# 实现电话号码表的增删查改功能

该电话簿可以实现可以添加联系人、删除联系人、寻找联系人和修改联系人的信息。

* 添加联系人的时候是默认添加到电话号码表的末尾；
* 寻找联系人就是对电话号码表进行一次简单的遍历；
* 删除联系人的话，是将最后一个联系人放在要删除的联系人的位置；
* 修改联系人的话，就是寻找联系人和添加联系人功能的结合；

[实现电话号码表的增删查改功能](#实现电话号码表的增删查改功能)  
[一. 具体功能实现：](#一-具体功能实现)  
 [1.0 电话号码表在内存中的结构](#X9475916ce5626cbaf3b0ce996ad8840728ddc86)  
 [1.1 实验程序的整体架构](#X65133114c169c405477c983a820035d3305cf91)  
 [1.2 查询联系人功能](#X377df039d1465db05edd69b27946498ed0197f1)  
 [1.3 添加联系人的功能](#Xe3e8618e37cfbe4116528f640b13efe3c5707dd)  
 [1.4 删除联系人的功能实现](#Xd14f07454f39d03ab7e2f133a8bcdf65cff1ff3)  
 [1.5 修改联系人信息功能实现](#Xa2ad6ec6b4d13a0d1d42a6b1208e05f36ee7ab6)  
 [1.6 输出整个电话表的信息](#Xab97d06c4382da72ba727222403aafc202ad4c6)  
[附录1:输出结果](#附录1输出结果)  
 [添加联系人：](#添加联系人)  
 [输出所有信息：](#输出所有信息)  
 [寻找联系人：](#寻找联系人)  
 [修改联系人信息：](#修改联系人信息)  
 [删除联系人信息：](#删除联系人信息)  
[附表2：具体代码实现](#附表2具体代码实现)

# 一. 具体功能实现：

## 1.0 电话号码表在内存中的结构

电话号码表中联系人的相关联系人的信息在内存中的结构如下如所示：

| 076C：0018 | ..kirito ..17698266260 | -------------->38Bytes |
| --- | --- | --- |
| **076C：003E** | **..Sam ..12345678909** | **-------------->38Bytes** |
| **076C：0064** | **..Kite ..12345343434** | **-------------->38Bytes** |
| **076C：008A** | **..Siri ..12345343434** | **-------------->38Bytes** |

代码中定义：

tel\_tab db 80 dup(20,?,20 dup(' '),32,12,?,12 dup(0),32);存储人名和电话号码

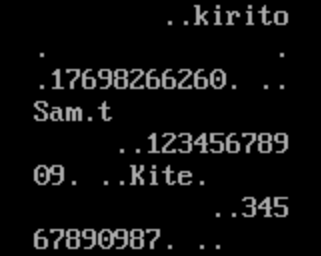


image-20221030150346971

## 1.1 实验程序的整体架构

主函数所做的事情就是一直循环输出功能列表，用户选择相应的功能之后开始执行相关的操作。

该部分很简单：就是首先屏幕会输出一个列表。上面是功能介绍，本次实验的具体功能有：

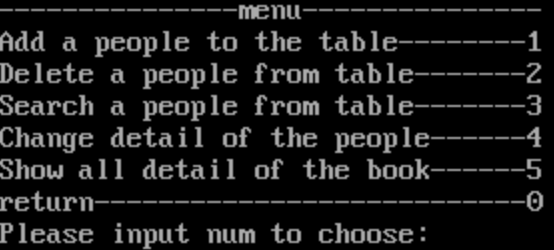


image-20221030151111207

每一次会提示用户输入一个字符，接着会根据输入的具体的字符进行判断而去执行不同的子程序。

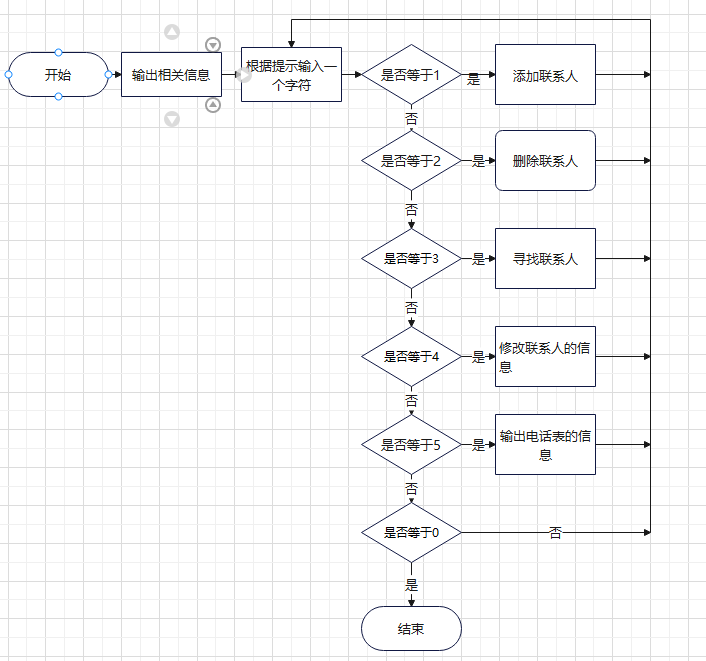


image-20221030152232884

## 1.2 查询联系人功能

首先会提示用户输入要查询的联系人的姓名，本次实验中只提供了根据姓名去寻找指定联系人的功能。

接着遍历即可。

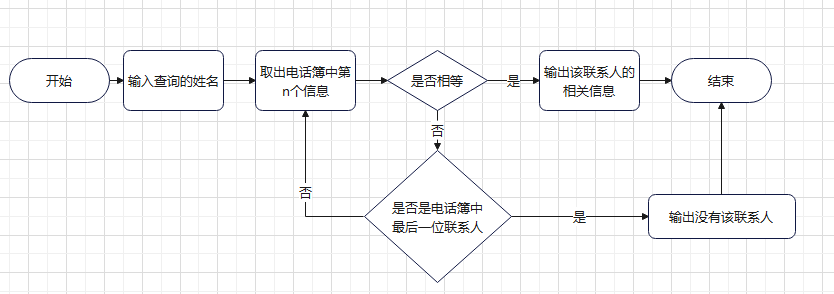


image-20221030152737946

## 1.3 添加联系人的功能

本程序中，其中有一个全局变量来存储电话簿中存储的联系人的总数，很多功能的都是根据这个变量来实现的，除此之外还有一个变量来存储该电话簿最多存储的联系人的总数。

我们添加联系人的时候，首先先确定我们即将添加的联系人准备存储的内存位置，然后将数据存入其中，最后将当前电话簿的联系人的总数加一。

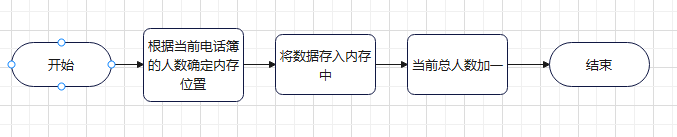


image-20221030153134938

## 1.4 删除联系人的功能实现

该功能实现的时候借助于前面已经造好的轮子，即联系人寻找功能。首先会提示输入要删除的联系人的姓名，接着会在电话簿中遍历寻找该联系人，如果未找到的话，说明该电话簿中不存在该联系人，结束程序。如果找到的话，计算最后一位联系人的内存位置，并且将该联系人的数据放在我们要删除的联系人的内存位置，替换掉，当前总人数减一即可。

注意一点就是，如果联系人总人数为1的话，我们可以直接将当前总人数减一即可完成删除联系人的目的。

相关流程图如下:

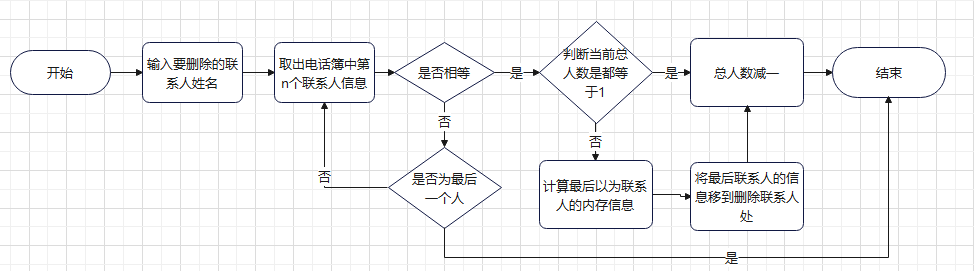


image-20221030154124211

## 1.5 修改联系人信息功能实现

该处功能的实现就是查询联系任何添加联系人的功能的结合，较为简单，暂不展开阐述。

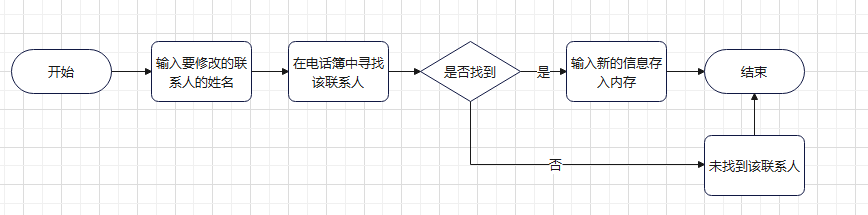


image-20221030154452809

## 1.6 输出整个电话表的信息

根据当前电话簿中的总人数来进行输出和每一个联系人的内存位置的查找。

如果当前人数为0的话，输出电话簿为空，否则就先取出第一个联系人的内存地址，将其中的信息一个一个字符进行输出，电话号码和姓名直接用空格隔开。

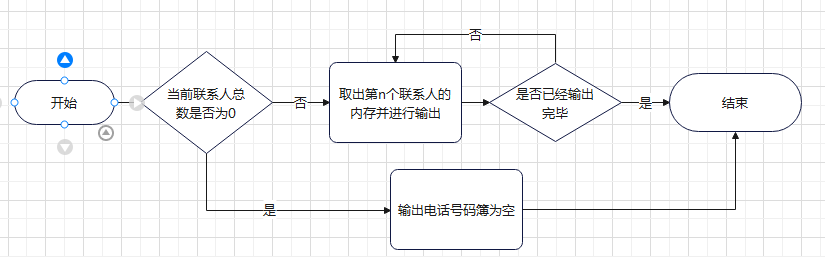


image-20221030154911177

# 附录1:输出结果

## 添加联系人：

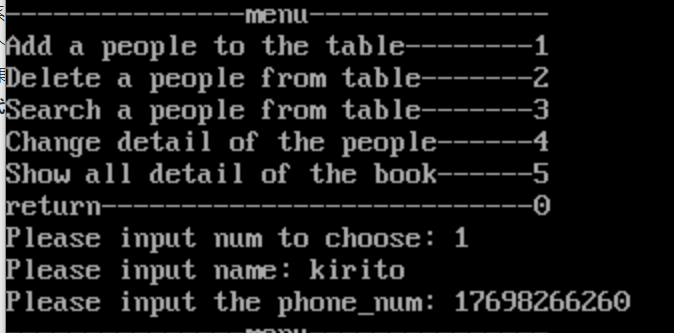


image-20221030155012182

## 输出所有信息：

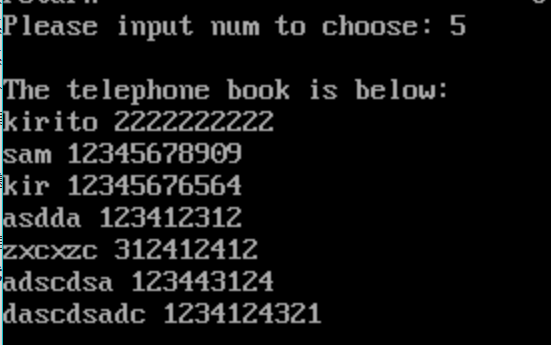


image-20221030155439201

## 寻找联系人：

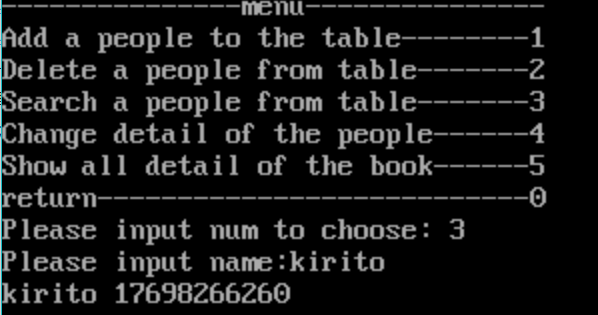


image-20221030155126949

## 修改联系人信息：

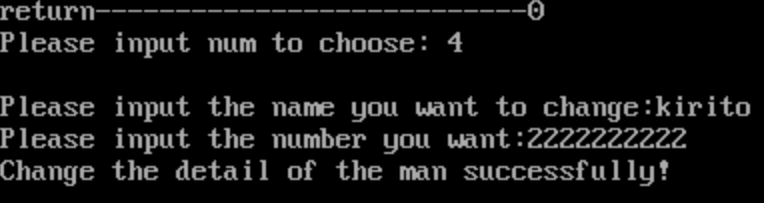


image-20221030155331381

## 删除联系人信息：

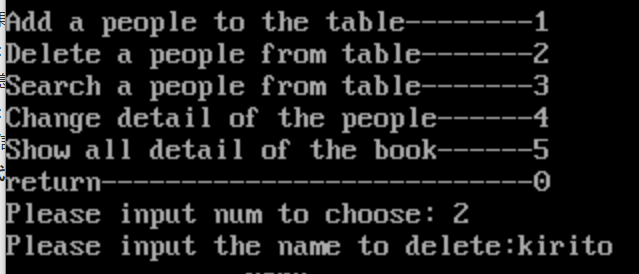


image-20221030155516035

可以从下图看出该联系人的信息已经删除。

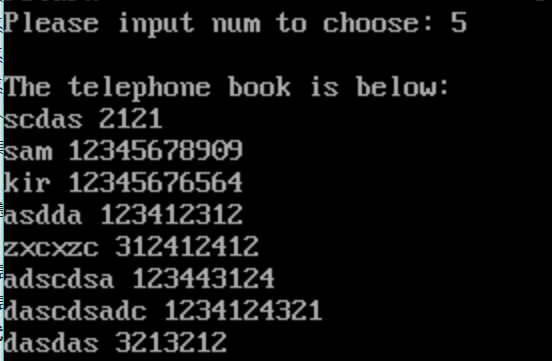


image-20221030155540585

# 附表2：具体代码实现

; 该程序实现了电话簿的基本功能：
  
; 增删查改
  
; 输出整个电话簿
  
assume ds:data,cs:code,ss:stacks
  
data segment
  
 sum dw 0 ;记录当前电话簿中有多少联系人
  
 namemax db 20 ;用来存储临时输入的联系人姓名
  
 \_length db ?
  
 \_name db 20 dup(' ')
  
 tel\_tab db 80 dup(20,?,20 dup(' '),32,12,?,12 dup(0),32);存储人名和电话号码
  
 menu db 13,10,"---------------menu-----------------",13,10
  
 menu1 db "--Add a people to the table--------1",13,10
  
 menu2 db "--Delete a people from table-------2",13,10
  
 menu3 db "--Search a people from table-------3",13,10
  
 menu6 db "--Change detail of the people------4",13,10
  
 menu7 db "--Show all detail of the book------5",13,10
  
 menu4 db "--exit-----------------------------0",13,10
  
 menu5 db "Please input num to choose (0-5): ",'$'
  
 temp db "Please input name: ","$"
  
 tel\_num db 13,10,"Please input the phone\_num(11-digit): ","$"
  
 CRLF db 13,10,'$'
  
 method db "Please input name:",'$'
  
 method\_0 db "Please input the name to delete:",'$'
  
 temp6 db 13,10,"The table is empty!",'$'
  
 temp7 db 13,10,"Failed to find the people!",'$'
  
 temp8 db 13,10,"The telephone book is below:",13,10,'$'
  
 temp9 db 13,10,"Please input the name you want to change:",'$'
  
 temp10 db 13,10,"Please input the number you want:",'$'
  
 temp11 db 13,10,"Change the detail of the man successfully!",13,10,'$'
  
  
data ends
  
stacks segment stack
  
 dw 20 dup(?)
  
stacks ends
  
  
; 定义宏，输出菜单
  
show\_menu macro
  
 mov dx,offset menu
  
 mov ah,09h
  
 int 21H
  
endm
  
  
;定义宏指令，用来清屏用
  
clear macro a,b,c,d
  
 mov al,0
  
 mov bh,7
  
 mov ch,a
  
 mov cl,b
  
 mov dh,c
  
 mov dl,d
  
 mov ah,6
  
 int 10h
  
endm
  
;设置光标在 (y，x)位置，设置光标行列位置
  
post\_c macro y,x
  
 mov bh,0
  
 mov ah,2
  
 mov dh,y
  
 mov dl,x
  
 int 10h
  
endm
  
  
; 定义宏，回车
  
CF macro
  
 mov dx,offset CRLF
  
 mov ah,09h
  
 int 21H
  
endm
  
  
code segment
  
start:
  
 mov ax,data
  
 mov ds,ax
  
 mov ax,stacks
  
 mov ss,ax
  
main\_loop:
  
 clear 0,0,24,79 ; 清屏
  
 post\_c 0,0
  
main\_loop0:
  
 show\_menu ; 输出菜单
  
 mov ah,1 ;从键盘输入数字选择
  
 int 21H
  
 push ax
  
  
 block\_loop:
  
 mov ah,0
  
 int 16h
  
 cmp ah,1ch
  
 jne block\_loop
  
  
 CF ;输出回车
  
 pop ax
  
 xor ah,ah
  
 cmp ax,0030h
  
 je return ;如果输入的是0，则结束程序
  
 cmp ax,0031h
  
 je loop1 ;如果输入的是1，2，3，4,5，则进入子程序
  
 cmp ax,0032h
  
 je loop2
  
 cmp ax,0033h
  
 je loop3
  
 cmp ax,0034h
  
 je loop4
  
 cmp ax,0035h
  
 je loop5
  
 jmp main\_loop
  
  
return:
  
 mov ah,4CH
  
 int 21H
  
loop1:
  
 call add\_man
  
 jmp main\_loop
  
loop2:
  
 call del\_man
  
 jmp main\_loop0
  
loop3:
  
 call sear\_man
  
 jmp main\_loop0
  
loop4:
  
 call change
  
 jmp main\_loop0
  
loop5:
  
 call show\_book
  
 jmp main\_loop0
  
  
;输出所有联系人
  
show\_book proc near
  
 mov dx,offset temp8
  
 mov ah,09h
  
 int 21h
  
  
 push sum
  
 cmp sum,0
  
 je show\_empty ;如果电话簿为空的话，输出相应信息
  
 mov cx,sum
  
  
  
 mov si,offset tel\_tab
  
show\_big:
  
 push cx ;外层循环，遍历每一位联系人
  
 add si,1
  
 mov cl,ds:[si]
  
 xor ch,ch
  
 add si,1
  
 push si
  
show\_name: ;输出姓名
  
 mov dl,ds:[si]
  
 mov ah,02h
  
 int 21H
  
 inc si
  
 loop show\_name
  
  
 pop si
  
 add si,22
  
 mov cl,ds:[si]
  
 xor ch,ch
  
 add si,1
  
 push si
  
  
 mov dl,' '
  
 mov ah,02h
  
 int 21H
  
show\_phonenum: ;输出电话号码
  
 mov dl,ds:[si]
  
 mov ah,02h
  
 int 21H
  
 inc si
  
 loop show\_phonenum
  
 CF
  
  
 pop si
  
 add si,13
  
 pop cx
  
 loop show\_big
  
 jmp show\_ret
  
  
show\_empty:
  
 mov dx,offset temp6
  
 mov ah,09h
  
 int 21h
  
show\_ret:
  
 pop sum
  
 ret
  
show\_book endp
  
  
add\_man proc near ;添加联系人
  
 lea dx,temp
  
 mov ah,09h
  
 int 21H
  
 mov dx,offset tel\_tab
  
 push sum
  
init: ;初始化该联系人存放的内存位置
  
 cmp sum,0
  
 je add\_loop
  
 add dx,38
  
 sub sum,1
  
 jmp init
  
add\_loop:
  
 pop sum
  
 mov ah,0ah
  
 int 21H
  
 push dx
  
  
 mov dx,offset tel\_num
  
 mov ah,09h
  
 int 21H
  
  
 pop dx
  
 add dx,23
  
 mov ah,0ah
  
 int 21h
  
  
 inc sum ;计数器加一
  
 ret
  
add\_man endp
  
  
;删除联系人
  
del\_man proc near
  
 push sum;暂存联系人的人数
  
  
 mov dx,offset method\_0
  
 mov ah,09h
  
 int 21H
  
  
 ;输入要删除的联系人
  
 mov dx,offset namemax
  
 mov ah,0ah
  
 int 21H
  
  
 cmp sum,1
  
 je dele\_brige2
  
  
 ;取出比较的次数
  
 mov cl,\_length
  
 xor ch,ch
  
 ;初始化电话簿的地址
  
 mov si,offset tel\_tab
  
 add si,2
  
 mov di,seg \_name
  
 mov es,di
  
 mov di,offset \_name
  
 mov bx,0 ;计数器，记录我们要删除的联系人的位置
  
 cmp sum,0
  
 je dele\_brige ;如果sum一开始就为0，即为空的，肯定找不到
  
dele\_loop4:
  
 cmp sum,0 ;此时肯定不会满足条件，所以开始寻找
  
 je dele\_brige1 ;由于跳转的位置距离此处太远所以需要跳板
  
 inc bx
  
dele\_cmp:
  
 push bx
  
 mov al,ds:[si]
  
 mov bl,es:[di]
  
 cmp al,bl
  
 pop bx
  
 jne dele\_nomatch
  
 inc si
  
 inc di
  
 loop dele\_cmp
  
 jmp dele\_suce
  
dele\_nomatch:
  
 add si,38
  
 mov di,offset \_name
  
 sub sum,1
  
 jmp dele\_loop4
  
dele\_brige:
  
 jmp dele\_empty
  
dele\_brige1:
  
 jmp dele\_fail
  
dele\_brige2:
  
 jmp dele\_only
  
;查找成功
  
;找到并初始化我们要删除的联系人的位置
  
dele\_suce:
  
 mov cx,bx
  
 sub cx,1
  
 mov si,offset tel\_tab
  
dele\_init:
  
 cmp cx,0
  
 je dele\_init\_
  
 add si,38
  
 loop dele\_init
  
  
dele\_init\_:
  
;初始化最后一位联系人的位置
  
 pop cx;取出当前联系人的总人数
  
 push cx
  
  
 mov di,seg \_name
  
 mov es,di
  
 mov di,offset tel\_tab
  
 sub cx,1;如果当前总人数为1的话，就不需要循环初始化
  
 cmp cx,0
  
 je \_exchange
  
dele\_loop:
  
 add di,38
  
 loop dele\_loop
  
 push si
  
 push di
  
;交换最后一位联系人和我们要删除的联系人的姓名位置
  
\_exchange:
  
 mov cx,22
  
\_xchange:
  
 mov al,es:[di]
  
 mov byte ptr ds:[si],al
  
 inc di
  
 inc si
  
 loop \_xchange
  
 ;交换相应联系人电话号码
  
 pop di
  
 pop si
  
 add si,23
  
 add di,23
  
 mov cx,14
  
\_xchange\_num:
  
 mov al,es:[di]
  
 mov byte ptr ds:[si],al
  
 inc di
  
 inc si
  
 loop \_xchange\_num
  
dele\_exit:
  
 pop sum
  
;删除一个联系人之后，总人数减少
  
 sub sum,1
  
 ret
  
dele\_only:
  
 mov dx,offset temp11
  
 mov ah,09h
  
 int 21H
  
 pop sum
  
 sub sum,1
  
 ret
  
;查找失败
  
dele\_fail:
  
 mov dx,offset temp7
  
 mov ah,09h
  
 int 21H
  
 pop sum
  
 ret
  
;表为空
  
dele\_empty:
  
 mov dx,offset temp6
  
 mov ah,09h
  
 int 21H
  
 pop sum
  
 ret
  
del\_man endp
  
  
;寻找联系人
  
sear\_man proc near
  
 push sum;暂存联系人的人数
  
 mov dx,offset method
  
 mov ah,09h
  
 int 21H
  
  
 ;输入要寻找的联系人
  
 mov dx,offset namemax
  
 mov ah,0ah
  
 int 21H
  
  
 ;取出比较的次数
  
 mov cl,\_length
  
 xor ch,ch
  
 ;初始化电话簿的地址
  
 mov si,offset tel\_tab
  
 add si,2
  
 mov di,seg \_name
  
 mov es,di
  
 mov di,offset \_name
  
 mov bx,0
  
 cmp sum,0
  
 je sear\_empty;如果sum一开始就为0，即为空的，肯定找不到
  
sear\_loop4:
  
 cmp sum,0;此时肯定不会满足条件，所以开始寻找
  
 je sear\_fail
  
 inc bx
  
sear\_cmp:
  
 push bx
  
 mov al,ds:[si]
  
 mov bl,es:[di]
  
 cmp al,bl
  
 pop bx
  
 jne nomatch
  
 inc si
  
 inc di
  
 loop sear\_cmp
  
 jmp sear\_suce
  
nomatch:
  
 add si,38
  
 mov di,offset \_name
  
 sub sum,1
  
 jmp sear\_loop4
  
  
;查找成功
  
sear\_suce:
  
 mov cx,bx
  
 sub cx,1
  
 mov si,offset tel\_tab
  
\_init:
  
 cmp cx,0
  
 je \_output
  
 add si,38
  
 loop \_init
  
  
  
\_output:
  
 add si,25
  
 push si
  
 mov si,offset \_name
  
 mov cl,\_length
  
 xor ch,ch
  
 CF
  
\_out\_name:
  
 mov dl,ds:[si]
  
 mov ah,02h
  
 int 21H
  
 inc si
  
 loop \_out\_name
  
  
 pop si
  
 mov cx,11
  
 mov dl,' '
  
 mov ah,02h
  
 int 21H
  
\_out\_number:
  
 mov dl,ds:[si]
  
 mov ah,02h
  
 int 21H
  
 inc si
  
 loop \_out\_number
  
sear\_exit:
  
 pop sum
  
 ret
  
;查找失败
  
sear\_fail:
  
 mov dx,offset temp7
  
 mov ah,09h
  
 int 21H
  
 jmp sear\_exit
  
;表为空
  
sear\_empty:
  
 mov dx,offset temp6
  
 mov ah,09h
  
 int 21H
  
 jmp sear\_exit
  
sear\_man endp
  
  
;修改联系人信息
  
change proc near
  
 push sum;暂存联系人的人数
  
 mov dx,offset temp9
  
 mov ah,09h
  
 int 21H
  
  
 ;输入要寻找的联系人
  
 mov dx,offset namemax
  
 mov ah,0ah
  
 int 21H
  
  
 ;取出比较的次数
  
 mov cl,\_length
  
 xor ch,ch
  
 ;初始化电话簿的地址
  
 mov si,offset tel\_tab
  
 add si,2
  
 mov di,seg \_name
  
 mov es,di
  
 mov di,offset \_name
  
 mov bx,0
  
 cmp sum,0
  
 je change\_empty;如果sum一开始就为0，即为空的，肯定找不到
  
change\_loop4:
  
 cmp sum,0;此时肯定不会满足条件，所以开始寻找
  
 je change\_fail
  
 inc bx
  
change\_cmp:
  
 push bx
  
 mov al,ds:[si]
  
 mov bl,es:[di]
  
 cmp al,bl
  
 pop bx
  
 jne change\_nomatch
  
 inc si
  
 inc di
  
 loop change\_cmp
  
 jmp change\_suce
  
change\_nomatch:
  
 add si,38
  
 mov di,offset \_name
  
 sub sum,1
  
 jmp change\_loop4
  
;查找成功
  
;找到并初始化我们要修改的联系人的位置
  
change\_suce:
  
 mov cx,bx
  
 sub cx,1
  
 mov si,offset tel\_tab
  
change\_init:
  
 cmp cx,0
  
 je change\_main
  
 add si,38
  
 loop change\_init
  
 push si;si保存的是我们要修改的联系人的内存位置
  
  
change\_main:
  
 mov dx,offset temp10
  
 mov ah,09h
  
 int 21H
  
  
 mov dx,si
  
 add dx,23
  
 mov ah,0ah
  
 int 21H
  
  
 mov dx,offset temp11
  
 mov ah,09h
  
 int 21H
  
  
change\_exit:
  
 pop sum
  
 ret
  
;查找失败
  
change\_fail:
  
 mov dx,offset temp7
  
 mov ah,09h
  
 int 21H
  
 jmp change\_exit
  
;表为空
  
change\_empty:
  
 mov dx,offset temp6
  
 mov ah,09h
  
 int 21H
  
 jmp change\_exit
  
  
change endp
  
code ends
  
end start