

STOS ADDRESS BOOK

Welcome to this tutorial on writing a database program in STOS. A simple database that will store information in the way of names, addresses, and other details. I will call this program "STOS Address Book". Note there are no prizes for guessing a better name.

First, what is a database? Well, it's a program that stores information entered by the user, and allows them to view it later. But why write a database to store names and addresses when you can quite simply use an address book to write in? Well, there are certain advantages.....for example, a program can find the info quicker than someone searching through a book. Also, should you wish to delete someone's entry at any time then you can do it quite easily without having blotted out entries.

Each entry is entered into a field. In other words, an element of an array. So, let's first set the screen up.

```
5 rem —ADDRESS BOOK
```

```
10 key off : curs off : hide on : flash off : mode 1
```

```
15 rem—SET UP ARRAYS FOR FILE INFO
```

```
20 dim SURNAME$(1000) : dim NAME$(1000) : dim ADDR$(1000) :
```

```
dim TOWN$(1000) : dim POST$(1000) : dim PHONE$(1000) :
```

```
dim INFO$(1000)
```

```
25 rem—SET UP VARIABLES
```

```
30 FILE=0 : FILES=0
```

Most databases are displayed in the medium resolution which makes them look more professional. There are five fields: Surname, Names, Addresses, Towns, Postcode, Phone, and Info. Note the way the TOWN array is spelled. The O is a nought. This stops STOS from interpreting the word as to WN\$

Each field has space reserved for 1000 entries. The more entries we have, the more memory is used. Now, let's set up the rest of the screen.

```
35 rem—OPEN FIRST WINDOW AND ENTER TITLE"
```

```
40 windopen 1,0,0,80,3,3,1
```

```
50 cdown : centre "ADDRESS BOOK"
```

```
55 rem—OPEN SECOND WINDOW AND ENTER OPTIONS"
```

```
60 windopen 2,59,3,21,17,3,1
```

```
70 locate 1,2 : print "(C)REATE A FILE"
```

```
80 locate 1,4 : print "(D)ELETE A FILE"
```

```
90 locate 1,6 : print "(F)IND A FILE"
```

```
100 locate 1,8 : print "(L)IST ALL FILES"
```

```
110 locate 1,10 : print "(P)RINT A FILE"
```

```
120 locate 1,12 : print "(Q)UIT AND SAVE"
```

```
125 rem—OPEN THIRD WINDOW FOR PROGRAM INFO
```

```
130 windopen 3,0,20,80,3,3,1
```

```
140 centre "CHOOSE AN OPTION"
```

This part of the program opens three windows which displays the title, program options, and the bottom window is used for program information, meaning that we use it prompting the user what to do.

The next thing to do is to use the blank part of the screen so we can print the field names but first, we have to leave the windows.

```
145 rem—LEAVE WINDOW AND GOTO SCREEN
```

```
150 qwindow 0
```

This is an undocumented feature of STOS. The manual tells us that when using the window commands the values range from 1 to 15 while nought is the STOS system window. You can however leave the windows completely without going to another or deleting them by using a "qwindow 0". Now we can print the field names on-screen without affecting the windows.

```
155 rem—PRINT FIELD NAMES
```

```
160 locate 0,4 : print "SURNAME:"
```

```
170 locate 0,6 : print "NAME:"
```

```
180 locate 0,8 : print "ADDRESS:"
```

```
190 locate 0,10 : print "TOWNS:"
```

```
200 locate 0,12 : print "POSTCODE:"
```

```
210 locate 0,14 : print "PHONE:"
```

```
220 locate 0,16 : print "INFO:"
```

```
225 locate 0,18 : print "FILES: ";FILES
```

Everything's set up. Now we just have to wait for the user to select an option.

```
230 rem—WAIT FOR AN INPUT
```

```
240 K$=input$(1)
```

```
250 rem—CHECK WHICH KEY HAS BEEN PRESSED
```

```
260 if K$="C" or K$="c" then goto 400
```

```
270 if K$="D" or K$="d" then goto 500
```

```
280 if K$="F" or K$="f" then goto 600
```

```
290 if K$="L" or K$="l" then goto 700
```

```
300 if K$="P" or K$="p" then goto 800
```

```
310 if K$="Q" or K$="q" then goto 900
```

```
320 K$="" : goto 240
```

The variable K\$ is used to store the user's selection. At the moment he only chose C to create a file...the program will then go to line 400 which is the input routine. Line 320 clears the variable and returns to line 240 ready to await another selection.

```
400 rem—CREATE A FILE
```

```
410 curs on : inc FILES
```

```
420 rem—GET SURNAME
```

```
430 X=8 : Y=4 : gosub 2000 : SURNAME$(FILES)=K$
```

```
440 rem—GET NAME
```

```
450 X=5 : Y=6 : gosub 2000 : NAME$(FILES)=K$
```

```
460 rem—GET ADDRESS
```

```
470 X=8 : Y=8 : gosub 2000 : ADDR$(FILES)=K$
```

```

480 rem—GET TOWNS
490 X=6 : Y=10 : gosub 2000 : TOWN$(FILES)=K$
500 rem—GET POSTCODE
510 X=9 : Y=12 : gosub 2000 : POST$(FILES)=K$
520 rem—GET PHONE NUMBER
530 X=6 : Y=14 : gosub 2000 : PHONE$(FILES)=K$
540 rem—GET INFO
550 X=5 : Y=16 : gosub 2000 : INFO$(FILES)=K$
555 rem—FILE CREATED SO GO BACK AND WAIT FOR NEXT INPUT
560 K$="" : curs off : gosub 2070 : goto 225

```

This is a kind of setup to enter the information needed. The X and Y variables hold the X and Y coordinates of the text cursor. Notice at the start of this part we turn the cursor on and add one to the FILES variable. This is the number of the file we are currently creating.

The X and Y variables are needed to position the text cursor just after each field name...IE SURNAME, NAME, etc... The routine then does a subroutine to the routine that enters the information into a variable. Once the file has been created the program turns off the cursor, does a subroutine to clear the wording entered off the screen then goes back to wait for another selection from the user.

```

2000 rem—INPUT ROUTINE
2010 locate X,Y
2020 input "";K$
2030 if K$="" then K$="UNKNOWN"
2040 return

```

The actual input routine. Note how we use the locate command to position the cursor at the right place for every field name. Line 2020 allows you to type in the information while line 2030 checks if the users just press the enter key without typing anything in.

```

2070 rem—CLEAR INFO OFF SCREEN

```

```

2080 wait 30
2090 locate 8,4 : print space$(50)
2100 locate 5,6 : print space$(53)
2110 locate 8,8 : print space$(50)
2120 locate 6,10 : print space$(52)
2130 locate 9,12 : print space$(49)
2140 locate 6,14 : print space$(52)
2150 locate 5,16 : print space$(53)
2160 return

```

The final part clears the wording off the screen once the file is created. The space\$ command just prints a line of spaces to clear it.

Now we are adding a Delete File function to our Address Book program. Here is the main routine for it.

```

600 rem—DELETE A FILE
610 if FILES=0 then goto 310
620 locate 5,6 : curs on : input "";N$ : curs off : if N$="" then goto 310
630 rem—LOOK FOR FIRST THREE LETTERS OF N$ IN NAME$ ARRAY
635 FILE=0
640 FOUND=0 : repeat
650 inc FILE : if left$(NAME$(FILE),3)=left$(N$,3) then FOUND=1 : goto 690
660 until FILE=FILES
670 rem—IF THE FILE IS'NT FOUND THEN GO BACK TO INPUT
680 if FOUND=0 then qwindow 3 : clw : curs off : centre "FILE NOT FOUND" : wait 100 : qwindow 0 : locate 5,6 : print space$(53) : qwindow 3 : clw : goto 200
690 rem—FILE FOUND SO PRINT INFO
700 locate 8,4 : print SURNAME$(FILE)

```

```

710 locate 5,6 : print NAME$(FILE)
720 locate 8,8 : print ADDR$(FILE)
730 locate 6,10 : print TOWN$(FILE)
740 locate 9,12 : print POST$(FILE)
750 locate 6,14 : print PHONE$(FILE)
760 locate 5,16 : print INFO$(FILE)
770 locate 0,18 : print "FILES:";FILES
780 rem—ASK USER IF HE WANTS TO DELETE THE FILE
790 qwindow 3 : clw : curs off : centre "DELETE THIS FILE (Y/N)?" :
D$=input$(1)
800 if D$="n" or D$="N" then qwindow 0 : gosub 3000 : goto 640
810 rem—DELETE FILE AND SORT THEM INTO ORDER
820 for X=FILE to FILES
830 SURNAME$(X)=SURNAME$(X+1)
840 NAME$(X)=NAME$(X+1)
850 ADDR$(X)=ADDR$(X+1)
860 TOWN$(X)=TOWN$(X+1)
870 POST$(X)=POST$(X+1)
880 PHONE$(X)=PHONE$(X+1)
890 INFO$(X)=INFO$(X+1)
900 next X : dec FILES
910 rem—GO BACK TO OPTIONS
920 qwindow 0 : gosub 3000 : qwindow 3 : clw : goto 200

```

What this routine does is it first checks if there are any files in memory and if so asks the user for the first name of the person they wish to delete. The routine finds the name then asks the user if he wants to delete it.

Note that only the first three letters of the name are searched for, this increases speed when searching. If the user doesn't wish to delete the first file, then the program will continue to list the other files under that name until either it reaches the end of the user deletes a file.

The program also uses window three to display the information for the function. It also does a gosub to a clearing routine to get rid of the excess wording on screen. It then deletes the selected record by doing a loop from FILE which is the file number to be deleted, to FILES which is the number of records held in memory. After this it subtracts the FILE variable by one then goes back to the selection routine. As a file has been removed the program brings all the other records back to replace the deleted file.

I have added a FIND FILE function to the database program. This allows us to enter the name of the person whose details we wish to find and the program will list every name it finds under that input, allowing us to step through each one till we find the one we want.

```
1000 rem—FIND A FILE
1010 if FILES=0 then goto 310
1020 locate 5,6 : curs on : input "" ; N$ : curs off : if N$="" then goto 310
1030 rem—LOOK FOR FIRST THREE LETTERS OF N$ IN NAME$ ARRAY
1040 FILE=0
1050 FOUND=0 : repeat
1060 inc FILE : if left$(NAME$(FILE),3)=left$(N$,3) then FOUND=1 : goto 1100
1070 until FILE=FILES
1080 rem—IF THE FILE ISN'T FOUND THEN GO BACK TO INPUT
1090 if FOUND=0 then qwindow 3 : clw : curs off : centre "FILE NOT FOUND" : wait 100 : qwindow 0 : locate 5,6 : print space$(53) : qwindow 3 : clw : goto 200
1100 rem—FILE FOUND SO PRINT INFO
1110 locate 8,4 : print SURNAME$(FILE)
1120 locate 5,6 : print NAME$(FILE)
```

```

1130 locate 8,8 : print ADDR$(FILE)
1140 locate 6,10 : print TOWN$(FILE)
1150 locate 9,12 : print POST$(FILE)
1160 locate 6,14 : print PHONE$(FILE)
1170 locate 5,16 : print INFO$(FILE)
1180 locate 0,18 : print "FILES:";FILES
1190 rem—ASK USER IF HE WANTS TO CONTINUE THE LISTING
1200 qwindow 3 : clw : curs off : centre "FILE NO:"+str$(FILE)+" NEXT
FILE (Y/N)" : D$=input$(1)
1210 if FILE=FILES then qwindow 0 : gosub 3000 : goto 3100
1220 if D$="Y" or D$="y" and FILE<FILES then qwindow 0 : gosub 3000
: goto 1050

```

This is almost the same as the Delete file function only it allows you to step through the records stored so you can read them. Notice how the program only checks the first three letters of the name for speed. The final part of the program checks if the user wants to continue and only goes back to list another record if the variable FILE (no of present file) is lower than the variable FILES (total no of files stored). Now let's look at listing all the files.

```

1230 rem —LIST ALL FILES
1290 FILE=0 : if FILES=0 then goto 310
1300 for FILE=1 to FILES
1310 locate 8,4 : print SURNAME$(FILE)
1320 locate 5,6 : print NAME$(FILE)
1330 locate 8,8 : print ADDR$(FILE)
1340 locate 6,10 : print TOWN$(FILE)
1350 locate 9,12 : print POST$(FILE)
1360 locate 6,14 : print PHONE$(FILE)
1370 locate 5,16 : print INFO$(FILE)

```



```
1380 locate 0,18 : print "FILES:";FILES
1390 rem—ASK USER IF HE WANTS TO CONTINUE THE LISTING
1400 qwindow 3 : clw : curs off : centre "FILE NO:"+str$(FILE)+"
NEXT FILE (Y/N)" : D$=input$(1)
1410 if FILE=FILES then qwindow 0 : gosub 3000
1420 if D$="Y" or D$="y" and FILE<FILES then qwindow 0 : gosub 3000
else qwindow 0 : gosub 3000 : goto 3100
1430 next FILE
```

This routine uses a loop to step through each file in turn and print it on the screen. The user is then asked if he would like to continue listing or to terminate. If he continues then the loop continues. If the variable FILE equals the same as FILES (number of files stored) then the program terminates itself.

If you want to see this working then I have supplied the complete STOS Code rather than type out this listing. You can use the routines for other programs if you so wish.