ARAB REPUBLIC OF EGYPT

THE INSTITUTE OF
NATIONAL PLANNING

Memo No. (1501)

An Information System for Personnel Management System { using Advanced Techniques of dBASE III PLUS }

by

Dr. Abdalla El-Daoushy
Dr. M. Yehia Youssef

May, 1989

CAIRO
SALAH SALEM St-NASR CTTY

Foreword

This memo. emphasize on the practical side of the dataBase and information systems. It helps in getting the job done quickly and efficiently.

The memo. presents working dataBase and information systems that do not only perform useful information tasks, but also demonstrate advanced computer programming techniques that can be used in many dataBase and information applications.

The memo. is not intended for the computer novice, However, familiarity with the basic commands of dBase III PLUS will be sufficient background.

This memo. is divided into two main parts of three chapters. Part one which comprises chapter 1 discusses the programming considerations and emphasize on advanced techniques for maximizing the speed and performance of designing and building software systems.

Part two comprises chapters 2 and 3 and presents a custom software system for managing a Personnel Management System. The techniques presented in designing this software system can be used to manage any single dataBase system.

Part two also presents the basics of creating user-friendly, menu-driven systems, creating and using custom screens and reports, and using index files for maximum speed.

Contents

Chapter 1

ADVANCED TECHNIQUES OF dBASE III PLUS

1.1 DATABASE DESIGNS :

Deciding what to store in a dataBase is one of the first steps in designing any software system. Many software systems involve only a single dataBase and perhaps one or two index files. More sophisticated software systems may use several dataBase files interactively.

In this section, four of the most commonly used dataBase designs are discussed: single dataBase, single dataBase with Memo field, relational dataBase, and master-file/transaction-file dataBase systems.

1. Single dataBase File System :

In such case, notice that:

- Whenever you add new RECORDs, or make changes to dataBase through:

```
        EDIT        DELETE        PACK        BROWSE
```

make sure Index files are active.

- Consider the command:

```
        USE Mail INDEX Names , Zips
```

{ where Names indexed on L_NAME and F_NAME and Zips indexed on ZIP_CODE}

since Names is first-listed index file, the commands:

```
        DISPLAY      LIST      REPORT FORM
```

will display the RECORDs in alphabetical order by Last and First Names.

To display the RECORDs in ZIP_CODE order, just make Zips the first-listed index file as:

```
        USE Mail INDEX Zips , Names
```

2. Single dataBase File with Memo Field :

When you use the LIST command to view the RECORDs of a dataBase file including Memo field, the word Memo is displayed in the listing instead of the contents of the field itself.

However, if you specify the field names by typing ,for example,

```
     LIST OFF AUTHOR , TITLE , PUBLISHER , PUB_YEAR , PAGES , ABSTRACT

                                   { ABSTRACT is a field of type Memo }
```

the contents of the ABSTRACT field are included in the listing.

You can gain some control over the appearance of the Memo field using the SET MEMOWIDTH command. For example, to specify the width of the Memo field by 40 characters (the default width is 50), enter the following command:

```
           SET MEMOWIDTH TO 40
```

Another way to gain control over the appearance of the Memo field is to use a program. For example, the following program displays data including the Memo field ADDRESS from the Library MASTER.DBF file:

```
* Program..: Library.PRG
* Date.....: May, 1989
* Remark...: To print dataBase with Memo field.
*
USE MASTER
GO TOP
DO WHILE .NOT. EOF()
   ?
   ? "          Author:" , AUTHOR
   ?
   ? "           Title:" , TITLE
   ?
   ? "       Publisher:" , PUBLISHER
   ?
   ? "Publishing Date:" , PUB_YEAR
   ?
   ? "    No. of Pages:" , PAGES
   ?
   ? "Key Words:" , TOPICS
```

```
?
? "------------------------------------------------------------"
? ABSTRACT
?
?
?
*
SKIP    && very important to avoid looping
*
ENDDO
* eof()
```

3. Relational dataBases :

Suppose there are two Accounts Receivable dataBases: AR1.DBF and AR2.DBF with the following structure:

AR1.DBF Structure:

Field #	Field name	Type	Width
1	BILL_DATE	Date	8
2	AMOUNT	Numeric	9.2
3	VEND_CODE	Character	7

AR2.DBF Structure:

Field #	Field name	Type	Width
1	VEND_CODE	Character	7
2	VENDOR	Character	25
3	ADDRESS	Character	25
4	CITY	Character	25
5	GOVERNRATE	Character	25
6	ZIP	Character	10

To get the relationship between the two dataBases, the LOOKUP (i.e. the 2nd dataBase file) dataBase file must be INDEXed on the key field (VEND_CODE) that relates the two dataBases as follows:

```
    USE AR2
    INDEX ON VEND_CODE TO Vendor
```

Both dataBases must be opened simultaneously with the SELECT command and the RELATIONship command as follows:

```
    SELECT 1
    USE AR1
    SELECT 2
    USE AR2 INDEX Vendor
    SELECT 1
    SET RELATION TO VEND_CODE INTO AR2
```

If you were to use the LIST command now, you would still see the RECORDs from the AR1.DBF.

But,to see the associated Vendor names and addresses, use the LIST command again, but this time specify field names from the AR2.DBF, using the B -> specification. (Note that AR2 in the example above is the B dataBase because it was SELECTed 2nd. A 3rd file would be the C -> dataBase, a 4th would be the D -> dataBase, and so forth).

```
    LIST BILL_DATE , AMOUNT , B -> VENDOR , B -> ADDRESS
```

Then, the information is readily available from both dataBases simultaneously.

4. Master-File / Transaction-File DataBase System :

The 4th commonly used dataBase design involves Master Files and Transaction Files. These are most often used ,for example, in Inventory and Bookkeeping applications; where one dataBase file contains (up to the minute information about) the stock on hand or account balances, and additional dataBases contain RECORDs of individual transactions, such as goods sold and received.. etc.

A simple Inventory System might ,for example, use a Master dataBase file (MASTER.DBF) and two transaction dataBase files (SALES.DBF and NEWSTOCK.DBF).

The MASTER.DBF contains current information about the stock on hand.

It gets its information on individual sales transactions and newstock transactions from transaction files SALES.DBF and NEWSTOCK.DBF. In a sense, MASTER.DBF is a summary of the SALES.DBF and NEWSTOCK.DBF.

The benefit of this dataBase structure is that it allows to keep abreast (i.e., to be informed of the latest information) of the exact stock on hand at any given moment, while at the same time it provides a separate, permanent RECORD of every item sold and received.

The SALES.DBF contains information about individual sales transactions. The NEWSTOCK contains information about coming goods.

The structure of both MASTER.DBF and SALES.DBF are shown as follows:

MASTER.DBF Structure:

Field #	Field name	Type	Width
1	PART_NO	Character	7
2	TITLE	Character	20
3	ON_HAND	Numeric	4
4	COST	Numeric	8.2
5	REORDER	Numeric	4

SALES.DBF Structure:

Field #	Field name	Type	Width
1	PART_NO	Character	7
2	INVOICE_NO	Character	7
3	SALSPERSON	Character	25
4	CUSTOMER	Character	25
5	QTY	Numeric	4
6	PRICE	Numeric	8.2
7	DATE	Date	8

Notice that both the SALES.DBF and MASTER.DBF have the same field called PART_NO. This common field allows dBASE to relate RECORDs in both files.

dBASE uses the UPDATE command to keep the MASTER.DBF current as follows:

i, Index each dataBase file on PART_NO field:

```
USE MASTER
INDEX ON PART_NO TO MASTER
USE SALES
INDEX ON PART_NO TO SALES
```

ii, Open both dataBases with the SELECT command and use the UPDATE command to transfer information from the SALES.DBF to the MASTER.DBF as follows:

```
CLEAR ALL
SELECT 2
USE SALES INDEX SALES
SELECT 1
USE MASTER INDEX MASTER
UPDATE ON PART_NO FROM SALES REPLACE ON_HAND WITH ON_HAND-B-> QTY
```

Conclusion:

As you see, there are many ways to design dataBases and dataBase systems. Which design you use depends on what you need to do and the limits imposed by your hardware.

1.2 Power of Indexed Files :

Using indexed files can actually save hours from a program's processing time just by setting the appropriate index files to handle the job properly.

Displaying RECORDs in Sorted Order :

Four methods are compared to each other:

1. The SORT Method:

The 1st method (the slowest if you have more than 50 RECORDs) is simply to use the SORT command to create a new physically SORTed file. For example, suppose you had already SORTed a file called PERSONEL.DBF and then added some new RECORDs with the commands:

```
USE PERSONEL
APPEND
```

You can get the RECORDs back into alphabetical order using SORT command as follows:

```
USE PERSONEL
SORT ON F_NAME TO Temp
CLOSE DATABASES
ERASE PERSONEL.DBF
RENAME Temp.DBF TO PERSONEL.DBF
USE PERSONEL
LIST
```

To sort 1000 RECORDs back into the first name order will take an estimated time of 6 minutes on a Floppy-Disk System and about 4.5 minutes on a Hard-Disk System.

2. The APPEND and INDEX Method:

```
USE PERSONEL
APPEND
INDEX ON F_NAME TO NAMES
LIST
```

This method requires about 45 seconds to re-SORT the 1000 RECORDs back into first name order on a Floppy-Disk System and about 23 seconds on a Hard-Disk System.

3. The LOCATE and INSERT Method:

This method would be to LOCATE the position in the SORTed dataBase file where each new RECORD belongs and INSERT the new RECORD directly into the proper alphabetical location.

```
USE PERSONEL
LOCATE FOR F_NAME = "Abdalla"
INSERT
LIST
```

The time consumed would be long since it has to find the appropriate place to INSERT each new RECORD plus about 50 seconds for each separate INSERT command to INSERT the RECORD on a Floppy-Disk System or about 30 seconds on a Hard-Disk System.

4. Creating an Index File and APPEND Command:

```
USE PERSONEL INDEX NAMES
```

Any changes to the PERSONEL.DBF, whether they are made through:

```
APPEND, EDIT, BROWSE, DELETE, PACK, or READ
```

are made to the index file (NAMES) as well, and the index file is automatically resorted and adjusted. For example,

```
USE PERSONEL INDEX NAMES
APPEND
LIST
```

dBASE automatically puts the index back into the first name order in less than one second and the LIST command immediately displays the RECORDs in last name order.

The following table shows the restoring times for the 4 techniques just described:

Method	Commands used	time required (in seconds)	
		Hard-Disk System	Floppy-Disk System
1	USE, APPEND, and SORT	270	360
2	USE, APPEND, and INDEX ON	23	45
3	USE, LOCATE, and INSERT	30	50
4	USE, INDEX file and APPEND	1	1

Clearly, the 4th method using an active index file while adding the new RECORDs is the fastest.

Faster Searching :

Consider the following two different programs for two different approaches and compare their processing time:

The 1st Approach:

This approach uses the standard LIST FOR approach to display all RECORDs with first name Abdalla.

```
*
* LIST FOR Approach
*
CLEAR
USE PERSONEL INDEX NAMES
ACCEPT "List all people with what first name? " TO Search
*
LIST FOR F_NAME = Search
* eof()
```

The time required to display 10 Abdallas is about 2 minutes on a Floppy-Disk System and 32 seconds on a Hard-Disk System.

The 2nd Approach:

This approach uses the FIND or SEEK commands to look up the first Abdalla in the NAMES index, then use the WHILE option to display the remaining Abdallas in the dataBase file.

```
*
* FIND, SEEK, and LIST WHILE Approach
*
CLEAR
USE PERSONEL INDEX NAMES
ACCEPT "List all people with what last name? " TO Search
SEEK Search
LIST WHILE F_NAME = Search
* eof()
```

The processing time here compared with the program above is 5 seconds on a Floppy-Disk System and less than 4 seconds with a Hard-Disk System.

The following table compares the processing time for the two different approaches:

		time required (in seconds)	
Method	Commands used	Hard-Disk System	Floppy-Disk System
1	LIST FOR	32	120
2	SEEK and LIST WHILE	3.79	5.57

Searching for Ranges:

. To create a small dataBase file of only those individuals whose names begin with the letters M through P from the PERSONEL.DBF file, you can use:

```
        USE PERSONEL
        COPY TO Temp FOR F_NAME >= "M" .AND F_NAME <= "P"
```

. An index file speeds the time considerably:

```
        USE PERSONEL INDEX NAMES
        FIND M
        COPY TO Temp WHILE F_NAME <= "P"
```

. To view all RECORDs that fall within a range of dates, the following program do so without the use of an index file:

```
*
* Display RECORDs within a Range of dates.
*
USE PERSONEL
CLEAR
STORE "        " TO Start , End
@ 10, 2 SAY "Enter start date " GET Start PICT "99/99/99"
@ 12, 2 SAY "  Enter end date " GET End PICT "99/99/99"
READ
*
STORE CTOD(Start) TO Start
STORE CTOD(End) TO End
*
LIST FOR DATE >= Start .AND. DATE <= End
* eof()
```

This program will require several minutes to list all RECORDs within the specified range of dates.

. As an alternative, you can create an index file of the DATE field:

```
     USE PERSONEL
     INDEX ON DATE TO DATES
```

so the program:

```
*
* Display RECORDs within a Range of dates.
*
USE PERSONEL INDEX DATES
CLEAR
STORE "          " TO Start , End
@ 10, 2 SAY "Enter start date " GET Start PICT "99/99/99"
@ 12, 2 SAY "  Enter end date " GET End PICT "99/99/99"
READ
*
STORE CTOD(Start) TO Start
STORE CTOD(End) TO End
*
SEEK Start
LIST WHILE DATE <= End
* eof()
```

This program will cut the searching time down to about 5 seconds compared with the first approach.

Technical Aspects of Index Files :

Let us discuss the technical side of indices to see why the FIND and WHILE approach always outperforms the FOR approach.

- when you use the FOR option to search a dataBase file, dBASE always start accessing the RECORDs from RECORD number 1 and reads every RECORD in the dataBase directly from the disk. For example, if you have a small dataBase file with 11 names in it, three of which are Abdalla, dBASE performs 11 disk accesses to display the three Abdallas.

- when you use the WHILE option rather than the FOR option to search for RECORDs, dBASE will take only 3 disk accesses to display out the three Abdalla.

In this small dataBase file, there would not be dramatic improvement in processing speeds, but if you had an un-indexed dataBase with 10 000 RECORDs in it, 10 of which had the first name Abdalla, the command:

```
LIST FOR F_NAME = "Abdalla"
```

would require 10 000 disk accesses, which could take well over 30 minutes. If you use an index file to FIND and LIST only the RECORDs in which the first name is Abdalla, dBASE will make only 10 disk accesses, thereby eliminating 9 990 unnecessary ones and about 29 minutes and only 55 seconds are needed.

Faster Mathematical Operations :

. To let dBASE counts the number of Abdallas in the dataBase file, the following commands may be used:

```
USE PERSONEL
COUNT FOR F_NAME = "Abdalla"
```

This method requires about <u>15 seconds</u> to display 10 Abdallas found in the dataBase file.

. This time can be cut down significantly to 1 second by using the commands:

```
USE PERSONEL INDEX NAMES
FIND Abdalla
COUNT WHILE F_NAME = "Abdalla"
```

The same technique can be used with the SUM and AVERAGE commands:

```
USE PERSONEL
SUM AMOUNT FOR F_NAME = "Abdalla"          { time = 17 seconds }

USE PERSONEL INDEX NAMES
FIND Abdalla
SUM AMOUNT WHILE F_NAME = "Abdalla"        { time = 2 seconds }
```

In the same manner,

```
USE PERSONEL INDEX NAMES
FIND Abdalla
AVERAGE AMOUNT WHILE F_NAME = "Abdalla"  { time = 2 seconds }
```

Faster Reports :

Suppose a formatted report called Mailist already created.

. To present the RECORDs for all the Abdallas in the report format, the following commands can do so:

```
USE PERSONEL INDEX NAMES
REPORT FORM Mailist FOR  F_NAME = "Abdalla"  {time = 30 seconds}
```

. If the following commands are used:

```
USE PERSONEL INDEX NAMES
FIND Abdalla
REPORT FORM Mailist WHILE F_NAME = "Abdalla"
```

the same job can be completed in about 6 seconds.

Faster Copying :

. The commands:

```
USE PERSONEL INDEX NAMES
COPY TO Temp FOR F_NAME = "Abdalla"
```

require about 30 seconds to process on Hard-Disk System

. To perform the same job using the commands:

```
USE PERSONEL INDEX NAMES
FIND Abdalla
COPY TO Temp WHILE F_NAME = "Abdalla"
```

copying process trim down to about 2 seconds.

Faster Editing :

.
```
    USE PERSONEL
    BROWSE
```

With this approach, you have to press PgDn many times to scroll through the file to find Abdalla.

. On the other hand, the following approach:

```
    USE PERSONEL INDEX NAMES
    FIND Abdalla
    BROWSE
```

BROWSE displays the 1st Abdalla in the dataBase file and all the remaining Abdallas immediately below it. So, there is no need to scroll through pages to find the Abdalla you wish to edit, because dBASE in this case displays all Abdallas immediately.

. Similarly, the following commands are used for quickly editing and changing a particular Abdalla:

```
    USE PERSONEL INDEX NAMES
    FIND Abdalla
    EDIT
```

Each press of the PgDn key will immediately position you at the next Abdalla in the dataBase file. So once again, you save time by not having to scroll through the entire dataBase file searching for the individual Abdallas.

Faster Sorting :

. You can use the SORT command to create a SORTed dataBase file called Temp from the PERSONEL dataBase file as follows:

```
USE PERSONEL
SORT ON F_NAME TO Temp
```

This approach takes from 3 to 7 minutes for a 1000 RECORDs dataBase file.

. But, if the NAMES index file already exits, you can achieve the same result with about 30 to 40% faster:

```
USE PERSONEL INDEX NAMES
COPY TO Temp
```

Moreover, when you copy an indexed file with the index active to another dataBase file, the RECORDs in the new dataBase are sorted physically.

Managing Multiple Index Files :

Consider the following command:

```
     USE PERSONEL INDEX NAMES , ZIPS
```

where ZIPS is indexed file on CITY+GOVERNRATE fields

With both index files specified in this fashion, all future modifications to the PERSONEL.DBF with the APPEND, EDIT, BROWSE, DELETE, PACK, or REPLACE command automatically update both indices, <u>and</u> <u>once again avoiding having to resort or reindex the dataBase file.</u>

<u>The order of the active index files in the use command plays an important role in how dBASE behaves. The 1st-listed index file is called the PRIMARY or MASTER index.</u> Whenever you use the LIST or REPORT commands or any other way to access the RECORDs in the dataBase, dBASE displays them in the sorted order of the PRIMARY index file. Also, the FIND and SEEK commands work only with the PRIMARY index file. So when you use the dataBases with the active index files listed with NAMES file 1st, the RECORDs will always appear sorted in first name order, and you will be able to use the FIND or SEEK command only to locate an individual by name. However, if you modify the dataBase in any way, whether it be through the APPEND, EDIT, BROWSE, READ, DELETE, PACK, or any other command, both index files will automatically be updated and resorted.

<u>dBASE III PLUS allows a total of 7 active index files with any given dataBase, and they are all updated simultaneously when you make changes to the dataBase. But for practical reasons, you may want to limit yourself to only 2 or 3.</u>

Conclusion :

There are some important points to keep in mind about indices and the SEEK and FIND commands.

1. The FIND Command works only on an INDEXed field. Notice that for a dataBase file in use with multiple INDEX files, FIND command works only with the PRIMARY index file.

2. If the data you want to look up are stored in a variable, you should use SEEK, rather than FIND, to locate the data in the index:

```
ACCEPT "Look up whom? " TO Search
SEEK Search
```

   However, you can optionally use FIND with a macro symbol (&) to tell dBASE you are looking for the contents of a memory variable:

```
ACCEPT "Look up whom? TO Search
FIND &Search
```

3. FIND and SEEK commands are used for simple searching and do not support operators. For example, you can not use commands such as these:

```
FIND Abdalla .OR. Yehia
SEEK F_NAME > Search
```

4. If you add or modify data in the dataBase file without all of the index file being active, the unopened index files will be corrupted. Later, when you use the dataBase file with a corrupted index file, dBASE will display an error message such as:

```
Record out of Range, or
End of File Found Unexpectedly.
```

If that occurs, you will need to re-create the index files. To do so, use the dataBase and make all index files active, then issue the REINDEX command:

```
USE PERSONEL INDEX NAMES , ZIPS
REINDEX
```

This will get everything back in shape, but you can avoid the problem entirely by always keeping all index files active when working with an indexed dataBase.

1.3 Programming in dBASE III PLUS :

Interacting with the User :

ACCEPT / TO :

This command presents a question to the user, waits for a response and stores that response to a Character-type-mwmory-variable.

For example:

```
ACCEPT "Send Report to Printer? (Y/N)" TO YN
```

The ACCEPT command always stores information as Character data. So, it may not be the best choice when the program needs to ask about Numeric data.

INPUT / TO :

It is similar to the ACCEPT, but the type of data entered determines the type of the variable created. For example:

```
INPUT "Enter your age: " TO Age
```

This command stores the answer to a Numeric memory variable named Age.

If data are to be stored as Character type, INPUT requires that response to be enclosed in quotation marks.

Important:

Be careful, not to use INPUT to get Date information from the user. For example:

```
INPUT "Enter to-day's date: " TO Date
```

This command accepts information in Date format (MM/DD/YY), but actually stores this Date as a quotient. For example, 03/31/89 would be stored as the quotient of 3 divided by 31 divided by 89 !!!.

WAIT and WAIT / TO :

i, It could be used without a prompt or memory variable:

```
WAIT
```

ii, It could be used with a prompt and memory variable:

```
WAIT "Send data to Printer? (Y/N) TO YN
```

Note:

The WAIT command ,similar to ACCEPT, always stores data as Character-type.

READ :

Unlike ACCEPT, INPUT, and WAIT, READ works only with Field- or Memory- input variable names that already exists. It is usually used in conjunction with the @....SAY....GET commands:

```
Choice = 0
@ 10, 5 SAY "Enter your choice(1-5): " GET Choice
READ
```

Looping with DO WHILE .... ENDDO :

- Every DO WHILE command in a program must have an ENDDO command associated with it.

- One of the most common uses for the DO WHILE ... ENDDO loop is to step through each RECORD in a dataBase file and perform some action on every RECORD.

  The command:

  ```
  DO WHILE .NOT. EOF()
  ```

  is often used for the purpose mentioned above.

Example :

```
CLEAR
SET TALK OFF
X = 1
DO WHILE X <= 20
  ? X
  X = X + 1
ENDDO
```

Making Decisions with IF ... ELSE ... ENDIF :

. Every IF command must have an ENDIF statement associated with it, but the ELSE statement is optional.

. An abbreviated form of the IF is the IIF Function, which can be used in a command line or even in a column definition in a REPORT Format or LABEL Format.

The basic syntax is:

```
IIF( This is TRUE , do this , Otherwise do this )
```

Example:

```
? IIF( X< 10 , "Less Than" , "Greater Than" )
```

This command line prints the words: Less Than if X < 10. Otherwise, it prints the words: Greater Than.

Example :

```
CLEAR
ACCEPT "Turn ON Printer? (Y/N)"  TO YN
IF UPPER(YN) = "Y"
   SET PRINT ON
   ?
   ? "You chose the Printer."
   ?
   EJECT
   SET PRINT OFF
ELSE
   CLEAR
   ?
   ? "You chose the Screen."
   ?
ENDIF
```

Note:

. The IIF Function is used where one condition leads to a single " either/or " result.

. The IF ... ELSE ... ENDIF can perform any number of steps based on the result of a condition.

Making Decision with DO CASE ...... ENDCASE :

Example :

```
* Program..: DoCase.PRG
* Remark...: To illustrate the DO CASE Command.
*
CLEAR
INPUT "Enter a number from 1 to 4 " TO X
DO CASE
   CASE X = 1
        ? "You entered 1. "
   CASE X = 2
        ? "You entered 2. "
   CASE X = 3
        ? "You entered 3. "
   CASE X = 4
        ? "You entered 4. "
   OTHERWISE
        ? "I told you from 1 to 4 !! "
ENDCASE
* eof()
```

Note that :

. The ENDCASE statement must be used to mark the end of the DO CASE clause; the OTHERWISE command is optional.

. The DO CASE command is more commonly used in Menu programs, where the program displays a list of options to the user, waits for a response, then decides what to do next based on the user's menu choice.

Structured Programming :

dBASE III PLUS enables programmers to write programs using the rules of structured programming. The basic goal of structured programming is to create programs that are self-documenting, easy to read and therefore easy to debug or modify in the future.

The following program is written using the rules of structured programming:

```
* Program..: Library.PRG
* Remark...: A Library System Main Menu..
*
USE MASTER
*
Choice = 0
Uline = REPLICATE( "_" , 70 )
*
DO WHILE Choice # 4
   CLEAR
   @ 1,20 SAY "Library Management System"
   @ 2, 5 SAY Uline
   ?
   ?
   TEXT

      Would you like to :
      -------------------

          1. Add new RECORDs

          2. Print Reports

          3. Edit Data

          4. Exit

   ENDTEXT
   *
   @15, 5 SAY "Enter your choice (1-4) " GET Choice RANGE 1,4
   READ
   *
   * --- Perform according to user's choice
   *
   DO CASE
      CASE Choice = 1
           APPEND
      CASE Choice = 2
           REPORT FORM Library
      CASE Choice = 3
           EDIT
    ENDCASE
ENDDO ( WHILE CHOICE # 4 )
*
* --- When Choice = 4, Exit
*
RETURN
* eof()
```

Notice that :

- All the programmer's comments are visible at a glance.

- There is a header at the top of the program that gives the name of the program and a brief description of what it does.

- It is easy to find the beginning and ending points of the loop.

- The program also uses the DO CASE .... ENDCASE command to decide which task to perform, based on the user's choice making the program easier to read.

- <u>In dBASE programming language, anything that you type to the right of an ENDDO, ENDIF, or ENDCASE command is assumed to be a programmer's comment.</u>

Debugging Techniques :

Murphy's Law dedicates that the total number of programs in the universe that run correctly the first time is always less than one.

dBASE III PLUS provides debugging tools that can be used to help find errors and correct them, thereby making the overall programming task a bit easier. For example, when dBASE encounters an error in a program, it displays the line with the error in it, the program(s) in which the error occurred, and the warning message:

```
Cancel, Ignore, or Suspend? (C, I, or S )
```

These options have the following effects:

. Cancel: Completely terminates the program and return to the dBASE dot prompt.

. Suspend: Temporarily terminates the program and returns to the dBASE dot prompt displaying the message: "Do Suspend".
- Private variables (those created within the program) are not erased.
- The program can be resumed at any time by entering the command: RESUME at the dot prompt.

. Ignore: Ignores the error and attempts to continue processing at the next line in the program.

DISPLAY Commands :

The various DISPLAY commands allow to view the status of memory variables, open files, active index files, and other useful items of information that may be the cause of errors.

DISPLAY MEMORY :

If you Suspend the program (rather than Cancel it or Ignore the error) and entered the command: DISPLAY MEMORY at the dot prompt, the names, contents, and data types of all active memory variables are displayed.

If necessary, use the RESUME command to return the program, press ESC to terminate the program, or use MODIFY COMMAND to edit the program.

DISPLAY MEMORY might also help in detecting the "Data Type Mismatch" error.

DISPLAY STRUCTURE :

This command displays the dataBase file under consideration including field names. This command can be used to check for the existence (and correcting spelling) of field names used in the program.

DISPLAY STATUS :

This command displays the names of all open dataBase files as well as the names and contents of all active (opened) index files.

SET ECHO ON : (very important)

<u>This command allows to see each line in a program while it is running. To see the entire program echoed, just type SET ECHO ON from the dot prompt before running the program. Or, if you know that an error is somewhere in a small part of the program, put the SET ECHO ON command right before that spot in the program. (Remember to use SET ECHO OFF when you have finished debugging).</u>

SET STEP ON :

With both ECHO and SET STEP ON, dBASE displays each line of the program as it is being processed, pauses after each line, and displays this message:

Press Space to step, S to suspend, or ESC to Cancel...

THe STEP option allows to control the progress of the program, so you can watch the logic of the program unfold as dBASE executes each command. This is a good technique for finding those logical errors that let a program run without crashing, but also without doing exactly what you had in mind. If the STEP option does not let you solve the problem, you can break out the long program with the following DEBUG command.

SET DEBUG ON : (very important)

This command sends every echoed statement in the program to the printer. With DEBUG, it is best to use SET ECHO ON and SET STEP OFF. You can use the printed copy of the echoed lines to study each step in the program in a concentrated fashion.

Chapter 2

Personnel Management System Design

Writing a custom software system is much like writing a book: it is a highly creative task that usually starts as a very general idea and finally grows into a polished working product.

Any software project, large or small, can be broken into a series of about 6 steps, starting with a basic idea and ending with a finished product.

The 6 different steps are:

1. Define the goal of the project and the user level (Project Definition).
2. Specify the input and output (I/O Specification).
3. Design the dataBase structure (dataBase Design).
4. Isolate specific program functions (Modular Program Design).
5. Write the individual programs (Modular Program Writing).
6. Test and make corrections (Modular Testing and Debugging).

Each step will be discussed in more detail as follows:

1. Project Definition :

Project definition for a software project usually starts out as a vague description of the actual project, such as "Create a Personnel Management System". We will need ,however, to be a bit more specific if we want project development to go smoothly. For example, who is going to use this system ?. The experience level of the end user is a key element in system design, so the project definition can be refined to include it. For example, "Create a Personnel Management System that can be used by an individual with little or no computer experience".

To define the project more specifically, think in terms of the specific tasks that a personnel dataBase management system can perform :

- Add information to a dataBase
- Sort data into a meaningful order
- Search for sets of data by type or range
- Display data in any report format designed, including calculations and summaries.
- Allow changes and deletions to a dataBase file
- Check for duplications in the dataBase file
- View selected data in the dataBase file

It becomes more easier to define the task more specifically as follows:

Create a personnel management system that allows an inexperienced user to do the following:

- Add new names and their information to the dataBase file
    - Use a custom screen rather than the usual APPEND screen.
- Sort the data into alphabetical order by first and last name or by birth date for bulk mailing
- Select specific data by name, birth date, city, or hiring date
- Print mailing labels and a directory, and make a merge file for printing form letters
- Make changes or delete names and associated information to the dataBase file.
- Check the dataBase file for duplicate RECORDs
    - Compare first names, last names, address, and zip codes for each RECORD with every other RECORD in the dataBase, and allow the user to remove duplicates.
- View selected data in the dataBase file.

Now, the large task of creating a personnel management system is clearly specified and, more importantly, broken down into smaller tasks that are relatively easy to accomplish.

2. I/O Specification :

The next step in developing a custom software system is to think about the project in terms of input and output.

In the Personnel Management System, we want the computer to produce three items:

1. Mailing labels, which contain (for example):

   Name,
   Company (Center, or Department),
   Address,
   City, and
   Governrate.

2. A Directory, which contains:

   Name,
   Company,
   Address,
   City,
   Governrate,
   Phone number, and
   Hiring date.

3. A File for Form Letters. A form letter file contains:

   Name,
   Company,
   Address,
   City, and
   Governrate.

Now defining the output really determines the input, so after removing the redundancies in the output, the input (according to this example) must be:

   Name,
   Company,
   Address,
   City,
   Governrate,
   Phone number, and
   Hiring date.

3. DataBase Design :

After deciding what information needs to be stored, it could be easy to begin designing the database.

At this stage of process, go ahead and load dBASE and create the dataBase file:

```
. CREATE PERSONEL
```

The following shows the dataBase structure of PERSONEL.DBF :

Field	Field name	Type	Width	Dec
1	ID_CARD	Character	7	
2	F_NAME	Character	10	
3	M_NAME	Character	10	
4	L_NAME	Character	10	
5	HIR_DATE	Character	8	
6	TITLE	Character	20	
7	JOB	Character	20	
8	CENTERDEPT	Character	20	
9	SALARY	Numeric	6	2
10	ALLOWANCES	Numeric	6	2
11	M_STATUS	Character	8	
12	NO_OF_CHLD	Numeric	2	0
13	ADDRESS	Character	50	
14	CITY	Character	25	
15	GOVERNRATE	Character	15	
16	HPHONE	Character	12	
17	WPHONE	Character	3	
18	DEGREE	Character	15	
19	AWARD_DATE	Date	8	
20	MAJOR_FLD	Character	20	
21	BIRTH_DATE	Date	8	
22	BIRTH_PLAC	Character	25	
23	NOTES	Memo	10	

Now, we need to consider what index files we need. We will need the following index files for the PERSONEL dataBase file:

```
USE PERSONEL
INDEX ON UPPER(trim(F_NAME)) TO FNAMES


USE PERSONEL
INDEX ON UPPER(trim(L_NAME)) TO LNAMES


USE PERSONEL
INDEX ON BIRTH_DATE  TO BDATES


USE PERSONEL
INDEX ON UPPER(trim(GOVERNRATE))+UPPER(trim(CITY)) TO ZIPS


USE PERSONEL
INDEX ON UPPER(TITLE)+UPPER(F_NAME) TO TITLE

USE PERSONEL
INDEX ON UPPER(JOB)+UPPER(F_NAME) TO JOB
```

Note:

The UPPER function is used to make all names upper case, for more accurate sorting and easier searching.

. Modular Program Design :

The easiest way to develop any software system is to break it down nto small programs, each program for one task.

To make ,for example,the Personel Management System easy to use, the irst program, the main menu program. displays when it runs a menu of ptions as follows:

```
-------------------------------------------------------------------------
 Personel Management System Main Menu          04/19/89 12:58:19        |
 ---------------------------------------------------------------------  |
                                                                        |
        Would you like to :                                             |
        -------------------                                             |
                                                                        |
              1. Add new members                                        |
                                                                        |
              2. Change/Delete data                                     |
                                                                        |
              3. Print system information                               |
                                                                        |
              4. Check for duplicates                                   |
                                                                        |
              5. View selected data                                     |
                                                                        |
                                                                        |
              6. Exit the system                                        |
                                                                        |
      Enter your choice (1-6)                                           |
                                                                        |
-------------------------------------------------------------------------
```

In modular programming design, it could be better to draw a block diagram for the various tasks of the software structure to see how the various parts of the system are related to each other as follows:

```
                    ------------------------------
                   | Main Menu                    |
                   | ---------                    |
                   |  1. Add new members          |
                   |  2. Change/Delete data       |
                   |  3. Print system information |
                   |  4. Check for duplicates     |
                   |  5. View selected data       |
                   |  6. Exit                     |
                    ------------------------------
                                 |
      ----------------------------------------------------------------
      |         |             |                        |             |
 ----------     |    -------------------         ------------    ----------
| Add new  |    |   | Select sort         |     | Check for  |  | View     |
| members  |    |   | and search options, |     | duplicate  |  | selected |
 ----------     |   | then print the data |     | names and  |  | data     |
                |    -------------------      | information|   ----------
                |                               ------------
       -----------------
      | Edit or Change  |
      | the system data |
       -----------------
```

We can see from this block diagram that the Personel Management Syste will consist of 5 programs :

1. The Main Menu program (the master program),

2. A program to add new members,

3. A program to allow changes and deletions, and

4. A program to select and print data in a chosen order and format,

5. A program to check for duplicates.

6. A program for viewing selected data.

5-6. Modular Program Writing,Testing, and Debugging :

In this phase, the different programs of the entire system are created. Then, they will be tested and debugged to be sure they actually work. Chapter 3 will be dealt with these tasks.

Chapter 3

Modular Program Writing, Testing , and Debugging

3.0 Main Menu Program :

A main menu needs to perform 3 main tasks :

1. Set up system parameters.
2. Present a list of options to user and wait for a selection.
3. Branch to the appropriate program to perform the task the user selected.

Main Menu Program's Algorithm :

```
- Set up system parameters

- Set up loop to perform tasks and then repeat menu

     . Clear

     . Present menu of options:

        1. Add new members

        2. Change/Delete data

        3. Print system information

        4. Check for duplicates

        5. Exit the system

     . Wait for user selection

     . Branch to appropriate program

        . If Choice = 1, branch to Append program

        . If Choice = 2, branch to Editing program

        . If Choice = 3, branch to Printing Reports program
```

. If Choice = 4, branch to Check Duplicates program

. If Choice = 5, Exit this system

. On return to Main Menu program, re-display the menu

Program Writing :

In this instance, you can use the dBASE MODIFY COMMAND editor, or any text editor, or word processor. But be forewarned that a large program barely fits into the MODIFY COMMAND editor. If you must use MODIFY COMMAND, you can conserve memory by leaving out programmer comments, blank lines, and even indentations if necessary. (The program might also run a little faster if you do so).

```
*
* Program..: PMenu.PRG
* Date.....: April, 1989
* Remark...: Main menu for the Personnel Management System
*
SET CONFIRM ON
SET STATUS OFF
SET TALK OFF
SET HELP OFF
SET BELL OFF
SET SCOREBOARD OFF
SET HEADING OFF
*
* Create underline variable, Uline
*
Uline = REPLICATE("-",70)
*
* Display menu and get user's choice
*
CLEAR
Choice =0
DO WHILE Choice # 6
   *
   @ 2, 5 SAY "Personnel Management System Main Menu"
   @ 2,55 SAY DTOC(DATE()) + "     " + TIME()
   @ 3, 5 SAY Uline
   *
   *
   TEXT


          1. Add new members

          2. Change/Delete data

          3. Print system information

          4. Check for duplicates

          5. View specific information


          6. Exit the system

   ENDTEXT
   *
   @20,5 SAY "Enter your choice (1-6)" GET Choice PICT "9";
                                                RANGE 1,6
   READ
   *
```

```
 *
 * Branch to the appropriate program
 *
 DO CASE
    CASE Choice = 1
         DO AddNew
    CASE Choice = 2
         DO EditDel
    CASE Choice = 3
         DO Reports
    CASE Choice = 4
         DO DupCheck
    CASE Choice = 5
         DO VUINFO
 ENDCASE
 *
ENDDO (WHILE Choice # 6)
*
CLOSE DATABASES
CLEAR
* QUIT
* eof()
```

```
                                                         05/07/89      00:15:42
-------------------------------------------------------------------------------
Personnel Management System Main Menu
-------------------------------------------------------------------------------

            1. Add new members

            2. Change/Delete data

            3. Print system information

            4. Check for duplicates

            5. View specific information

            6. Exit the system

      Enter your choice (1-6) ]
```

Note that :

- The program begins by a name of the program and a brief description of what it does, then set the various dBASE parameters to be used in the personnel management system. (Remember that the lines preceded by asterisks are programmer comments. These are just information and do not affect the program in any way)

- The SET commands control the various system parameters:

  . SET CONFIRM ON : Determines that pressing the RETURN-Key is necessary after filling a screen prompt.

  . SET STATUS OFF : removes the status bar from the bottom of the screen.

  . SET DEFAULT TO B : makes drive B the default drive for accessing the disk.

  . The TALK, HELP, and BELL parameters control miscellaneous dBASE screen messages that would not be needed in the present personnel management system, so all of them are SET OFF.

- The first 3 @ ... SAY statements create a screen heading to identify the menu and display the current system date (DTOC(DATE())) and Time (TIME()).

- The TEXT segment displays the actual menu choices.

- The user's response is stored in the memory variable Choice (GET Choice).

- The PICTure statement ensures that only a single numeral will be accepted by the program, and the RANGE option ensures that the number will be between 1 and 5 (inclusive).

- The READ statement activates the GET Choice command and waits for the user to enter a number.

- With the DO CASE clause, the program will branch to the appropriate program, perform the selected task, and return to the main menu program.

- By closing the DO WHILE loop (ENDDO), the program re-display the menu.

- The last 3 statements in the program:

```
        CLOSE DATABASES
        CLEAR
        *QUIT
```

are used to close the dataBase files, clear the screen, and quit dBASE.

The asterisk in front of the QUIT command is placed to inactivate it for now, since it will be more convenient during development of the system to return to the dBASE dot prompt than to the DOS prompt. After all the programs are written and tested, this asterisk will be removed.

APPEND and EDIT/CHANGE Programs :

3.1.a APPEND Program
AddNames.FMT Program :

```
* Program..: AddNames.FMT
* Date.....: June, 1989
* Remark...: Custom Screen for Adding New Names with their Information
*
@ 2, 4 SAY "Personnel dataBase Management System:  Add New Members"
@ 2,69 SAY "Screen 1"
@ 4, 1 SAY "Identification Card:"
@ 4,22 GET PERSONEL->ID_CARD PICT "@!"
@ 4,33 SAY "Center/Department:"
@ 4,52 GET PERSONEL->CENTERDEPT PICT "@!"
*
@ 6, 1 SAY "First Name:"
@ 6,13 GET PERSONEL->F_NAME PICT "@!"
@ 6,27 SAY "Middle Name:"
@ 6,40 GET PERSONEL->M_NAME PICT "@!"
@ 6,55 SAY "Last Name:"
@ 6,66 GET PERSONEL->L_NAME PICT "@!"
*
@ 8, 1 SAY "Hiring Date:"
@ 8,14 GET PERSONEL->HIR_DATE
@ 8,24 SAY "Title:"
@ 8,31 GET PERSONEL->TITLE PICT "@!"
@ 8,53 SAY "Job:"
@ 8,58 GET JOB PICT "@!"
*
@10, 1 SAY "Marital Status:"
@10,17 GET PERSONEL->M_STATUS PICT "@!"
@10,27 SAY "Number of children:"
@10,47 GET PERSONEL->NO_OF_CHLD PICT "99"
@10,51 SAY "Salary:"
@10,59 GET PERSONEL->SALARY PICT "999.99"
*
@12, 5 SAY "Birth Date:"
@12,17 GET PERSONEL->BIRTH_DATE PICT "99/99/99"
@12,31 SAY "Place of Birth:"
@12,47 GET PERSONEL->BIRTH_PLAC
*
@14, 1 SAY "Allowances:  (1) "
@14,18 GET PERSONEL->ALLOWANCES PICT "999.99"
*
*
```

```
*
@15, 1 SAY "Deductions:  (1)"
*
*
*
*
@18, 2 SAY "CURSOR"
@18,26 SAY "DELETE"
@18,51 SAY "RECORD"
*
@19, 4 SAY "Character: "+CHR(26)+" "+CHR(27)
@19,28 SAY "Character: Del"
@19,53 SAY "Previous Record: PgUp"
*
@20, 9 SAY "Word: Home End"
@20,32 SAY "Field: ^Y"
@20,57 SAY "Next Record: PgDn"
*
@21, 8 SAY "Field: "+CHR(24)+" "+CHR(25)
@21,31 SAY "Record: ^U"
@21,59 SAY "Done/Save: ^End"
*
@22, 2 SAY "Insert Mode: INS"
@22,61 SAY "Abandon: Esc"
*
@ 1, 0 TO  3,79
@17, 0 TO 23,79
@18,24 TO 22,24
@18,49 TO 22,49
*
*
*
READ
*
```

```
*
@ 2, 4 SAY "Personnel dataBase Management System:  Add New Members"
@ 2,69 SAY "Screen 2"
*
@ 4, 1 SAY "Home Address:"
@ 4,15 GET PERSONEL->ADDRESS
*
@ 6, 9 SAY "City:"
@ 6,15 GET PERSONEL->CITY PICT "@!"
@ 6,37 SAY "Governrate:"
@ 6,49 GET PERSONEL->GOVERNRATE PICT "@!"
@ 6,66 SAY "Code:"
@ 6,72 GET PERSONEL->ZIP_CODE
*
@ 8, 3 SAY "Home Phone:"
@ 8,15 GET PERSONEL->HPHONE
@ 8,37 SAY "Work Phone:"
@ 8,49 GET PERSONEL->WPHONE
*
@10, 1 SAY "Last Certificate:"
@10,19 GET PERSONEL->DEGREE PICT "@!"
@10,36 SAY "Award Date:"
@10,48 GET PERSONEL->AWARD_DATE PICT "99/99/99"
*
@12, 6 SAY "Major Field:"
@12,19 GET PERSONEL->MAJOR_FLD PICT "@!"
*
@15, 1 SAY "Notes:"
@15, 8 GET NOTES
*
*
*
@18, 2 SAY "CURSOR"
@18,26 SAY "DELETE"
@18,51 SAY "RECORD"
*
@19, 4 SAY "Character: "+CHR(26)+" "+CHR(27)
@19,28 SAY "Character: Del"
@19,53 SAY "Previous Record:  PgUp"
*
@20, 9 SAY "Word: Home End"
@20,32 SAY "Field: ^Y"
@20,57 SAY "Next Record: PgDn"
*
@21, 8 SAY "Field: "+CHR(24)+" "+CHR(25)
@21,31 SAY "Record: ^U"
@21,59 SAY "Done/Save: ^End"
*
@22, 2 SAY "Insert Mode: INS"
@22,61 SAY "Abandon: Esc"
```

```
*
@ 1, 0 TO  3,79
@17, 0 TO 23,79
@18,24 TO 22,24
@18,49 TO 22,49
*
*
* eof()
```

3.1.b AddNew.PRG Program :

That we have a custom screen for adding new RECORDs to the PERSONEL.DBF file,we can create a program that uses this custom screen. This program will be called AddNew.PRG.

```
* Program..: AddNew.PRG
* Date.....: April, 1989
* Remark...: To add new members using the ADDNAMES.FMT
*
USE PERSONEL INDEX FNAMES , LNAMES , BDATES
*
SET FORMAT TO ADDNAMES
APPEND
*
CLOSE FORMAT
REINDEX
CLEAR
RETURN
* eof()
```

```
 Personnel dataBase Management System:  Add New Members          Screen 1

Identification Card:                 Center/Department:

First Name:                Middle Name:                  Last Name:

Hiring Date:   /  /    Title:                           Job:

Marital Status:            Number of children:        Salary:      .

    Birth Date:   /  /        Place of Birth:

Allawances:   (1)        .
Deductions:   (1)

 CURSOR                    | DELETE                    | RECORD
   Character:  ←    →      |   Character: Del          |   Previous Record: PgUp
        Word: Home End     |       Field: ^Y           |       Next Record: PgDn
       Field: ↑↓           |      Record: ^U           |        Done/Save: ^End
 Insert Mode: INS          |                           |          Abandon: Esc
```

```
 Personnel dataBase Management System:  Add New Members          Screen 2

Home Address:

        City:                     Governrate:                   Code:

  Home Phone:          .          Work Phone:

Last Certificate:                Award Date:   /  /

      Major Field:

Notes: memo

 CURSOR                    | DELETE                    | RECORD
   Character: ←    →       |   Character: Del          |   Previous Record:  PgUp
        Word: Home End     |       Field: ^Y           |       Next Record: PgDn
       Field: ↑ ↓          |      Record: ^U           |        Done/Save: ^End
 Insert Mode: INS          |                           |          Abandon: Esc
```

```
+--------------------------------------------------------------------------------+
|  Personnel dataBase Management System:  Edit/Change Records        Screen 1    |
+--------------------------------------------------------------------------------+
Identification Card: M-6974     Center/Department: OPERATIONS RESEARCH

First Name: ABDALLA        Middle Name: ABDEL-AZIZ       Last Name: EL-DAOUSHY

Hiring Date: 01/01/66  Title: ASSISTANT PROFESSOR   Job: ASSTANT PROFESSOR

Marital Status: MARIED     Number of children:  2   Salary: 200.00

    Birth Date: 02/20/41        Place of Birth:  MENOUFIA

Allawances:  (1) 999.99
Deductions:  (1)

+--------------------------------------------------------------------------------+
| CURSOR                 | DELETE                  | RECORD                      |
|   Character: -->  <--  |   Character: Del        |   Previous Record: PgUp     |
|        Word: Home End  |       Field: ^Y         |       Next Record: PgDn     |
|       Field: ↑↓        |      Record: ^U         |     Done/Save: ^End         |
| Insert Mode: INS       |                         |        Abandon: Esc         |
+--------------------------------------------------------------------------------+
```

```
+--------------------------------------------------------------------------------+
|  Personnel dataBase Management System:  Edit/Change Records        Screen 2    |
+--------------------------------------------------------------------------------+
Home Address: 24 ABU HAYYAN EL-TAWHIDI STREET

        City: NASR CITY                  Governrate: CAIRO               Code:

  Home Phone: 2622149                    Work Phone: 603166

Last Certificate: PH.D.                 Award Date: 09/05/78

    Major Field: OPERATIONS RESEARCH


Notes: memo

+--------------------------------------------------------------------------------+
| CURSOR                 | DELETE                  | RECORD                      |
|   Character: -->  <--  |   Character: Del        |   Previous Record:  PgUp    |
|        Word: Home End  |       Field: ^Y         |     Next Record: PgDn       |
|       Field: ↑↓        |      Record: ^U         |     Done/Save: ^End         |
| Insert Mode: INS       |                         |        Abandon: Esc         |
+--------------------------------------------------------------------------------+
```

Note that :

1- Graphics characters (such as arrow keys) are used in this custom screen program by using the appropriate CHR code. For example, to print a ← , use CHR(27). The → is CHR(26). CHR(24) is a ↑ , and CHR(25) is a ↓ .
Also, you can create a reasonable facsimile of the Return (or Enter) key using CHR(17)+"-".

The main trick is to place these special characters in the FORmat (.FMT) program without disrupting the alignment of things.

2. By the way, for a quick look at all the ASCII characters available on your screen, you can use the following program:

```
* Program..: ASCII.PRG
* Remark...: To display all ASCII codes.
* Date.....: April, 1989
*
SET TALK OFF
I = 0
DO WHILE I <= 255
   ?? STR(I,3) , CHR(I) + " "
   I = I + 1
ENDDO (WHILE I <= 255)
* eof()
```

3.2.a EditDel.PRG Program
      Editing/Changing/Deleting Program :
===

```
* Program..: EditDel.PRG
* Date.....: April, 1989
* Remark...: Edit and delete members using EdNames.FMT
*
USE PERSONEL INDEX LNAMES
*
More = .T.
DO WHILE More
   CLEAR
   @ 2, 5 SAY "Edit/Delete RECORDs"
   @ 2,55 SAY DTOC(DATE()) + "   " + TIME()
   @ 3, 5 SAY Uline
   *
   STORE SPACE(20) TO Mem_F_NAME , Mem_L_NAME
   *
   @10, 5 SAY "First name :" GET Mem_F_NAME
   @12, 5 SAY " Last name :" GET Mem_L_NAME
   @14,10 SAY "Enter name of dataBase to Edit/Delete, or RETURN to Exit"
   READ
   *
   * Exit if no First name  entered
   *
   IF Mem_F_NAME = " "
      EXIT
   ENDIF
   *
   * If Name entered, create search string....
   *
   Search = TRIM(UPPER(Mem_L_NAME))
   *
   * Try to find that individual...
   *
   SEEK Search
   *
   * If found, Ask for Edit for Change or Delete
   *
```

```
*
IF FOUND()
   *
   *
   CLEAR
   @ 5, 1
   DISP ALL TRIM(F_NAME)+" "+TRIM(L_NAME) FOR TRIM(L_NAME)=Search
   *
   WAIT
   @20,0 CLEAR
   CD = SPACE(1)
   DO WHILE CD = SPACE(1)
      *
      @20, 5 SAY "Option{ C(hange) or D(elete) } :";
                                               GET CD PICT "A"
      READ
      *
      DO CASE
          CASE UPPER(CD)="C"   && Edit for change
               CHUZ = "Change"
          CASE UPPER(CD)="D"
               CHUZ = "Delete"
          OTHERWISE
               LOOP
      ENDCASE
   ENDDO  ( DO WHILE CD=" " )
   *
   Rec_No = SPACE(3)
   DO WHILE Rec_No = SPACE(3)
      *
      @22, 5 SAY "Which RECORD to &CHUZ. :" GET Rec_No
      READ
      *
      IF Rec_No = " " .OR. VAL(Rec_No) > RECCOUNT()
         @23,10 SAY "No such Record No., Please try again!"
         WAIT
        CLEAR
        LOOP
      ENDIF
   ENDDO  ( DO WHILE Rec_No = SPACE(3) )
   *
   GO VAL(Rec_No)
   DO CASE
      CASE CHUZ = "Change"
           *
           * Edit this RECORD...
           *
           SET FORMAT TO EDNAMES
           EDIT VAL(Rec_No)          && ! ! ! Edit only one! how?
           SET FORMAT TO
           CLEAR
           LOOP
          CASE CHUZ = "Delete"
                DELETE
                @23, 5 SAY "This RECORD is Deleted !"
                WAIT
               CLEAR
               LOOP
       ENDCASE
       *
     ELSE   && IF FOUND()
       *
       @14, 0 CLEAR
       @15,10 SAY "Not found ! "
       ? CHR(7)
       WAIT
    ENDIF  ( FOUND() )
    *
ENDDO (WHILE More = .T.)
*
* Done Editing. Ask about PACKing the dataBase.
*
CLEAR
YesNo = " "
@10, 5 SAY "PACK RECORDs marked for deletion now? (Y/N) ";
                 GET YesNo PICT "!"
READ
*
IF YesNo = "Y"
   SET TALK ON
   PACK
   SET TALK OFF
ENDIF
CLEAR
RETURN
* eof()
```

3.2.b Edit/Change Program
      EdNames.FMT Program :

```
* Program..:
* Date.....: June, 1989
* Remark...: Custom Screen for Adding New Names with their Information
*
@ 2, 4 SAY "Personnel dataBase Management System:  Edit/Change Records "
@ 2,69 SAY "Screen 1"
@ 4, 1 SAY "Identification Card:"
@ 4,22 GET PERSONEL->ID_CARD PICT "@!"
@ 4,33 SAY "Center/Department:"
@ 4,52 GET PERSONEL->CENTERDEPT PICT "@!"
*
@ 6, 1 SAY "First Name:"
@ 6,13 GET PERSONEL->F_NAME PICT "@!"
@ 6,27 SAY "Middle Name:"
@ 6,40 GET PERSONEL->M_NAME PICT "@!"
@ 6,55 SAY "Last Name:"
@ 6,66 GET PERSONEL->L_NAME PICT "@!"
*
@ 8, 1 SAY "Hiring Date:"
@ 8,14 GET PERSONEL->HIR_DATE
@ 8,24 SAY "Title:"
@ 8,31 GET PERSONEL->TITLE PICT "@!"
@ 8,53 SAY "Job:"
@ 8,58 GET JOB PICT "@!"
*
@10, 1 SAY "Marital Status:"
@10,17 GET PERSONEL->M_STATUS PICT "@!"
@10,27 SAY "Number of children:"
@10,47 GET PERSONEL->NO_OF_CHLD PICT "99"
@10,51 SAY "Salary:"
@10,59 GET PERSONEL->SALARY PICT "999.99"
*
@12, 5 SAY "Birth Date:"
@12,17 GET PERSONEL->BIRTH_DATE PICT "99/99/99"
@12,31 SAY "Place of Birth:"
@12,47 GET PERSONEL->BIRTH_PLAC
*
@14, 1 SAY "Allowances:  (1) "
@14,18 GET PERSONEL->ALLOWANCES PICT "999.99"
*
*
```

```
*
@15, 1 SAY "Deductions:  (1)"
*
*
*
*
@18, 2 SAY "CURSOR"
@18,26 SAY "DELETE"
@18,51 SAY "RECORD"
*
@19, 4 SAY "Character: "+CHR(26)+" "+CHR(27)
@19,28 SAY "Character: Del"
@19,53 SAY "Previous Record: PgUp"
*
@20, 9 SAY "Word: Home End"
@20,32 SAY "Field: ^Y"
@20,57 SAY "Next Record: PgDn"
*
@21, 8 SAY "Field: "+CHR(24)+" "+CHR(25)
@21,31 SAY "Record: ^U"
@21,59 SAY "Done/Save: ^End"
*
@22, 2 SAY "Insert Mode: INS"
@22,61 SAY "Abandon: Esc"
*
@ 1, 0 TO  3,79
@17, 0 TO 23,79
@18,24 TO 22,24
@18,49 TO 22,49
*
*
*
READ
*
@ 2, 4 SAY "Personnel dataBase Management System:  Edit/Change Records"
@ 2,69 SAY "Screen 2"
*
@ 4, 1 SAY "Home Address:"
@ 4,15 GET PERSONEL->ADDRESS
*
@ 6, 9 SAY "City:"
@ 6,15 GET PERSONEL->CITY PICT "@!"
@ 6,37 SAY "Governrate:"
@ 6,49 GET PERSONEL->GOVERNRATE PICT "@!"
@ 6,66 SAY "Code:"
@ 6,72 GET PERSONEL->ZIP_CODE
*
@ 8, 3 SAY "Home Phone:"
@ 8,15 GET PERSONEL->HPHONE
@ 8,37 SAY "Work Phone:"
@ 8,49 GET PERSONEL->WPHONE
*
```

```
*
@10, 1 SAY "Last Certificate:"
@10,19 GET PERSONEL->DEGREE PICT "@!"
@10,36 SAY "Award Date:"
@10,48 GET PERSONEL->AWARD_DATE PICT "99/99/99"
*
@12, 6 SAY "Major Field:"
@12,19 GET PERSONEL->MAJOR_FLD PICT "@!"
*
@15, 1 SAY "Notes:"
@15, 8 GET NOTES
*
*
*
@18, 2 SAY "CURSOR"
@18,26 SAY "DELETE"
@18,51 SAY "RECORD"
*
@19, 4 SAY "Character: "+CHR(26)+" "+CHR(27)
@19,28 SAY "Character: Del"
@19,53 SAY "Previous Record:  PgUp"
*
@20, 9 SAY "Word: Home End"
@20,32 SAY "Field: ^Y"
@20,57 SAY "Next Record: PgDn"
*
@21, 8 SAY "Field: "+CHR(24)+" "+CHR(25)
@21,31 SAY "Record: ^U"
@21,59 SAY "Done/Save: ^End"
*
@22, 2 SAY "Insert Mode: INS"
@22,61 SAY "Abandon: Esc"
*
@ 1, 0 TO  3,79
@17, 0 TO 23,79
@18,24 TO 22,24
@18,49 TO 22,49
*
*
* eof()
```

```
Edit/Delete RECORDs                                  05/07/89   00:27:51
------------------------------------------------------------------------


First name :

 Last name :

     Enter name of dataBase to Edit/Delete, or RETURN to Exit
```

```
Edit/Delete RECORDs                                  05/07/89   00:27:51
------------------------------------------------------------------------


First name : Abdalla

 Last name : El-Daoushy

     Enter name of dataBase to Edit/Delete, or RETURN to Exit




  4  ABDALLA EL-DAOUSHY
 14  ABDALLA EL-DAOUSHY
 15  ABDALLA EL-DAOUSHY

Press any key to continue...
```

3.3.a Mailing Labels and Current Directory :

One of the main functions of the current system is to create mailing labels and current directory.

In this section, the format files for the labels and directory will be developed; then in the next section, the formats will be used in a program that allows presorting and searching.

dBASE and Report Generator :

1. Mailing Labels :

To create a mailing label format for the system, the format file must has a name. Two-across format (two columns of labels on each sheet) will be created, so let this format file be called TwoCol.LBL.

- Now, use the following commands:

```
. USE PERSONEL
. CREATE LABEL  TwoCol
```

then, follow the on screen instructions to create the desired file.

- To display the labels on the screen, type the command:

```
. LABEL FORM TwoCol
```

The labels will appear on the screen as ,for example, follows :

```
Options                Contents                    Exit  01:54:01 am
                 ┌──────────────────────────────────────────────────┐
                 │ Label contents 1:  TRIM(F_NAME)+" "+L_NAME        │
                 │                2:  TRIM(CENTERDEPT)               │
                 │                3:  TRIM(ADDRESS)                  │
                 │                4:  TRIM(CITY)                     │
                 │                5:  (GOVERNRATE)                   │
                 └──────────────────────────────────────────────────┘
```

To send the label output to the printer, just type :

```
. LABEL FORM TO PRINT
```

Both of the above LABEL FORM commands will be used in the Reports.PRG program discussed in the section after the next one.

2. Custom Reports :

The present system allows to print a directory of the members.

There are two alternatives to create this directory: either by using the dBASE III PLUS Report Generator or by dBase programming techniques as in the next section.

To start designing the report, call the file Director.FRM nd use the following dBASE commands:

```
. USE PERSONEL INDEX FNAMES , LNAMES , ZIPS
. CREATE REPORT Director
```

then follow the on screen instructions to create the desired report.

Hint:

- Highlight the Groups option on the top menu. Select the "Group on expression" option. Enter the following expression:

```
TRIM(F_NAME)+" "+TRIM(L_NAME)+SPACE(40-LEN(TRIM(F_NAME))+
                              LEN(TRIM(M_NAME))+1))+CENTERDEPT
```

- Press ^PgDn to zoom in and see more of the formula at the bottom of the screen.

- Press ^PgUp to zoom back out.

- The middle portion of the expression above :

```
+SPACE(40-(LEN(TRIM(F_NAME))+LEN(TRIM(M_NAME))+1))
```

ensures that the CENTERDEPT field always starts on the 40th column of the Report.

How?:

By adding a certain number of spaces, calculated by subtracting the combined length of the first and last names (with blanks removed) plus 1 for the space. Hence, if a person's name contains 30 characters (with a space between the first and last names), this portion of the expression places 10 spaces (40 minus 30) after the person's name. The last portion of the expression: CENTERDEP prints the center/department name to the right of the name and calculated number of spaces (always beginning in the 40th column).

To test the report, enter these commands:

```
. USE PERSONEL INDEX FNAMES , LNAMES
. REPORT FOR Director
```

The directory will be seen similar .for example, to the following one:

```
Report Options                                       05/09/89   01:45:13
------------------------------------------------------------------------

             1. Mailing Labels

             2. Directory

             3. Form-letter file

             4. None (RETURN to Main Menu)


Enter choice (1-4)  2



Sort options                                         05/09/89   01:45:43
------------------------------------------------------------------------

             1. Alphabetical order by name

             2. Governrate and City order

             3. Original (Unsorted) order


Enter choice (1-3)  1

                        (A)ll RECORDs or (Q)uery?  A
```

```
                                                  05/09/89  Page  1
System Directory
------------------------------------------------------------------------

ABDALLA EL-DAOUSHY
OPERATIONS RESEARCH
24 ABU HAYYAN EL-TAWHIDI STREET                    2622149         603166
NASR CITY,CAIRO                                                    01/01/66

ABDEL-KADER HAMZA
OPERATIONS RESEARCH / INP
24 Mokhles El-Alfi street                          2611534         603166
NAST CITY,CAIRO                                                    01/01/64

MOHAMMED YOUSSEF
INDUSTRIAL PLANNING
Oxford street                                      2473875         603166
HELIOPOLIS,CAIRO                                                    /  /

ABDELHAMID EL-KASSAS
OPERATIONS RESEARCH
55 El-shaheed El-Haggar street                                     2622149
EL-ANDALOS,EL-NAKHL,CAIRO                                           /  /
```

To display the report on the printer rather than on the screen, use this command:

```
    . REPORT FORM Director TO PRINT
```

You can also specify that only certain RECORDs be included in the report. For example, to display the Director report for people whose birth date were in the year 1941, use the following command:

```
    . REPORT FORM Director FOR YEAR(BIRTH_DATE) = 41
```

Modifying the Report Format :

Once a Report Format has been created, it can be modified any time by using the appropriate dataBase and the MODIFY REPORT command. For example, to change the format of the Director report, enter these commands:

```
     . USE PERSONEL INDEX FNAMES , LNAMES , ZIPS
     . MODIFY REPORT Director
```

and use all the same techniques used to create the report.

3.3.b Sorting and Searching :

The Reports.PRG Program :

The program here is going to perform the following functions:

1. Display a menu of report options on the screen as follows:

    1. Mailing Labels

       This option will be based on the format created earlier in TwoCol.LBL file.

    2. Directory

       This option based on the Director.FRM created earlier.

    3. Form-letter file

       This option creates a special file to create form-letters.

    4. None (Return to Main Menu)

2. Once the user selects a report format, a second menu appears to ask the user how to organize the report as follows:

    1. Alphabetical order by name

    2. Governrate and City order (sometimes zip-code order)

    3. Original (Unsorted) order

3. After the user selects a sort order, the screen displays this the prompt:

   (A)ll RECORDs or (Q)uery?

- If the user types the letter A, all records from the dataBase will be displayed on the report.

- If the user types the letter Q, a dBASE III PLUS QUERY FORM appears on the screen. The user can select the File name, Operator, Constant/Expression, and Connect options in the usual manner to create a Filter condition. For example, the following Query form:

```
Set Filter              Nest              Display              Exit  02:32:35

  Field Name              HIR_DATE
  Operator                Less than or equal
  Constant/Expression     03/31/86
  Connect

  Line Number             1
```

Line	Field	Operator	Constant/Expression	Connect
1	HIR_DATE	Less than or equal	03/31/86	
2				
3				
4				
5				
6				
7				

will display only those RECORDs that have HIRing_DATEs in March, 1986.

Highlighting Exit on the top menu and selecting Save completes the query and filters out all RECORDs that do not match the query criteria.

The present program (Reports.PRG) will use the query temporarily to print the mailing labels, directory, or form-letter file, then immediately "Unfilter" the dataBase.

4. Finally, the program will ask the user whether the data should be displayed on the screen or printed:

Send data to Printer? (Y/N)

Program Algorithm :

```
- CLear screen

- Display menu of report options:

     . Select report format
           1. Mailing labels
           2. Directory
           3. Form-letter file
           4. None (return to main menu)

- Get user's menu choice
  If options 4 requested,
     return to main menu
  Otherwise,

     . Display menu of sort order:
           1. Alphabetical order by name
           2. Governrate and City order
           3. Original (Unsorted) order
     . Get user's menu choice
       If option 1 selected,
          USE INDEX of Last and First names (LNAMES.NDX)
       If option 2 selected
          USE INDEX of Governrate and City order (ZIPS.NDX)
       IF option 3 selected,
          USE PERSONEL dataBase without index file

- Ask if (A)ll or (Q)uery
  IF Query,
          MODIFY QUERY FORM

- Ask about hard copy
  IF going to printer
     Have user prepare printer
     Set up macro for sending report to printer
     Leave out RECORDs marked for deletion
     Execute the selection
  IF mailing labels requested,
     Print mailing labels
  IF directory report requested,
     Print directory
  If form-letter file requested,
     Ask form-letter file
```

- Done, pause screen

- Set up NAMES and ZIPS index files again for future use

- Remove filters

- Return to main menu.

Reports.PRG Program :

```
* Program..: Reports.PRG
* Date.....: April, 1989
* Remark...: To set up sort orders and search conditions,
*            then print the appropriate report.
*
USE PERSONEL INDEX FNAMES , ZIPS
*
CLEAR
@ 2, 5 SAY "Report Options"
@ 2,56 SAY DTOC(DATE()) + "    " + TIME()
@ 3, 5 SAY Uline
?
?
TEXT

              1. Mailing Labels

              2. Directory

              3. Form-letter file


              4. None (RETURN to Main Menu)

ENDTEXT
*
MChoice = 0
@20, 5 SAY "Enter choice (1-4) " GET MChoice PICT "9" RANGE 1,4
READ
*
IF MChoice = 4
   RETURN
ENDIF
*
* Ask for sort order
*
CLEAR
@ 2, 5 SAY "Sort options"
@ 2,56 SAY DTOC(DATE()) + "    " + TIME()
@ 3, 5 SAY Uline
?
?
```

```
*
TEXT

            1. Alphabetical order by name

            2. Governrate and City order

            3. Original (Unsorted) order

ENDTEXT
*
SChoice = 0
@20, 5 SAY "Enter choice (1-3) " GET SChoice PICT "9" RANGE 1,3
READ
*
* Set up appropriate sort order
*
DO CASE
   CASE SChoice = 1
        SET INDEX TO FNAMES
   CASE SChoice = 2
        SET INDEX TO ZIPS
   CASE SChoice = 3
        USE PERSONEL
ENDCASE
*
* Ask about query
*
CLEAR
AllSome = " "
@ 5, 5 SAY "(A)ll RECORDs or (Q)uery? " GET AllSome PICT "!"
READ
*
* Respond to query choice
*
IF AllSome = "Q"
   MODIFY QUERY PERSONEL
ENDIF
*
* Print report based on previous MChoice.
*
CLEAR
STORE " " TO Printer , PMacro
*
```

```
*
* If not making a form-letter file, ask about printer.
*
IF MChoice < 3
   @15, 5 SAY "Send data to Printer? (Y/N) " GET Printer PICT "!"
   READ
   IF Printer = "Y"
      PMacro = "TO PRINT"
      WAIT "Prepare Printer, then press any keyy to continue..."
   ENDIF
ENDIF
*
* Leave out RECORDs marked for deletion.
*
SET DELETED ON
CLEAR
*
DO CASE
   *
   * Print Mailing Labels...
   *
   CASE MChoice = 1

        LABEL FORM TwoCol &PMacro.

        *
        * Print Directory...
        *
   CASE MChoice = 2
        * REPORT FORM Director &PMacro    . . . ! ! !
        DO PrintDir
        *
        * Make a Form-letter file...
        *
   CASE MChoice = 3
        Filename = SPACE(14)
        SET CONFIRM ON
        @ 5, 0 CLEAR
        @15, 5 SAY "Enter name of Form-letter file (e.g., B:MMerge.TXT)";
                                                     GET Filename
        READ
        SET CONFIRM OFF
        COPY TO &Filename DELIMITED WITH "
ENDCASE
```

```
*
* Done. Return to Report Menu.
*
IF Printer = "Y"
   EJECT
ENDIF
*
WAIT "Press any key to RETURN to Report Menu..."
SET DELETED OFF
SET FILTER TO
CLEAR
RETURN
* eof()
```

Note that :

- Option 3 creates a special file that can be used with the WordStar MailMerge and MicroSoft Word programs to create form-letters.

  If the user requests option 3 ,to create a form-letter file, the program must ask for a name for the file, then copy the selected contents of the PERSONEL.DBF dataBase file to a Text file with the appropriate format for interfacing with Word or WordStar.

- The DELIMITED WITH " option used with the COPY command will ensure the appropriate format.

- The variable Filename is set to 14 SPACES to allow for a drive-specifier and an extension.

- When you ask to print mailing labels, the SAMPLE option will display two false labels as rows of asterisks. These shows how the actual labels will be printed. The prompt:

        Do you want more samples? (Y/N)

  will appear on the screen. If the labels are not properly aligned, adjust them in the printer and select Y(es) to print more samples. Repeat this process until the labels are properly aligned, then select N(o) to quit printing samples. The program will then print all the mailing labels in the system.

The Directory Program :

One program may be found with dBASE III PLUS. This problem is that the REPORT command is sometimes unused when formatting a report from a large dataBase with many fields. For this reason, a program called <u>PrintDir.PRG is developed here to print a formatted report without the use of the REPORT command.</u>

Program Algorithm :

```
- Set up LineCounter, PageCounter, and PageTitle.

- Start at the top of dataBase file

- Print the report title (PageTitle)

- Loop through each RECORD in the dataBase

      . Print first and last name
      . Print Center/Department
      . Print Address and Phone numbers
      . Format CITY, GOVERNRATE
      . Print City, Governrate, and Hiring Date
      . Print a blank line
      . Increment LineCounter by 5
      . IF report is being printed, handle pagination:
          . start on new page
          . increment PageCounter
          . print report title
          reset LineCounter

      . Skip to next dataBase RECORD.

- When Done, return to Reports Menu.
```

PrintDir.PRG Program :

```
* Program..: PrintDir.PRG
* Date.....: April, 1989
* Remark...: Print Directory for Personnel Management System
*            This program is used instead of the REPORT FORM command
*
* Initialize LineCount and Title variable
*
LineCount = 4
PageCount = 1
PageTitle = "System Directory"
*
* Start at top of dataBase file
*
GO TOP
*
IF Printer = "Y"      && comes from Reports.PRG
   *
   * Allow for Printer Abort.
   *
   CLEAR
   @20, 5 SAY "Press any key to Abort print job"
   *
   CLEAR TYPEAHEAD
   *
   ON KEY DO PrintStop
   *
   SET CONSOLE OFF
   SET PRINT ON
ENDIF
*
* Print title
*
? PageTitle + SPACE(37) + DTOC(DATE()) + "  Page " + STR(PageCount,2)
? Uline
?
?
*
```

```
*
* Loop through each RECORD in dataBase.
*
DO WHILE .NOT. EOF()
   ? TRIM(F_NAME)+" "+L_NAME
   ? TRIM(CENTERDEPT )
   x = trim(address)
   x = x +space(50-len(x))
   ? x, hphone, " " , wphone
   *
   FullCsz = TRIM(CITY)+","+GOVERNRATE
   FULLCsz = FullCsz + SPACE(60-LEN(FullCsz))+"      "+DTOC(HIR_DATE)
   ? FullCsz
   ?
   LineCount = LineCount + 5
   *
   * If Report is being printed, handle pagination.
   *
   IF Printer = "Y" .AND. LineCount >= 50
      EJECT
      PageCount = PageCount + 1
      ? PageTitle + SPACE(37) + DTOC(DATE()) + "  Page " + ;
                           STR(PageCount , 2)
      ? Uline
      ?
      ?
      LineCount = 4
   ENDIF
   SKIP
ENDDO (WHILE .NOT. EOF())
*
* Done. RETURN to Reports.PRG
*
SET CONSOLE ON
ON KEY
CLEAR TYPEAHEAD
SET PRINT OFF
RETURN
* eof()
```

Important Notes :

Stopping the Printer :

In the above program, a feature is added to stop the printer by pressing any key. In the program, the screen displays this message while the report is being printed:

```
    Press any key to Abort print job
```

Printing any key displays the message:

```
    Print job Aborted . . .
```

and returns control to the system main menu.

The CLEAR TYPEAHEAD command clears out any extraneous keystrokes from the typeahead buffer, since every time you press a key, it first goes into a "holding tank" called the "TYPEAHEAD buffer". To ensure that the next command in the program (ON KEY) works, it is best to make sure the TYPEAHEAD buffer is clear.

The SET CONSOLE OFF command keeps the printed report from appearing on the screen so that the "Press any key to Abort print job" message does not disappear.

The ON KEY command at the bottom of the program disables the previous "ON KEY DO PrinStop" command, so the next key press does not run the PrinStop program.

The CLEAR TYPEAHEAD again clears out any extraneous keystrokes, which is just a precautionary measure in this case.

The PrinStop.PRG Program :

```
* Program..: PrinStop.PRG
* Date.....: April, 1989
* Remark...: TO halt printer and return to Main Menu.
*
SET CONSOLE ON
SET PRINT OFF
CLEAR
? "Print job Aborted..."
SET FILTER TO
SET DELETED OFF
SET INDEX TO FNAMES , LNAMES , ZIPS
*
CLEAR TYPEAHEAD
ON KEY
RETURN
* eof()
```

This program is run only if the user presses any key to Abort printing. It is immediately sets the ÇONSOLE back on, turns OFF the printer, and clears the screen. Then it displays the message:

```
Print job Aborted . . .
```

The SET FILTER TO command removes any filter conditions set by the QUERY FORM (back in the Reports.PRG).

The SET DELETED OFF command "unhides" the RECORDs marked for deletion.

ABDALLA EL-DAOUSHY
OPERATIONS RESEARCH
24 ABU HAYYAN EL-TAWHIDI STREET
NASR CITY
CAIRO

ABDELHAMID EL-KASSAS
OPERATIONS RESEARCH
CAIRO

AMAL EL-SAYED
OPERATIONS RESEARCH
18 El-Horriyya street
EL-MATARIA
CAIRO

FATHEIA ZAGHLOL
OPERATIONS RESEARCH
CAIRO

MOHAMMED ABDEL-AAL
INFORMATION CENTER
9 ABDEL-RAZEK EL-SANHORI STREET
NASR CITY
CAIRO

SALEH EL-ADAWY
OPERATIONS RESEARCH
4 Ibn Hagar El-Askalani street
HELIOPOLIS
CAIRO

ABDEL-KADER HAMZA
OPERATIONS RESEARCH / INP
24 Mokhles El-Alfi street
NAST CITY
CAIRO

AFAF NAKHLA
OPERATIONS RESEARCH
Rabaa Square, Apt# 603 Building #
NASR CITY
CAIRO

AMANI OMAR
OPERATIONS RESEARCH
47 EL-Giza street
HELIOPOLIS
CAIRO

MOHAMMED YOUSSEF
INDUSTRIAL PLANNING
HELIOPOLIS
CAIRO

MOHAMMED EL-KAFRAWY
OPERATIONS RESEARCH
NASR CITY
CAIRO

YEHIA YOUSSEF
INDUSTRIAL PLANNING
HELIOPOLIS
CAIRO

3.4 Checking for Duplicate Data Entry :

The final program in the present system is needed to check the dataBase file for duplicate RECORDs based on identical names, addresses, and zip codes.

Rather than actually deleting RECORDs, the program displays a report of existing duplications.

The easiest way to check for duplicates in a dataBase file is to first SORT the data into some order, then check for matching pairs. For example, the program will INDEX the PERSONEL.DBF file on last and first names, city, and governrate. Then the program will display all RECORDs that match these fields.

Program Algorithm :

```
- Use PERSONEL.DBF

- INDEX ON F_NAME+L_NAME+ADDRESS+CITY+GOVERNRATE TO Temp

- Clear screen

- Print Report title

- Loop through the dataBase file

      . Store Name, Address, City, and Governrate

      . Skip down one RECORD

      . See if identical match occurs, if so,
            Skip back one RECORD and
            List RECORDs with identical match
         otherwise,
            Continue at next RECORD

- Continue Loop

- Return to Main Menu.
```

The DupCheck.PRG Program :

```
*
* Program.: DupCheck.PRG
* Date....: April,1989
* Remark..: To scan dataBase for possible duplications.
*
SET DELETED ON
SET SAFETY OFF
*
CLEAR
Printer = " "
@15, 5 SAY "Send possible duplicates to printer? (Y/N) ";
                    GET Printer PICT "!"
READ
*
IF Printer = "Y"
   WAIT "Prepare printer, then press any key to continue..."
   SET PRINT ON
ENDIF
*
* Display resorting message and create index files...
*
CLEAR
@ 5,10 SAY "Resorting: Please Wait..."
*
USE PERSONEL
INDEX ON TRIM(F_NAME)+TRIM(L_NAME)+HPHONE TO TEMP
*
CLEAR
? "Duplications found for " + SPACE(30) + DTOC(DATE())
? "---------------------- " , SPACE(28) , "--------"
?
*
* Loop through dataBase until eof(), and compare RECORDs.
*
DO WHILE .NOT. EOF()
   Compare =UPPER(TRIM(F_NAME))+UPPER(TRIM(L_NAME))+HPHONE
   SKIP
   IF UPPER(TRIM(F_NAME))+UPPER(TRIM(L_NAME))+HPHONE = Compare SKIP-1
      LIST WHILE UPPER(TRIM(F_NAME))+UPPER(TRIM(L_NAME))+HPHONE = Compa
          TRIM(F_NAME)+" "+TRIM(L_NAME)+"        "+HPHONE
      ?
   ENDIF (Equal RECORDs)
ENDDO (WHILE .NOT. EOF())
*
```

```
*
IF Printer = "Y"
   EJECT
   SET PRINT OFF
ENDIF (Printer)
*
* Done. Erase Temporary INDEX File..
*
CLOSE DATABASES
ERASE TEMP.NDX
*
SET DELETED OFF
SET SAFETY ON
*
* the possibility of deleting the duplicated RECORDs.
*
@20, 2 SAY "If you like to delete these duplicated RECORDs,"
@21, 2 SAY "write on paper these Names and select option 2 of next menu"
@22, 2 SAY "------------------------------------------------------------"
WAIT
CLEAR
RETURN
* eof()
```

Note that :

Since Duplication-checks would not performed very often, it is wiser to just create a new temporary INDEX file (Temp.NDX) each time needed. That is what happen in the program above.

```
Duplications found for                                      05/09/89
----------------------                                      --------

     16
     17


     15   ABDALLA EL-DAOUSHY          2622149

     11   ABDEL-KADER HAMZA          2611534




  If you like to delete these duplicated RECORDs,
  write on paper these Names and select option 2 of next menu
-------------------------------------------------------------
Press any key to continue...
```

Viewing Selected Data Program :

In this program, a general procedure for viewing selected data will be established. This program will be called VUINFO.PRG.

VUINFO.PRG Program :

```
* Program.: VUINFO.PRG
* Date....: June, 1988
* Notes...: This program Views Selected Information
*
USE PERSONEL
*
DO WHILE .T.
   *
   CLEAR
   @ 2, 5 SAY "Personnel System: View Menu"
   @ 2,55 SAY DTOC(DATE()) + "   " + TIME()
   @ 3, 5 SAY Uline
   *
   * Initialize variables
   *
   YN = SPACE(1)
   MTitle = SPACE(20)
   MJob = SPACE(20)
   MLetter = SPACE(1)
   ML_NAME = SPACE(10)
   *
   * Set View Menu
   *
   TEXT


               M.  RETURN to the Main Menu

               T.  VIEW by Title

               J.  VIEW by Job

               L.  VIEW by Letter

               N.  VIEW by Last Name

               A.  VIEW All

   ENDTEXT
   *
```

```
*
Q = SPACE(1)
@20, 5 SAY "Enter Your Selection:" GET Q PICTURE "A"
READ
*
DO CASE
   *
   CASE UPPER(Q) = "M"
        CLEAR
        RETURN
        *
   CASE UPPER(Q) = "T"
        SET INDEX TO TITLE     && TITLE+F_NAME
        CLEAR
        *
        @12,5 SAY "Enter Title:" GET MTitle PICT "@!"
        READ
        WAIT
        CLEAR
        DISP ALL TRIM(F_NAME)+" "+L_NAME,TRIM(TITLE),TRIM(JOB);
                  FOR TRIM(TITLE)=TRIM(MTitle)
        *
        @22, 5 SAY " (Nothing found) or No more RECORDs . . ."
        WAIT
        CLOSE INDEX
        CLEAR
        *
   CASE UPPER(Q) = "J"
        SET INDEX TO JOB      && TRIM(JOB)+TRIM(F_NAME)
        CLEAR
        @22, 5 SAY "Enter Job:" GET MJob PICT "@!"
        READ
        WAIT
        CLEAR
        *
        DISP ALL TRIM(F_NAME)+" "+L_NAME,TRIM(TITLE),TRIM(JOB);
                   FOR TRIM(JOB) = TRIM(MJob)
        *
        @22, 5 SAY "No more RECORDs (or nothing found). . ."
        WAIT
        CLOSE INDEX
        CLEAR
        *
   CASE UPPER(Q) = "L"
        SET INDEX TO LNAMES  && Indexed on L_NAME
        CLEAR
        @22, 5 SAY "Enter First Letter of Last name:";
                        GET MLetter PICT "!"
        READ
        WAIT
        *
```

```
    *
    IF MLetter = " "
       CLEAR
       CLOSE INDEX
       LOOP
    ENDIF
    *
    * Search for L_NAME started with MLetter. . .
    *
    SEEK TRIM(MLetter)
    CLEAR
    *
    * Loop through all records with the desired first
    * letter of the last name
    *
    EXT = .T.
    DO WHILE TRIM(L_NAME) = TRIM(MLetter) .AND. EXT
       DISP TRIM(F_NAME)+" "+L_NAME,TRIM(TITLE),WPHONE
       @22, 5 SAY "Do you wish to continue? (Y/N): ";
                        GET YN PICTURE "!"
       READ
       IF YN="Y"
          SKIP
          CLEAR
          YN = ' '
          LOOP
        ENDIF
        EXT = .F.
        CLOSE INDEX
     ENDDO
     *
     IF EXT
        @22, 5 SAY "No more name(s) (or nothing)"
        @22,34 SAY "beginning with &MLetter."
        WAIT
     ELSE
        WAIT
     ENDIF
     *
CASE UPPER(Q) = "N"
    CLEAR
    @22, 5 SAY "Enter Last name:" ;
                         GET ML_NAME PICT "@!"
    READ
    WAIT
    CLEAR
    SET INDEX TO LNAMES   &&  L_NAME
    SEEK TRIM(ML_NAME)
    *
```

```
          *
          IF .NOT. FOUND()
             *
             @15, 5 SAY "Not founded !."
             @17, 5 SAY "No such Last name (&ML_NAME.), Try again"
             WAIT
             CLOSE INDEX
             CLEAR
             LOOP
          ENDIF
          *
          *  founded . . .
          *
          CLEAR
          DISP ALL TRIM(F_NAME)+" "+L_NAME,TRIM(TITLE);
                                  FOR TRIM(L_NAME) = TRIM(ML_NAME)
          @22, 5 SAY "No more (or nothing found) !"
          WAIT
          CLOSE INDEX
          *
     CASE UPPER(Q) = "A"
          SET INDEX TO FNAMES
          CLEAR
          DISP ALL TRIM(F_NAME)+" "+L_NAME,TRIM( TITLE),;
                                          TRIM(HPHONE),JOB
          WAIT
          CLOSE INDEX
          *
     OTHERWISE
          LOOP
   ENDCASE
   *
   CLEAR
ENDDO
* eof()
```

When the user selects VIEW by TITLE, we want the computer to display:

First name, Last name, Title, and Job

On the other hand, when the user selects VIEW by JOB, we want the computer to display:

First name, Last name, Title, and Job

The intended of the View by Letter option is to enable the user to view all those RECORDs in which the first letter of the last name is specified. For example, the user might wish to view all those RECORDs having last name beginning with M.

The View by Letter option can be done in several ways, but one fairly simple way would be to use the SEEK or FIND command to get to the first occurrence of the appropriate letter, and then use the SKIP command to find other occurrences. SKIPping can be done by using a DO WHILE loop.

When the user selects VIEW ALL, we want the computer to display:

First name, Last name, Title, Home phone, Work phone, and Job.

On the other hand, if he selects VIEW by letter, we want the computer to display:

First name, Last name, Title, Home phone, and Work phone.

Finally, if the user selects VIEW by Last name, the computer displays:

First name, Last name, Title, Home phone, and Work phone.

The program should return to the Main Menu if the user enter nothing of the options of the VIew Menu.

REFERENCES

1. Alan Simpson,

Advanced Techniques in dBASE III Plus
San francisco , Paris , Dusseldorf , London

2. Ashton-Tate,

dBASE III Plus

3. Abdalla El-Daoushy,

First course in dataBase Analysis and Programming with dBASE III Plus, Memo. no. 880, 1988
Institute of National Planning, Cairo, Egypt

4. Abdalla El-Daoushy

INP dataBase Management System ,Memo no. 1475, 1988
Institute of National Planning, Cairo, Egypt

5. Abdalla El-Daoushy, M. Y. Youssef

An Information System for Inventory Management (A Computer Oriented), Memo no. 1498, 1989
Institute of National Planning, Cairo, Egypt

6. C. J. Date,

An Introduction to Database Systems, Third Edition
Addison-Wesley Publishing Company
Reading,Massachusctts . Amsterdam . London
Manila . Singapore . Sydney . Tokyo

7. James Martin,

Principles of Data-Base Management
Printic-Hall, Inc. Englewood Clffs. New jersey

تعتبر عملية تصميم وبناء قواعد البيانات وشبكات المعلومات من الأساليب الحديثة التى تسهم فى التحكم فى ثورة المعلومات المعاصره التى تتزايد وتتضاعف من حيث الكم والنوع ٠

ومن الاحساس المتزايد فى مصر بأهمية المعلومات ودورها فى تطور وتقدم الأمم والشعوب ،صدر قرار جمهورى رقم ٦٢٧ لسنة ١٩٨١ بانشاء مراكز للمعلومات فى الأجهزة الادارية للدولة والهيئات العامة ،وكان هذا القرار خطوه جيده من جانب الحكومه فى هذا المجال الحيوى ٠ كذلك اهتمت الجامعات المصرية ومعاهد البحوث العلميه بالمعلومات كعلم وقامت بتطوير مناهجه المختلفة ٠

وهذه المذكره من جانبنا تهدف لتقديم واحد من تطبيقات نظم المعلومات وذلك بتصميم وبناء قاعدة معلومات لشئون الافراد بالمعهد باستخدام الحاسب الشخصى وهى بالاضافه لتصميم قاعدة البيانات تعرض الأساليب المتقدمة فى ال DBASE III PLUS الذى يستخدم فى هذا المجال والتى تساعد فى سرعة خزن واسترجاع المعلومات ٠