表有明显的减小。


输入码
汉字问组
0
1
2
3
n
（C）变长结构

- 以20902个汉字的纵横码对照表为例，
  - 若采用定长结构的对照表，则需要(2＋2)×20902＝83608字节（2个字节输入码，2个字节汉字内码）；
  - 若采用变长结构的对照表，则需要20902×2＋6123×3＝60173字节（20902个汉字内码和6123个输入码的长度与重码字计数器），比定长结构的对照表减少了近23K字节；
- 这种结构的对照表存在的问题是，由于其表项不等长，所以必须采用扫描法查找，所以从理论上讲，其检索时间相对比较长。

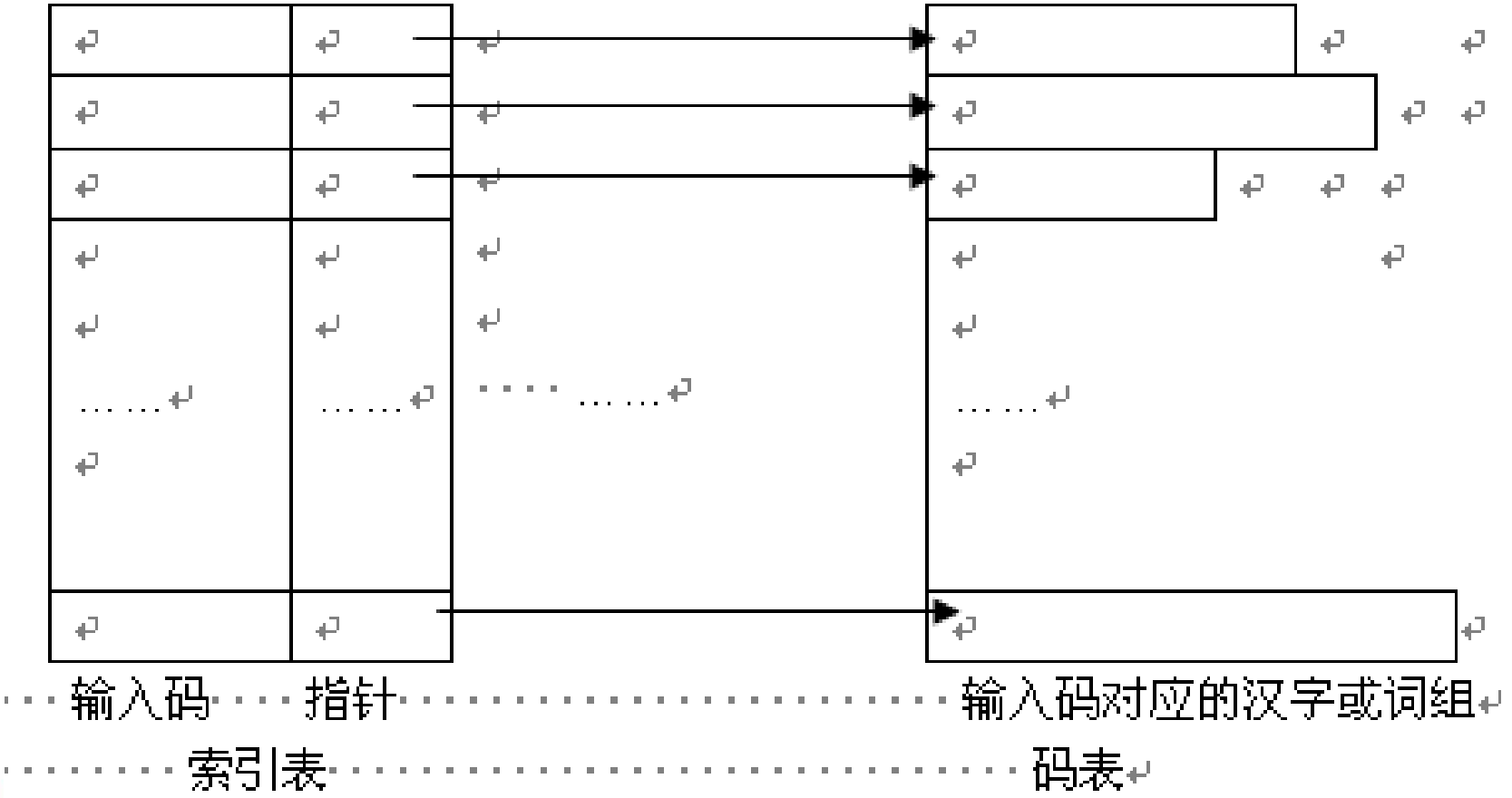
- 纵横码的不等长对照表结构描述：
  - <对照表>::=<表项>[<表项>]
  - <表项>::=<输入码><汉字集>
  - <输入码>::=<2个字节的BCD码>
  - <汉字集>::=<1个字节汉字集长度><重码汉字>
  - <重码汉字>::=<2个字节的汉字内码>[< 2个字节的汉字内码>]
- 假设需要检索的输入码为S，对照表表项数为K，分别用ICi，LENi，HZi表示第i表项的输入码、重码汉字词集长度和重码汉字词集合。


开始，设置 ptr 指向对照表头部
i<K?
n
检索结束，
此输入码无对应的汉字
y
ICi=*ptr, LENi=*(ptr+2)
S=ICi?
n
ptr=ptr+2+LENi
y
从(ptr+3)开始长度为 LENi 的汉字串送 HZi
结束，缓冲区中存放的
就是此输入码对应的汉字

- 假设输入码的个数为K，那么最大的比较次数为K，最小的比较次数为1，所以其平均比较次数为(K+1)/2；
- 由于K比N小，所以其比较次数要比等长结构的对照表少，以纵横码为例：
  - 变长结构对照表的平均比较次数是(6123+1)/2=3062次；
  - 等长结构对照表采用扫描法的平均比较次数为(20902＋1)/2＝10452，是变长结构的3倍！

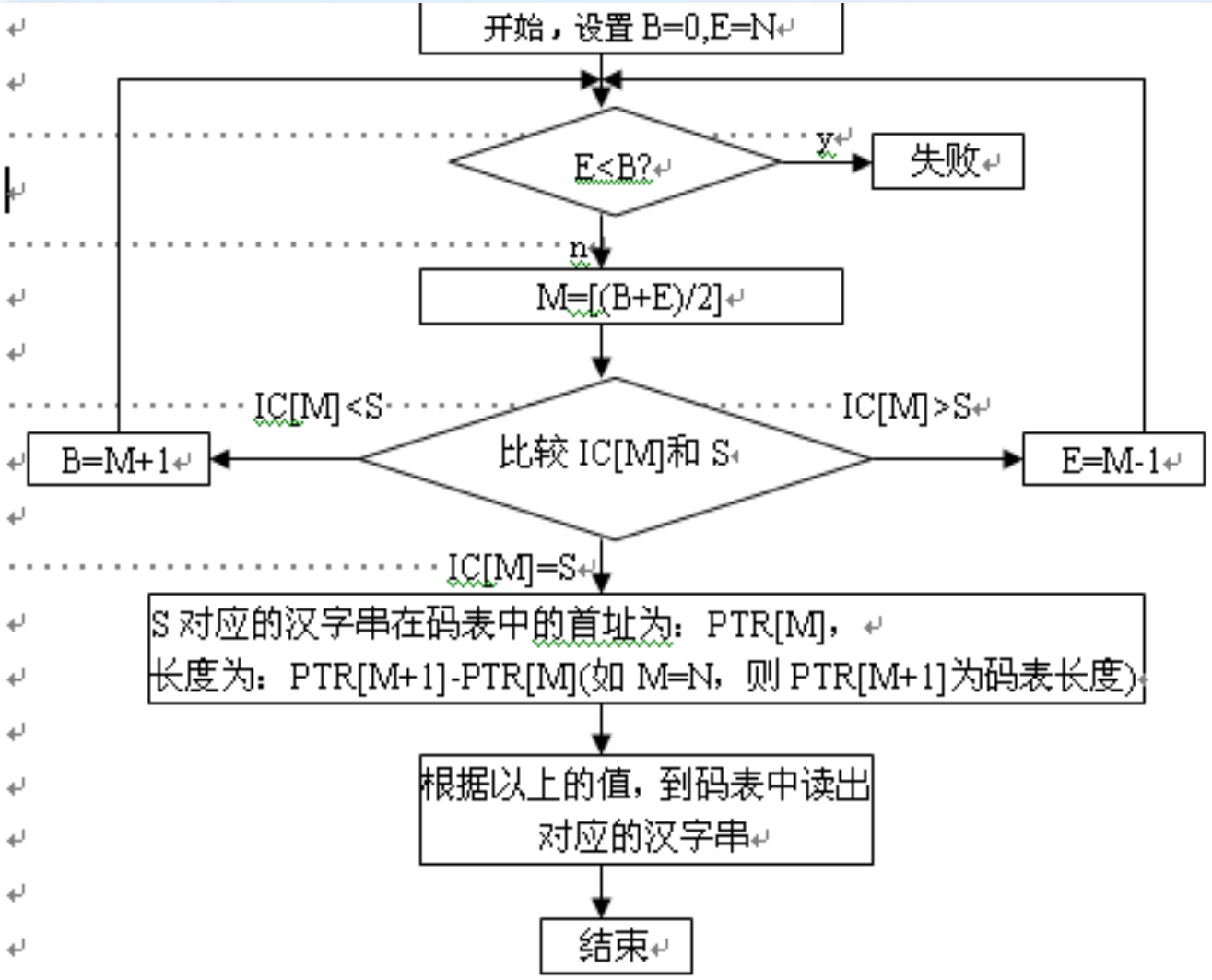
六、索引结构对照表

- 索引结构是一种最常用的对照表结构，也是一种高效率的检索结构；
- 索引结构的对照表分成两个部分—索引表和码表：
  - 索引表的表项由两部分组成，包括输入码和指针，指针指向码表中该输入码对应的汉字和词组集的首址；
  - 码表的结构与变长结构对照表的右半部分相同（存放重码汉字）；


输入码
指针
输入码对应的汉字或词组
索引表
码表

- 每个索引表表项对应于一个码表表项；
- 索引表中的表项是等长的，而码表中的表项是不等长的。

- 索引结构结合了等长结构和变长结构的优点，通过把输入码和汉字串分开的方法，使得索引表为等长结构，而码表为变长结构，从而实现了既提高检索效率（可以对索引表实施二分法检索），又节省了存储空间。


开始，设置 B=0,E=N
E<B?
y
失败
n
M=[(B+E)/2]
IC[M]<S
IC[M]>S
比较 IC[M]和 S
B=M+1
E=M-1
IC[M]=S
S 对应的汉字串在码表中的首址为：PTR[M]，
长度为：PTR[M+1]-PTR[M](如 M=N，则 PTR[M+1]为码表长度)
根据以上的值，到码表中读出
对应的汉字串
结束

七、各种结构对照表的性能比较

- 以拼音码为例，来对等长、变长和索引结构的对照表的存储空间和检索速度作一个比较；
- 拼音可以归纳为共有23个声母和36个韵母，每个汉字的读音由一个声母和一个韵母组成（有少数汉字只有韵母没有声母）；
- 采用压缩方法来存放输入码，用1—23表示23个声母，可以用5位来表示，用1—36表示36个韵母，可以用6位来表示，共计11位，为了方便起见，用两个字节存放输入码（如果无声母，用0表示）。

声母	无声母（0） b(1) p(2) m(3) f(4) d(5) t(6) n(7) l(8) g(9) k(10) h(11) j(12) q(13) x(14) zh(15) ch(16) sh(17) r(18) z(19) c(20) s(21) y(22) w(23)
韵母	a(1) o(2) e(3) ai(4) ei(5) ao(6) ou(7) an(8) en(9) ang(10) eng(11) ong(12) i(13) ia(14) ie(15) iao(16) iou(17) ian(18) in(19) iang(20) ing(21) iong(22) u(23) ua(24) uo(25) uai(26) uei(27) uan(28) uen(29) uang(30) ueng(31) v(32) ve(33) van(34) vn(35) er(36)

- 等长结构：
  - <对照表>::=<表项>[<表项>]
  - <表项>::=<输入码><汉字>
  - <输入码>::=<1个字节的声母><1个字节的韵母>
  - <汉字>::=<2个字节内码>
- 变长结构：
  - <对照表>::=<表项>[<表项>]
  - <表项>::=<输入码><汉字集>
  - <输入码>::=<1个字节的声母><1个字节的韵母>
  - <汉字集>::=<1个字节汉字集长度><重码汉字>
  - <重码汉字>::=<2个字节的汉字内码>[< 2个字节的汉字内码>]

- 索引结构：
  - <对照表>::=<索引表><码表>
  - <索引表>::=<表项>[<表项>]
  - <表项>::=<输入码><指针>
  - <输入码>::=<1个字节的声母><1个字节的韵母>
  - <指针>::=<2个字节的整数>
  - <码表>::=<1个字节汉字集长度><重码汉字>
  - <重码汉字>::=<2个字节的汉字内码>[< 2个字节的汉字内码>]

三种结构对照表的性能比较

对照表结构	码本大小	平均检索次数
等长结构	20902×4＝83608	$[\log_2 20902]-1=14$ (二分法)
变长结构	20902×2＋417×（2＋1）＝43055	（417＋1）/2＝209
索引结构	20902×2＋417×（2＋2＋1）＝43889	$\log_2 417-1=8$

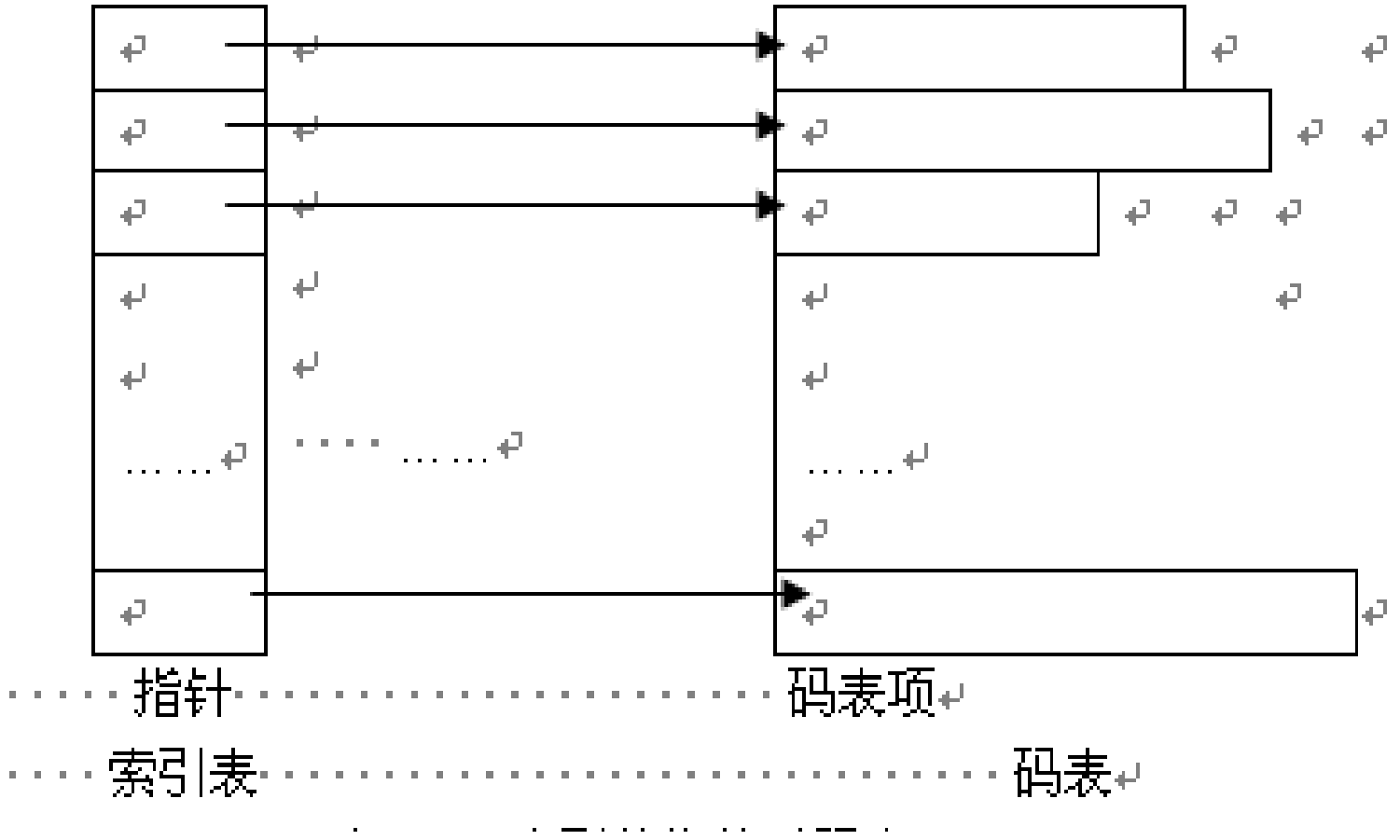
八、稀疏索引结构对照表

- 前面的索引表中，每个表项对应于一个输入码，称为稠密索引结构；
- 稀疏索引结构就是省略了索引表表项中的输入码部分，而索引表的表项则对应于整个编码空间，也就是说，即使这个输入码没有对应的汉字，则仍有其对应的表项存在于索引表中：

- 例如，某个输入法的码元集合为26个字母，码长为4，则其对照表有26×26×26×26＝456976个表项；
- 表项按照其对应的输入码顺序来排列，也就是从“aaaa”开始，一直排到“zzzz”为止；
- 假如某个输入码是$S_1S_2S_3S_4$，我们可以用下列公式计算出该输入码所对应的索引表表项的序号i：

$$i = (S_1\text{-'a'}) \times 26^3 + (S_2\text{-'a'}) \times 26^2 + (S_3\text{-'a'}) \times 26 + (S_4\text{-'a'})$$

- 把序号乘以索引表表项的长度就得到相应的码表指针的地址。注意：不需要检索。


指针
码表项
索引表
码表

- 存在问题：
  - 可能会导致索引表过大；
  - 例如，在上面所举的例子中，如果采用稀疏索引结构的对照表，每个表项长度为2个字节，那么索引表长度为456976×2字节，约要占900K字节的空间，显然太大了；
- 为了节省空间，可以采用部分索引的技术，即取输入码中的部分码元进行稀疏索引。

- 假设码元集合中的码元数为m，输入码的长度为n，每个索引表项的长度为2（假设指针大小为2个字节），如果要对输入码的i（i≤n）个码元作索引，索引表的长度L为：$L=m^i\times 2$
- 则有 $i=\log_m(L/2)$
- 如果把L的大小定在30k左右，当采用数字编码时m=10，则i=4；当采用字母编码时m=26，则i=3；
- 如果把L的大小定在2k左右，当采用数字编码时m=10，则i=3；当采用字母编码时m=26，则i=2；

稀疏索引结构对照表举例

- 仍以拼音码为例，采用2字节压缩拼音，其结构描述如下：

<对照表>::=<索引表><码表>

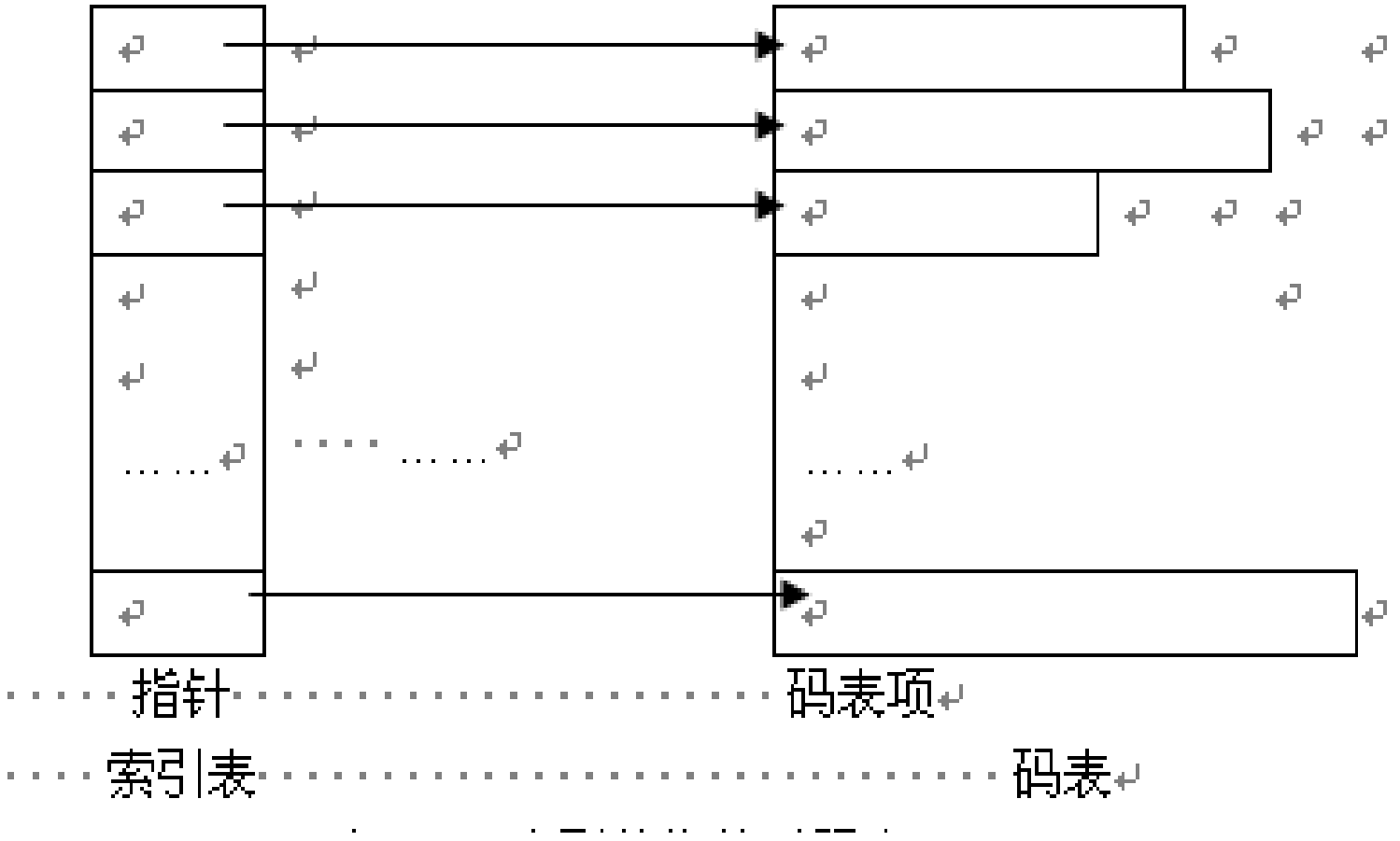
<索引表>::=<表项>[<表项>]

<表项>::=<指针>

<指针>::=<2个字节的整数>

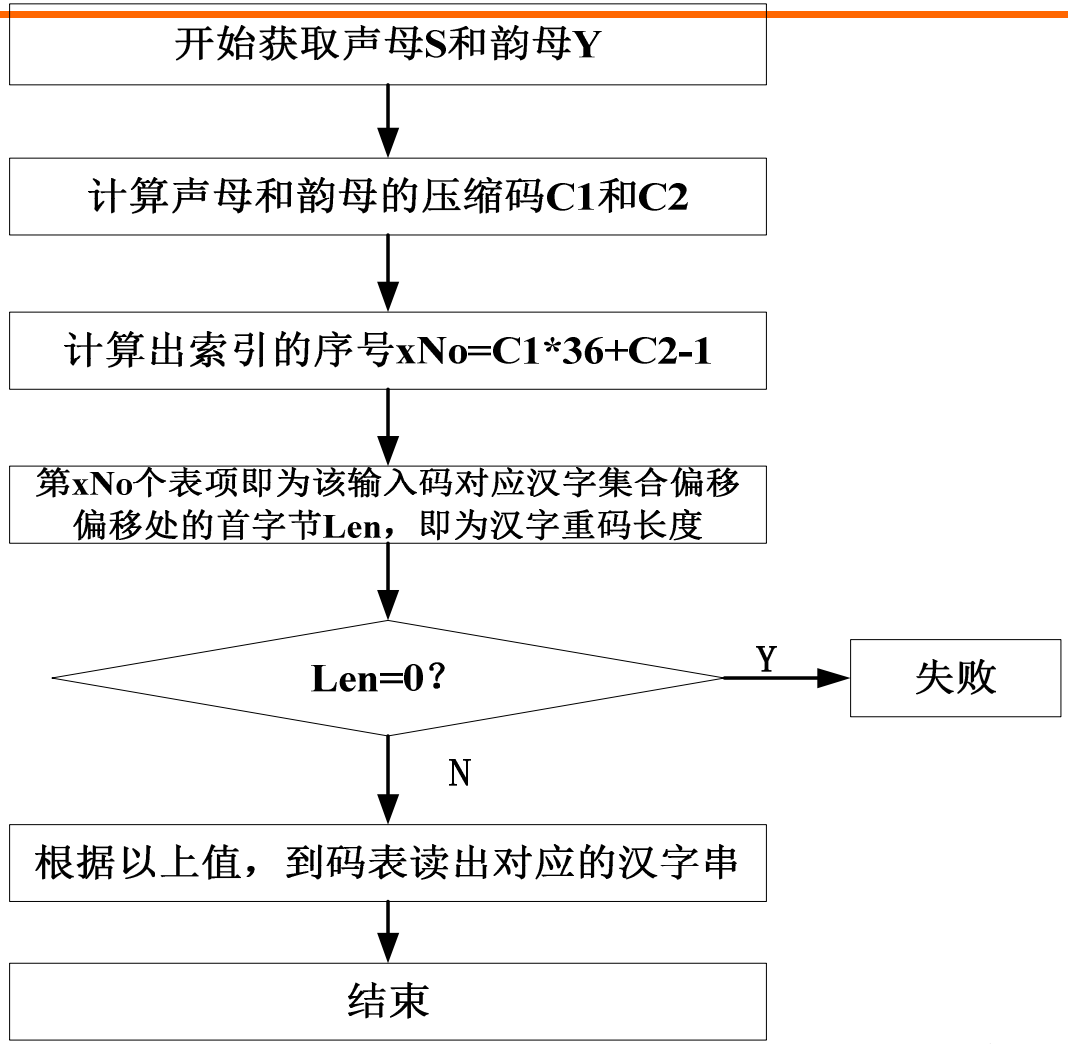
<码表>::=<1个字节汉字集长度><重码汉字>

<重码汉字>::=<2个字节的汉字内码>[< 2个字节的汉字内码>]


指针
码表项
索引表
码表

- 此结构索引表的大小为：
  - 24×36×(2+1)＝2592个字节
- 采用的是稀疏索引，每个声母和韵母的组合都包含在索引表中
  - 索引表有864个表项，比前面例子的417个表项要多一倍以上；
  - 不存在的拼音声韵组合也有其对应的索引表项，如：“cei”、“cua”、“quang”等；
- 索引表的表项按照拼音字母顺序排列。

相关算法


开始获取声母S和韵母Y
计算声母和韵母的压缩码C1和C2
计算出索引的序号xNo=C1*36+C2-1
第xNo个表项即为该输入码对应汉字集合偏移
偏移处的首字节Len，即为汉字重码长度
Len=0？
Y
失败
N
根据以上值，到码表读出对应的汉字串
结束

两种索引结构对照表性能比较

- 稠密索引结构对照表：
  - 占用空间： (20902×2)+(417×5)=43889字节；
  - 平均查找次数8次；
- 稀疏索引结构对照表：
  - 占用空间：(20902×2)+(864×3)=44396字节；
  - 平均查找次数0次，采用计算定位。