

第七章 实用程序设计

- 7.1 线性表的检索程序
- 7.2 排序程序
- 7.3 串操作指令与加密解密程序



School of computer, Aiping XU



7.1 线性表的检索程序

检索：就是在数据结构中查找满足某种条件的记录。
常用的方法有：

- 顺序检索 ➡
- 二分法检索 ➡
- 分块检索

- ① 要求线性表分成若干块，块与块之间必须排序
- ② 建立块的索引表
- ③ 检索时分两步：先在索引表中查找，再到相应块中顺序检索

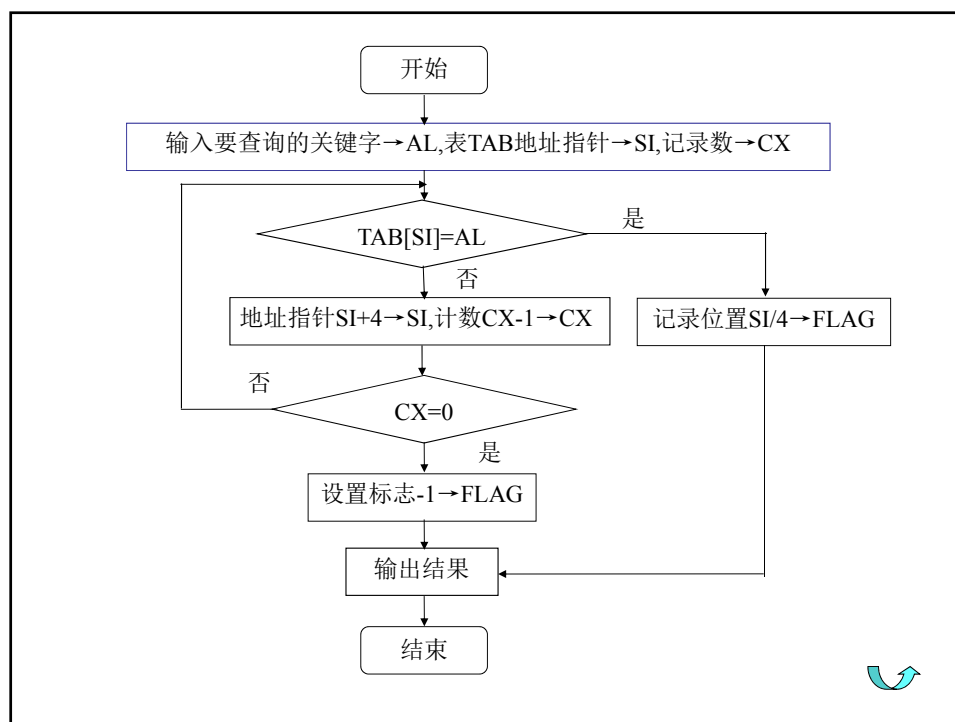


顺序检索：就是用待查的关键字与表中各记录的关键字逐个比较，直到找到满足条件的记录则检索成功，或找遍所有记录均不满足条件则检索失败。

- 例7.1 在表中查找某字符关键字，如果查到，记录该关键字出现在表中的第几项，否则用不同于表中元素的值(例如-1)作为标志。表中的每一个记录占四个字节，关键字占一个字节，其他信息占三个字节。

分析

- ✓ 要定义表，表长，记录检索结果的标志以及提示信息。
- ✓ 每个记录占四个字节，所以每比较一个记录指针加4。



例7.1源程序清单（1）

- .386
- STACKS SEGMENT USE16 STACK 'STACK'
- DB 256 DUP(?)
- STACKS ENDS
- DATAS SEGMENT USE16
- TAB DB '3F23', '1M22', '7M14', '4F52'
- N = (\$-TAB)/4 ; 表的记录数
- MESSAGE DB 'PLEASE INPUT THE KEY:\$'
- FLAG DW ? ; 存放查询结果
- RESULT DW ? ; 存放检索结果在表中的偏移位置
- BUF DB 6 DUP(?), 13, 10, '\$' ; 显示结果（符号+5位数值位）
- CONST DW 10000,1000,100,10,1
- DATAS ENDS
- CODES SEGMENT USE16
- ASSUME CS:CODES, DS:DATAS, SS:STACKS
- START: MOV AX, DATAS
- MOV DS, AX

例7.1源程序清单（2）

- **CALL FINDP** ; 调用顺序检索子程序，检索结果送往FLAG
- MOV DL, 0DH ; 显示回车符
- MOV AH, 2
- INT 21H
-
- MOV DL, 0AH ; 显示换行符
- MOV AH, 2
- INT 21H
-
- MOV AX, FLAG
- MOV RESULT, AX ; 检索结果送RESULT
- **CALL CVD16** ; 调用二进制转换成十进制子程序，结果送往BUF
- LEA DX, BUF ; 显示输出结果
- MOV AH, 9
- INT 21H
- MOV AH, 4CH
- INT 21H

例7.1源程序清单（3）

```

■ 子程序名: FINDP
■ 功能描述: 在表中顺序检索, 找到对应关键字所在表中的位置
■ 出口参数: FLAG中存放查询结果
■ FINDP PROC
■     LEA DX, MESSAGE ; 输出提示信息
■     MOV AH, 9
■     INT 21H
■     MOV AH, 1        ; 输入关键字
■     INT 21H
■     MOV SI, 0        ; 指针指向表中第一个记录的关键字
■     MOV CX, N        ; 表中的记录数作为循环控制次数
■     LOOPS: CMP AL, TAB[SI] ; 在表中顺序查找和输入KEY相同的记录
■             JE FOUND
■             ADD SI, 4
■             LOOP LOOPS
■             MOV FLAG, -1 ; 未找到, 置标志-1
■             JMP EXIT
■     FOUND: MOV CL, 2
■             SHR SI, CL    ; SI/4→找到记录在表中相对位置
■             MOV FLAG, SI
■     EXIT: RET
■ FINDP ENDP

```

例7.1源程序清单（4）

```

■ ; 子程序名: CVD16
■ ; 功能描述: 二进制转换成十进制子程序
■ ; 入口参数: RESULT存放等转换的二进制数
■ ; 出口参数: BUF存放转换后的十进制数
■ CVD16 PROC
■     MOV AX, RESULT
■     OR AX, AX        ; 判断正负号
■     JNS PLUS
■     NEG AX
■     MOV BUF, '-'
■     JMP SHORT CVD
■ PLUS: MOV BUF, '+'
■ CVD:  MOV SI, 1        ; 保存结果的缓冲区指针
■       MOV CX, 5        ; 设置转换的次数
■       LEA BX, CONST    ; 二转十常数表首址

```

例7.1源程序清单（5）

```

■      CWD
■      IDIV WORD PTR[BX] ; 商->AX, 余数->DX
■      ADD AL, 30H
■      MOV BUF[SI], AL
■      INC SI
■      MOV AX, DX
■      ADD BX, 2          ; 指向常数表的下一项
■      LOOP CVDL
■      RET
■  CVD16 ENDP
■  CODES ENDS
■      END START
■  运行结果：如KEY=4，则输出显示：+00003

```



二分法检索：首先用要检索的关键字与中间位置的记录的关键字值比较，这个中间记录把线性表分成了两个子表，比较结果如果相等则检索完成，否则再根据要检索的关键字与中间记录关键字的大小确定下一步检索在哪个子表中进行。这样递归进行下去，直到找到满足条件的记录，或者确定表中没有这样的记录。

■ **要求检索前表中记录已经按关键字的大小排序**

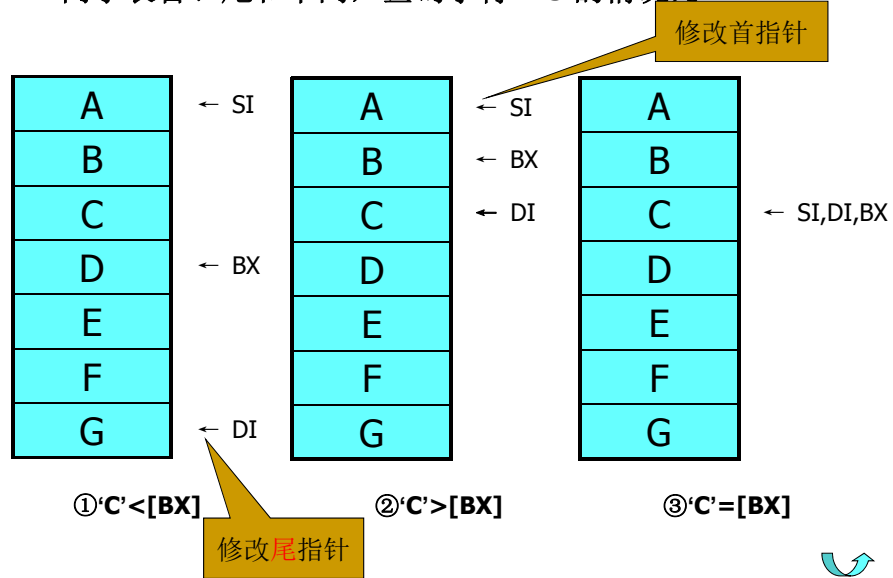
● 例7.2 用二分查找法编制查找字符表中某个字符的程序

分析

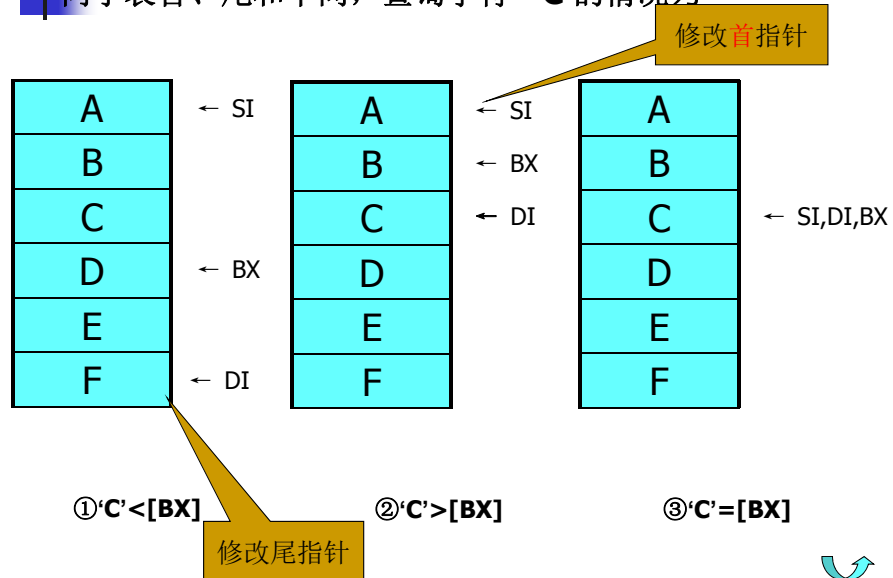
- ✓ 定义表，表长以及检索结果的信息。
- ✓ 两个指针分别指向子表的首和尾，确定子表范围
- ✓ 首指针大于尾指针表示检索失败

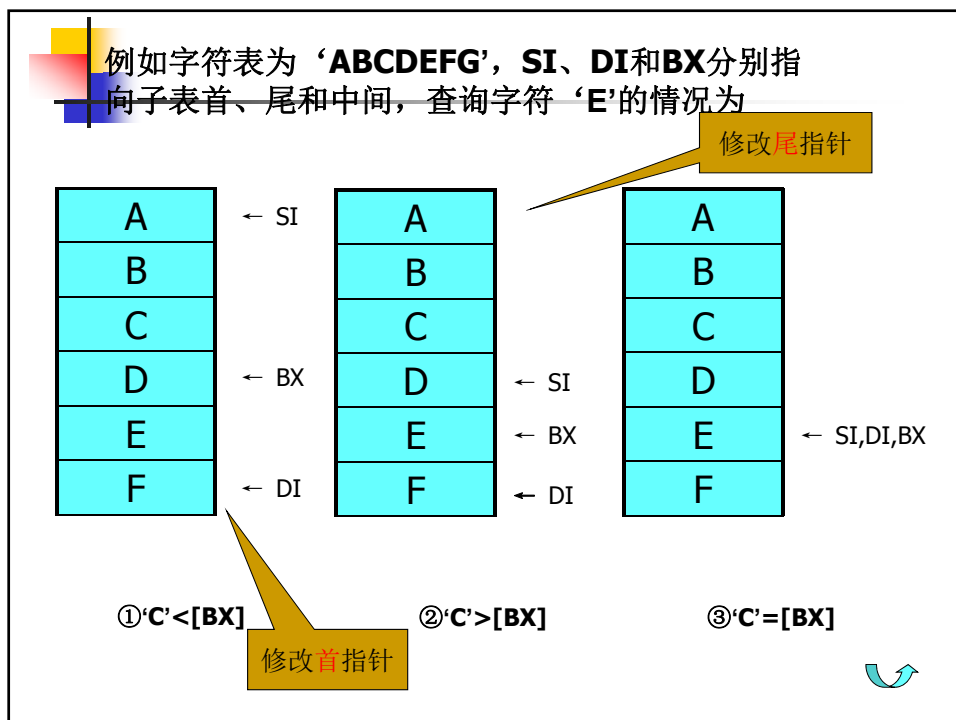
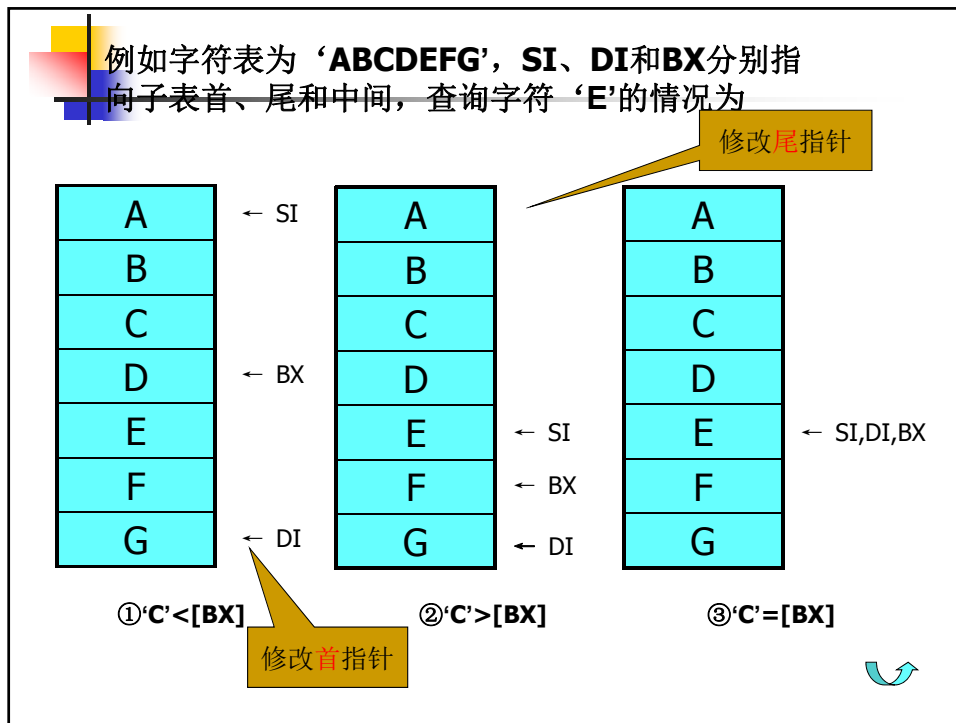


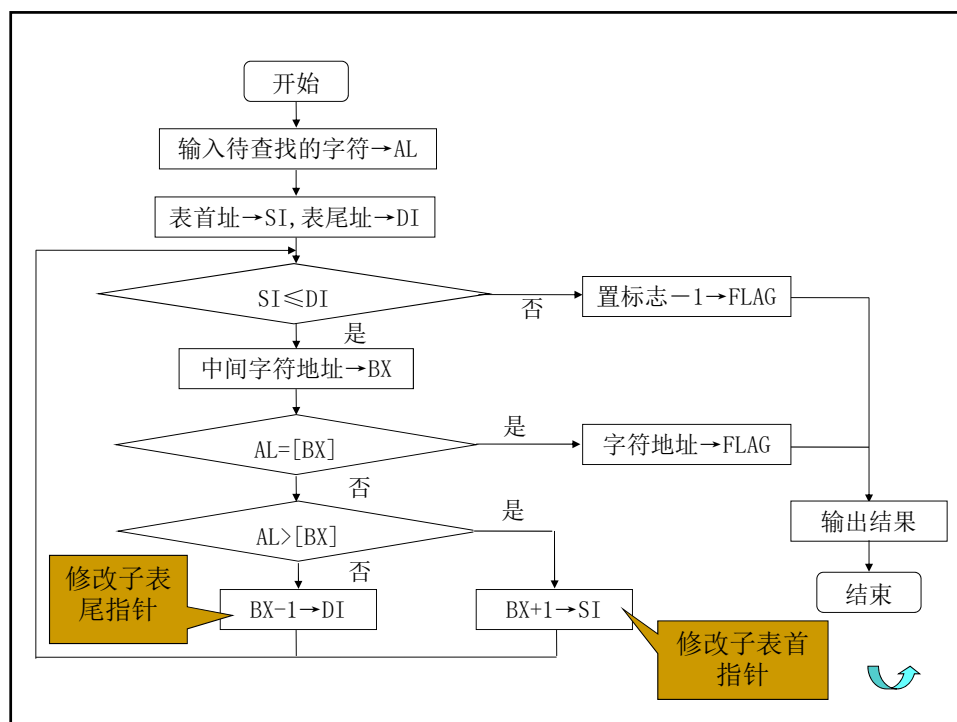
例如字符表为 'ABCDEFGG', SI、DI和BX分别指向子表首、尾和中间, 查询字符 'C'的情况为



例如字符表为 'ABCDEF', SI、DI和BX分别指向子表首、尾和中间, 查询字符 'C'的情况为







- 设SI=2000H DI=2006 ; 7个数
- $BX = (2000H + 2006H) / 2 = 2003H$

- 设SI=2000H DI=2005 ; 6个数
- $BX = (2000H + 2005H) / 2 = 2003H$

例7.2 源程序清单 (1)

```

■ INOUTF MACRO X,Y           ; 定义输入输出的宏
■     MOV X,Y
■     INT 21H
■     ENDM
■ STACKS SEGMENT PARA STACK 'STACK'
■     DB 256 DUP(?)
■ STACKS ENDS
■ DATAS SEGMENT PARA PUBLIC 'DATA'
■     PROMPT DB 'PLEASE INPUT THE KEYWORD:$'
■     TABLE DB 'ABCDEFGH'
■     COUNT EQU ($-TABLE)-1    ; 表的记录数减1
■     KEYW DW ?                ; 存放KEYWORD
■     OKFIND DB ?, 'Is FOUND!', 13, 10, '$'
■     NOFIND DB ?, 'Is NOT FOUND!', 13, 10, '$'
■     FLAG DW ?                ; 存放标志
■ DATAS ENDS
■ CODES SEGMENT PARA PUBLIC 'CODE'
■     ASSUME CS:CODES, DS:DATAS, SS:STACKS
■ START: MOV AX, DATAS
■     MOV DS, AX

```

例7.2源程序清单 (2)

```

■ LEA DX, PROMPT
■ INOUTF AH, 9                ; 提示输入字符
■ INOUTF AH, 1                ; 提示输入要查找的字符
■ MOV KEYW, AL                ; 保存要查找的字符
■ MOV OKFIN, AL
■ MOV NOFIN, AL
■ MOV AH, 2                   ; 显示回车换行符
■ INOUTF DL, 13
■ INOUTF DL, 10
■ CALL SEARCHP                ; 调用二分查找子程序
■ INOUTF AH, 9
■ MOV AX, 4C00H
■ INT 21H

```

例7.2源程序清单（3）

```

; 子程序名: SEARCHP
; 功能描述: 单字节字符串中查找关键字的二分查找子程序
; 入口参数: KEYW存放要查找的字符
; 出口参数: DX中存放显示查找信息的首地址
SEARCHP PROC
    MOV     SI, OFFSET TABLE    ; 表首址送SI
    MOV     DI, COUNT
    ADD     DI, SI               ; DI中存放表末址
    MOV     AL, KEYW
LEFT:    CMP     SI, DI
    JG      NOTFIND
    MOV     BX, SI
    ADD     BX, DI
    SHR     BX, 1                ; 计算出中间地址
    CMP     AL, [BX]
    JZ      FOUND
    JG      RIGHT                ; 如果大于, 在表的后半部分查找
    DEC     BX                   ; 如果小于, 在表的前半部分查找
    MOV     DI, BX
    JMP     LEFT

```

例7.2源程序清单（4）

```

RIGHT:   INC     BX
    MOV     SI, BX
    JMP     LEFT
FOUND:   MOV     FLAG, BX
    LEA     DX, OKFIND
    JMP     EXIT
NOTFIND: MOV     FLAG, -1
    LEA     DX, NOTFIND
EXIT:    RET
SEARCHP ENDP
END START

运行结果: 如输入B, 查找结果为B Is FOUND!

```





7.2 排序程序

排序：就是将记录按排序码不增或不减的次序排列起来。常用的方法有：

- 插入排序

每步将一个待排序的记录，按其排序码值的大小插到前面已经排序的表中适当位置，直到全部插入为止。

- 选择排序

每步从待排序的记录中选出排序码最小的记录，顺序放在已排序的记录序列的最后，直到全部排完。



- 交换排序



两两比较待排序记录的排序码，并交换不满足顺序要求的偶对，直到全部满足为止。

- 归并排序

把待排序的表分成若干个子表，每个子表内的记录是排序的，将这些已排序的子表进行合并，得到完全排序的表。



● 例7.3 编写将数组ARRAY中的n个数据，按递增顺序排列在原数组中的程序SORT.ASM。

分析

- ✓ 本例选用交换排序中的一种冒泡排序
- ✓ 先比较第1个数和第2个数，如果第1个数大于第2个数，则两数交换，否则不交换；然后比较第2个数和第3个数，按同样规则决定是否交换，重复此过程直到处理完第n-1个数和第n个数。这样n-1次比较和交换的过程称为一次冒泡，这一步将最大的数传到最后一个位置。
- ✓ 在一次冒泡中，用一个标志表示本次冒泡是否有交换，如果没有则表示已经达到排序要求，可停止处理，否则重复执行冒泡过程。

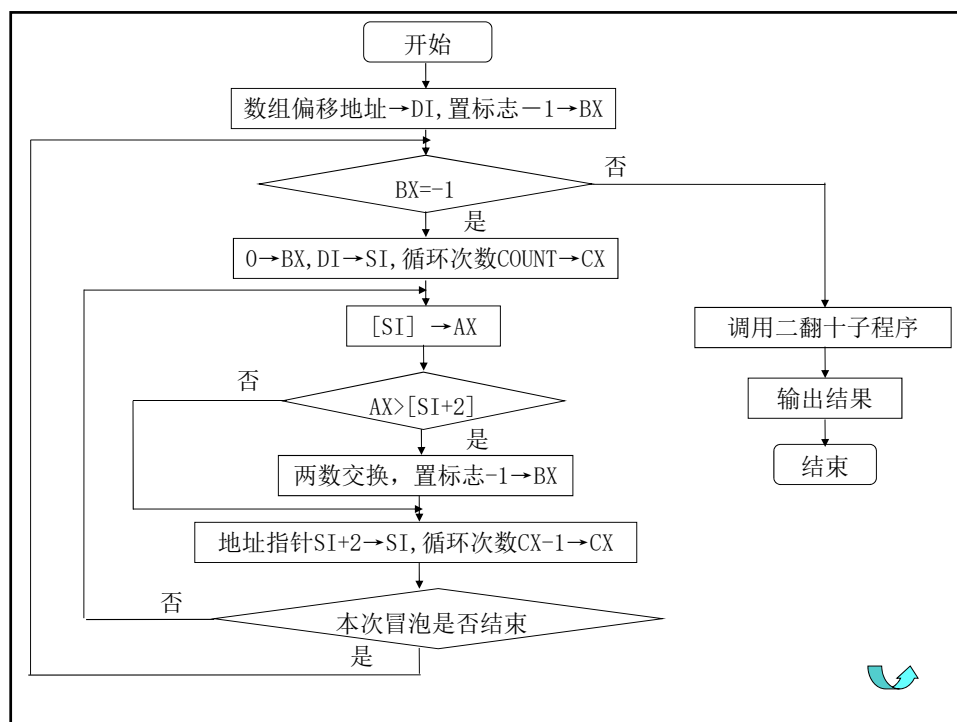


例如待排序的数据为8, 2, 1, 5, 0, 其冒泡排序的过程为：

第一次冒泡	8	2	2	2	2
	2	8	1	1	1
	1	1	8	5	5
	5	5	5	8	0
	0	0	0	0	8
第二次冒泡	2	1	1	1	1
	1	2	2	2	2
	5	5	5	0	0
	0	0	0	5	5
	8	8	8	8	8
第三次冒泡	1	1	1	1	1
	2	2	0	0	0
	0	0	2	2	2
	5	5	5	5	5
	8	8	8	8	8
第四次冒泡	1	0	0	0	0
	0	1	1	1	1
	2	2	2	2	2
	5	5	5	5	5
	8	8	8	8	8

第五次冒泡无交换发生，则结束





例7.3 源程序清单 (1)

```

STACKS SEGMENT PARA STACK 'STACK'
    DB 256 DUP(?)
STACKS ENDS
DATAS SEGMENT PARA PUBLIC 'DATA'
    ARRAY DW 8, 2, 1, 5, 0 ; 定义数组
    LEN = ($-ARRAY)/2 ; 数组元素个数
    COUNT DW LEN-1
    RESULT DW ? ; 待转换二进制数
    BUF DB 6 DUP(?), 2 DUP(','), '$' ; 转换后待显示的十进制数
    CONST 10000, 1000, 100, 10, 1
DATAS ENDS
CODES SEGMENT PARA PUBLIC 'CODE'
    ASSUME CS:CODES, DS:DATAS, SS:STACKS
START: MOV AX, DATAS
    MOV DS, AX
    CALL SORTP ; 调用排序子程序
  
```

例7.3 源程序清单（2）

```

MOV CX, LEN
MOV SI, OFFSET ARRAY
OUTSTR: MOV AX, [SI] ; 取排序之后首元素
MOV RESULT, AX
PUSH CX
ADD SI, 2
PUSH SI
CALL CVD16 ; 调用二转十子程序（同例7.1）
LEA DX, BUF
MOV AH, 9
INT 21H
POP SI
POP CX
LOOP OUTSTR
MOV AX, 4C00H
INT 21H

```


例7.3 源程序清单（3）

```

; 子程序名: SORTP
; 功能描述: 实现数组冒泡排序
; 入口参数: ARRAY待排序数组
; 出口参数: ARRAY已排序数组
SORTP PROC
MOV DI, OFFSET ARRAY
MOV BX, -1 ; 设置标志
LOOPOUT: CMP BX, -1
JNE SORTEND ; 标识不为-1则排序完成
XOR BX, BX
MOV CX, COUNT ; 每次冒泡中比较次数
MOV SI, DI ; 数组偏移地址送SI
LOOPIN: MOV AX, [SI] ; 两两比较
CMP AX, [SI+2]

```

例7.3源程序清单（4）

- **JLE NONCHANGE** ; 如果小于等于, 则不交换
- **XCHG [SI+2], AX**
- **MOV [SI], AX**
- **MOV BX, -1** ; -1表示此次冒泡有交换
- **NONCHANGE: ADD SI, 2**
- **LOOP LOOPIN** ; 判断本次冒泡结束否
- **JMP LOOPOUT**
- **SORTEND: RET**
- ... ; 二转十子程序代码
- **CODES ENDS**
- **END START**
- 运行结果: +00000 +00001 +00002 +00005 +00008 

7.3 串操作指令与加解密程序

7.3.1 串操作指令的共性

- ◆ 源串偏移地址由SI或ESI指出, 默认段址在DS中;
- ◆ 目的串偏移地址由DI或EDI指出, 段址只能在ES中;
- ◆ 隐含操作数为AL/AX/EAX;
- ◆ 标志位DF=0, 操作数地址自动增量;
标志位DF=1, 操作数地址自动减量;
- ◆ CX/ECX作为重复计数器;
- ◆ 指令操作码最后两个字符是SB/SW/SD, 表示
字节/字/双字串操作。



串操作指令的重复前缀

REP

;CX-1→≠0则重复,直到CX=0为止
;ECX-1→≠0则重复,直到ECX=0为止

REPE/REPZ

; CX-1→≠0且ZF=1则重复,直到CX=0或ZF=0为止
; ECX-1→≠0且ZF=1则重复,直到ECX=0或ZF=0为止

REPNE/REPNZ

; CX-1→≠0且ZF=0则重复,直到CX=0或ZF=1为止
; ECX-1→≠0且ZF=0则重复,直到ECX=0或ZF=1为止

◆ 1. 串传送指令

MOVS OPD,OPS

MOVS ; 传送字节串

MOVS ; 传送字串

MOVS ; 传送双字串

将DS: SI/ESI指向的源串中的一个字节/字/双字传送至ES: DI/EDI指定的目的串, 并自动修改指针, 不影响标志位。



例7.4 将以SOURCE为首址的字节存储区中存放的字符串传送到以DEST为首址的字节存储区，串长为50。

分析	MOV	AX, DATA ; 假设仅一个数据段,
✓ 设置段基址DS和ES	MOV	DS, AX ; 段名DATA
✓ 设置偏移地址SI/ESI和DI/EDI	MOV	ES, AX
	LEA	SI, SOURCE ; 源串首址送SI
	LEA	DI, DEST ; 目的串首址送DI
✓ 设置标志位	MOV	CX, 50 ; 串长送CX
✓ 设置重复计数	CLD	; DF=0
	REP	MOVSB ; 传送50个字符

重复前缀REP：无条件重复CX指定的次数



```

MOV    AX, DATA ; 假设仅一个数据段,
MOV    DS, AX    ; 段名DATA
MOV    ES, AX
LEA    SI, SOURCE ; 源串首址送SI
LEA    DI, DEST  ; 目的串首址送DI
MOV    CX, 50    ; 串长送CX
L1:    MOV    AL, [SI]
        MOV    [DI], AL
        INC    SI
        INC    DI
        LOOP   L1
        ;CLD                ; DF=0
        ; REP MOVSB

```

◆ 2. 串比较指令

CMPS OPD,OPS

CMPSB ; 比较字节串

CMPSW ; 比较字串

CMPSD ; 比较双字串

将源串减目的串，根据结果修改标志位ZF、OF、AF、CF、PF、SF，并自动修改指针，但不改变任何操作数。

● 例7.5 对两个字类型的串STRING1和STRING2进行比较，如果相等，0送BX，否则0FFFFH送BX。设串长为COUNT。

重复前缀
REPE/REPZ:
CX/ECX≠0且
ZF=1时重复,
否则终止

```

LEA    SI, STRING1 ; 源串首址送SI
LEA    DI, STRING2 ; 目的串首址送DI
MOV    CX, COUNT
CLD
REPZ   CMPSW       ; CX≠0且ZF=1时继续比较
JNZ    NEQU        ; ZF=0则不相等
MOV    BX, 0        ; 相等BX置0
JMP    EXIT
NEQU:  MOV    BX, 0FFFFH ; 不相等BX置0FFFFH
EXIT:  MOV    AX, 4C00H
INT    21H

```



◆ 串搜索指令

SCAS **OPD**

SCASB ; 搜索字节串

SCASW ; 搜索字串

SCASD ; 搜索双字串

将隐含操作数AL/AX/EAX减ES: DI/EDI指向的目的串中的一个字节/字/双字，根据结果修改标志位ZF、CF、SF、OF、AF、PF，并自动修改指针，但不改变任何操作数。



● 例7.6 在长度为COUNT的字符串STRING中搜索字符'\$'。

分析

✓ 设置段基址DS

✓ 设置目的串偏移地址

DI/EDI

✓ 设置隐含操作数

✓ 设置标志位

✓ 设置重复计数

```
LEA    DI, STRING    ; 串首址送DI
MOV    AL, '$'        ; 要搜索的字符送AL
MOV    CX, COUNT      ; 串长送CX
CLD
REPNE  SCASB ; CX≠0且ZF=0时继续搜索
JZ     OK             ; 串中搜索到字符转OK
```

重复前缀REPNE/REPNZ:
CX/ECX≠0且ZF=0时重复,
否则终止



4. 从源串取数指令

LDS OPS

LODSB ; 从字节串取数

LODSW ; 从字串取数

LODSD ; 从双字串取数

将DS: SI/ESI指向的源串中的一个字节/字/双字送隐含操作数AL/AX/EAX，并自动修改指针，不影响标志位。

LODSW 相当: `MOV AX, [SI]`
 `ADD SI, 2`

LODSD 相当: `MOV EAX, [ESI]`
 `ADD ESI, 4`

此指令不跟前缀

5. 往目的串存数指令

STOS OPD

STOSB ; 往字节串存数

STOSW ; 往字串存数

STOSD ; 往双字串存数

将隐含操作数AL/AX/EAX送ES: DI/EDI指向的目的串，并自动修改指针，不影响标志位。

STOSW 相当: `MOV [SI], AX`
 `ADD SI, 2`

STOSD 相当: `MOV [ESI], EAX`
 `ADD ESI, 4`



- 例7.7 将串长为10的双字串BUF初值均置为0。

```

MOV    AX, DATA
MOV    ES, DATA
LEA    EDI, BUF      ; 目的串首址送EDI
MOV    ECX, 10       ; 串长送ECX
MOV    EAX, 0
CLD
REP    STOSD         ; 传送10个双字

```



7.3.2 加密解密程序

- 例7.9 将从键盘输入的一串数字变成密文后，存入内存，再将该密文进行解密，并显示译文。

分析

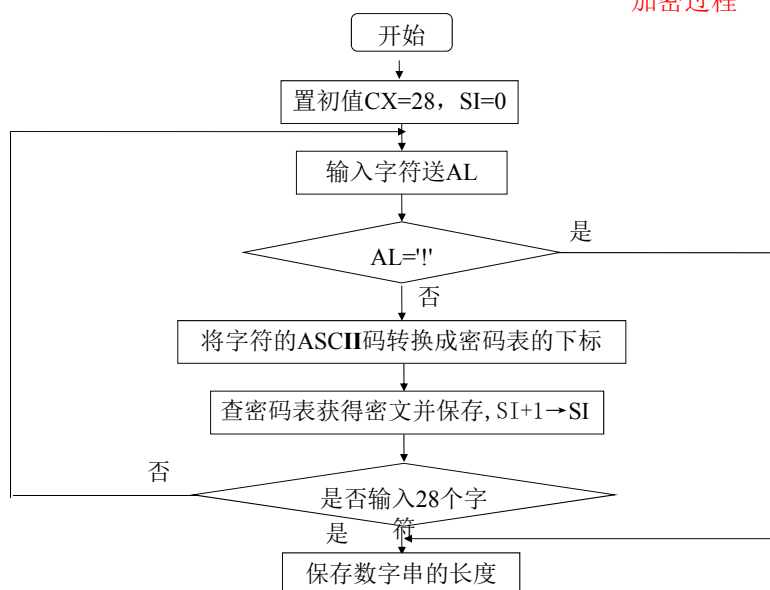
- ✓ 要定义密码表和解密表
- ✓ 定义明文的结束标志

本例中设定明文最大长度为28，若不足28则以“!”结束

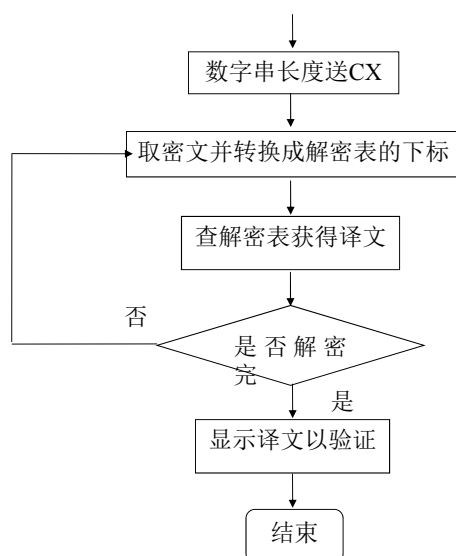
- ✓ 加密过程中，将输入的数字转化为该数字在密码表中的偏移量，然后通过查密码表得到密文
- ✓ 在解密过程中，根据密码查找解密表，得到译文



加密过程



解密过程



例7.9源程序清单（1）

```

.386
■ IO      MACRO X, Y          ; 定义输入输出字符串的宏
■          PUSH AX
■          LEA  DX, X
■          MOV  AH, Y
■          INT  21H
■          POP  AX
■          ENDM
■ STACKS SEGMENT PARA STACK USE16 'STACK'
■          DB  512 DUP(?)
■ STACKS ENDS
■ DATAS SEGMENT PARA PUBLIC USE16 'DATA'
■ COUNT   = 28
■          ; 0123456789ABCDEF
■ MLAB    DB  '7501264398CFAEBD' ; 密码表
■ JLAB    DB  '2347615098CEAFDB' ; 解密表
■ MCODE   DB  COUNT DUP(' '), 13, 10, '$' ; 存放密文
■ JCODE   DB  COUNT DUP(' '), 13, 10, '$' ; 存放译文
■ MSG     DB  'PLEASE INPUT CODE,"!" IS EOJ', 13, 10, '$'
■ NUMBER  DW  0 ; 存放密文长度
■ CR_LF   DB  13, 10, '$'
■ DATAS ENDS

```



例7.9源程序清单（2）

```

■ CODES SEGMENT PARA PUBLIC USE16 'CODE'
■          ASSUME CS:CODES, DS:DATAS, SS:STACKS
■ START: MOV  AX, DATAS
■          MOV  DS, AX
■          CALL MP      ; 调用加密子程序
■          IO   CR_LF, 9 ; 显示回车换行
■          IO   MCODE, 9 ; 显示密文
■          CALL JP      ; 调用解密子程序
■          IO   JCODE, 9 ; 显示译文
■          MOV  AX, 4C00H
■          INT  21H

```



; 子程序名: MP ; 功能描述: 加密并保存密文 ; 出口参数: MCODE存放密文
 MP PROC
 IO MSG, 9
 MOV CX, COUNT
 MOV SI, 0
 INPUT: MOV AH, 7 ; 输入密码, 每输入一个, 显示一个星号
 INT 21H ; 直到输入一个! 号为止
 PUSH AX
 MOV DL, '*'
 MOV AH, 2
 INT 21H
 POP AX
 CMP AL, '!'
 JE EXIT
 SUB AL, 30h
 CMP AL, 9
 JBE L1
 SUB AL, 7
 L1: LEA BX, MLAB
 XLATB
 MOV MCODE[SI], AL
 INC SI
 LOOP INPUT
 EXIT: MOV NUMBER, SI
 RET
 MP ENDP

例7.9源程序清单 (4)

; 子程序名: JP ; 功能描述: 解密 ;
 ; 入口参数: NUMBER密文长长, MCODE密文 ; 出口参数: JCODE存放译文
 JP PROC
 CLD
 MOV CX, NUMBER
 LEA SI, MCODE
 LEA DI, JCODE
 LEA BX, JLAB
 J: LODSB ; 取密文
 SUB AL, 30H
 CMP AL, 9
 JBE L2
 SUB AL, 7
 L2: XLATB ; 查密码表获得译文
 STOSB ; 保存译文
 LOOP J
 RET
 JP ENDP
 CODES ENDS
 END START

例7.9源程序清单（5）

- 运行结果：
 - C:>MASM\MPJP.EXT
 - PLEASE INPUT CODE, “!” IS EOJ
 - *****
 - 74467795
 - 19951120



本课程结束

