A FORM BASED PERSONALIZED BULK MAILING SYSTEM

by

LIN, HUI-LING

B.A., Kansas State University, 1982

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1984

Approved by :

Major Professor

LD
2668
.R4
1984
L56
c. 2

TABLE OF CONTENTS

A11202 662126

Page

LIST OF DIAGRAMS ii

LIST OF FIGURES iii

ACKNOWLDGEMENT v

CHAPTER 1 INTRODUCTION 1

1.1 Background 1
1.2 Evaluation of Wordstar Mailmerge System . . . 5
1.3 Approach 10
1.4 Organization of report 12

CHAPTER 2 LOGICAL DESIGN AND IMPLEMENTATION 13

CHAPTER 3 USER GUIDE 33

3.1 Overview 33
3.2 Getting Started 33

3.2.1 System requirements 34
3.2.2 Activating the computer 34
3.2.3 Preparing the new disks for use 34
3.2.4 Copying a file or an entire disk . . . 35

3.3 The MAIL Program 36

3.3.1 The opertional features of the MAIL . . 36
3.3.2 Files required in the MAIL 37
3.3.3 Starting the MAIL program 38
3.3.4 Create Menu function 39
3.3.5 Mail Menu function 52
3.3.6 Delete Menu function 59
3.3.7 Print Menu function 62

CHAPTER 4 CONCLUSIONS 76

BIBLIOGRAPHY 78

APPENDIX

SOURCE CODE 80

LIST OF DIAGRAMS

Page

Warnier-Orr Diagrams

2.1 Bulk Mailing System Screen 14

2.2 Create Criterion Categories 16

2.3 Add Address Record 18

2.4 Display on Screen 20

2.5 Delete Address Record 21

2.6 Delete File 23

2.7 Mail Labels 24

2.8 Mail Letters 26

2.9 Add Letter 27

2.10 Data Flow 29

LIST OF FIGURES

Page

FIGURES

1.1 Form Letter File of the MailMerge System 6

1.2 Data File of the MailMerge System 7

1.3 Message on the CRT of the MailMerge System . . . 7

1.4 The WardStar Commands 9

3.1 Main Menu 40

3.2 Main Menu Help 41

3.3 Create Menu 42

3.4 Create Menu Help 43

3.5 Create Criteria Menu 45

3.6 Create Criteria Menu Help 46

3.7 Create Address Record Menu 48

3.9 Create Address Record Menu Help 49

3.10 Add Letter Menu 51

3.11 Add Letter Menu Help 53

3.12 Mail Menu 54

3.13 Mail Menu Help 56

3.14 Mail Letter Menu 57

3.15 Mail Letter Menu Help 58

3.16 Mail Label Menu 60

3.17 Mail Label Menu Help 61

3.18 Delete Menu 63

Page

FIGURES

3.19 Delete Menu Help 64

3.20 Delete Address Record Menu 65

3.21 Delete Address Record Menu Help 66

3.22 Print Menu 68

3.23 Print Menu Help 69

3.24 Display Letter File on Screen 70

3.25 Display Letter File on Screen Help 71

3.26 Display Address File on Screen 72

3.27 Display Address File on Screen Help 73

3.28 Search ID Function in Display Address Menu . . . 74

ACKNOWLEDGEMENTS

I wish to express my most grateful and sincere appreciation to my major professor, Dr. Elizabeth Unger, for her guidance and direction in completion of this report. Thanks are also given to Dr. Rodney Bates and Dr. Richard McBride for serving on my committee.

Additionally, sincere thanks to my parents for their support and encouragement.

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Over the past two decades, office automation activity has focused on the mechanization of the single task. The usual goal was typing, and the job was approached through the creation, refinement, and spread of word processors as another piece of equipment intended to improve office productivity. As stated by Chorafas (1), the basic issues have changed. The objective now is to develop a "single system" by combining word processing, data processing, image handling, and message transmission services. This will supplant the need for typewriters, reproduction centers, mailrooms and office file cabinets; thus, integrating a myriad of office functions in a single computer-based terminal via user-friendly commands.

There are people who feel that the paperless plunge will pay off. This is particularly true for major corporations and government agencies which, snowed under with paper, want to get in step with the paperless pace promised by automation. For instance, forms have been proposed and used in the design of at least three office automation systems (Office Talk, Odyssey, and OFS) (1). Forms help ease the transition from a manual office system based on paper forms to a computer office

system based on electronic forms. Electronic forms have many advantages, e.g., they can be copied, annotated, stacked on an electronic desk, and personalized. In addition, forms can be traced and can serve as a high-level protocol for information communication. Therefore, forms become important for office automation systems of the future in that they provide a better interface for facets of the automated office. (2)

Forms can be viewed as both text and formatted data. Forms consist of form fields in conjunction with dispersed printed text. The fields are the blank spaces in forms where required or requested information is to be inserted. Tsichritzis (3) stated that " A better office automation approach would have been to have a form's approach to data base query." Hence, the form based systems separate data into two categories, i.e., data on forms and data in a DBMS. However, there is no need for such a separation. There should be a natural way to access data on forms through data base application programs or data base commands. The data management and form processing should be integrated into the same system.

Today, almost every business or organization uses mailing lists to send advertising, junk mail, newsletters, and special information directly to customers. Writing, printing, folding, addressing, and mailing this literature constitutes a

lot of work that most business people would rather avoid because it is tedious, repetitious, and non-productive. However, we can choose the form based concept used for the integration of automated personalized bulk mailing systems to free valuable workers from "busy work" and permit them to accomplish more productive tasks.

Many letters received today have been addressed by a computer. Microcomputers can be used in the same way as large commercial computers to speed up the preparation of letters and other documents, store them for future use, type them quickly and accurately. Then, they can be sent to one or even hundreds of people (bulk mailing), with the computer also printing the envelopes or mailing labels.

The microcomputer, with its low initial investment and upkeep, is bringing the advantages of computerization to all levels of business. Many departments within large businesses have not previously used computers because the additional work would have exceeded the capacity of the existing computer installation. They can now use one or more microcomputers to computerize the operations within a single department. (4) Also, microcomputers can be used by the thousands of small business organizations that have not had the capital to afford a mainframe computer. Hence, the bulk mailing system developed in this report will be implemented on an

IBM-look-alike compatible Columbia Data Products microcomputer, at the Department of Computer Science, Kansas State University.

The bulk mailing system is a research project and, as such, will be implemented in a high level language to the fullest extent possible. The existing compilers for the Columbia microcomputer in our department are Ada/Janus, MS/BASIC, CP/M-86 Pascal, IBM MS/Pascal and TURBO Pascal. The new language Ada employs a structure similar to Pascal, but with many extensions and improvements. However, from my research, some of these features are not implemented in Ada/Janus as yet (e.g. no tasks). BASIC is easy to learn but since there are so many versions of this language, a program written in BASIC is not portable. An additional criticism of BASIC is that it is hard to document, hence, if very complicated programs are written in it, they are difficult to read and maintain by others. The language Pascal embodies the principles of structured programming so that it is easy for programmers to understand a program written by others. For this reason and those above, I have chosen Pascal as the language for my implementation. Since TURBO Pascal has its own editor and takes less compile time than other Pascal compilers for microcomputers, it will be the particular version of Pascal used.

1.2 EVALUATION OF AN EXISTING MAILING SYSTEM - WORDSTAR MAILMERGE

An existing bulk mailing system for personal computers called MailMerge, has become a popular program on such microcomputers as the Apple and the IBM - PC. The MailMerge program is available as an option to the WordStar system, a word processing system marketed by Micropro International Corporation. For example, with MailMerge one can create a basic form letter and then personalize each copy of this letter with information, such as name, address, date and price, etc. This information can be maintained in a separate data file by using WordStar, or it can be entered by an operator while running the MailMerge program. For instance, if the letter was designed to be mailed to anyone who is past due in paying money owed the firm (figure 1.1), the past due amount can be either stored in a data file for retrieval (figure 1.2) or can be entered interactively by the operator during MailMerge process (figure 1.3). When the basic form letter is merge-printed with operator input or input from the separate file containing the personalized data, each letter is individually typed and the information automatically inserted

```
..EXAMPLE
.MT 10
.CS
.SV DATE,March 1, 1984
.OP
.DF DATAFILE
.RV NAME,COMPANY,STREET,CITY-STATE-ZIP,COUNTRY,SALUTATION
.DM PRINTING LETTER TO: &NAME&
.DM NAME OF COMPANY: &COMPANY&
.AV "ENTER AMOUNT DUE: ",AMOUNT


&DATE&



&NAME/O&
&COMPANY/O&
&STREET/O&
&CITY-STATE-ZIP/O&
&COUNTRY/O&



DEAR &SALUTATION/O&


Though we mailed your statement over six weeks ago, we
have not yet received your payment.  Please, if you have
not done so already, send us the past due amount of
&AMOUNT& immediately.


Sincerely yours,



Katherine Carlone
Accounts receivable department
```

Figure 1.1 FORM LETTER FILE

```
B: DATAFILE FC=73 FL=2 COL 01          INSERT ON


Mr. Fred Smith,United Inc.,100 Elm Street,"Menlo,CA
10000",,Mr. Smith
Ms. Ann White,,200 Oxford Street, London WIP 6JD,
England,Ms. White
```

Figure 1.2 DATAFILE

```
MailMerge-printing     B:EXAMPLE          not editing

   P = Stop PRINT


   PRINTING LETTER TO: Mr. Fred Smith

   NAME OF COMPANY: United Inc.

   ENTER AMOUNT DUE:

   DIRECTORY OF DISK B:
    DATAFILE     EXAMPLE
```

Figure 1.3 MESSAGES ON THE SCREEN

at the appropriate place. The mailing labels can also be quickly generated using MailMerge. With a mailing list entered and maintained in a MailMerge data file, one can have up-to-date labels printed in any format desired, whenever needed. One will also save the cost of an expensive mailing label machine or hours of typing time. (5)

MailMerge is ideal for preparing and mailing contracts, purchase orders, or any other documents with standard text and names, addresses, and other data that change from document to document. MailMerge is a powerful tool for increasing one's direct mail communication, advertising, product promotion, publicity, and every other aspect of business communications where the attention of a relatively large number of people is required. However, MailMerge suffers from the lack of flexibility and user-friendliness.

MailMerge is attractive to experienced users but seems complicated to beginners. The user's first response is, "there are so many commands, how can I learn them all?". (5) The easiest way to learn MailMerge is to have a WordStar book in hand and follow through each step until you become familier with each command. This is a sophisticated system of 160 one, two and three key stroke commands. (figure 1.4) This complexity may discourage some and frustrate others who do not have enough patience.

WORDSTAR COMMANDS

OPENING MENU COMMANDS

DESCRIPTION	COMMAND
Preliminary Commands	
Change logged disk drive	L
File directory ON/OFF	F
Set help level	H
Commands to Open a File	
Open a document file	D
Open a non-document file	N
File Commands	
Print a file	P
Rename a file	E
Copy a file	O
Delete a file	Y
System Commands	
Run a program	R
Exit to system	X
WordStar Options	
Run MailMerge	M
Run SpellStar	S

OTHER MENUS

(From main menu only)	
Help menu	^J
Block menu	^K
Onscreen menu	^O
Print menu	^P
Quick menu	^Q
(Space bar returns you to main menu)	

MOVE CURSOR

Right one character	^D
Left one character	^S
Up one line	^E
Down one line	^X
Right one word	^F
Left one word	^A
To tab right	^I
To top of screen	^QE
To bottom of screen	^QX
To top of file	^QR
To end of file	^QC
To right end of line	^QD
To left end of line	^QS
To top of block	^QB
To end of block	^QK
To next misspelling	^QL
To position before previous command	^QP
To start of last find/replace	^QV
Find replace text again	^L
To marker 0-9	^Q0-Q9

SCROLL (MOVE SCREEN)

Up one line	^W
Down one line	^Z
Up one screen	^R
Down one screen	^C
Up screen continuously	^QW
Down screen continuously	^QZ

FORMAT

Paragraph tab	^OG
Variable tabbing ON/OFF	^OV
Center text	^OC
Set left margin	^OL
Set right margin	^OR
Release margins	^OX
Set margins and tabs from text line	^OF
Set tabs	^OI
Clear tabs	^ON
Justification ON/OFF	^OJ
Set line spacing	^OS
Page break display ON/OFF	^OP
Ruler display ON/OFF	^OT
Word wrap ON/OFF	^OW
Print control display ON/OFF	^OD
Hyphen help ON/OFF	^OH
Soft hyphen entry ON/OFF	^OE
Reform paragraph	^B

DELETE AND INSERT

Delete character	^G
Delete character left	DEL
Delete word right	^T
Delete line	^Y
Delete to end of line	^QY
Delete to beginning of line	^Q DEL
Delete a block	^KY
Delete a file	^KJ
Insert ON/OFF	^V
Insert carriage return (blank line)	^N

FIND AND REPLACE

Find text	^QF
Find and replace text	^QA
Find/replace text again	^L
Cursor to start of last find/replace	^QV

SAVE FILES

Save and resume edit	^KS
Save and return to opening menu	^KD
Save and exit to system	^KX
Abandon file without saving	^KQ

MISCELLANEOUS

Set help level	^JH
Repeat following command or character	^QQ
Interrupt command	^U

FILE AND BLOCK OPERATIONS

Column mode ON/OFF	^KN
Mark/unmark block beginning	^KB
Mark/unmark block end	^KK
Hide/show marked block	^KH
Copy block	^KC
Delete block	^KY
Move block	^KV
Write block into another file	^KW
Read file into document	^KR
Set/remove marker 0-9	^K0-9
Copy file	^KO
Rename file	^KE
Print file	^KP
Delete file	^KJ
File directory ON/OFF	^KF
Change logged disk drive	^KL

DESIGN THE PRINTED PAGE

Use In Pairs	
Boldface	^PB
Doublestrike	^PD
Underline	^PS
Strikeout	^PX
Subscript	^PV
Superscript	^PT
Use Alone	
Strikeover	^PH
Non-break space	^PO
Phantom space	^PF
Phantom rubout	^PG
Strikeover line	^P RETURN
Alternate pitch	^PA
Standard pitch	^PN
Print pause	^PC
Ribbon color change	^PY
User-defined function	^PQ, ^PW, ^PE, ^PR

DOT COMMANDS

Bidirectional printing	.BP
Microjustification	.UJ
Page offset, left margin	.PO
Character width	.CW
Comment (not printed)	.IG or . .
Conditional page	.CP
Footing	.FO
Heading	.HE
Heading margin	.HM
Footing margin	.FM
Line height	.LH
Margin at top	.MT
Margin at bottom	.MB
New page (page break)	.PA
Omit page number	.OP
Page number	.PN
Page number column	.PC
Subscript/superscript roll	.SR
Paper length	.PL

MAILMERGE

DOT COMMANDS

Define file	.DF
Read variables	.RV
Repeat	.RP
Set variables	.SV
Ask for variables	.AV
Display message	.DM
Clear screen	.CS
File insert	.FI
Print time line forming	.PF
Output justification	.OJ
Input justification	.IJ
Conditional command "except"*	.EX
Conditional command "if"*	.IF
End command*	.EF

*Only available for CP/M-80 and MP/M-80 operating systems

For alphabetical listing of commands, see index.

There are several points of inflexibility in MailMerge. It has no selection criteria building function to allow a user to specify the selection of names and addresses of only those customers who satisfy a certain set of conditions. Everytime the MailMerge is invoked it will print the entire mailing list or selected contiguous subsets thereof. MailMerge does not provide a function to allow a user to modify the form letter (preprinted text) interactively. In other words, whenever one wants to insert new information field into the form of the letter, one has to add a data field into data file and then start the MailMerge process or one has to enter the information interactively while running MailMerge program. This is certainly not very convenient for the companies that need to generate customized forms for a given mailing.

1.3 APPROACH

The mailing system developed in this report will overcome the disadvantages of inflexibility and user-unfriendliness mentioned above. In order to make the system easy to use, this system provides a facility for automating office procedures that can be used by office workers, as opposed to computer professionals. As such, it is inclined toward a natural language (English) rather than cryptic programming instructions. Chorafas (1) states " The best interface between an operator and the system they work with is English

words, edited, stored in memory, retrieved, and displayed on an easily read soft copy." The mailing system should include a menu of supported commands applicable to the letters or mailing lists that are visible in the window. All a user has to do is to key in one of the displayed commands and let the mailing system perform the complicated task for the user.

The mailing system will allow each user to set up selection codes before the mailing list is created in the data file. These selection codes may be used for the selection criteria of the mailing process (under selection menu screen). For example, if a book company is trying to sell a computer book, it will select the people who are working for computer organizations or the students and faculty who are in computer science departments. Hence, these selection criteria allow the company to effectively and efficiently insure that a particular targeted population receives the appropriated mailing. Moreover, the mailing system contains its own simple editor whose edit function can be used to modify the structure of the letter. This editing may be done interactively and, thus, there is no need to change the data structure of the mailing list. In addition, this mailing system will be more user-friendly and more flexible than the MailMerge system of WordStar.

1.4 ORGANIZATION OF REPORT

This chapter presented the background for this project, a description of the problem and the method to be used to solve it. The second chapter is an overview of the system design, with an explanation of the internal data structures. The third chapter is a simple user guide for the mailing system. The last chapter summarizes the results and suggests future work.

CHAPTER 2

LOGICAL DESIGN AND IMPLEMENTATION

This chapter will describe the design, data structures and implementation of the bulk mailing system. The structured system design, like structured programming, is hierarchical, modular, and logically organized. While various system design tools were considered, the Warnier-Orr diagram approach was chosen an ideal method for developing logically correct program design structure from the primitive data structures to be used. Higgins (7) stated that the various forms of the Warnier-Orr diagram are useful in many stages of a software development project, from requirements definition all the way through the operation and use of the delivered project. Also, Warnier-Orr Diagrams can enhance understanding of computer process, data relationships, and increase programmer and analyst productivity as well. Therefore, Warnier-Orr diagrams will be used throughout this chapter for describing the structure both of the design phase and implementation phase of the Bulk Mailing System.

The flow of control of the Bulk Mailing System is illustrated in the Warnier-Orr diagram of diagram 2.1. The main menu consists of Creation (C), Deletion (D), Mailing (M),

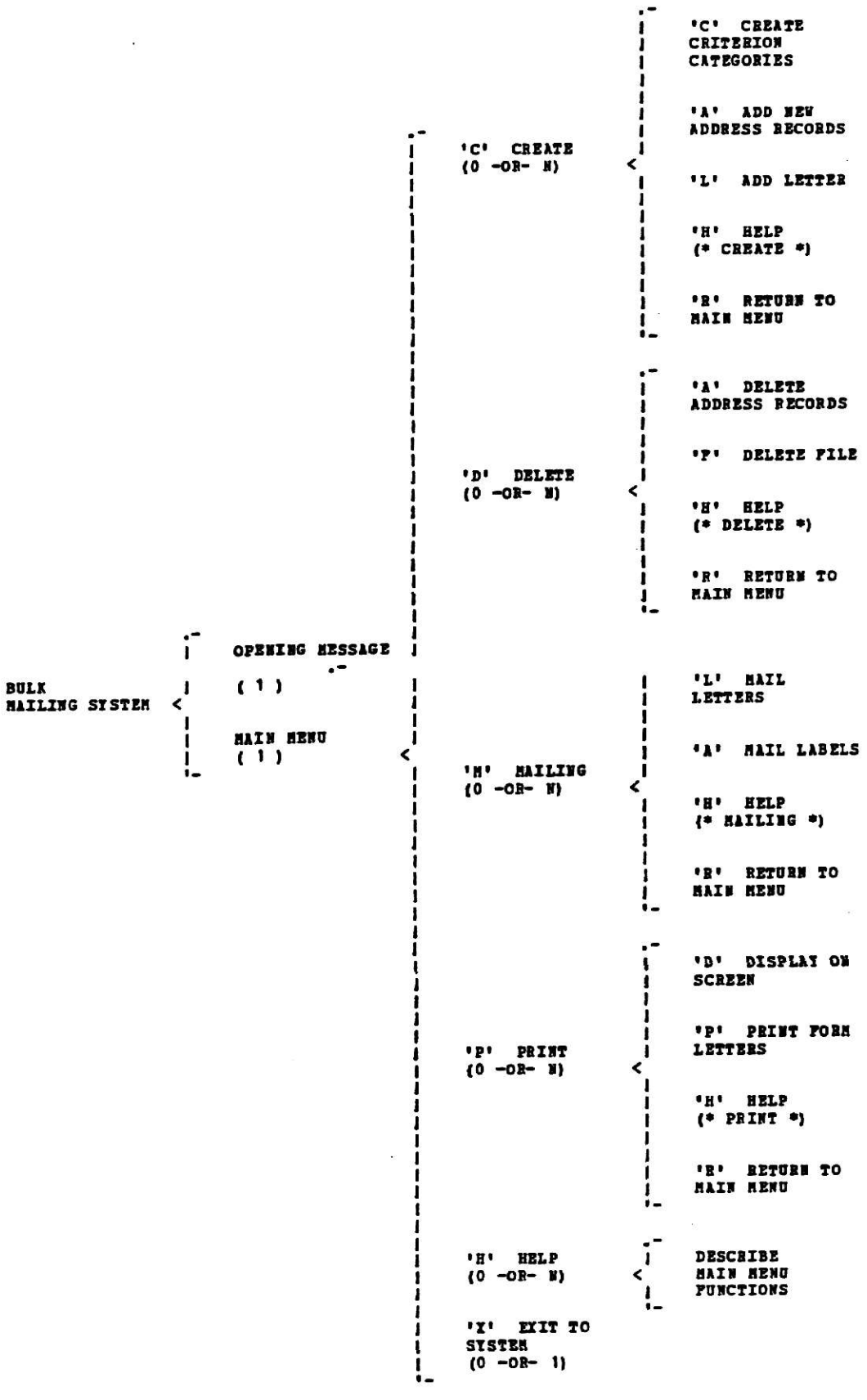

BULK MAILING SYSTEM
OPENING MESSAGE
(1)
MAIN MENU
(1)
'C' CREATE (0 -OR- N)
'C' CREATE CRITERION CATEGORIES
'A' ADD NEW ADDRESS RECORDS
'L' ADD LETTER
'H' HELP (* CREATE *)
'R' RETURN TO MAIN MENU
'D' DELETE (0 -OR- N)
'A' DELETE ADDRESS RECORDS
'F' DELETE FILE
'H' HELP (* DELETE *)
'R' RETURN TO MAIN MENU
'M' MAILING (0 -OR- N)
'L' MAIL LETTERS
'A' MAIL LABELS
'H' HELP (* MAILING *)
'R' RETURN TO MAIN MENU
'P' PRINT (0 -OR- N)
'D' DISPLAY ON SCREEN
'P' PRINT FORM LETTERS
'H' HELP (* PRINT *)
'R' RETURN TO MAIN MENU
'H' HELP (0 -OR- N)
DESCRIBE MAIN MENU FUNCTIONS
'X' EXIT TO SYSTEM (0 -OR- 1)

Diagram 2.1

Printing (P), Help menu (H) and eXit mailing system (X). These functions can be performed as many times as desired by entering the appropriate keystrokes. Since this system was designed to be "user-friendly", the selection of system screens only requires a single keystroke. Also, each screen has a "Help" menu to tell a user what functions can be performed with each screen. The user can simply press the "H" key to get the help description instead of looking for help from the user's manual, and press "R" to return to that screen. Navigation through the system was implemented so that a user can initiate the desired program by simply pressing the appropriate key. If an incorrect key is pressed by the user, an error message appears stating that the user must make another attempt.

The procedure (local to Create_Menu) "Create_Criterion_Categories" (see diagram 2.2) creates a criterion file that can be used as a reference when a user enters an address record to an address file. These criteria information in the address file will be used for the mailing process by specifying a specific criterion. Since the system only allows 20 criterion codes, the program will generate an error message if a user tries to enter the 21st code. Also, the range of each criterion number is 1 through 10. If a

CREATE CRITERION CATEGORIES ROUTINE

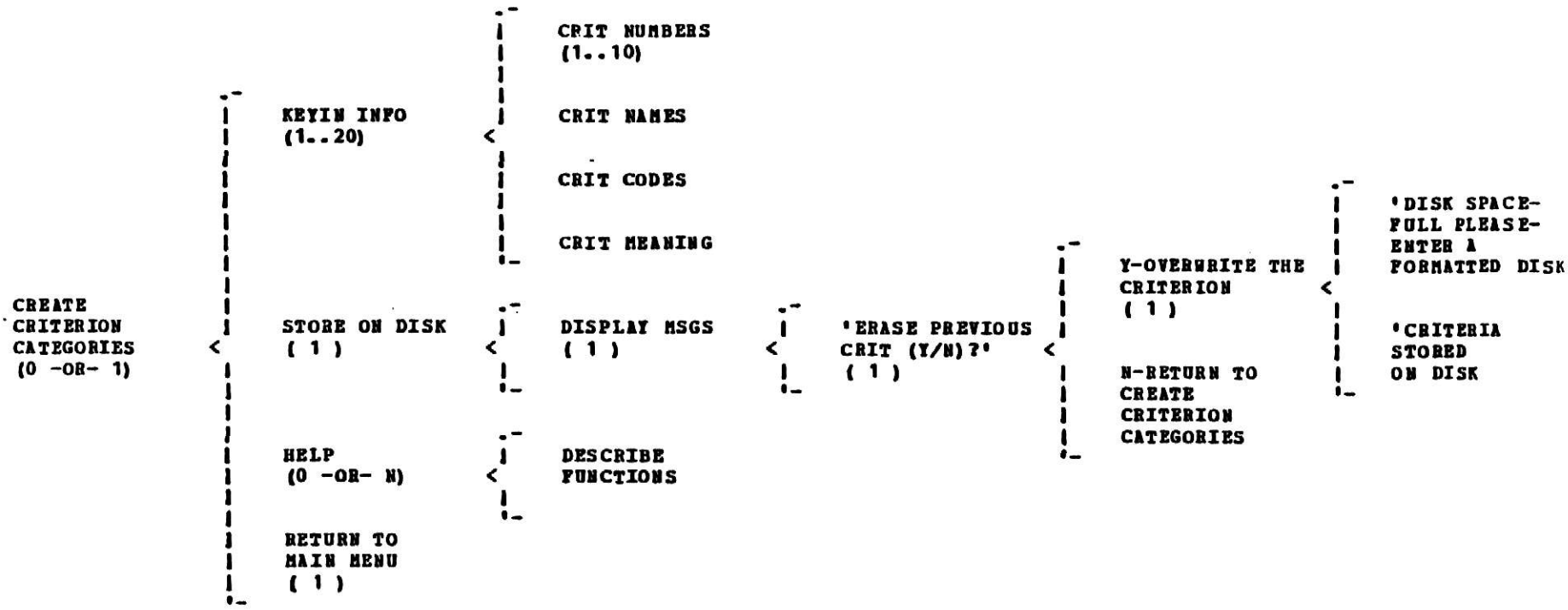


Diagram 2.2

user's input is not in this range, an error message appears with a request to enter a correct one. The criterion number is declared as a string type instead of integer type to prevent an automatic run-time error if a non-integer is accidentally entered by a user. If the name of the criterion file entered by the user already exists on the disk, the system will display a warning message to the user to insure that the existing file is not inadvertently overwritten.

The procedure (local to Create_Menu) "Add_New_Address_Record" (see diagram 2.3) allows a user to create a new address file or add a new address to an old address file. When a user enters the name of the address file, this procedure will check to see if the file exists, or not. If it exists, the procedure lets a user add a new address record onto the existing file. When a user enters an ID number, this procedure calls a hash function to determine a storage index for this ID number (Key). If this record already exists then a warning is generated so that the existing record is not accidently overwritten. If it is a new record, the new address record will be stored into a file by using a predefined procedure "Seek" (random access to a file) to find the correct position and the new record will be written to the address file.

The procedure (local to Print_Menu) "Display_On_Screen",

ADD NEW ADDRESS RECORDS ROUTINE

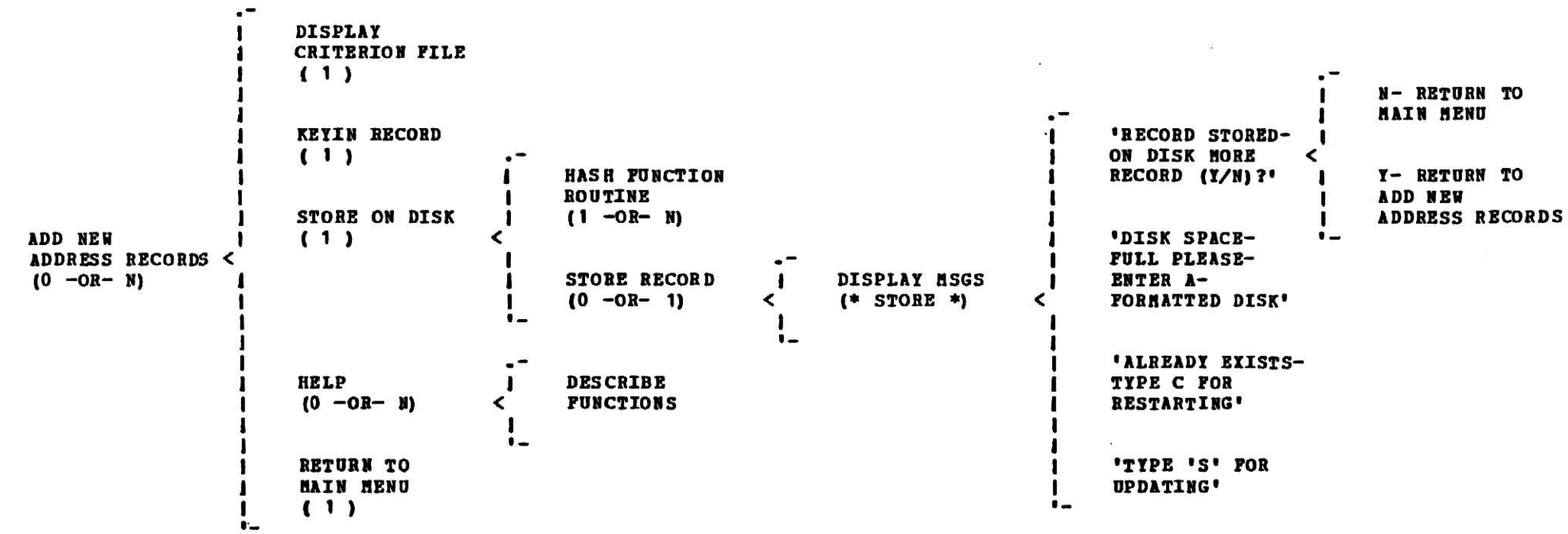


Diagram 2.3

(see diagram 2.4) allows a user to visually display either the letter file or address file on the CRT. This procedure will first determine whether the file named exists or not and will next determine whether the file is an address file or a letter file. If the file does not exist or is not of the proper type (i.e., address or letter) then an error message is generated. If the file to be displayed is an address file, the procedure invokes a local procedure "Display_Address". This procedure gives a user the option of asking for help (H), return to main menu (R), search the desired address record (S) or check all address records (C). The local procedure "Display_Next" uses the predefined "seek function" to get the next existing address record from the address file and display it on the screen. Also, a message will be shown when a user is trying to get the next record and the end-of-file is reached. The subprogram "Do_Search" searches the address file for the particular record whose ID number matches the ID number (Key) entered by a user. If this record does not exist then a message to that effect is generated; otherwise, the desired record is displayed on the screen. If the file to be display is a letter file the procedure "Display_Letter" will be invoked and the desired letter file will be displayed on the screen.

The procedure (local to Delete_Menu) "Delete_Address_Record" (see diagram 2.5) allows a user to

DISPLAY ON SCREEN ROUTINE

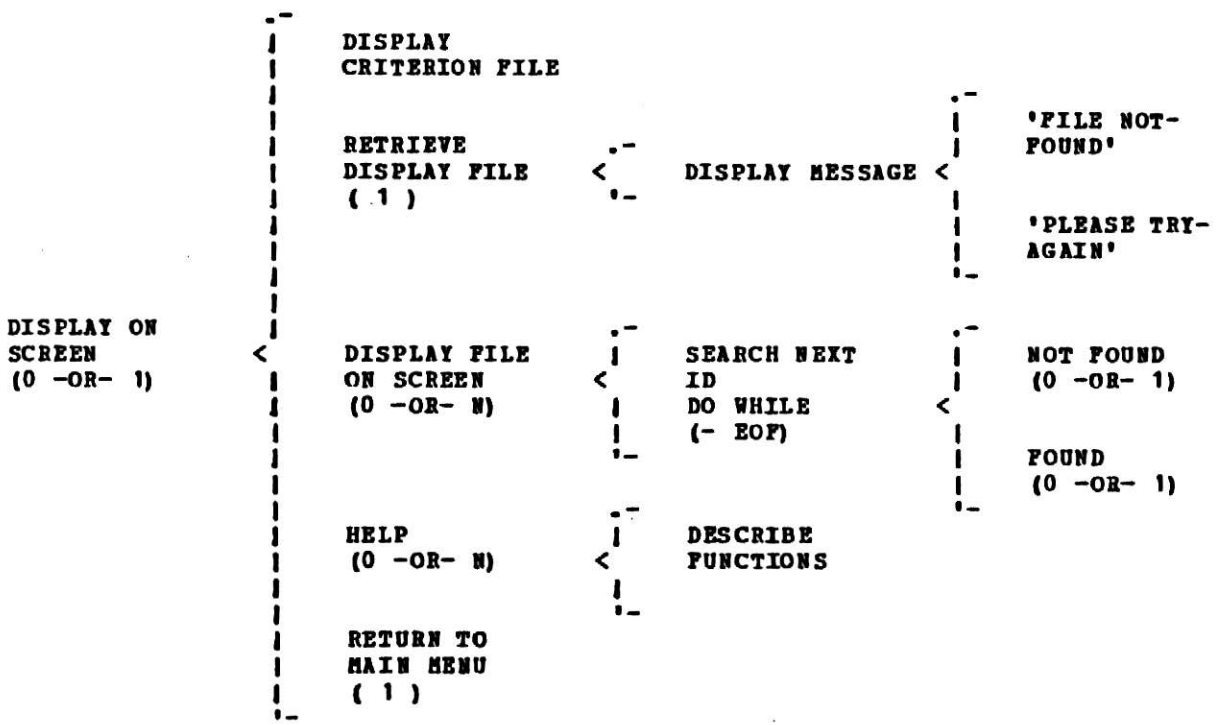


Diagram 2.4

DELETE ADDRESS RECORDS ROUTINE

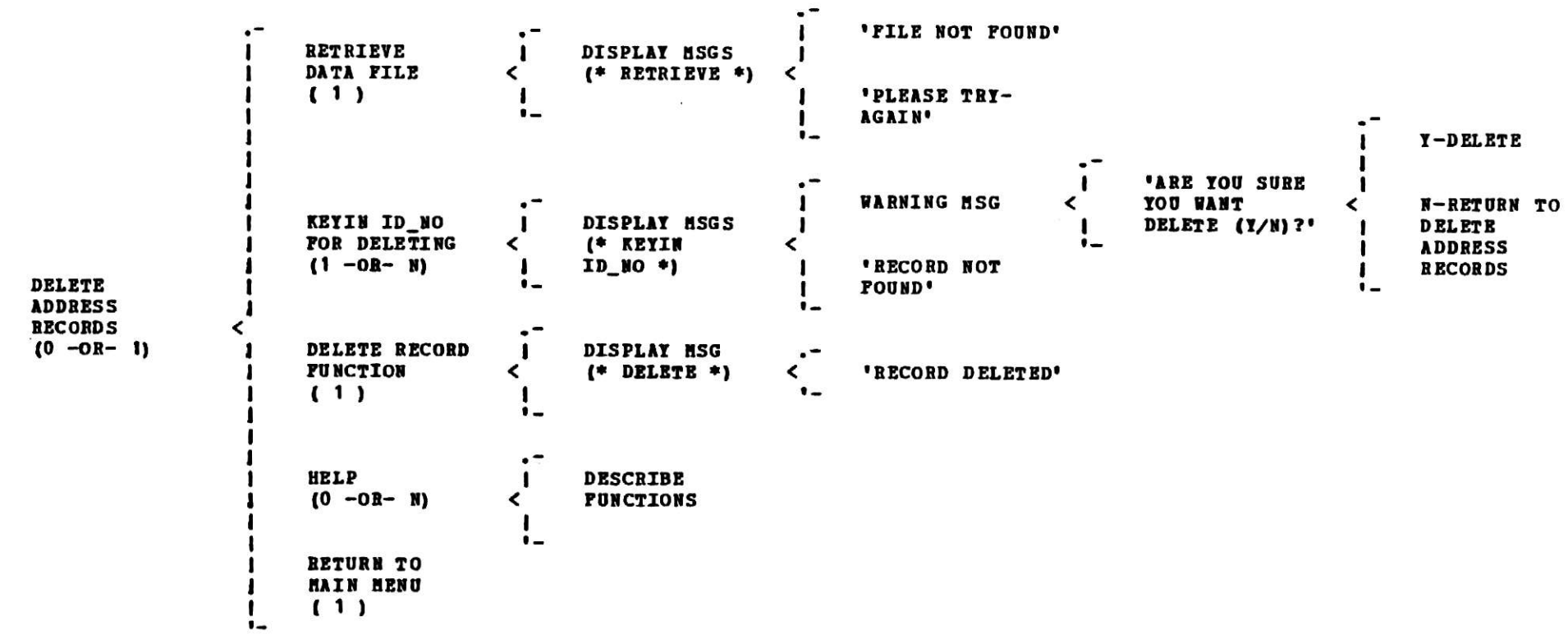


Diagram 2.5

delete a record from an address file. If the record is not in the address file, a message to that effect will be displayed; otherwise, a warning message is issued to give the user a chance to reconsider so that an inadvertent deletion is not made.

The procedure (local to Delete_Menu) "Delete_File" (see diagram 2.6) allows a user to delete an existing file. The predefined procedure "Erase(Filename)" was used to physically delete an existing file from the disk. A warning message is displayed to give the user a chance to reconsider before the deletion is actually performed.

The procedure (local to Mailing_Menu) "Mail_Labels" (see diagram 2.7) allows a user to produce a mailing list by specifying a set of selection criteria. The procedure first verifies that the criteria entered by a user are syntactically correct. If the criteria are incorrect the system will display an error message with a request to the user to either re-enter the criteria or exit the Mail_Labels procedure. If the criteria are correct, the procedure uses the criteria to perform the address selection from the address file. All of the selected addresses will be printed by the line printer. The procedure "Do_One_Criterion" (local to Mail_Labels) is invoked if the user only enters one criterion and the procedure "Do_Two_Criteria" (local to Mail_Labels) is invoked

DELETE FILE ROUTINE

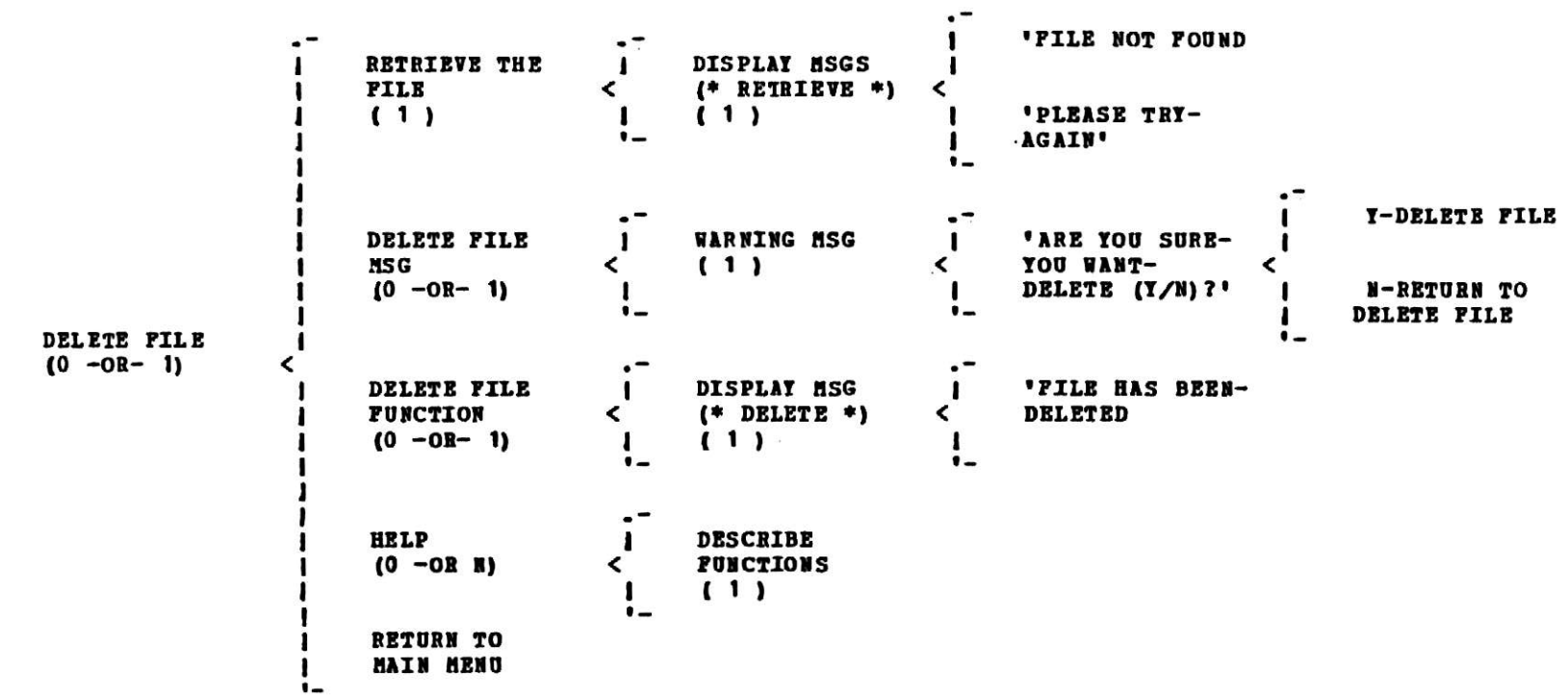


Diagram 2.6

MAIL LABELS ROUTINE

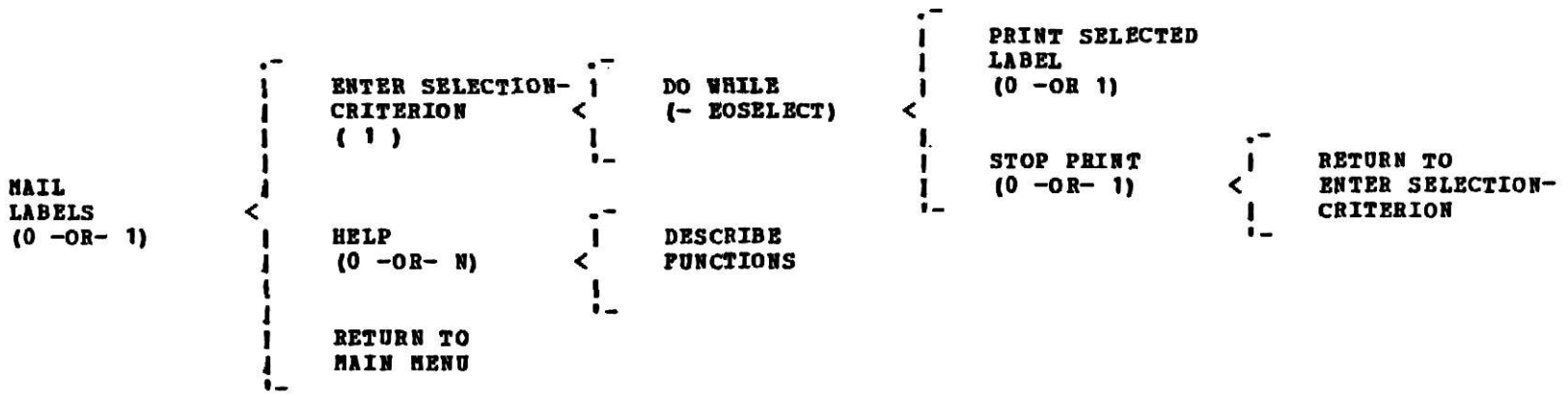


Diagram 2.7

if the user furnishes two criteria. A "case statement" takes care of any combination of operators or relationships that two criteria may generate.

The procedure (local to Mailing_Menu) "Mail_Letter" (see diagram 2.8) allows a user to produce a personalized bulk mailing by specifying a set of selection criteria. The procedure does what the "Mailing_Labels" procedure did except that after selecting an address record, it inserts the name and address information into the address block of the letter or the body of the letter (i.e., salutation appears in the body of the letter). The letter is then displayed on the screen. The user can now either personalize that letter by using the system's editing functions or enter "N" to skip the addressee. If the user skips that address record then the next selected address record will be chosen and the letter will be brought to the screen again. This process will be continued until the end-of-file is reached and no more selected records are to be prepared for mailing.

The procedure (local to Create_Menu) "Add_Letter" (see diagram 2.9) allows a user to create a form letter and store it onto the disk. The form letter is declared as a file of records which contains a two dimension array (20 by 75) of character (this constitutes a single screen). This procedure may invoke several subprograms such as "Move_Up", figure 2.8

MAIL LETTER ROUTINE

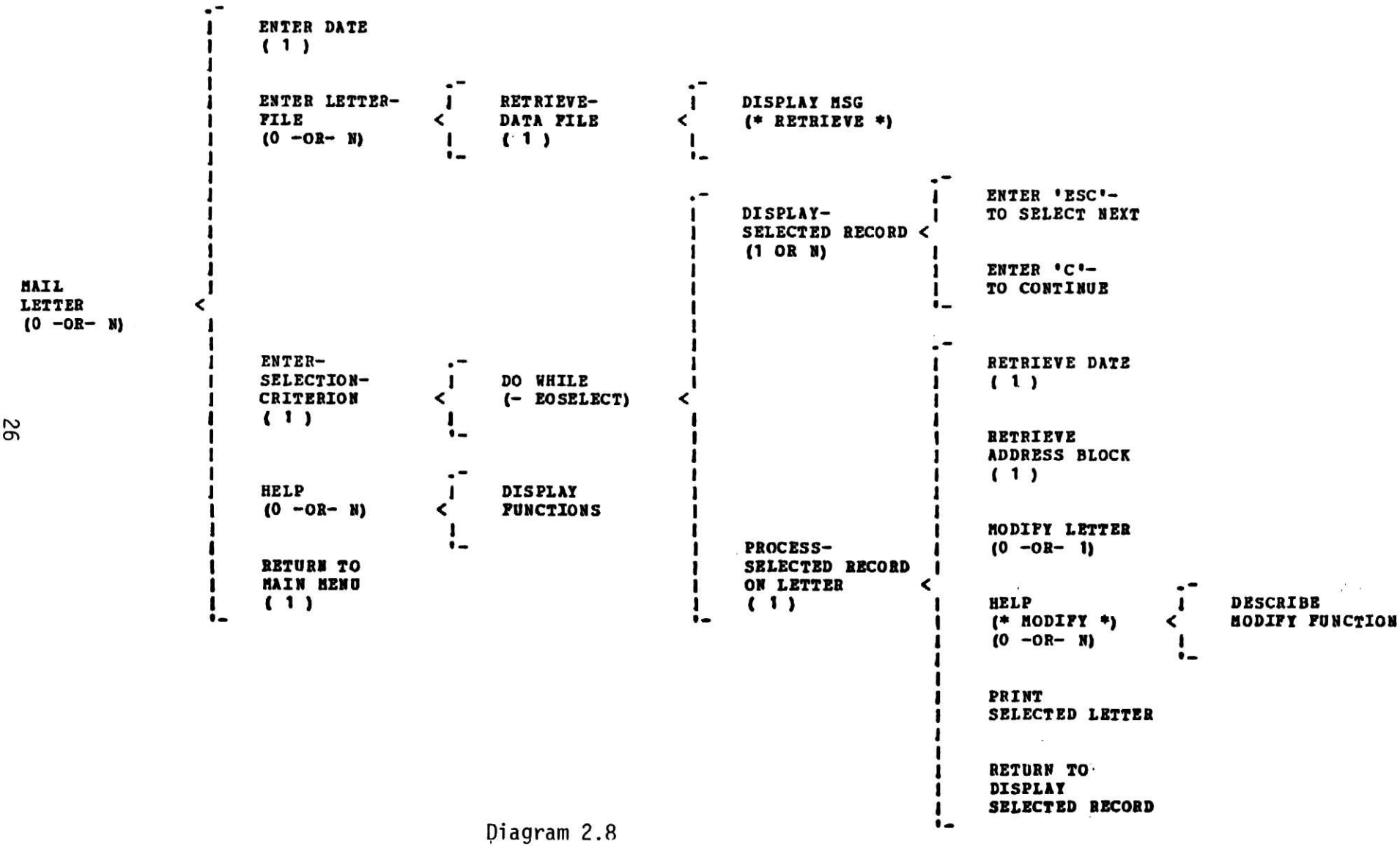


Diagram 2.8

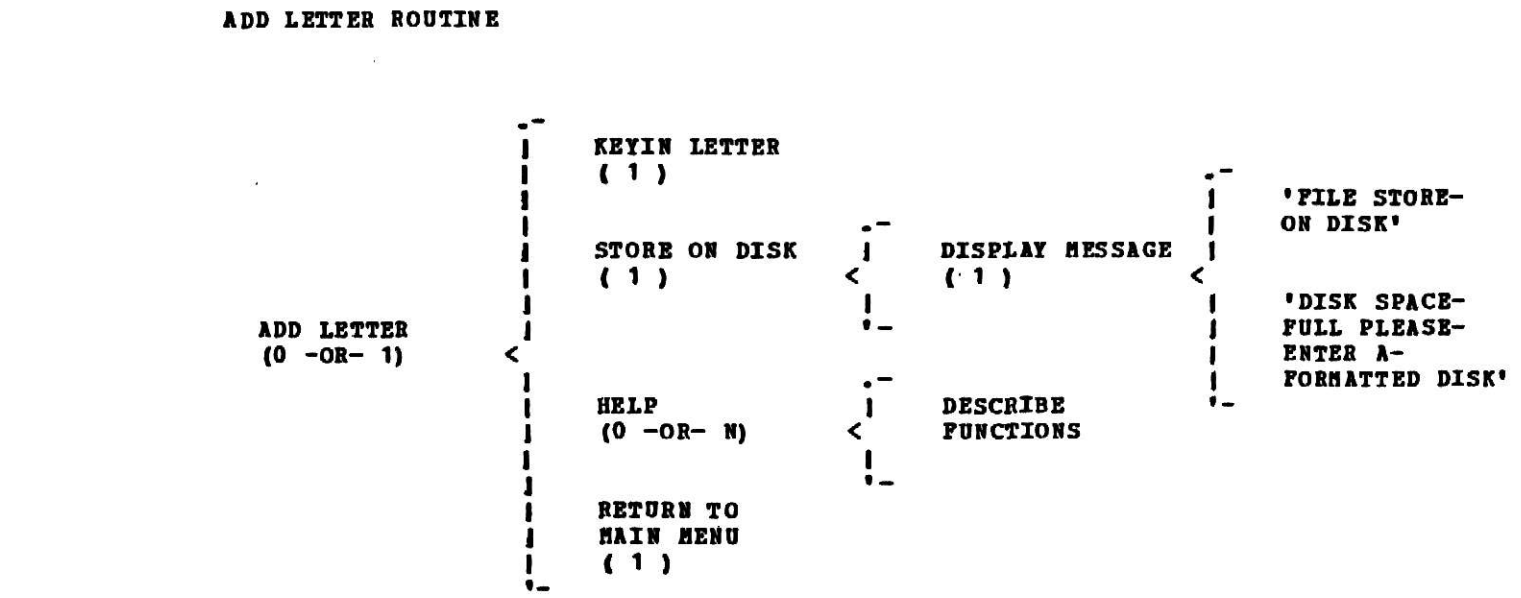

ADD LETTER ROUTINE
ADD LETTER
(0 -OR- 1)
KEYIN LETTER
(1)
STORE ON DISK
(1)
HELP
(0 -OR- N)
RETURN TO
MAIN MENU
(1)
DISPLAY MESSAGE
(1)
DESCRIBE
FUNCTIONS
'FILE STORE-
ON DISK'
'DISK SPACE-
FULL PLEASE-
ENTER A-
FORMATTED DISK'

Diagram 2.9

"Move_Left", "Move_Right", "Insert" and "Delete" if a user requests one or more of these editing features. The local procedure "Insert" allows a user to insert characters into a letter. At this point, the insert procedure saves the insertion pointer (index) that is the position at which a user starts to insert. The insert procedure then reads the insertion characters and the rest of the characters of a letter (after insertion point) into a linear array. The "Reformat" procedure is invoked to analyze that linear array and to reformat it correctly into properly spaced sentences and paragraphs. The reformatted portion of the letter is plugged back into the appropriate cells of the "page" array. During the above process, the procedure checks to insure that the number of inserted characters does not exceed the available field width. If the field width is exceeded, a warning message is displayed for the user. The "Delete" procedure allows a user to delete characters from an existing form letter. The process will be the same as the "Insert" procedure except it now removes some characters from the letter. Before returning to the main menu, the procedure "Add_Letter" checks the saving flag. It will display a warning message to a user if s/he is trying to exit this menu without saving the file.

The data flow diagram of the Bulk Mailing System is shown in diagram 2.10. The Bulk Mailing System stores the standard

DATA FLOW DIAGRAM OF BULK MAILING SYSTEM

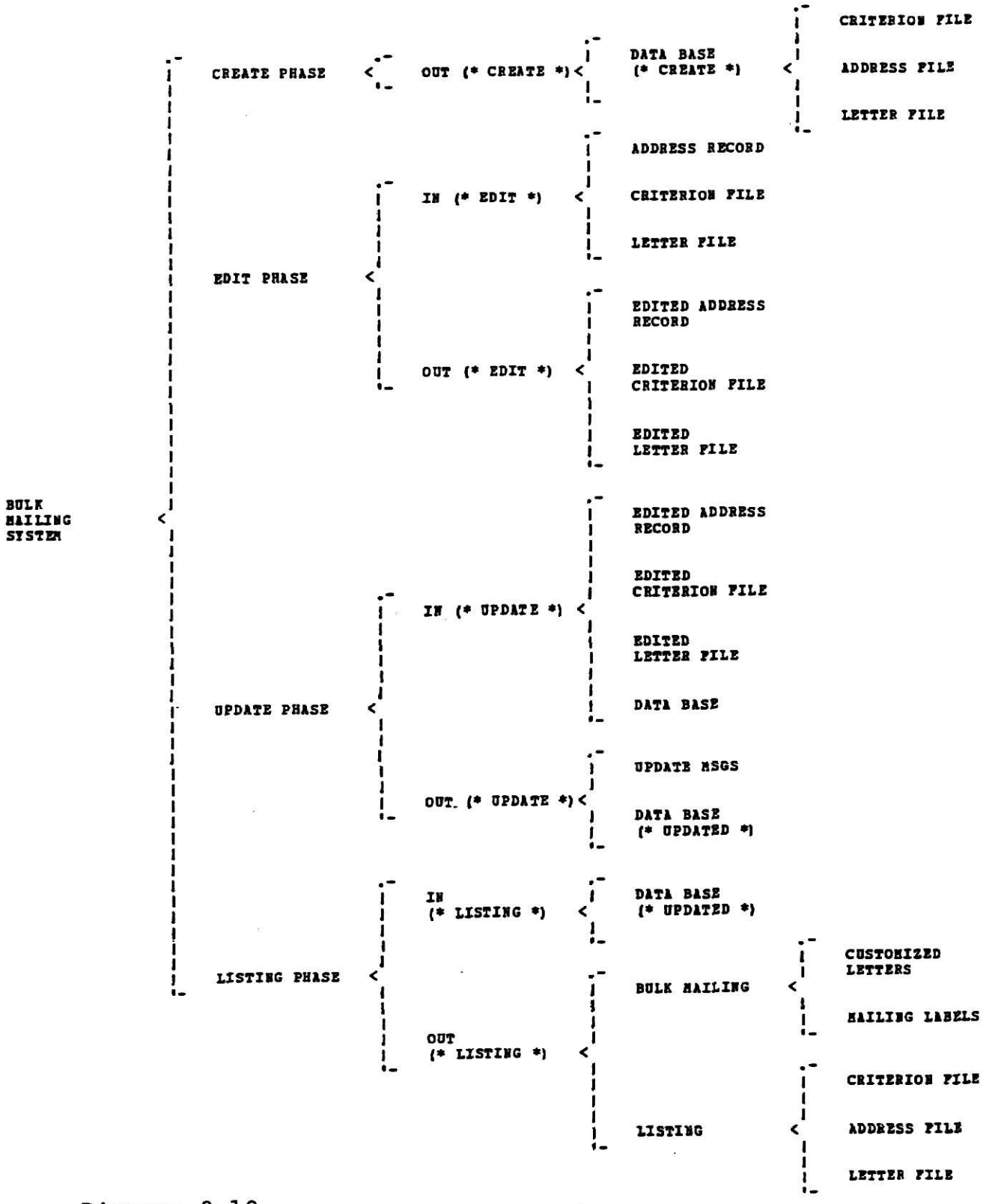


Diagram 2.10

form letters in one data base and the values of the form fields or the values of the criteria fields in one or more additional data bases. The term "Data Base" used here will actually be a file-management system that allows a user to name and specify files with records and fields in a limited way. The file-management system is used because it is so popular and the existing data base packages are quite expensive and limited. Each file has a unique key (file name) which is automatically assigned at creation time. A form letter can be retrieved by means of this key. Then, after a user specifies a set of criteria, the application program will retrieve the information from the address file to complete a form instance.

Since the TURBO Pascal is not a standard Pascal, it is necessary to list some of the built-in procedures or functions of TURBO Pascal that do not exist in standard Pascal and are used in the Bulk Mailing System.

1. Delay(Time)

   The Delay procedure creates a loop which runs for approximately as many milliseconds as defined by its argument which must be an integer. This procedure was used for displaying messages on the screen so that they remained a short period of time before being cleared.

2. ClrEol

   Clears all characters from the cursor position to the end of the line without moving the cursor.

3. ClrScr

   Clears the screen and places the cursor in the upper-left-hand corner. It is used whenever the navigation from the current screen to the next screen is desired.

4. GoToXY(Xpos,Ypos)

   This procedure moves the cursor to any addressable point on the screen. It is used for displaying messages, for input routines and for displaying each menu.

5. Keypressed

   This function is used to check if a key has been pressed at the console.

6. Assign(Filevar,Filename)

   The Filename is assigned to the file variable Filevar, and all further operation on Filevar will operate on the disk file Filename.

7. Seek(Filevar,N)

   This procedure can be used to move the file pointer to the N'th component of the file denoted by Filevar. It is used

in the "Bulk Mailing System" to find the specific point in a file to begin writing or reading the address record.

8. Erase(Filename)

   This procedure is used in "Delete_File" routine to erase the existing disk file associated with Filevar.

9. Copy(Str,Pos,Num)

   Copy returns a string containing Num characters from Str starting at position Pos. This procedure is used to check the file name and determine whether the file is an address file or letter file before next action.

CHAPTER 3

USER GUIDE

3.1 OVERVIEW

This manual explains how to use the mailing program to produce mailing labels or a personalized bulk mailing. This chapter consists of two parts :

.Part 1 - Getting started.

.Part 2 - The MAIL program.

The first part describes what must be done before one can begin to use the MAIL program. This part includes the system requirement of the MAIL program, activating the computer, preparing a diskette for use and making a spare or backup copy of it. These instructions should be enough to prepare one to use the MAIL program. If knowledge of some other DOS commands not listed in part 1 is required, then a user needs to refer to a DOS manual or DOS quick reference card. The second part provides an overview of the MAIL program and directions for its use. This part includes a description of the MAIL program, the files required, and the starting procedures. In addition, the create menu function, the delete menu function, the mail menu function and the print menu function are also described.

3.2 GETTING STARTED

3.2.1 SYSTEM REQUIREMENTS

In order for the Bulk Mailing System to operate properly, a system with the following attributes is required :

a) 8088 based microprocessor system (IBM or IBM compatible)
b) 64 K bytes (or more) of main memory
c) MS/DOS operating system (version 2.0)
d) 2 disk drives
e) A cursor addressable CRT device (24 lines by 80 column CRT)
f) Keyboard
g) Text printer
h) Soft sector and flexible diskettes

3.2.2 ACTIVATING THE COMPUTER

1) Place a system disk (a DOS disk or user's own disk with system commands on it) in drive A;
2) Turn on the computer;
3) Reset the calendar and the clock (this is optional);
4) When the system prompt (A>) appears, the system is booted and ready for use.

3.2.3 PREPARING THE NEW DISKS FOR USE

Before a new disk can receive information from DOS, it must be formatted using the "format" command. This is accomplished as follows :

1) Start the system as described in 3.2.2 above or strike the "Alt", "Ctrl" and Del" keys simultaneously to restart the system;
2) Place the DOS disk in drive A and a new disk in drive B;
3) Use the following command - A> format b:/s/v. The b: directs the computer to format the disk in drive B. The /s option directs it to copy the "command.com" file from the DOS disk to user's disk so that s/he can use this disk as a system disk to boot the system at a later time. The /v option directs it to prompt the user for a volume label as a name for the user's disk. S/he may use initials or last name and a two digit number as the volume label (i.e., HLL01).

Don't worry if the disk review informs a user that s/he has a bad sector on the disk. It will not effect the ability of the disk to accept data. S/he may reformat the disk if desired as this often eliminates a bad sector problem on a new disk.

3.2.4 COPYING A FILE OR AN ENTIRE DISK

Since a file that is inadvertently erased or overwritten is gone forever, it is highly recommended that a backup copy be made and retained in a safe place. The copying of a file or an entire disk may be accomplished as follows :

1) Copying a disk : Insert the DOS system disk into drive A, type "diskcopy a: b:", and press the enter (<—) key. The system displays a message directing a user to insert

the source disk in drive A and the target disk in drive B. After this is done s/he may then strike any key to start. When the copy has been made a message "copy complete" is displayed with a request "copy another (Y/N) ?". The "Y" key is pressed if a user wants to format another disk; otherwise, the "N" key is pressed. This will cause the system prompt (A>) to be displayed.

2) Copying a file : Place the source disk in drive A and the target disk in drive B. The user may use the "dir" command followed by a: or b: to check the files on disk A or disk B respectively. If the source file is on disk A then one may type "copy <file name> b:" to copy that file onto disk B. When the copy has been made, a message "1 file(s) copied" is displayed on the screen.

3) Copying more than one file : The command "copy *.* b:" will copy all the files from the diskette in default drive A to the drive B. Its function is the same as diskcopy except that the "diskcopy" command can only be done if the A disk has a "diskcopy.com" command file.

3.3 THE MAIL PROGRAM

3.3.1 THE OPERATIONAL FEATURES OF THE MAIL PROGRAM

The operational features of the MAIL program will now be discussed. For a business that requires frequent large mailings of a customer list, or to only certain customers on that list, the MAIL program can speed up the preparation of

such mailings and allows complete control of the entire operation. For instance, let us assume that a secretary needs to send out personalized invitations to a product demonstration and that one invitation is needed for every account that does over $30,000 worth of business with the company. The demonstrations will be held in three different locations (East, West, Midwest) on different dates, and the correct information has to be on each invitation.

Further assume that a list of all of the companies is stored on a diskette. It contains the names, addresses and the selection criteria codes of hundreds of customers. The criteria fields includes the information about their credit rating, the volume of business they present, and other key information. By preparing a selection specification s/he can have the MAIL program search the file and generate a list of customers who have done over $30,000 worth of business with the company. S/he then begins by retrieving and displaying a invitation from an existing form file. The address block will be inserted automatically from the newly generated customer list. S/he can now customize the invitation by looking at the address block and thus determine and insert the appropriate location for the demonstration. At this point the invitation is complete and ready for printing. The MAIL program should save time, reduce costs and improve written communication with customers.

3.3.2 FILES REQUIRED IN THE MAIL PROGRAM

Basically, a file is a collection of information residing on a mass storage device that contains the user's data. The information can be stored in or retrieved from the file. Files for this system can be grouped into 3 basic types with each type concerned with a particular feature of the MAIL program. All of the MAIL program files have a name field of eight characters and a file type used in the MAIL program as follows:

- .CRI - Designates a criterion file that contains 20 criteria IDs, their associated criteria codes and the meaning of each code.
- .ADS - Designates an address file that contains 10 criteria fields and address fields which are required in a letter.
- .LTR - Designates a file that contains a prewritten form letter.

Since the user's data file is stored on drive B, it is essential that the user specify drive B followed by the filename.filetype. Otherwise, the MAIL program will search for the file on default drive A and will give the user a warning message that the file does not exist.

3.3.3 STARTING THE MAIL PROGRAM

If the user has been using DOS or running other programs, s/he can easily switch to the MAIL program. Before doing so, however, s/he must be sure to exit from the other program so

that the screen shows the system prompt (A>). The user then puts the disk which contains the MAIL program (MAIL.COM file) into drive A and the data file disk in drive B. Type in "mail" and press the enter (<—) key. The MAIL program will be loaded into main memory and the main menu of MAIL will appear. The user is now ready to begin using MAIL.

The main menu (see figure 3.1) has 6 functions. A user can use this menu to select the function s/he wants MAIL to perform. S/he needs to fill in the selection type that s/he wants to perform (i.e., M selects Mailing, X selects eXit MAIL program, etc).

To assist new or infrequent MAIL program users, each submenu has a help option. If the user selects a function that s/he is unsure of, simply pressing the "H" key will cause the MAIL program to display a brief explanation of that function as well as a reference to the manual that will give the user detailed instructions. This feature is especially useful to the "casual" user or to the person who can not recall how to perform an infrequently needed function. Also, the MAIL program will accept either upper case or lower case letters as valid input so that in this respect it is quite user-friendly.

3.3.4 CREATE MENU FUNCTION

The create menu function (see figure 3.3) has five options. Pressing the "H" key will display the help menu (see figure 3.4), and the "R" key will return a user to the main

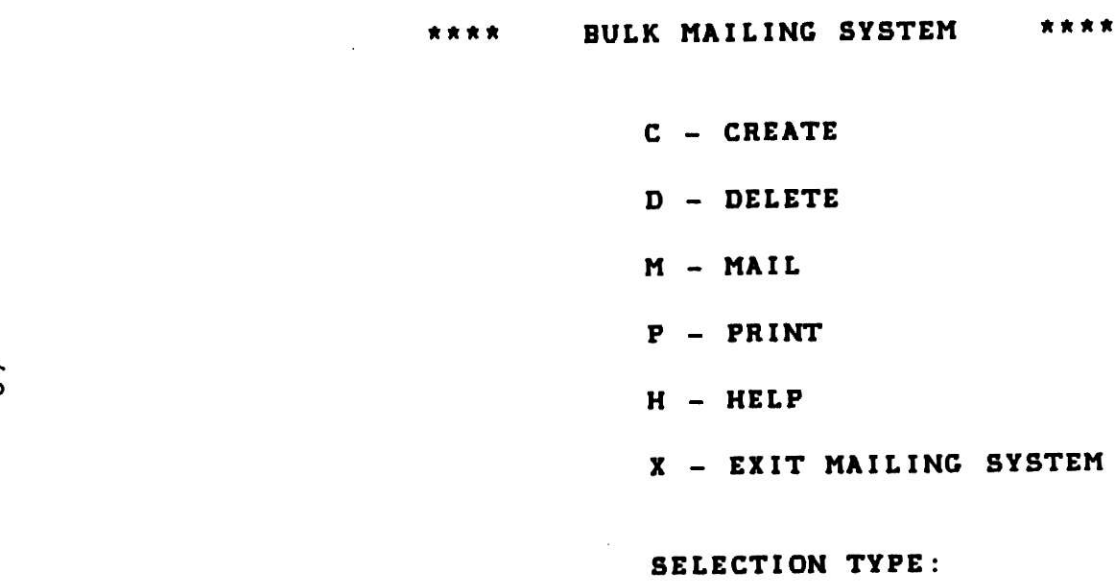

**** BULK MAILING SYSTEM ****
C - CREATE
D - DELETE
M - MAIL
P - PRINT
H - HELP
X - EXIT MAILING SYSTEM
SELECTION TYPE:

Figure 3.1

**** BULK MAILING SYSTEM HELP ****

C - allows the creation of 1) new address file, 2) new letter file, 3) new address in an old file, 4) criteria file

D - allows the deletion of an address, address file or letter file

M - allows the creation of an address label or a bulk mailing

P - allows the printing of the address file or letter file on the screen or printer.

X - Exit from mailing system to main system.

press "R" to return you to mailing system menu -->

Figure 3.2

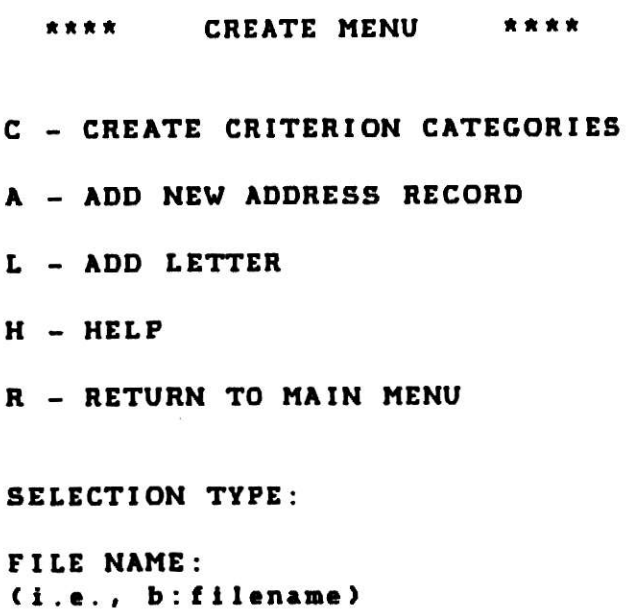

**** CREATE MENU ****
C - CREATE CRITERION CATEGORIES
A - ADD NEW ADDRESS RECORD
L - ADD LETTER
H - HELP
R - RETURN TO MAIN MENU
SELECTION TYPE:
FILE NAME:
(i.e., b:filename)

Figure 3.3

**** CREATE MENU HELP ****

C - allows the creation of selection criteria codes that may be used to categorize stored information for mailing process later.

A - allows the creation or updating of the address file

L - allows the creation of a letter file

FILENAME - The file you are going to add or create having the format b:filename where b disk and filename is 1-8 characters

press "R" to return you to "create menu" -->

Figure 3.4

menu. The "C", "A", and "L" keys will bring the cursor down to "FILE NAME:" after a user selects one. The file name is the key of the file. Since the system will generate the file type for a user by the user's selection type, a user does not have to key in the file type. S/he needs to type "b:" which means B drive and the 1-8 characters of the "FILE NAME". Following is the explanation of these 3 options :

"C" - Enables a user to create a criterion category file (see figure 3.5) which allows a user to set up criteria codes. This criterion file will be used as a reference when a user is creating or displaying an address file. The file name entered under "CREATE MENU" is shown on the right hand bottom corner of the display, so that the user will always know what file s/he is dealing with at any point. The system generates a file type "CRI" for the criterion file. Also, a user can observe several options at the bottom of the display. Pressing the "R" key will return to the main menu, whereas pressing the "H" key will display the help menu (see figure 3.6). The "A" key must be pressed if a user wants to add information into a new criterion file. If the file already exists on the diskette, then the prompt "Do you want to overwrite the previous file (N/Y) ?" is displayed and the system waits for the user's response. If the user enters "Y" the cursor will move to the first field of this screen and s/he can now start to enter the contents of the criterion file by typing each field. Remember that

```
****     CREATE CRITERION CATEGORIES     ****

                    CRIT   CODE   MEANING
CRIT 1   :          sex      f     female
CRIT 1   :                   m     male
CRIT 2   :          occp     s     staff
CRIT 2   :                   f     faculty
CRIT 00  :
CRIT     :
CRIT     :
CRIT     :
CRIT     :
CRIT     :
CRIT     :
CRIT     :
CRIT     :
CRIT     :
CRIT     :
CRIT     :
CRIT     :
CRIT     :
CRIT     :
CRIT     :
-->a
R - Return to main  A - Add contents    H - Help   00 - Store  hII.CRI
```

Figure 3.5

**** CREATE CRITERION CATEGORIES HELP ****

The number of codes of criterion may not be exceed 20. Enter codes as illustrated below. Crit allows 1-4 chars, codes allows 1 and meaning allows 1-7 chars. Type "00" when you are ready to store file.

		CRIT	CODES	MEANING
CRIT	1	sex	f	female
CRIT	1		m	male
CRIT	2	dept	1	000-100
CRIT	2		2	101-200
CRIT	2		3	201-300

To change to a new criterion, type over the criterion number with the next number in sequence.

		CRIT	CODES	MEANING
CRIT	1	sex	f	female
CRIT	1		m	male
CRIT	2	dept	1	000-100
CRIT	2		2	101-200
CRIT	2		3	201-300
CRIT	3	occp	f	faculty
CRIT	3		s	student
CRIT	4			

Press "R" to returns you to create criterion menu -->

Figure 3.6

the criteria number must be in the range 1 through 10, the "CRIT" field will accept 1 through 4 characters, the "CODE" field will accept one character and the "MEANING" field will accept 1 through 7 characters. When the end of each field has been reached, the enter (<—) key must be pressed. This action will cause the cursor to jump to the next field. If a user finds that a typing error has been made and that the cursor is still in the same field, a correction can be made by simply backspacing to the error and re-keying the field. If the mistake is not discovered until after a user has progressed to a subsequent field, s/he can type "00" (zeros) to exit the creation mode and start over again. Only 20 criteria codes are allowed. If a user tries to enter the 21st code, a warning message is displayed and the 20 existing criteria will be automatically stored onto the disk. If the user has entered fewer than 20 criteria codes and is ready to exit the creation mode, s/he can simply type "00" (zeros) to store the file onto the disk and exit the mode.

"A" - Enables a user to create an address file (see figure 3.7) which allows a user to create and store the criteria information, names and addresses on the disk. The system will generate the file type "ADS" for the file and again this information will be displayed on the bottom line of the screen for reference. The option "R" will return to the main menu or the option "H" will display the help menu (see

```
                    ****     ADD NEW ADDRESS RECORD     ****
                                                          CRIT   CODE   MEANING
     CRIT 1     :                                  CRIT 1  :sex     f     female
     CRIT 2     :                                  CRIT 1  :        m     male
     CRIT 3     :                                  CRIT 2  :dept    1     000-100
     CRIT 4     :                                  CRIT 2  :        2     101-200
     CRIT 5     :                                  CRIT 2  :        3     201-300
     CRIT 6     :                                  CRIT    :
     CRIT 7     :                                  CRIT    :
     CRIT 8     :                                  CRIT    :
     CRIT 9     :                                  CRIT    :
     CRIT 10    :                                  CRIT    :
     ID_NO      :                                  CRIT    :
     NAME       :                                  CRIT    :
     ADDRESS1   :                                  CRIT    :
     ADDRESS2   :                                  CRIT    :
     CITYSTZIP  :                                  CRIT    :
     COUNTRY    :                                  CRIT    :
     SALUTATION:                                   CRIT    :
     CLOSING    :                                  CRIT    :
                                                   CRIT    :
                                                   CRIT    :

-->509828407              Please enter Id_No - 9 digits
R  - Return to main A - Add  S - Store  H - Help  C - restart  address.ADS
```

Figure 3.7

**** ADD NEW ADDRESS HELP ****

You may 1) Create or update the address record by entering "A"
then answer the request by keyin the name of the criterion file

2) When entering new data, follow the code listed for Crit 1 to Crit 10 on the right side of the screen. If there are no codes listed, create selection criterion categories before you add new address into the file.

3) Update the old records by answering the prompt "y" if the address already exists, and type "n" if you want to restart instead of updating that record.

4) Press ↑ PrtSc keys to get a listing from the printer.

press "R" to return you to add new record menu -->

Figure 3.8

figure 3.8). The user may enter the "A" key to add an address record into the address file. The prompt "Please enter the name of the criterion file" is displayed on the screen. If the requested criterion file exists, the values of the criteria in the file are displayed on the right side of the screen. The user can now enter a 9 digit ID number. After the ID number has been entered one of two things can occur. The system could inform the user that this person is a new record. If this is the case, the user can start to enter the contents of each field followed by the enter (<—) key. If a user finds that s/he has made a typing error, and s/he is out of the field, s/he must travel to the bottom of the screen and enter the "C" key for a restart. If the record is correct, typing "S" will store it onto the disk. On the other hand, if the record already exists, the system will display a message "record already exists, overwrite (N/Y) ?" to the user. The answer "Y" moves the cursor to the first field and the system allows a user to enter information for that person. By pressing the "N" key this action will be ignored. In order to be user-friendly, if the user designates the same criterion file for each record, s/he does not have to type the name of the criterion file every time. S/he may simply enter "Y" for the prompt : "Do you want to use the same criterion file (N/Y) ?".

"L" - Enables a user to create a letter file (see figure 3.9) which allows a user to create a form letter and store

```
New Letter - test.LTR     Type \ for commands
\ restore Quit, S - Save, P - Print, H - Help, R - Return,
```

Figure 3.9

it on the disk. The system will generate the file type "LTR" for the file and this information will be displayed on the bottom line of the screen for reference. If the file name entered by a user already exists, this file will be retrieved and displayed on the screen and the user can make any changes if desired. If the file does not exist, a message will be displayed to inform the user that it is a new letter. This will be followed by the file name. The user can then begin typing a new form letter. There are several options in this menu. Pressing the "\" key will display a menu of commands for creating or manipulating a letter file. A second "\" brings the user back to the create (or edit) mode, pressing the "S" key will save the file on the disk, pressing the "P" key will inform the user how to get a printout from the text printer, pressing the "H" key will display a help menu (see figure 3.10), and pressing the "R" key will return to the main menu. If a user presses the "R" key before s/he saves the file, the system will display a warning message to remind the user to save the file. S/he may either press the "Y" key to save the file or the "N" key to exit the mode without saving the file. When the user is typing a letter s/he can use the four arrow, Del and Ins keys on the keyboard to edit the letter. When s/he enters the "Ins" key, s/he is in the insert mode and entering the "Ins" key a second time will return a user to the create (or edit) mode.

**** ADD LETTER HELP ****

You may write a letter like this:

@date@

@name@
@address1@
@address2@
@city@, @state@ @zip@
@country@

Dear @salutation@:

Here it is, the first date of 1984. Remember, your past due amount is *****.

@closing@

your name

1) The words and phrases between the @ @ are variable names. A variable name can be either upper or lower case. The value of the date is entered by the operator, and the rest of the values for the variables are substituted from the address file automatically while running the mailing process. The **** mark is the place for modifying a personalizing letter during the mailing process. Make sure the variables in the letter is entered exactly same as above. When the mailing system substitutes a variable value for a variable name it also removes "@@" that surround it. If it cannot find the variable value in the data file, the variable field will not be substituted.

2) Press "R" to return you to "add letter menu" -->

Figure 3.10

3.3.5 MAIL MENU FUNCTION

The Mailing menu shown in figure 3.11 gives a user four options. Pressing the "L" key will produce the bulk mailing, pressing the "A" key will produce the labels, pressing the "H" key asks for help (see figure 3.12), and pressing the "R" key will return the user to the main menu. If a user chooses either the "L" or the "A" option the system will request that the name of the address file be entered. The user can follow the example shown on the screen to enter the file name. If the user enters the option "L", the submenu mailing letter shown in figure 3.13 will appear. The user can enter the "H" key to display the help menu on the screen (see figure 3.14) or press the "R" key to return to the main menu. Otherwise, the selection "L" will start the mailing process and the cursor will move to the first query "ENTER DATE:". The date (the date of the letter to be written) entered here will appear on the letter. The system then prompts the user to enter the file name of the letter which s/he wants to retrieve from the disk for the mailing process. If the file does exist, the system requests that the user either enter "Y" to personalize each letter or "N" to produce the bulk mailing without personalizing each letter. The "SELECTION CRITERIA:" of figure 3.14 allows a user to produce a mailing list by specifying a set of selection criteria. The system gives the user the ability to enter either one or two criteria. The syntax of the selection criteria first specifies the criterion

```
****     MAILING MENU     ****

  L - MAIL LETTERS

  A - MAIL LABELS

  H - HELP

  R - RETURN TO MAIN MENU

  SELECTION TYPE:

  FILE NAME :
  (i.e., b:filename.ext)
```

Figure 3.11

```
****     MAIL MENU HELP     ****

A - allows the creation of bulk mailing

L - allows the creation of mailing labels

FILENAME - The name of the address file
           b: is disk drive, filename is 1-8 chars and ext is 3 chars


press "R" to return you to mail menu -->
```

Figure 3.12

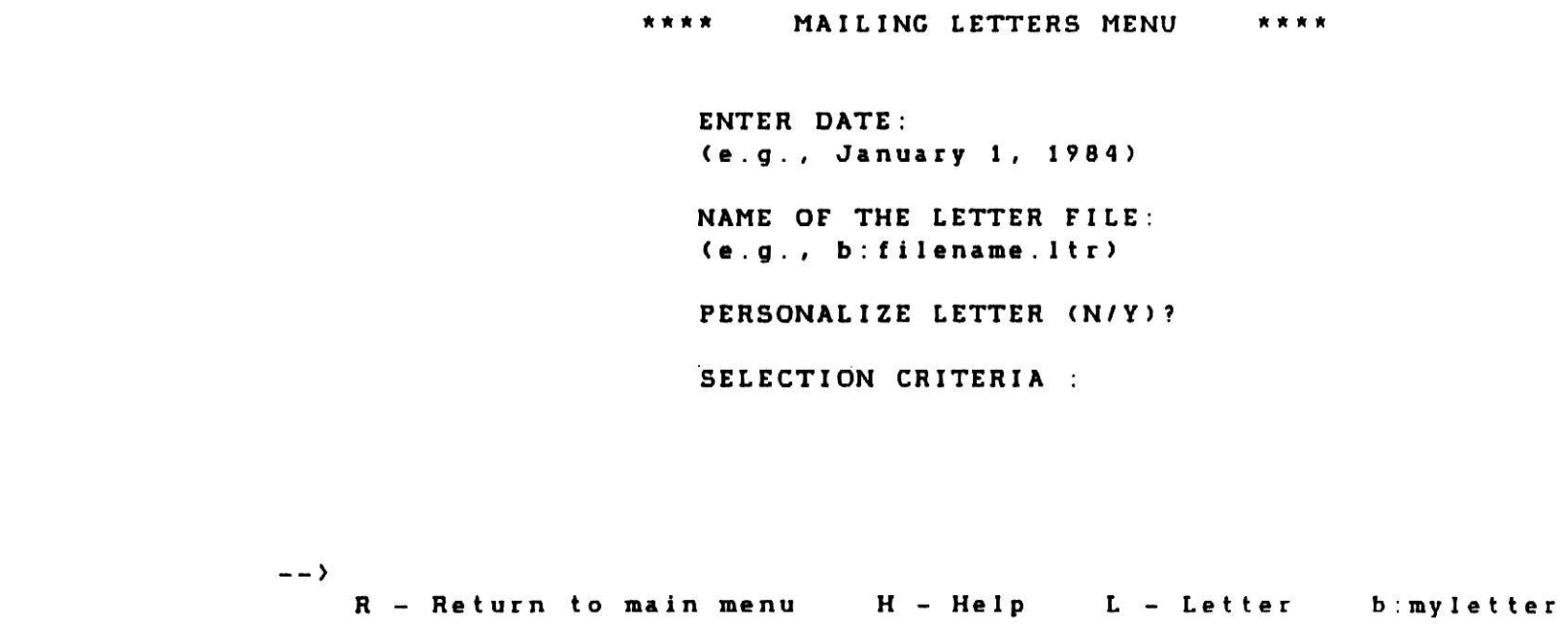

**** MAILING LETTERS MENU ****
ENTER DATE:
(e.g., January 1, 1984)
NAME OF THE LETTER FILE:
(e.g., b:filename.ltr)
PERSONALIZE LETTER (N/Y)?
SELECTION CRITERIA :
-->
R - Return to main menu
H - Help
L - Letter
b:myletter

Figure 3.13

**** MAILING LETTER HELP ****

You may create a bulk mailing by following these steps :

1) Enter the date of the letter
2) Enter the name of the letter file
3) You may personalize the letter if you want, type "y" to change the letter, or type "n" to continue
4) Enter one or two selection criteria, i.e., 01=F/02=4 where "/" means "or" and "," means "and" or enter "all" to select all records
5) If no records are selected then the system display the messages said "no records match the condition". Or if there are some records match the criteria, then the system will check that if you want to change the letter contents. If answer is "y", the screen brought you a letter which has the data fields inserted to appropriated places already. Otherwise, the system will ask you to strike "p" to start printing the bulk mailing letter and display messages when the printing is finished.
6) If you want to personalize the letter, you may change the contents of the letter interactively, then press ↑ PrtSc keys to print the letter.
7) Enter "C" to continue processing the next selected record or press "N" to skip to next selected record or enter "R" to returns you to main menu

Press "R" to returns you to mailing letter menu -->

Figure 3.14

code number (in range 01, 02 ..., 10), then the operator (">" for greater than, "<" for less than and "=" for equal), and finally the criterion code (1 character or 1 digit) or simply enter "all" to select all of the address records from the address file. If there are two selection criteria, a user must enter a relational operator symbol ("," means "and", "/" means "or") before s/he enters the second criterion. Once this is done, the system selects the records that match the criteria entered by the user and retrieves the name and address for the letter. A user can personalize that letter if s/he previously asked to do so; otherwise, the system will ask the user to strike "P" to start printing the letters and will display a message when the printing is completed. When a user is personalizing the letter, s/he may enter "C" to process the next selected record or press "N" to skip to next selected record.

If a user chooses the option "A" to generate the mailing labels, the mailing labels menu will be displayed on the screen (see figure 3.15). The user can either press the "H" key to ask for help (see figure 3.16) or the "L" key to generate the labels. Once this is done, the user needs to enter the selection criteria (same as mailing letters option). If the criteria entered by a user are not syntactically correct, the system will display an error message with a request to the user to either re-enter the criteria or press

```
****     MAILING LABELS MENU     ****


SELECTION CRITERIA :
(i.e., 01=f,10>2)


-->I
    R - Return to main menu     H - Help     L - Label    address.ads
```

Figure 3.15

**** MAILING LABELS HELP ****

You may produce a mailing labels by giving one or two criteria,i.e., (01=M,10=2) where "01" is criterion 1, M is code, "," means "and" ("/" means "or") and "10=2" is the second criterion; or enter "all" to select all the records. After the selection criteria are entered, the printing program is invoked to print the selected records.

press "R" to return you to mailing labels menu -->

Figure 3.16

the "R" key to exit the mailing labels function.

3.3.6 DELETE MENU FUNCTION

This section explains how to delete the address records from an address file or to erase any existing files that are no longer needed. The delete menu is shown in figure 3.17. Pressing the "H" key will cause the help menu to be displayed on the screen (see figure 3.18) whereas, pressing the "R" key will cause a return to the main menu. A user may enter "delete address records function" by typing "A". Once this is done, the submenu shown in figure 3.19 will appear. To begin a deletion, enter a "D" and the cursor will move to the proper place where the 9 digit ID number is to be entered. After the ID number has been entered, the system displays one of two messages to inform the user either that "the record is not found" (record does not exist) or "are you sure you want to delete this record (N/Y) ?" (record exists). The user needs to be sure that s/he wants to delete the record before s/he enters the "Y" key; otherwise, the record is deleted. The help menu of "delete address menu" is shown in figure 3.20.

If the user selects the "F" option in the delete menu (see figure 3.17), the system will ask the user to enter the name of the file s/he wants to delete. If the file exists, a warning message is displayed on the screen that gives the user a chance to reconsider before s/he actually destroys the file.

```
****     DELETE MENU     ****

 A - DELETE ADDRESS

 F - DELETE FILE

 H - HELP

 R - RETURN TO MAIN MENU

 SELECTION TYPE:

 FILE NAME:
 (i.e., b:filename.ext)
```

Figure 3.17

```
****     DELETE MENU HELP     ****

A - allows the deletion of address records

F - allows the deletion of a file

FILENAME - The file you are going to delete
           or the address file for deleting address record.
           b: is disk drive, filename is 1-8 chars and ext is 3 chars

press "R" to return you to delete menu -->
```

Figure 3.18

```
****     DELETE ADDRESS MENU     ****

ID NUMBER TO BE DELETED:

-->d

R  - Return to main        D - Delete     H - Help    address.ads
```

Figure 3.19

**** DELETE ADDRESS FILE HELP ****

You may 1) Enter the "D" then an ID number of the record to delete an address from an address file.

press "R" to returns you to delete address menu --)

Figure 3.20

3.3.7 PRINT MENU FUNCTION

This section describes how to use the print function to produce a hard copy (paper) or soft copy (on the screen) of the information stored in the data file. When a user uses this function, the submenu will be displayed on the screen (see figure 3.21). A user may enter the "R" key to return to the main menu or press the "H" key to ask for help (see figure 3.22). If the user wants the file to be displayed on the screen, the "D" key must be pressed followed by the name of the file. The system will then check the file type to determine which submenu will be displayed next. The file type "LTR" displays the letter on the screen (see figure 3.23) and the file type "ADS" displays an address file on the screen (see figure 3.25). In order to display the address record, the user needs to enter the "D" key. The system then requests that a user enter the name of the criterion file. If the user does not want the criterion file to be displayed s/he may just press the enter (<—) key to continue. At this point, the first address record is displayed on the screen. The user may enter the "H" key if help is needed (see figure 3.26) or return to the main menu by pressing the "R" key. A user also may check all the address records by entering successive "C"s or individually check an address by pressing the "S" (search) key and filling in the 9 digit identification number following "NEXT ID -->". (see figure 3.27)

```
****     PRINT MENU     ****

 D - DISPLAY ON SCREEN

 P - PRINT

 H - HELP

 R - RETURN TO MAIN MENU


 SELECTION TYPE:

 FILE NAME:
 (i.e., b:filename.ext)
```

Figure 3.21

```
****     PRINT MENU HELP     ****

D - allows the display of the address file or letter
    file on the screen.

P - allows the printing of the address file or letter
    file from the printer

FILENAME - The file you are going to display
           or print. b: is disk drive, filename is 1-8 chars and ext is 3.


press "R" to return you to print menu -->
```

Figure 3.22

```
@date@

@name@
@addrs1@
@addrs2@
@citystzip@
@country@

Dear @salutation@:

Please send us the past due amount of ********** immediately.

                                    @closing@


                                    Hui-Ling Lin
                                    Accounts receivable department



-->
        R - Return to main menu   H - Help      b:a.ltr
```

Figure 3.23

**** DISPLAY LETTER FILE HELP ****

You may 1) Check the contents of the letter but you will not be able to change the letter. You will have to go to the "add letter file menu" next.

2) Print the letter on screen by pressing ↑ PrtScs.

press "R" to returns you to display letter file -->

Figure 3.24

```
                    ****     DISPLAY FILE     ****

CRIT 1      : f                                  CRIT 1   :sex     f      female
CRIT 2      : 2                                  CRIT 1   :        m      male
CRIT 3      :                                    CRIT 2   :dept    1      000-100
CRIT 4      :                                    CRIT 2   :        2      101-200
CRIT 5      :                                    CRIT 2   :        3      201-300
CRIT 6      :                                    CRIT     :
CRIT 7      :                                    CRIT     :
CRIT 8      :                                    CRIT     :
CRIT 9      :                                    CRIT     :
CRIT 10     :                                    CRIT     :
ID_NO       : 987654321                          CRIT     :
NAME        : Mr. John Doe                       CRIT     :
ADDRESS1    : Dept. of Computer Science          CRIT     :
ADDRESS2    : Kansas State University            CRIT     :
CITYSTZIP   : Manhattan, KS 66506                CRIT     :
COUNTRY     :                                    CRIT     :
SALUTATION: John                                 CRIT     :
CLOSING     : yours                              CRIT     :
                                                 CRIT     :
                                                 CRIT     :

-->c
C - Continue R - Return main H - Help S - Search next D - Display b:address.ads
```

Figure 3.25

**** DISPLAY ADDRESS FILE HELP ****

You may 1) Display the address file by entering "D".

2) Individually check an address by entering "S" and filling in the identification number following "NEXT ID -->".

3) Print the information on the screen by pressing ↑ PrtSc at the same time.

4) Check all addresses by pressing successive "C"s.

press "R" to returns you to display address menu -->

Figure 3.26

```
                    ****     DISPLAY FILE     ****

CRIT 1      : f                                  CRIT   CODE   MEANING
CRIT 2      : 2                    CRIT 1   :sex     f      female
CRIT 3      :                      CRIT 1   :         m      male
CRIT 4      :                      CRIT 2   :dept     1      000-100
CRIT 5      :                      CRIT 2   :         2      101-200
CRIT 6      :                      CRIT 2   :         3      201-300
CRIT 7      :                      CRIT     :
CRIT 8      :                      CRIT     :
CRIT 9      :                      CRIT     :
CRIT 10     :                      CRIT     :
ID_NO       : 987654321            CRIT     :
NAME        : Mr. John Doe         CRIT     :
ADDRESS1    : Dept. of Computer Science  CRIT     :
ADDRESS2    : Kansas State University    CRIT     :
CITYSTZIP   : Manhattan, KS 66506        CRIT     :
COUNTRY     :                      CRIT     :
SALUTATION: John                   CRIT     :
CLOSING     : yours                CRIT     :
                                   CRIT     :
                                   CRIT     :

NEXT ID -->
C - Continue R - Return main H - Help S - Search next D - Display b:address.ads
```

Figure 3.27

If the "P" option is selected under the print menu (see figure 3.21), the existing file will be printed to the text printer (hard copy).

CHAPTER 4

CONCLUSION

The Bulk Mailing System is "menu-driven", hence the selection of system screens only requires a single keystroke. Also, each screen has a "Help" menu to assist a user in determining what functions can be performed with each screen. Therefore, the Bulk Mailing System has achieved one of its goals : user-friendliness. The other objective of the Bulk Mailing System (flexibility) is achieved by allowing a user to set up selection codes before the mailing list is created. These selection codes can then be used to effectively and efficiently prepare the mail for a particular targeted population.

The source code of the Bulk Mailing System takes 80382 bytes and the executable code "MAIL.COM" takes 50934 bytes. The total compile time required is 2 minute and 41 seconds. The TURBO Pascal compile speed of approximate 15 lines per second lives up to its name "turbo", as it is significantly faster than other Pascal compilers for microcomputers.

Future work of this project might include the following :

.To link the MAIL system into the Volkswriter wordprocessor in order to enhance the editing function.

.To allow an user to delete the address record interactively by using the editing function key.

.To integrate a commerical DBMS into the MAIL program to build the address file from an existing data base.

.To develop a multipage (multiscreen) form letter for the MAIL program.

.To provide a tutorial program to teach the potential user (electronic teacher).

BIBLIOGRAPHY

1. Chorafas, Dimitris N.. Office Automation - The Productivity Challenge, Printice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.
2. Gehani, Narain H.. "The Potential of Forms in Office Automation," IEEE Transaction on Communications, Vol. Com-30, No. 1, Junuary 1982.
3. Tsichritzis, D.. "OFS : An Integrated Form Management System," Computer System Research Group, University of Toronto, 1980.
4. Veit, Stanley S.. Using Mircocomputers in Business, Hayden Book Company, Inc., Rochelle Park, New Jersey, 1983.
5. Curtin, Dennis P.. The WardStar Hand Book, Curtin and London, Inc. and Van Nostrand Reinhold Company, 135 West 50th Street, New York, 1983.
6. Orr, Kenneth T. Introducing Structured Systems Design, Infotech International Limited, Maidenhead, England.
7. Higgins, Daid A. "Warnier-Orr Diagrams of Auerbach Computer Programming in Management, Auerbach Publishers Inc.
8. Goldberg, Kenneth P. Microcomputers, John Wiley and Sons, INC., 1983.
9. Cherry, Geroge W. Personal Pascal : Compiled Pascal for the IBM Personal Computer, A Prentice-Hall Company,

Reston, Virginia 22090, 1984.

10. IBM Disk Operating System Reference Manual, Microsoft, INC., Boca Raton, Florida 33432, 1983.

11. TURBO Pascal Reference Manual, Boralnd International Inc., 4808 Scotts Valley Drive, Suite 1, Scotts Valley, CA 95066, 1983.

APPENDIX

```
(*****************************************************************)
(*                                                               *)
(*   - BULK MAILING SYSTEM (MAIL.PAS).                           *)
(*                                                               *)
(*****************************************************************)

Program Mailing;
{$V-}
Const
   Max_Adds = 299;
   Hash_Prime = 293;
   FxMax = 74;
   FyMax = 20;
   Max_Crit = 20;
   Bell = ^G;
type
   Anystring = string[75];
   Code_Array = array [1..10] of char;
   AddsRecs = record
                Crit_Code   : Code_Array;
                Id_No       : string[9];
                Name        : string[25];
                Addrs1      : string[25];
                Addrs2      : string[25];
                CityStZip   : string[25];
                Country     : string[15];
                Salutation  : string[20];
                Closing     : string[20];
              end;
   AFiles = file of AddsRecs;
   Critrecs = record
                 No : string[2];
                 Id : string[4];
                 Code : char;
                 Meaning : string[7];
              end;
   CFiles = file of Critrecs;
   LettRecs = record
                Ch : char
              end;
LFiles = File of LettRecs;
var
   Letter : array [0..FxMax,0..FyMax] of LettRecs;
   Crit : array[1..Max_Crit] of Critrecs;
   Critfile : CFiles;
   Lettfile : LFiles;
   Ch : char;
   First, Add, Found, Free, Type_Ok, Store : boolean;
   Filen, Filename : string[15];
   Addsfile : AFiles;
   BlankAdds, TempAdds, InAdds : AddsRecs;
   TempId_No : string[9];
   Fx, Fy, Index : integer;
   Value, Value1, Operator1, Operator, Relation : char;
```

```
    Crit_Index1, Crit_Index : integer;
    Error, Select_All, One_Criterion : boolean;

{$I menuhelp.pas}

{*****************************************************************}
{*   Following are the 23 procedures for displaying the          *}
{*   screens of the MAIL program.                                *}
{*****************************************************************}


Procedure Display_Main_Menu;
begin
    ClrScr;
    gotoxy(18,3);
    write('****     BULK MAILING SYSTEM     ****');
    gotoxy(29,6);
    write('C - CREATE');
    gotoxy(29,8);
    write('D - DELETE');
    gotoxy(29,10);
    write('M - MAIL');
    gotoxy(29,12);
    write('P - PRINT');
    GoToXY(29,14);
    write('H - HELP');
    gotoxy(29,16);
    write('X - EXIT MAILING SYSTEM');
    gotoxy(29,19);
    write('SELECTION TYPE:');
end;   (*  Display_Main_Menu  *)

Procedure Display_M_Menu;
begin
    ClrScr;
    gotoxy(22,4);
    write('****     MAILING MENU     ****');
    gotoxy(24,7);
    write('L - MAIL LETTERS');
    gotoxy(24,9);
    write('A - MAIL LABELS');
    gotoxy(24,11);
    write('H - HELP');
    gotoxy(24,13);
    write('R - RETURN TO MAIN MENU');
    gotoxy(24,16);
    write('SELECTION TYPE:');
    gotoxy(24,18);
    write('FILE NAME :');
    gotoxy(24,19); write('(i.e., b:filename.ext)');
end;   (*  Display_M_Menu  *)

Procedure Help_Main;
var
    ch : char;
```

```
begin
   ClrScr;
   GoToXY(20,3); write('****    BULK MAILING SYSTEM HELP    ****');
   GoToXY(1,6);
   write('C - allows the creation of 1) new address file, ');
   write('2) new letter');
   GoToXY(5,7);
   write('file, 3) new address in an old file, 4) criteria file');
   GoToXY(1,9);
   write('D - allows the deletion of an address, address file');
   GoToXY(5,10);
   write('or letter file');
   GoToXY(1,12);
   write('M - allows the creation of an address label or a bulk ');
   GoToXY(5,13);
   write('mailing');
   GoToXY(1,15);
   write('P - allows the printing of the address file or letter');
   GoToXY(5,16);
   write('file on the screen or printer.');
   GoToXY(1,18);
   write('X - Exit from mailing system to main system.');
   GoToXY(8,21);
   write('press "R" to return you to mailing system menu -->');
   repeat
     GoToXY(58,21);
     read(Trm,Ch);
     if not (ch in ['r','R']) then
        begin
          GoToXY(10,24); write('PLEASE TRY AGAIN !! ');
          write(Bell); Delay(1000);
          GoToXY(58,21); ClrEol;
          GoToXY(10,24); ClrEol;
        end;    (*   else   *)
   until (Ch in ['r','R']);
   ClrScr;
end;    (*    Help_Main   *)


Procedure Display_C_Menu;
begin
   ClrScr;
   gotoxy(24,4);
   write('****    CREATE MENU    ****');
   gotoxy(22,7);
   write('C - CREATE CRITERION CATEGORIES');
   gotoxy(22,9);
   write('A - ADD NEW ADDRESS RECORD');
   gotoxy(22,11);
   write('L - ADD LETTER');
   gotoxy(22,13);
   write('H - HELP');
   gotoxy(22,15);
   write('R - RETURN TO MAIN MENU');
   gotoxy(22,18);
```

```
   write('SELECTION TYPE:');
   GoToXY(22,20); write('FILE NAME:');
   GoToXY(22,21); write('(i.e., b:filename)');
end;  (*  Display_C_Menu  *)

Procedure Help_Create;
var
   ch : char;
begin
   ClrScr;
   GoToXY(22,6); write('****     CREATE MENU HELP     ****');
   GoToXY(1,9);
   write('C - allows the creation of selection criteria codes ');
   write('that may');
   GoToXY(5,10);
   write('be used to categorize stored information for mailing ');
   GoToXY(5,11); write('process later.');
   GoToXY(1,13);
   write('A - allows the creation or updating of the ');
   write('address file');
   GoToXY(1,15);
   write('L - allows the creation of a letter file');
   GoToXY(1,17);
   write('FILENAME - The file you are going to add or create ');
   write('having the ');
   GoToXY(12,18);
   write('format b:filename where b disk and filename is 1-8 ');
   write('characters');
   GoToXY(8,22);
   write('press "R" to return you to "create menu" -->');
      repeat
       gotoxy(52,22);
       read(trm,ch);
       if not (ch in ['r','R']) then
       begin
          GoToXY(10,24); write('PLEASE TRY AGAIN !! ');
          write(Bell); Delay(1000);
          GoToXY(52,22); clreol;
          gotoxy(10,24); clreol;
       end;
      until (ch in ['r','R']);
   ClrScr;
end;  (*   Help_Create     *)

Procedure Display_P_Menu;
begin
   ClrScr;
   gotoxy(22,5);
   write('****     PRINT MENU     ****');
   gotoxy(23,8);
   write('D - DISPLAY ON SCREEN');
   gotoxy(23,10);
   write('P - PRINT');
   gotoxy(23,12);
   write('H - HELP');
```

```
    gotoxy(23,14);
    write('R - RETURN TO MAIN MENU');
    gotoxy(23,18);
    write('SELECTION TYPE:');
    GoToXY(23,20); write('FILE NAME:');
    GoToXY(23,21); write('(i.e., b:filename.ext)');
end;  (*  Display_P_Menu  *)

Procedure Display_D_Menu;
begin
    ClrScr;
    gotoxy(22,5);
    write('****     DELETE MENU     ****');
    gotoxy(23,8);
    write('A - DELETE ADDRESS');
    gotoxy(23,10);
    write('F - DELETE FILE');
    gotoxy(23,12);
    write('H - HELP');
    gotoxy(23,14);
    write('R - RETURN TO MAIN MENU');
    gotoxy(23,17);
    write('SELECTION TYPE:');
    GoToXY(23,19); write('FILE NAME:');
    GoToXY(23,20); write('(i.e., b:filename.ext)');
end;  (*  Display_D_Menu  *)

Procedure Help_Mail;
var
    ch : char;
begin
    ClrScr;
    GoToXY(22,6); write('****     MAIL MENU HELP     ****');
    GoToXY(3,9);
    write('A - allows the creation of bulk mailing');
    GoToXY(3,11);
    write('L - allows the creation of mailing labels');
    GoToXY(3,13); write('FILENAME - The name of the address file');
    GoToXY(14,14);
    write('b    s disk drive, filename is 1-8 chars and ext ');
    wr  e('is   ch rs');
    GoToXY(3,17);
    write('press "R" to return you to mail menu -->');
    repeat
      GoToXY(43,17);
      read(Trm,Ch);
      if not (ch in ['r','R']) then
      begin
           GoToXY(10,22); write('PLEASE TRY AGAIN !! ');
           write(Bell); Delay(1000);
           GoToXY(43,17); ClrEol;
           GoToXY(10,22); ClrEol;
      end;     (*   else   *)
    until (Ch in ['r','R']);
    ClrScr;
```

```
end;  (*   Help_Mail     *)

Procedure Help_Print;
var
   ch : char;
begin
   ClrScr;
   GoToXY(22,5); write('****     PRINT MENU HELP     ****');
   GoToXY(3,8);
   write('D - allows the display of the address file or letter');
   GoToXY(7,9);
   write('file on the screen.');
   GoToXY(3,11);
   write('P - allows the printing of the address file or letter');
   GoToXY(7,12);
   write('file from the printer');
   GoToXY(3,14);
   write('FILENAME - The file you are going to display');
   GoToXY(14,15);
   write('or print. b: is disk drive, filename is 1-8 chars ');
   write('and ext is 3.');
   GoToXY(3,18);
   write('press "R" to return you to print menu -->');
   repeat
     GoToXY(44,18);
     read(Trm,Ch);
     if not (ch in ['r','R']) then
     begin
          GoToXY(10,22); write('PLEASE TRY AGAIN !! ');
          write(Bell); Delay(1000);
          GoToXY(44,17); ClrEol;
          GoToXY(10,22); ClrEol;
     end;     (*   else   *)
   until (ch in ['r','R']);
   ClrScr;
end;  (*   Help_Print     *)

Procedure Help_Delete;
var
   ch : char;
begin
   ClrScr;
   GoToXY(17,6); write('****     DELETE MENU HELP     ****');
   GoToXY(3,9);
   write('A - allows the deletion of address records');
   GoToXY(3,11);
   write('F - allows the deletion of a file');
   GoToXY(3,13);
   write('FILENAME - The file you are going to delete');
   GoToXY(14,14);
   write('or the address file for deleting address record.');
   GoToXY(14,15);
   write('b: is disk drive, filename is 1-8 chars and ext ');
   write('is 3 chars');
   GoToXY(3,17);
```

```
    write('press "R" to return you to delete menu -->');
    repeat
      GoToXY(45,17);
      read(Trm,Ch);
      if not (ch in ['r','R']) then
      begin
           GoToXY(10,22); write('PLEASE TRY AGAIN !! ');
           write(Bell); Delay(1000);
           GoToXY(45,15); ClrEol;
           GoToXY(10,22); ClrEol;
      end;     (*   else   *)
  until (Ch in ['r','R']);
  ClrScr;
end;  (*   Help_Delete    *)

Procedure Help_Add_Address;
var
   ch : char;
begin
   ClrScr;
   GoToXY(18,3); write('****     ADD NEW ADDRESS HELP     ****');
   GoToXY(1,6);
   write('You may 1) Create or update the address record ');
   write('by entering "A"');
   GoToXY(12,7);
   write('then answer the request by keyin the name of the ');
   write('criterion file');
   GoToXY(9,9);
   write('2) When entering new data, follow the code listed ');
   write('for Crit 1 to');
   GoToXY(12,10);
   write('Crit 10 on the right side of the screen.  If there ');
   write('are no codes');
   GoToXY(12,11);
   write('listed, create selection criterion categories ');
   write('before you add new');
   GoToXY(12,12); write('address into the file.');
   GoToXY(9,14);
   write('3) Update the old records by answering the prompt ');
   write('"y" if the');
   GoToXY(12,15);
   write('address already exists, and type "n" if you want ');
   write('to restart');
   GoToXY(12,16); write('instead of updating that record.');
   GoToXY(9,18);
   write('4) Press ! PrtSc keys to get a listing from ');
   write('the printer.');
   GoToXY(3,22);
   write('press "R" to return you to add new record menu -->');
   repeat
     GoToXY(53,22);
     read(Trm,Ch);
     if not (ch in ['r','R']) then
     begin
          GoToXY(10,24); write('PLEASE TRY AGAIN !! ');
```

```
            write(Bell); Delay(1000);
            GoToXY(53,20); ClrEol;
            GoToXY(10,24); ClrEol;
        end;     (*   else   *)
    until (Ch in ['r','R']);
    ClrScr;
end;  (*   Help_Add_Address     *)

Procedure Display_Address_Menu (File_Id : anystring);
begin
    ClrScr;
    gotoxy(20,1);
    write('****     DISPLAY FILE     ****');
    GoToXY(55,2); write('CRIT  CODE  MEANING');
    GoToXY(6,3);  write('CRIT 1      :');
    GoToXY(46,3); write('CRIT     :');
    GoToXY(6,4);  write('CRIT 2      :');
    GoToXY(46,4); write('CRIT     :');
    GoToXY(6,5);  write('CRIT 3      :');
    GoToXY(46,5); write('CRIT     :');
    GoToXY(6,6);  write('CRIT 4      :');
    GoToXY(46,6); write('CRIT     :');
    GoToXY(6,7);  write('CRIT 5      :');
    GoToXY(46,7); write('CRIT     :');
    GoToXY(6,8);  write('CRIT 6      :');
    GoToXY(46,8); write('CRIT     :');
    GoToXY(6,9);  write('CRIT 7      :');
    GoToXY(46,9); write('CRIT     :');
    GoToXY(6,10); write('CRIT 8      :');
    GoToXY(46,10); write('CRIT     :');
    GoToXY(6,11);  write('CRIT 9     :');
    GoToXY(46,11); write('CRIT     :');
    GoToXY(6,12);  write('CRIT 10    :');
    GoToXY(46,12); write('CRIT     :');
    GoToXY(6,13);  write('ID_NO      :');
    GoToXY(46,13); write('CRIT     :');
    GoToXY(6,14);  write('NAME       :');
    GoToXY(46,14); write('CRIT     :');
    GoToXY(6,15);  write('ADDRESS1   :');
    GoToXY(46,15); write('CRIT     :');
    GoToXY(6,16);  write('ADDRESS2   :');
    GoToXY(46,16); write('CRIT     :');
    GoToXY(6,17);  write('CITYSTZIP :');
    GoToXY(46,17); write('CRIT     :');
    GoToXY(46,18); write('CRIT     :');
    GoToXY(46,19); write('CRIT     :');
    GoToXY(6,18);  write('COUNTRY    :');
    GoToXY(46,20); write('CRIT     :');
    GoToXY(6,19);  write('SALUTATION:');
    GoToXY(46,21); write('CRIT     :');
    GoToXY(6,20);  write('CLOSING    :');
    GoToXY(46,22); write('CRIT     :');
    GoToXY(1,24);  write('-->');
    GoToXY(1,25);  write('C'); LowVideo;
    write(' - Continue'); GoToXY(14,25); HighVideo;
```

```
    write('R'); LowVideo; write(' - Return main');
    HighVideo; GoToXY(30,25); write('H');
    LowVideo; write(' - Help');
    GoToXY(39,25); HighVideo; write('S');
    LowVideo; write(' - Search next');
    HighVideo; GoToXY(55,25); write('D');
    LowVideo; write(' - Display');
    HighVideo; GoToXY(67,25); write(File_Id);
    GoToXY(4,24);
end;    (*    Display_Address_Menu    *)

Procedure Display_Add_Address_Menu(File_Id : Anystring);
begin
    ClrScr;
    gotoxy(22,1);
    write('****     ADD NEW ADDRESS RECORD     ****');
    GoToXY(55,2);  write('CRIT  CODE  MEANING');
    GoToXY(6,3);   write('CRIT 1     :');
    GoToXY(46,3);  write('CRIT     :');
    GoToXY(6,4);   write('CRIT 2     :');
    GoToXY(46,4);  write('CRIT     :');
    GoToXY(6,5);   write('CRIT 3     :');
    GoToXY(46,5);  write('CRIT     :');
    GoToXY(6,6);   write('CRIT 4     :');
    GoToXY(46,6);  write('CRIT     :');
    GoToXY(6,7);   write('CRIT 5     :');
    GoToXY(46,7);  write('CRIT     :');
    GoToXY(6,8);   write('CRIT 6     :');
    GoToXY(46,8);  write('CRIT     :');
    GoToXY(6,9);   write('CRIT 7     :');
    GoToXY(46,9);  write('CRIT     :');
    GoToXY(6,10);  write('CRIT 8     :');
    GoToXY(46,10); write('CRIT     :');
    GoToXY(6,11);  write('CRIT 9     :');
    GoToXY(46,11); write('CRIT     :');
    GoToXY(6,12);  write('CRIT 10    :');
    GoToXY(46,12); write('CRIT     :');
    GoToXY(6,13);  write('ID_NO      :');
    GoToXY(46,13); write('CRIT     :');
    GoToXY(6,14);  write('NAME       :');
    GoToXY(46,14); write('CRIT     :');
    GoToXY(6,15);  write('ADDRESS1   :');
    GoToXY(46,15); write('CRIT     :');
    GoToXY(6,16);  write('ADDRESS2   :');
    GoToXY(46,16); write('CRIT     :');
    GoToXY(6,17);  write('CITYSTZIP :');
    GoToXY(46,17); write('CRIT     :');
    GoToXY(46,18); write('CRIT     :');
    GoToXY(46,19); write('CRIT     :');
    GoToXY(6,18);  write('COUNTRY    :');
    GoToXY(46,20); write('CRIT     :');
    GoToXY(6,19);  write('SALUTATION:');
    GoToXY(46,21); write('CRIT     :');
    GoToXY(6,20);  write('CLOSING    :');
    GoToXY(46,22); write('CRIT `   :');
```

```
    GoToXY(1,24);  write('-->');
    GoToXY(1,25);  write('R');
    LowVideo;      write(' - Return to main');
    HighVideo;  GoToXY(21,25);
    write('A'); LowVideo; write(' - Add');
    HighVideo;  GoToXY(30,25); write('S');
    LowVideo;   write(' - Store');
    HighVideo;  GoToXY(41,25); write('H');
    LowVideo;   write(' - Help');
    HighVideo;  GoToXY(51,25); write('C');
    LowVideo;   write(' - restart');
    HighVideo;  GoToXY(64,25); write(File_Id);
    GoToXY(4,24);
end;    (*    Display_Add_Address_Menu    *)

Procedure Help_Delete_Address;
var
    ch : char;
begin
    ClrScr;
    GoToXY(17,6); write('****    DELETE ADDRESS FILE HELP   ****');
    GoToXY(1,9);
    write('You may 1) Enter the "PgDn" key to page through ');
    write('each record and');
    GoToXY(12,10);
    write('then enter "Del" key to delete an address from an ');
    write('address file.');
    GoToXY(3,13);
    write('press "R" to returns you to delete address menu -->');
    repeat
      GoToXY(54,13);
      read(Trm,Ch);
      if not (ch in ['r','R']) then
      begin
           GoToXY(10,22); write('PLEASE TRY AGAIN !! ');
           write(Bell); Delay(1000);
           GoToXY(54,13); ClrEol;
           GoToXY(10,22); ClrEol;
      end;    (*   else   *)
  until (Ch in ['r','R']);
  ClrScr;
end;  (*   Help_Delete_Address_file     *)

Procedure Display_Delete_Address_Menu (File_Id : anystring);
begin
    ClrScr;
    gotoxy(20,1);
    write('****    DELETE ADDRESS FILE MENU    ****');
    GoToXY(6,3); write('CRIT 1     :');
    GoToXY(6,4); write('CRIT 2     :');
    GoToXY(6,5); write('CRIT 3     :');
    GoToXY(6,6); write('CRIT 4     :');
    GoToXY(6,7); write('CRIT 5     :');
    GoToXY(6,8); write('CRIT 6     :');
    GoToXY(6,9); write('CRIT 7     :');
```

```
    GoToXY(6,10); write('CRIT 8     :');
    GoToXY(6,11); write('CRIT 9     :');
    GoToXY(6,12); write('CRIT 10    :');
    GoToXY(6,13); write('ID_NO      :');
    GoToXY(6,14); write('NAME       :');
    GoToXY(6,15); write('ADDRESS1   :');
    GoToXY(6,16); write('ADDRESS2   :');
    GoToXY(6,17); write('CITYSTZIP  :');
    GoToXY(6,18); write('COUNTRY    :');
    GoToXY(6,19); write('SALUTATION:');
    GoToXY(6,20); write('CLOSING    :');
    GoToXY(1,24); write('-->');
    GoToXY(1,25); write('R'); LowVideo; write(' - Return main');
    GoToXY(18,25); HighVideo; write('Del');
    LowVideo; write(' - Delete'); HighVideo; GoToXY(34,25);
    write('PgDn'); LowVideo; write(' - Next record'); HighVideo;
    GoToXY(55,25); write('H'); LowVideo; write(' - Help');
    HighVideo; GoToXY(67,25); write(File_Id);
    GoToXY(4,24);
end;     (*     Display_Delete_Address_Menu     *)

Procedure Help_Display_Address_File;
var
    ch : char;
begin
    ClrScr;
    GoToXY(17,3); write('****     DISPLAY ADDRESS FILE HELP     ****');
    GoToXY(1,6); write('You may 1) Display the address file ');
    write('by entering "D".');
    GoToXY(9,8);
    write('2) Individually check an address by entering "S" ');
    write('and filling in');
    GoToXY(12,9); write('the identification number following ');
    write('"NEXT ID -->".');
    GoToXY(9,11);
    write('3) Print the information on the screen by pressing ');
    write('! PrtSc');
    GoToXY(12,12);
    write('at the same time.');
    GoToXY(9,13);
    write('4) Check all addresses by pressing successive "C"s.');
    GoToXY(3,15);
    write('press "R" to returns you to display address menu -->');
    repeat
      GoToXY(55,15);
      read(Trm,Ch);
      if  not (ch in ['r','R']) then
      begin
           GoToXY(10,22); write('PLEASE TRY AGAIN !! ');
           write(Bell); Delay(1000);
           GoToXY(55,15); ClrEol;
           GoToXY(10,22); ClrEol;
      end;     (*   else   *)
   until (Ch in ['r','R']);
   ClrScr;
```

```
end;  (*   Help_Display_Address_file      *)


Procedure Help_Display_Letter_File;
var
   ch : char;
begin
   ClrScr;
   GoToXY(17,6);
   write('****    DISPLAY LETTER FILE HELP    ****');
   GoToXY(1,9);
   write('You may 1) Check the contents of the letter but ');
   write('you will not');
   GoToXY(12,10);
   write('be able to change the letter.  You will have to go');
   GoToXY(12,11);
   write('to the "add letter file menu" next.');
   GoToXY(9,13);
   write('2) Print the letter on screen by pressing ¦ PrtScs.');
   GoToXY(3,16);
   write('press "R" to returns you to display letter file -->');
   repeat
     GoToXY(54,16);
     read(Trm,Ch);
     if not (ch in ['r','R']) then
     begin
          GoToXY(10,22); write('PLEASE TRY AGAIN !! ');
          write(Bell); Delay(1000);
          GoToXY(54,16); ClrEol;
          GoToXY(10,22); ClrEol;
     end;     (*   else   *)
  until (Ch in ['r','R']);
  ClrScr;
end;  (*   Help_Display_Letter_file      *)

Procedure Help_Mail_Label;
var Ch : char;
begin
   ClrScr;
   gotoxy(20,7);
   write('****    MAILING LABELS HELP     ****');
   gotoxy(5,12);
   write('You may produce a mailing labels by giving one or ');
   write('two criteria, i.e.,');
   GoToXY(1,13);
   write('(01=M,10=2) where "01" is criterion 1, M is code, ');
   write('"," means "and"');
   GoToXY(1,14);
   write('("/" means "or") and "10=2" is the second ');
   write('criterion; or enter ');
   write('"all" to');
   GoToXY(1,15);
   write('select all the records.  After the selection ');
   write('criteria are entered,');
   GoToXY(1,16);
```

```
    write('the printing program is invoked to print the ');
    write('selected records.');
    GoToXY(1,20);
    write('press "R" to return you to mailing labels menu -->');
    repeat
      GoToXY(51,20);
      read(Trm,Ch);
      if not (ch in ['r','R']) then
      begin
           GoToXY(10,23); write('PLEASE TRY AGAIN !! ');
           write(Bell);
           write(Bell); Delay(1000);
           GoToXY(51,20); ClrEol;
           GoToXY(10,23); ClrEol;
      end;     (*   else   *)
  until (Ch in ['r','R']);
  ClrScr;
end;  (*   Help_Mail_Label     *)

Procedure Display_Mail_Labels_Menu(File_Id : Anystring);
begin
    ClrScr;
    gotoxy(20,7);
    write('****     MAILING LABELS MENU     ****');
    gotoxy(23,14);
    write('SELECTION CRITERIA :');
    GoToXY(23,15);
    write('(i.e., 01=f,10>2)');
    GoToXY(1,22); write('-->');
    GoToXY(5,23);
    write('R'); LowVideo; write(' - Return to main menu');
    HighVideo; write('     H'); LowVideo; write(' - Help');
    HighVideo; write('     L'); LowVideo; write(' - Label');
    HighVideo; write('     ',File_Id);
    GoToXY(4,22);
end;  (*   Display_Mail_Labels_Menu     *)

Procedure Display_Mail_letter_Menu(File_Id : Anystring);
begin
    ClrScr;
    gotoxy(20,5);
    write('****     MAILING LETTERS MENU     ****');
    gotoxy(23,8);
    write('ENTER DATE:');
    gotoxy(23,9);
    write('(e.g., January 1, 1984)');
    gotoxy(23,11);  write('NAME OF THE LETTER FILE:');
    gotoxy(23,12);  write('(e.g., b:filename.ltr)');
    gotoxy(23,14);
    write('PERSONALIZE LETTER (N/Y)?');
    gotoxy(23,16);
    write('SELECTION CRITERIA :');
    gotoxy(1,22); write('-->');
    gotoxy(5,23);
    write('R'); LowVideo; write(' - Return to main menu');
```

```
   HighVideo; write('     H - Help');
   LowVideo; write('     L - Letter');
   HighVideo; write('     ',File_Id)
end;  (*   Display_Mail_Letter_Menu     *)

Procedure Help_Mailing_Letter;
var
   ch : char;
begin
   ClrScr;
   GoToXY(22,1); write('****     MAILING LETTER HELP     ****');
   GoToXY(1,3);
   write('You may create a bulk mailing by following ');
   write('these steps : ');
   GoToXY(1,5); write('1)  Enter the date of the letter');
   GoToXY(1,6); write('2)  Enter the name of the letter file');
   GoToXY(1,7);
   write('3)  You may personalize the letter if you want, ');
   write('type "y" to change');
   GoToXY(5,8);
   write('the letter, or type "n" to continue'); GoToXY(1,9);
   write('4)  Enter one or two selection criteria, i.e., ');
   write('01=F/02=4 where "/"');
   GoToXY(5,10);  write('means "or" and "," means "and"');
   write('or enter "all" to select all records.');
   GoToXY(1,11);
   write('5)  If no records are selected then the system ');
   write('display the messages');
   GoToXY(5,12);
   write('said "no records match the condition".  Or if ');
   write('there are some');
   GoToXY(5,13);
   write('records match the criteria, then the system will ');
   write('check that if you');
   GoToXY(5,14);
   write('want to change the letter contents.  If answer is ');
   write('"y", the screen');
   GoToXY(5,15);
   write('brought you a letter which has the data fields
   write('inserted to');
   GoToXY(5,16);
   write('appropriated places already.  Otherwise, the system ');
   write('will ask you');
   GoToXY(5,17);
   write('to strike "p" to start printing the bulk mailing ');
   write('letter and display');
   GoToXY(5,18);
   write('messages when the printing is finished.');
   GoToXY(1,19);
   write('6)  If you want to personalize the letter, ');
   write('you may change the');
   GoToXY(5,20);
   write('contents of the letter interactively, then press ');
   write('! PrtSc keys to');
   GoToXY(5,21);  write('print the letter.');
```

```
    GoToXY(1,22);
    write('7)  Enter "C" to continue processing the next ');
    write('selected record or');
    GoToXY(5,23);
    write('press "N" to skip to next selected record or ');
    write('enter "R" to returns');
    GoToXY(5,24); write('you to main menu');
    GoToXY(1,25);
    write('Press "R" to returns you to mailing letter menu -->');
    repeat
      GoToXY(52,25);
      read(Trm,Ch);
      if not (ch in ['r','R']) then
      begin
           write(Bell);   Delay(1000);
           GoToXY(52,25); ClrEol;
      end;     (*   else   *)
  until (Ch in ['r','R']);
  ClrScr;
end;  (*   Help_Mailing_Letter     *)

Procedure Display_Create_Crit_Menu(File_Id : Anystring);
begin
    ClrScr;
    gotoxy(14,1);
    write('****     CREATE CRITERION CATEGORIES     ****');
    GoToXY(26,3); write('CRIT  CODE  MEANING');
    GoToXY(13,4); write('CRIT     :');
    GoToXY(13,5); write('CRIT     :');
    GoToXY(13,6); write('CRIT     :');
    GoToXY(13,7); write('CRIT     :');
    GoToXY(13,8); write('CRIT     :');
    GoToXY(13,9); write('CRIT     :');
    GoToXY(13,10); write('CRIT     :');
    GoToXY(13,11); write('CRIT     :');
    GoToXY(13,12); write('CRIT     :');
    GoToXY(13,13); write('CRIT     :');
    GoToXY(13,14); write('CRIT     :');
    GoToXY(13,15); write('CRIT     :');
    GoToXY(13,16); write('CRIT     :');
    GoToXY(13,17); write('CRIT     :');
    GoToXY(13,18); write('CRIT     :');
    GoToXY(13,19); write('CRIT     :');
    GoToXY(13,20); write('CRIT     :');
    GoToXY(13,21); write('CRIT     :');
    GoToXY(13,22); write('CRIT     :');
    GoToXY(13,23); write('CRIT     :');
    GoToXY(1,24);  write('-->');
    GoToXY(1,25);
    write('R'); LowVideo; write(' - Return to main');
    HighVideo; GoToXY(21,25); write('A'); LowVideo;
    write(' - Add contents');
    HighVideo; GoToXY(41,25); write('H'); LowVideo;
    write(' - Help'); HighVideo;
    GoToXY(52,25); write('00'); LowVideo; write(' - Store');
```

```
   HighVideo; GoToXY(64,25); write(File_Id);
   GoToXY(4,24);
end;    (*    Display_Create_Crit_Menu    *)

Procedure Help_Create_Criteria;
var
   ch : char;
begin
   ClrScr;
   GoToXY(16,1);
   write('****    CREATE CRITERION CATEGORIES HELP   ****');
   GoToXY(3,3);
   write('The number of codes of criterion may not be ');
   write('exceed 20.  Enter codes');
   GoToXY(1,4);
   write('as illustrated below.  Crit allows 1-4 chars,  ');
   write('codes allows 1 and');
   GoToXY(1,5);
   write('meaning allows 1-7 chars. Type "00" when ');
   write('you are ready to store file.');
   GoToXY(18,7); write('CRIT        CODES          MEANING');
   GoToXY(8,8);
   write('CRIT  1   sex          f              female');
   GoToXY(8,9);
   write('CRIT  1                m             male');
   GoToXY(8,10);
   write('CRIT  2   dept         1              000-100');
   GoToXY(8,11);
   write('CRIT  2                2              101-200');
   GoToXY(8,12);
   write('CRIT  2                3              201-300');
   GoToXY(3,13);
   write('To change to a new criterion, type over the criterion');
   GoToXY(1,14);
   write('number with the next number in sequence.');
   GoToXY(18,16); write('CRIT        CODES          MEANING');
   GoToXY(8,17);
   write('CRIT  1   sex          f             female');
   GoToXY(8,18);
   write('CRIT  1                m              male');
   GoToXY(8,19);
   write('CRIT  2   dept         1              000-100');
   GoToXY(8,20);
   write('CRIT  2                2              101-200');
   GoToXY(8,21);
   write('CRIT  2                3              201-300');
   GoToXY(8,22);
   write('CRIT  3   occp         f              faculty');
   GoToXY(8,23);
   write('CRIT  3                s              student');
   GoToXY(8,24);
   write('CRIT  4                                      ');
   GoToXY(7,25);
   write('Press "R" to returns you to create criterion menu -->');
  repeat
```

```
      GoToXY(60,25);
      read(Trm,Ch);
      if not (ch in ['r','R']) then
      begin
           write(Bell); Delay(1000);
           GoToXY(60,25); ClrEol;
      end;     (*   else  .*)
  until (Ch in ['r','R']);
  ClrScr;
end;  (*   Help_Create_Criteria     *)


(*****************************************************************)
(*   This procedure determines whether the file entered by a     *)
(*   user exists on the disk or not.                             *)
(*****************************************************************)


Function Exist(FileN: AnyString) : boolean;
var F : file;
begin
   {$I-}
   assign(F,FileN);
   reset(F);
   {$I+}
   if IOResult <> 0 then Exist := false
   else Exist := true;
end;

Procedure Get_File_Name(var Line:Anystring);
 var Ch_In : char;
begin
   Line := ' ';
   read(con,Line);
end;    (*     Get_File_Name     *)

{$I crit.pas}

(*****************************************************************)
(*  This procedure  creates the  criterion file  entered by a  *)
(*  user.  Also, it  checks the  range of  criterion  numbers  *)
(*  before the criterion record is written to the disk file.   *)
(*****************************************************************)


Procedure Create_Crit_File(File_Id : Anystring; var F : CFiles);
var
   Row, I, J : integer;
   Pre_No : string[2];
   Select : set of char;
begin
   Row := 3; Pre_No := '1'; GoToXY(18,Row+1); write(Pre_No);
   for j := 1 to 20 do
   begin
      Crit[J].No := '  ';  Crit[J].Code := ' ';
```

```
   Crit[J].Id := '    '; Crit[J].Meaning := '          ';
end;
I := 1;
Crit[I].No := '1'; assign(F,File_Id);
if exist(File_Id) then
   begin
      GoToXY(10,24);
      write('erase previous criteria (N/Y) ?');
      GoToXY(41,24); read(Trm,Ch);
      Select := ['n','N','y','Y'];
      if not (Ch in Select) then
           repeat
             write(Bell); GoToXY(10,24); ClrEol;
             GoToXY(41,24); read(Trm,Ch);
           until Ch in Select;
      GoToXY(10,24); ClrEol
   end;    (*    if    *)
if (not exist(File_Id)) or (Ch in ['y','Y']) then
begin
gotoxy(15,24);
rewrite(F);
   repeat
      if I = 1 then
      begin
        GoToXY(26,Row+1);
        read(Con,Crit[I].Id);
      end
      else
      begin
        if Pre_No = Crit[I].No then Crit[I].Id := ' '
        else begin
               GoToXY(26,Row+I);
               read(Con,Crit[I].Id);
             end    (*    else    *)
      end;    (*    else    *)
      GoToXY(33,Row+I); read(Con,Crit[I].Code);
      GoToXY(38,Row+I); read(Con,Crit[I].Meaning);
      write(F,Crit[I]);
      Pre_No := Crit[I].No;
      I := I + 1; Crit[I].No := '  ';
      GoToXY(18,Row+I); read(Con,Crit[I].No);
      while not (((ord(Crit[I].No[1]) in [49..57]) and
                  (ord(Crit[I].No[2]) = 32)) or
                 ((ord(Crit[I].No[1])=49) and
                   (ord(Crit[I].No[2])=48)) or
                 (Crit[I].No = '00')) do
         begin
           GoToXY(15,24);
           write('Criteria no. is not in range 1..10');
           write(Bell); Delay(1500); GoToXY(18,Row+I);
           write('   '); GoToXY(15,24);
           ClrEol; GoToXY(18,Row+I);
           Crit[I].No := '  '; read(Con,Crit[I].No)
         end;
   until (I ) 20) or (Crit[I].No = '00');
```

```
      close(F);
   if I > 20 then
      begin
         GoToXY(4,24); ClrEol; GoToXY(15,24);
         write('Sorry, only 20 criteria allowed');
         write('and file is stored');
         write(Bell); Delay(2000); GoToXY(15,24); ClrEol;
      end    (*   if    *)
   else begin
          GoToXY(15,24); write('criterion file stored on disk');
          Delay(2000); GoToXY(15,24); ClrEol
        end    (*    else    *)
   end     (*    if    *)
end;   (*    Create_Crit_File    *)


(*****************************************************************)
(*  This  procedure allows a user to create a  criterion file,  *)
(*  ask for help or return to the main menu.                     *)
(*****************************************************************)


Procedure Create_Criteria(var F : CFiles);
 var
   Ch : char;
   Display_Crit : boolean;
   Type_Ok : boolean;
   Sel_Type : set of char;
begin
   Display_Crit := true;
   ClrScr;
   Sel_Type := ['R','S','H','0','A'];
   while Display_Crit do
   begin
     Display_Create_Crit_Menu(Filename);
     GoToXY(4,24); read(Trm,ch);
     Ch := upcase(Ch);
     if Ch in Sel_Type then Type_Ok := true
     else begin
            repeat
              write(Bell); GoToXY(4,24); ClrEol;
              GoToXY(4,24); read(Trm,Ch);
              Ch := upcase(Ch);
            until Ch in Sel_Type
          end;    (*    else    *)
     if Type_Ok then
        case Ch of
            'H' : Help_Create_Criteria;
            'R' : Display_Crit := false;
            'A' : Create_Crit_File(Filename,F);
            '0' : repeat
                    write(Bell); GoToXY(4,24); ClrEol;
                    GoToXY(4,24); read(Trm,Ch); Ch := upcase(Ch);
                  until Ch <> 'S'
      end     (*    case    *)
```

```
   end;    (*     while     *)
  ClrScr
end;     (*     Create_Criteria     *)


{$I adds.pas}

(*****************************************************************)
(*     This procedure reads a record from the criterion file     *)
(*     and displays it on the screen.                            *)
(*****************************************************************)


Procedure Display_Crits(File_Id : Anystring);
var
   I, Row : integer;
   F : CFiles;
begin
   if File_Id <> '' then
   begin
      assign(F,File_Id);
      reset(F); I := 1; Row :=2;
      while not EOF(F) do
        begin
           read(F,Crit[I]);
           GoToXY(51,I+Row);     write(Crit[I].No);
           GoToXY(55,I+Row);     write(Crit[I].Id);
           GoToXY(62,I+Row);     write(Crit[I].Code);
           GoToXY(67,I+Row);     write(Crit[I].Meaning);
           I := succ(I)
       end;     (*   while     *)
   end;     (*     if     *)
   GoToXY(4,24); write(' ' : 15);
end;        (*   Display_Crits     *)


(*****************************************************************)
(*     This procedure requests a user to enter the name of the   *)
(*     message if the file does not exist; otherwise the         *)
(*     criterion file will be displayed on the screen.           *)
(*****************************************************************)


Procedure Read_Crit_File(var File_Id : Anystring);
begin
   GoToXY(15,24);  ClrEol;   GoToXY(25,24);
   write('PLEASE ENTER NAME OF CRITERIA FILE, ');
   write('i.e. b: filename.cri');
   GoToXY(4,24); write(' '); GoToXY(4,24);
   Get_File_Name(File_Id);
   if File_Id <> '' then if ( not Exist(File_Id)) then
   repeat
      GoToXY(10,24); ClrEol;
      GoToXY(25,24); write('FILE NOT FOUND !!');
      write(Bell); Delay(1000); GoToXY(4,24); ClrEol;
```

```
      GoToXY(25,24);
      write('PLEASE ENTER NAME OF CRITERIA FILE, ');
      write('i.e. b: Filename.cri');
      GoToXY(4,24);
      Get_File_Name(File_Id);
   until exist(File_Id) or (File_Id = '');
   Display_Crits(File_Id)
end;     (*     read_Crit_File     *)


(*****************************************************************)
(*   This procedure uses the direct access function "seek" to    *)
(*   point to the desired position of the file and to store      *)
(*   the record into that file.                                  *)
(*****************************************************************)


Procedure Store_Adds_Rec(Index : integer; var F : AFiles);
begin
   reset(F);
   seek(F,Index);
   write(F,Tempadds);
   close(F);  GoToXY(15,24);  ClrEol;
   GoToXY(25,24); write('ADDRESS RECORD STORED ON DISK');
   Delay(2000)
end;       (*     Store_Adds_Rec     *)


(*****************************************************************)
(*   This procedure converts a string into an integer number     *)
(*   then does the hash function.                                *)
(*****************************************************************)


Procedure Do_Hashing(var F : AFiles; var I : integer;
                     var Found, Free : boolean);
var
   Key : integer;
begin
     Key := abs((ord(TempId_No[4])*10000)
            +(ord(TempId_No[5])*1000)+
            (ord(TempId_No[6])*100)+(ord(TempId_No[7])*10)+
            (ord(TempId_No[8])*10)+ord(TempId_No[9]));
   I := (Key mod Hash_Prime);
   repeat
      reset(F);
      seek(F,I);
      read(F,InAdds);
      if InAdds.Id_No = TempId_No then Found := true
      else If InAdds.Id_No = '          ' then Free := true
      else begin
             I := succ(I);
             if I = Max_Adds then I := 0;
           end;
   until (Found) or (Free);
```

```
   close(F)
end;      (*     Do_Hashing     *)


(*****************************************************************)
(*   This procedure  reads the contents of an address record     *)
(*   from the users input and either stores the record or re-    *)
(*   starts the record depending on the users desision.          *)
(*****************************************************************)


Procedure Add_Address_Contents(Index : integer; var F : AFiles);
var
  I,J : integer;
  Sel_Type : set of char;
  Ch : char;
  Done : boolean;
begin
   I := 1;
   Sel_Type := ['Y','N'];
   GoToXY(18,13); write(Con,TempId_No);
   with Tempadds do
   begin
     GoToXY(18,3);  read(Crit_Code[I]);
     if not (Crit_Code[I] in [' '..'z']) then
         Crit_Code[I] := ' ';
     GoToXY(18,4);  read(Crit_Code[I+1]);
     if not (Crit_Code[I+1] in [' '..'z']) then
         Crit_Code[I+1] := ' ';
     GoToXY(18,5);  read(Crit_Code[I+2]);
     if not (Crit_Code[I+2] in [' '..'z']) then
         Crit_Code[I+2] := ' ';
     GoToXY(18,6);  read(Crit_Code[I+3]);
     if not (Crit_Code[I+3] in [' '..'z']) then
         Crit_Code[I+3] := ' ';
     GoToXY(18,7);  read(Crit_Code[I+4]);
     if not (Crit_Code[I+4] in [' '..'z']) then
         Crit_Code[I+4] := ' ';
     GoToXY(18,8);  read(Crit_Code[I+5]);
     if not (Crit_Code[I+5] in [' '..'z']) then
         Crit_Code[I+5] := ' ';
     GoToXY(18,9);  read(Crit_Code[I+6]);
     if not (Crit_Code[I+6] in [' '..'z']) then
         Crit_Code[I+6] := ' ';
     GoToXY(18,10);  read(Crit_Code[I+7]);
     if not (Crit_Code[I+7] in [' '..'z']) then
         Crit_Code[I+7] := ' ';
     GoToXY(18,11);  read(Crit_Code[I+8]);
     if not (Crit_Code[I+8] in [' '..'z']) then
         Crit_Code[I+8] := ' ';
     GoToXY(18,12);  read(Crit_Code[I+9]);
     if not (Crit_Code[I+9] in [' '..'z']) then
         Crit_Code[I+9] := ' ';
     GoToXY(18,14);  read(Name);
     GoToXY(18,15);  read(Addrs1);
```

```
      GoToXY(18,16);  read(Addrs2);
      GoToXY(18,17);  read(CityStZip);
      GoToXY(18,18);  read(Country);
      GoToXY(18,19);  read(Salutation);
      GoToXY(18,20);  read(Closing);
      Id_No := TempId_No;
      GoToXY(25,24);  write('Please enter C or S'); ClrEol;
      GoToXY(4,24);   write('          ');
      GoToXY(4,24);   read(Trm,Ch); ClrEol
    end;  (*    with    *)
    Done := false;
    repeat
      case Ch of
        's','S' : begin
                    Store_Adds_Rec(Index,F);
                    Done := true
                  end;
        'c','C' : begin
                    GoToXY(15,24);
                    write('Do you want start again (N/Y) ?');
                    GoToXY(46,24); read(Trm,Ch);
                    Ch := upcase(Ch);
                    if not (Ch in Sel_Type) then
                        repeat
                          write(Bell); GoToXY(25,24); ClrEol;
                          GoToXY(46,24); read(Trm,Ch);
                          Ch := upcase(Ch);
                        until (Ch = 'Y') or (Ch = 'N');
                    if Ch = 'Y' then Done := true  (*  restart  *)
                  end;  (*    'C'    *)
        else repeat
                write(Bell);
                GoToXY(25,24); write('Please enter C or S ');
                GoToXY(4,24); write(' ');
                GoToXY(4,24); read(trm,Ch);
                Ch := upcase(Ch);
             until (Ch = 'C') or (Ch = 'S');

      end;    (*    case    *)
     GoToXY(15,24); ClrEol;
  until done
end;    (*    Add_Address_Contents    *)


(*****************************************************************)
(*   This procedure initializes the address file to blanks.      *)
(*****************************************************************)


Procedure Initialize_File(var F : AFiles);
var I : integer;
begin
   rewrite(F);
   with TempAdds do
   begin
```

```
      for I := 1 to 10 do
          Crit_Code[I] := ' ';
      Id_No := '          ';
      Name := '                              ';
      Addrs1 := '                              ';
      Addrs2 := '                              ';
      CityStZip := '                                 ';
      Country := '                    ';
      Salutation := '                          ';
      Closing := '                         ';
   end;     (*    with    *)
   for I := 0 to Max_Adds do
        write(F,TempAdds);
   close(F)
end; (*   Initialize_File   *)


(*****************************************************************)
(*   This procedure allows a user to display the criterion file*)
(*   on the screen; then according to the id number entered     *)
(*   by the user to determing whether the record exists or not.*)
(*   If the record is a new record the procedure Add_Address_   *)
(*   Contents will be invoked to store the new record.          *)
(*****************************************************************)


Procedure Create_Address_File(var F : AFiles;var Times : integer);
var
   Select : set of char;
   Free, Add, Found : boolean;
begin
   Add := false;
   Free := false;
   Found := false;
   if Times = 1 then Read_Crit_File(Filen)
   else
   begin
      GoToXY(20,24);
      write('Do you want to use the old criterion file (N/Y) ?');
      GoToXY(69,24); read(Trm,Ch);
      if not (Ch in ['n','N','y','Y']) then
      repeat
         GoToXY(69,24); ClrEol;
         GoToXY(69,24); read(Trm,Ch);
      until Ch in ['n','N','y','N'];
      if Ch in ['n','N'] then Read_Crit_File(Filen)
      else Display_Crits(Filen)
   end;     (*    if     *)
   if (Filen = '') then
   begin
      Times := pred(Times);
      GoToXY(15,24); ClrEol;
      GoToXY(25,24);
      write('PLEASE ADD CRITERIA FILE UNDER CREATE MENU FIRST !!');
      ClrEol;
```

```
      delay(2000);
    end
    else begin
          GoToXY(15,24); ClrEol;
          GoToXY(25,24); write('Please enter Id_No - 9 digits');
          ClrEol; delay(1200);
          GoToXY(4,24); write(' ');
          GoToXY(4,24); read(Con,TempId_No);
          GoToXY(25,24); write('checking'); ClrEol;
          assign(F,Filename);
          if (exist(Filename)) then
          begin
             Do_Hashing(F,Index,Found,Free);
             if Found then
             begin
               GoToXY(25,24);
               write('record already exists, overwrite (N/Y) ?');
               GoToXY(65,24); read(Trm,Ch);
               Select := ['n','N','y','Y'];
               if not (Ch in Select) then
                  repeat
                     write(Bell);
                     GoToXY(65,24); ClrEol;
                     GoToXY(65,24); read(Trm,Ch);
                  until Ch in Select;
               if (Ch = 'N') or (Ch = 'n') then
                  Add := false
               else Add := true;
            end    (*   if Found   *)
            else
             begin
               GoToXY(4,24); ClrEol; GoToXY(25,24);
               write('Please enter new record');
               Add := true
             end;    (*    else not found    *)
          end   (*    if file exists    *)
      else
          begin
           GoToXY(25,24); write('waiting  ');
           Initialize_File(F);
           Add := true;
           GoToXY(4,24); write(' ' : 10);
           GoToXY(25,24); write('Ok to enter a new record');
           ClrEol;
           Do_Hashing(F,Index,Found,Free)
          end;   (*    not exist    *)
      if Add then Add_Address_Contents(Index,F)
   end    (*   else   *)
end;    (*    Creat_Address_File    *)
```

```
(*****************************************************************)
(*    This procedure allows a user to ask for Help, return to    *)
(*    Main menu or add an address record to the address file.    *)
(*****************************************************************)

Procedure Add_Address(var F : AFiles);
 var
   Ch : char;
   Display_Adds : boolean;
   Sel_Type : set of char;
   Times : integer;
begin
   Times := 0;
   Display_Adds := true;
   ClrScr;
   Sel_Type := ['C','R','H','S','A'];
   while Display_Adds do
   begin
     Display_Add_Address_Menu(Filename);
     GoToXY(4,24); read(Trm,ch);
     Ch := upcase(Ch);
     if not (Ch in Sel_Type) then
     begin
         write(Bell);
         GoToXY(4,24); ClrEol;
     end      (*     if     *)
     else
        case Ch of
            'H' : Help_Add_Address;
            'R' : Display_Adds := false;
            'A' : begin
                    Times := succ(Times);
                    Create_Address_File(F,Times)
                  end;
            'S' : begin
                        write(Bell);
                        GoToXY(15,24);
                        write('add contents first');
                        Delay(2000);
                        GoToXY(4,24); ClrEol
                  end;    (*     'S'     *)
          'C' :   begin
                      write(Bell);
                      GoToXY(4,24); ClrEol;
                  end
   end     (*     case     *)
  end;
  ClrScr;
end;     (*     Add_Address     *)
```

```
(*****************************************************************)
(*    This procedure allows a user to delete an existing file    *)
(*    from the disk.                                             *)
(*****************************************************************)


Procedure Delete_File(File_Id : Anystring);
var
  Ch : char;
  Select : set of char;
  F : File;
begin
   GoToXY(20,25);
   write('Are you sure you want to delete this file (N/Y) ? ');
   GoToXY(70,25); read(Trm,Ch);  Ch := Upcase(Ch);
   Select := ['Y','N'];
   If not (Ch in Select) then
   repeat
      GoToXY(70,25); ClrEol;
      GoToXY(70,25); read (Trm,Ch);  Ch := Upcase(Ch);
   until Ch in Select;
   if Ch = 'Y' then
   begin
      assign(F,File_Id);
      close(F);
      erase(F);
      GoToXY(20,25); write('FILE IS DELETED'); ClrEol;
      Delay(2000)
   end
end;     (*    Delete_File    *)


{$I addlett.pas}

(*****************************************************************)
(*    This procedure displays the message (parameter S) on the   *)
(*    screen.                                                    *)
(*****************************************************************)


Procedure Flash(x,y: integer; S : Anystring);
begin
    GoToXY(x,y);
    write(S);
end;  (*    Flash    *)


(*****************************************************************)
(*    This procedure initializes the form  letter to blanks.     *)
(*****************************************************************)


Procedure Init;
```

```
var
   I, J : integer;
begin
   For I := 0 to FyMax do
       For J := 0 to FxMax do
           Letter[J,I].Ch := ' ';
   Flash(28,23,Filename); Flash(15,23,'New Letter - ');
   Flash(39,23,' Type \ for commands');
   Fx := 1; Fy := 1;  GoToXY(Fx,Fy)
end;     (*     Init     *)


(*************************************************************)
(*     This procedure  converts the input character into  an     *)
(*     IBM  character.                                           *)
(*************************************************************)


Procedure IBMch(var Ch : char);
begin
   case Ch of
        'H' : Ch := ^E;
        'P' : Ch := ^X;
        'M' : Ch := ^D;
        'K' : Ch := ^S;
        'S' : Ch := #127;
        'G',
        'I',
        'O',
        'Q' : Ch := #00;
    end    (*     case     *)
 end;      (*     IBMch     *)


(*************************************************************)
(*     This procedure allows a user to move the cursor up        *)
(*     one character.                                            *)
(*************************************************************)


Procedure Move_Up;
begin
   Fy := Fy - 1;
   if Fy < 1 then Fy := FyMax;
   GoToXY(Fx,Fy)
end;     (*     Move_Up     *)


(*************************************************************)
(*     This procedure allows a user to move the cursor ritht     *)
(*     one character.                                            *)
(*************************************************************)


Procedure Move_Right;
```

```
begin
   Fx := Succ(Fx);
   if Fx > FxMax then
   begin
      Fx := 1;
      Fy := Fy + 1;
      If Fy > FyMax then Fy := 1
   end;    (*    if    *)
   GoToXY(Fx,Fy)
end;     (*     Move_Right     *)


(*****************************************************************)
(*     This procedure allows a user to move the cursor left      *)
(*     one character.                                            *)
(*****************************************************************)


Procedure Move_Left;
begin
   Fx := Pred(Fx);
   if Fx < 1 then
   begin
      Fx := FxMax;
      Fy := Fy - 1;
      if Fy < 1 then Fy := FyMax
   end;     (*     if     *)
   GoToXY(Fx,Fy)
end;     (*     Move_Left     *)


(*****************************************************************)
(*     This procedure displays the form letter on the screen.    *)
(*****************************************************************)


Procedure Update;
var
   UFx,UFy : integer;
begin
   ClrScr;
   Flash(15,23,Filename);
   Flash(33,23,' Type \ for commands');
   for UFy := 0 to FyMax do
       For UFx := 0 to FxMax
           do if Letter[UFx,UFy].Ch <>  '' then
              begin
                 GoToXY(UFx+1,UFy+1);
                 write(Letter[UFx,UFy].Ch)
              end;
   GoToXY(Fx,Fy)
end;     (*     Update     *)
```

```
(*****************************************************************)
(*    This procedure allows a user to move the cursor down one   *)
(*    character.                                                 *)
(*****************************************************************)


Procedure Move_Down;
begin
     Fy := Fy + 1;
     if Fy > FyMax then Fy := 1;
     GoToXY(Fx,Fy)
end;     (*     Move_Down     *)


(*****************************************************************)
(*     This procedure allows a user to move the cursor to the    *)
(*     beginning of the next line.                               *)
(*****************************************************************)


Procedure Next_Line;
begin
    Fy := Fy + 1;
    if Fy > FyMax then Fy := 1;
    Fx := 1;
    GoToXY(Fx,Fy)
end;     (*     Next_Line     *)


(*****************************************************************)
(*    This procedure reads the input entered by a user and echo  *)
(*    prints it.  Also, the procedure moves the cursor to the    *)
(*    right.                                                     *)
(*****************************************************************)


Procedure Get_Next(Fx,Fy : integer);
var  I, J, K, L : integer;
begin
            Letter[Fx-1,Fy-1].Ch := Ch;
            GoToXY(Fx,Fy);
            write(Ch);
            Move_Right
end;     (*    Get_Next     *)


(*****************************************************************)
(*     This procedure displays the help menu on the screen.      *)
(*****************************************************************)


Procedure Help_Add_Letter;
var
```

```
   Ch : Char;
   Line : string[80];
   H : text;
   J : integer;
begin
   if exist('Addlett.hlp') then
   begin
      assign(H,'Addlett.hlp');
      reset(H);
      while not EOF(H) do
      begin
         ClrScr;
         repeat
            readln(H,Line);
            if (Copy(line,1,3)) <> '.PA' then writeln(Line);
         until EOF(H) or (Copy(line,1,3) = '.PA');
         if (Copy(Line,1,3) = '.PA') then
         begin
            GoToXY(37,24); read(kbd,Ch)
         end
      end;
      GoToXY(52,18);
      repeat
         read(Kbd,Ch);
      until Upcase(Ch) = 'R';
      update;
   end
   else begin
          GoToXY(10,25);
          write('Help file is not found');
          Update
        end    (*    else    *)
 end;    (*    Help_Add_Letter    *)


Procedure Load;
begin
   ClrScr;
   assign(Lettfile,Filename);
   reset(Lettfile);
   for Fx := 0 to FxMax do
       for Fy := 0 to FyMax do
            read(Lettfile,Letter[Fx,Fy]);
   Fx := 1; Fy := 1;
   Update
end;    (*    Load    *)


(*************************************************************)
(*   This procedure saves the form letter file onto the disk.  *)
(*************************************************************)


Procedure Save;
var
```

```
   I,J : integer;
begin
   if exist(Filename) then
   begin
      GoToXY(10,25);
      write('Letter file already exists, overwrite (N/Y) ?');
      GoToXY(55,25);
      repeat
        read(Kbd,Ch);
      until Upcase(Ch) in ['N','Y']
   end;
   if not (exist(Filename)) or (upcase(Ch) in ['Y']) then
   begin
      rewrite(Lettfile);
      for I := 0 to FxMax do
          for J := 0 to FyMax do
              write(Lettfile,Letter[I,J]);
      GoToXY(10,25);
      write('file stored on disk');
      Store := true;
      ClrEol;
      close(Lettfile);  GoToXY(10,24);    ClrEol;
      GoToXY(Fx,Fy)
   end
   else begin
          GoToXY(10,25);
          ClrEol
        end
end;     (*     Save     *)


Procedure Commands(var Create_Letter : boolean);
begin
   GoToXY(9,24);
   write('\  restore  Quit,  S - Save,  P - Print,  H  - Help,');
   write('R - Return,');
   Read(Kbd,Ch);
   Ch := Upcase(Ch);
   case Ch of
        'R' : Create_Letter := False;
        'S' : Save;
        'H' : Help_Add_Letter;
        'P' : begin
              Flash(10,25,
              'Please enter shift and PrtSc key to get a copy');
                Delay(2000); GoToXY(10,25); ClrEol
              end;
        '\' : Update;
     end     (*   case     *)
end;     (*     Command     *)
```

```
(*****************************************************************)
(*    This procedure allows a user  to create a form letter,     *)
(*    save the letter file, display the help menu or print the   *)
(*    form letter on the line printer.                           *)
(*****************************************************************)


Procedure Add_Letter;
var Create_Letter : boolean;
begin
   ClrScr;
   Store := false;
   if exist(Filename) then Load
   else init;
   Create_Letter := true;
   repeat
      read(Kbd,Ch);
      if Keypressed then
      begin
         read(Kbd,Ch);
         IBMch(Ch)
      end;
      case Ch of
          ^E : Move_Up;
          ^X,^J : Move_Down;
          ^D,^F : Move_Right;
          ^S,^A : Move_Left;
          '\' : Commands(Create_Letter);
          ^M  : Next_Line;
          else
             if Ch in [' '..'~',#127] then
                Get_Next(Fx,Fy)
        end;     (*    case    *)
   until not Create_Letter;
   if not Store then
   begin
      GoToXY(10,25);
      write('Letter is not save yet, save (N/Y) ?');
      repeat read(Kbd,Ch) until upcase(Ch) in ['N','Y'];
      if upcase(Ch) = 'Y' then Save
   end
end;    (*   Add_Letter    *)


{$I display.pas}

(*****************************************************************)
(*    This procedure searches the record that is requested by    *)
(*    a user.  A message will be displayed if the record is      *)
(*    not found.                                                 *)
(*****************************************************************)
```

```
Procedure Do_Search(var F : AFiles; var Displays : boolean);
var
   Not_In : boolean;
   I : integer;
begin
  found := false;
  Not_In := false;
  GoToXY(4,24); ClrEol;
  GoToXY(1,24);
  write('NEXT ID -->');
  GoToXY(12,24); read(TempId_No);
  repeat
   Do_Hashing(F,I,Found,Not_In);
   if (not Found) then
   begin
     write(Bell); GoToXY(22,24);
     write('Id not found, please enter another');
     write('Id or 9 zeros to quit');
     GoToXY(12,24); write('          ');
     GoToXY(12,24); read(TempId_No);
   end;
  until (Found) or (TempId_No = '000000000');
  GoToXY(1,24); write('-->'); ClrEol;
end;    (*   Do_Search    *)


(*****************************************************************)
(*    This procedure finds the next address record of the        *)
(*    address file if it exists.                                 *)
(*****************************************************************)


Procedure Display_Next(var F : AFiles);
var
   Exits : boolean;
begin
    GoToXY(25,24); ClrEol;
    Exits := false;
    reset(F);
     repeat
      Index := Index + 1;
      if Index <= Max_Adds then
      begin
        seek(F,Index);
        read(F,Inadds);
        if Inadds.Id_No <> '          ' then Exits := true;
      end
      else Exits := true
     until (Exits) or (Index > Max_Adds);
    close(F);
    if Index > Max_Adds then
    begin
       GoToXY(25,24); write(' ');
       GoToXY(25,24); write(' end of file reached');
       Delay(1000);
```

```
    end
end;    (*    Display_Next    *)


(*****************************************************************)
(*    This procedure clears the field value of the address       *)
(*    record that is displayed on the screen.                    *)
(*****************************************************************)


Procedure Clear_Contents;
var
   I : integer;
begin
     for I := 4 to 23 do
     begin
         GoToXY(18,I);
         write(' ' : 28);
     end    (*    for loop    *)
end;    (*    Clear_Contents    *)


(*****************************************************************)
(*    This procedure displays the contents of the address        *)
(*    record on the screen.                                      *)
(*****************************************************************)


Procedure Display_Contents(Adds : AddsRecs);
var
  I : integer;
begin
   I := 1;
   if First then
   begin
      GoToXY(25,24); write('The 1st record'); ClrEol;
      First := false
   end;
   with Adds do
   begin
     GoToXY(18,3);  write(Crit_Code[I]);
     GoToXY(18,4);  write(Crit_Code[I+1]);
     GoToXY(18,5);  write(Crit_Code[I+2]);
     GoToXY(18,6);  write(Crit_Code[I+3]);
     GoToXY(18,7);  write(Crit_Code[I+4]);
     GoToXY(18,8);  write(Crit_Code[I+5]);
     GoToXY(18,9);  write(Crit_Code[I+6]);
     GoToXY(18,10);  write(Crit_Code[I+7]);
     GoToXY(18,11);  write(Crit_Code[I+8]);
     GoToXY(18,12);  write(Crit_Code[I+9]);
     GoToXY(18,13);  write(Id_No);
     GoToXY(18,14);  write(Name);
     GoToXY(18,15);  write(Addrs1);
     GoToXY(18,16);  write(Addrs2);
     GoToXY(18,17);  write(CityStZip);
```

```
      GoToXY(18,18);  write(Country);
      GoToXY(18,19);  write(Salutation);
      GoToXY(18,20);  write(Closing)
    end     (*     with     *)
end;     (*     Display_Contents     *)


(*****************************************************************)
(*     This procedure reads a letter file from the disk and      *)
(*     displays it on the screen.                                *)
(*****************************************************************)


Procedure Display_Letter_On_Screen(File_Id : Anystring);
var
  Fx, Fy : integer;
  F : LFiles;
begin
  ClrScr;
  assign(F,File_Id);
  reset(F);
  For Fx := 0 to FxMax do
      For Fy := 0 to FyMax do
      begin
         read(F,Letter[Fx,Fy]);
         if Letter[Fx,Fy].Ch <> '' then
         begin
             GoToXY(Fx+1,Fy+1);
             write(letter[Fx,Fy].Ch)
         end
      end;
  GoToXY(1,24); write('-->');
  GoToXY(10,25); write('R - Return to main menu');
  GoToXY(36,25); write('H - Help');
  GoToXY(49,25); write(File_Id);
  close(F)
end;     (*     Display_Letter_On_Screen     *)


Procedure Display_Letter(File_Id : Anystring);
var
   Ch : char;
   Display_Letter : boolean;
begin
   Display_Letter := true;
   while Display_Letter do
   begin
      Display_Letter_On_Screen(File_Id);
      GoToXY(4,24); read(Trm,Ch);
      case Ch of
          'h','H' : Help_Display_Letter_File;
          'r','R' : Display_Letter := false;
          else
            repeat
               GoToXY(4,24); ClrEol;
```

```
                GoToXY(4,24); read(Trm,Ch);
              until Ch in ['r','R','h','H'];
        end  (*     case     *)
    end
end;      (*   Display_Letter     *)


(*****************************************************************)
(*  This procedure reads the criterion file from the disk and    *)
(*  displays it on the screen.                                   *)
(*****************************************************************)


Procedure Display_Crits_on_Screen;
var Filen : Anystring;
begin
    Read_Crit_File(Filen);
    If (Filen = '') then
    begin
      GoToXY(25,24); write('no criterian file will be displayed');
      ClrEol; write(Bell); Delay(2000)
    end     (*     if     *)
end;     (*     Display_Crits     *)


(*****************************************************************)
(*    This procedure  allows a user to search a record, display  *)
(*    the next existing record, display the Help menu or return  *)
(*    to Main menu.                                              *)
(*****************************************************************)


Procedure Display_Adds_On_Screen(File_Id : Anystring;
                                 var Display : boolean);
var
    F : AFiles;
    Ch : char;
    Sel_Type : set of char;
begin
    First := false;
    Index := -1;
    Display := true;
    Assign(F,File_Id);
    Sel_Type := ['S','D','H','R','C'];
    reset(F);
    repeat
      Index := Index + 1;
      seek(F,Index);
      read(F,Inadds);
    until (Inadds.Id_No <> '          ') or (Index = Max_Adds);
    close(F);
    GoToXY(25,24); write('The 1st record');
    while Display do
    begin
      Clear_Contents;
```

```
      Display_Contents(Inadds);
      GoToXY(4,24); read(Trm,Ch);
      Ch := Upcase(Ch);
      case Ch of
        'S' : Do_Search(F,Display);
        'C' : begin
                if Index > Max_Adds then
                begin
                  Index := -1;
                  First := true
                end;
                Display_Next(F);
              end;
        'H' : begin
                Help_Display_Address_File;
                Display_Address_Menu(File_Id);
                Display_Crits_On_Screen;
                GoToXY(25,24); ClrEol;
                Index := -1;
                reset(F);
                repeat
                   Index := Index + 1;
                   seek(F,Index);
                   read(F,Inadds);
                until Inadds.Id_No <> '          ';
                close(F); GoToXY(25,24);
                write('The 1st record');
              end;
        'R' : Display := false;
        else begin
               repeat
                 GoToXY(4,24); write(' '); GoToXY(4,24);
                 read(Trm, Ch); Ch := Upcase(Ch);
               until Ch in (Sel_Type - ['D']);
             end;
      end;     (*    case     *)
    end;      (*     Do while     *)
end;     (*     Display_Adds_On_Screen     *)

Procedure Display_Address(File_Id : Anystring);
 var
    Ch : char;
    Display_Address : boolean;
    Type_Ok : boolean;
    Sel_Type : set of char;
    Filen : Anystring;
begin
    Display_Address := true;
    ClrScr;
    Sel_Type := ['C','R','H','S','D'];
    while Display_Address do
    begin
      Display_Address_Menu(File_Id);
      GoToXY(4,24); read(Trm,ch);
      Ch := upcase(Ch);
```

```
      if not (Ch in Sel_Type) then
      begin
           write(Bell); GoToXY(4,24); ClrEol
      end     (*     if     *)
      else
         case Ch of
             'H' : Help_Display_Address_File;
             'R' : Display_Address := false;
             'D' : begin
                      Display_Crits_On_Screen;
                      GoToXY(25,24); ClrEol;
                      Display_Adds_On_Screen(File_Id,
                                             Display_Address);
                      gotoxy(15,24);
                      write('exit display_adds_on_screen')
                   end;
           'S','C' : begin
                        GoToXY(15,24);
                        write('enter D for display first');
                        write(Bell); Delay(1000);
                        GoToXY(4,24); ClrEol; GoToXY(15,24);
                        ClrEol
                      end    (*     'S','C'     *)
         end     (*     case     *)
   end;     (*     while     *)
  ClrScr;
end;     (*     Display_Address     *)


(*****************************************************************)
(*   This procedure checks the file name entered by a user to    *)
(*   see if the file is an address file or a letter file and     *)
(*   invokes the desired procedure.                              *)
(*****************************************************************)


Procedure Display_On_Screen(File_Id : Anystring);
var
   Leng, I : integer;
   File_Type : string[3];
   Founds : boolean;
begin
   I := 1;
   Leng := Length(File_Id);
   Founds := false;
   while (not Founds) and (I <= Leng) do
     if File_Id[I] = '.' then founds := true
     else I := I + 1;
   if Founds then
   begin
        File_Type := Copy(File_Id,I+1,3);
        File_Type[1] := Upcase(File_Type[1]);
        File_Type[2] := upcase(File_Type[2]);
        File_Type[3] := Upcase(File_Type[3]);
        if ((File_Type[1]='A') and (File_Type[2]='D') and
```

```
            (File_Type[3]='S'))
        then Display_Address(File_Id) else
        if ((File_Type[1]='L') and (File_Type[2]='T') and
            (File_Type[3]='R'))
        then Display_Letter(File_Id)
        else begin
                write(Bell);
                GoToXY(15,24);
                write('sorry, the system can not');
                write('display the file');
                GoToXY(15,25);
                write('that the file type is other');
                write('than "ads" and "ltr".');
                Delay(4500)
             end;     (*    else    *)
    end   (*    if    *)
    else begin
            write(Bell); GoToXY(25,24);
            write(File_Id,': ','error, ');
            write('File name does not have the file type');
            Delay(3500)
         end    (*    else    *)
end;    (*    Display_On_Screen    *)


{$I deladds.pas}

(*****************************************************************)
(*   This procedure displays the first record of the address     *)
(*   file on the screen.                                         *)
(*****************************************************************)


Procedure Display_First_Record(var F : AFiles);
begin
   First := false;
   Index := -1;
   reset(F);
   repeat
      Index := Index + 1;
      seek(F,Index);
      read(F,InAdds);
   until (InAdds.Id_No () '          ') or (Index = Max_Adds);
   close(F);
   if InAdds.Id_No () '          ' then
   begin
      GoToXY(25,24); write('The 1st record')
   end
end;    (*   Display_First_Record    *)
```

```
(*************************************************************)
(*   This procedure reads the input entered by the user and  *)
(*   converts the input into an IBM character.               *)
(*************************************************************)


Procedure Read_Users_Request(var Ch : char);
begin
   GoToXY(4,24); read(Kbd,Ch);
   if Keypressed then
   begin
      read(Kbd,Ch);
      case ch of
        'S' : Ch := #127;
        'Q' : Ch := #00;
      end;     (*  case  *)
   end;
   Ch := upcase(Ch)
end;     (*      Read_Users_Request     *)


(*************************************************************)
(*  This procedure displays a warning message on the screen. *)
(*************************************************************)


Procedure Warning_Msg(Var Ch : char);
var
   Select : set of char;
begin
   GoToXY(25,24);
   write('Are you sure you want to delete this record (N/Y)?');
   GoToXY(75,24); read(Trm,Ch);
   Select := ['Y','N'];
   if not (upcase(Ch) in Select) then
   repeat
      write(Bell);
      GoToXY(75,24); ClrEol; GoToXY(75,24); read(Trm,Ch);
   until upcase(Ch) in Select
end;     (*      Warning_Msg      *)


(*************************************************************)
(*  This procedure deletes an address record from the address *)
(*  file.                                                    *)
(*************************************************************)


Procedure Do_Delete(var F : AFiles);
var
   I : integer;
begin
   reset(F);
```

```
    with BlankAdds do
    begin
       for I := 1 to 10 do Crit_Code[I] := ' ';
       Id_No := '          ';
       Name := '                              ';
       Addrs1 := '                               ';
       Addrs2 := '                               ';       CityStZip := '
          ';
       Country := '                  ';
       Salutation := '                          ';
       Closing := '                          '
    end;
    seek(F,Index);
    write(F,BlankAdds);
    close(F);
    GoToXY(25,24); write('ADDRESS RECORD DELETED');
    ClrEol; Delay(1500)
end;     (*     Do_Delete     *)

Procedure Delete_Address(var F : AFiles);
var
  Display_Del_Adds : boolean;
  Sel_Type : set of char;

begin
  Display_Del_Adds := true;
  ClrScr;
  Display_Delete_Address_Menu(Filename);
  Assign(Addsfile,Filename);
  Sel_Type := ['H','R',#00,#127];
  Display_First_Record(F);
  while Display_Del_Adds do
  begin
     Clear_Contents;
     Display_Contents(InAdds);
     Read_Users_Request(Ch);
     if not (Ch in Sel_Type) then
     begin
        write(Bell);
        GoToXY(4,24);
        ClrEol
     end
     else
     case Ch of
        #127: begin
                 Warning_Msg(Ch);
                 If upcase(Ch) = 'Y' then
                    Do_Delete(F)
              end;
        #00 : begin
                 if Index > Max_Adds then
                 begin
                    Index := -1;
                    First := true
                 end;
                 Display_Next(F)
```

```
            end;
        'H' : begin
                Help_Delete_Address;
                Display_Delete_Address_Menu(Filename);
                Display_First_Record(F)
              end;
        'R' : Display_Del_Adds := false;
        else
            repeat
               write(Bell);
                GoToXY(4,24); write(' ');
               Read_Users_Request(Ch);
            until Ch in Sel_Type
        end      (*     case      *)
  end;     (*     do while     *)
  ClrScr
end;     (*     Delete_Address     *)


(*****************************************************************)
(*    This procedure allows a user to delete an existing file    *)
(*    from the disk.                                             *)
(*****************************************************************)


Procedure Delete_File(File_Id : Anystring);
var
  Ch : char;
  Select : set of char;
  F : File;
begin
   GoToXY(20,25);
   write('Are you sure you want to delete this file (N/Y) ? ');
   GoToXY(70,25); read(Trm,Ch);  Ch := Upcase(Ch);
   Select := ['Y','N'];
   If not (Ch in Select) then
   repeat
      GoToXY(70,25); ClrEol;
      GoToXY(70,25); read (Trm,Ch);  Ch := Upcase(Ch);
   until Ch in Select;
   if Ch = 'Y' then
   begin
      assign(F,File_Id);
      close(F);
      erase(F);
      GoToXY(20,25); write('FILE IS DELETED'); ClrEol;
      Delay(2000)
   end
end;     (*     Delete_File     *)


{$I label.pas}
```

```
(*****************************************************************)
(*  This procedure displays an error message on the screen and *)
(*  sets the error flag to true.                                 *)
(*****************************************************************)


Procedure Display_Error;
begin
   GoToXY(25,19);
   write('wrong criteria, please try again !!');
   write(Bell);
   Delay(2500);
   Error := true
end;    (*    Display_Error    *)


(*****************************************************************)
(*   This procedure determines whether the criteria entered by *)
(*   a user are syntactically correct or not.                    *)
(*****************************************************************)


Procedure Analize_Criteria(Line : Anystring; I : integer);
begin
   Error := false;
   if ((upcase(Line[1])='A') and (upcase(Line[2])='L') and
       (upcase(Line[3])='L')) then Select_All := true
   else
   if not ((Line[I] = '1') or (Line[I] = '0')) then Display_Error
   else begin
          if (Line[I] = '1') and (Line[I+1] = '0') then
              Crit_Index := 10
          else
            if (Line[I] = '0') and (Line[I+1] in ['1'..'9']) then
            begin
               case Line[I+1] of
                     '1' : Crit_Index := 1;
                     '2' : Crit_Index := 2;
                     '3' : Crit_Index := 3;
                     '4' : Crit_Index := 4;
                     '5' : Crit_Index := 5;
                     '6' : Crit_Index := 6;
                     '7' : Crit_Index := 7;
                     '8' : Crit_Index := 8;
                     '9' : Crit_Index := 9
               end    (*    case    *)
            end
            else Display_Error
        end;    (*  else    *)
   if not Error then
   begin
      Value := upcase(Line[I+3]);
      operator := upcase(Line[I+2])
```

```
   end
end;     (*  Analize_Criteria     *)

Procedure Do_Mail;
var I : integer;
    Line : Anystring;
begin
   I := 1;
   Line :='                                                          ';
   GoToXY(43,14); read(Line);
   Analize_Criteria(Line,I);
   Value1 := Value;
   Operator1 := Operator;
   Crit_Index1 := Crit_Index;
   if not Error then
     if Line[I+4] = ' ' then One_Criterion := true
     else begin
            One_Criterion := false;
            Relation := Line[I+4];
            I := I + 5;
            Analize_Criteria(Line,I);
          end
end;       (*     Do_Mail     *)


(*****************************************************************)
(*  This procedure prints a label on the line printer.           *)
(*****************************************************************)


Procedure Print_Labels;
var I : integer;
begin
     writeln(lst,' ': 30);
     with TempAdds do
          begin
            writeln(Lst,Name);
            if addrs1 <> ''then
               writeln(Lst,Addrs1);
            if Addrs2 <> '' then
               writeln(Lst,Addrs2);
            if CityStZip <> '' then
               writeln(Lst,CityStZip);
            if Country <> '' then
               write(Lst,Country)
          end;       (*   with    *)
          for I := 1 to 3 do writeln(Lst,' ':30);
          Index := succ(Index)
end;      (*     Print_Labels     *)
```

```
(*****************************************************************)
(*  This procedure matches the criteria entered by a user with *)
(*  the criteria in the address  file on  the  disk  and  then *)
(*  selects the address  record  for  printing  labels  if it  *)
(*  matches.                                                   *)
(*****************************************************************)


Procedure Do_Two_Criteria;
begin
   assign(AddsFile,Filename);
   reset(AddsFile);
   while not (EOF(AddsFile)) do
   begin
      read(AddsFile,TempAdds);
      with TempAdds do
        case Operator1 of
         '=': case relation of
                ',': case operator of
                       '=': if ((upcase(Crit_Code[Crit_Index1]))
                                =value1) and
                               ((upcase(Crit_Code[Crit_Index]))
                                 = value) then Print_Labels;
                       ')': if ((upcase(Crit_Code[Crit_Index1]))
                                 =value1) and
                               ((upcase(Crit_Code[Crit_Index]))
                                 ) value) then Print_Labels;
                       '(': if ((upcase(Crit_Code[Crit_Index1]))
                                 =value1) and
                               ((upcase(Crit_Code[Crit_Index]))
                                 ( value) then Print_Labels;
                       else Display_Error
                     end;      (*    case ','      *)
                '/': case operator of
                       '=': if ((upcase(Crit_Code[Crit_Index1]))
                                 =value1) or
                               ((upcase(Crit_Code[Crit_Index]))
                                 = value) then Print_Labels;
                       ')': if ((upcase(Crit_Code[Crit_Index1]))
                                 =value1) or
                               ((upcase(Crit_Code[Crit_Index]))
                                 ) value) then Print_Labels;
                       '(': if ((upcase(Crit_Code[Crit_Index1]))
                                 =value1) or
                               ((upcase(Crit_Code[Crit_Index]))
                                 ( value) then Print_Labels;
                       else Display_Error
                     end      (*    case '/'      *)
              else Display_Error
            end;     (*  case '='     *)
         '(': case relation of
                ',': case operator of
                       '=': if ((upcase(Crit_Code[Crit_Index1]))
```

```
                        (value1) and
                       ((upcase(Crit_Code[Crit_Index]))
                        = value) then Print_Labels;
                '>': if ((upcase(Crit_Code[Crit_Index1]))
                        (value1) and
                       ((upcase(Crit_Code[Crit_Index]))
                        > value) then Print_Labels;
                '<': if ((upcase(Crit_Code[Crit_Index1]))
                        (value1) and
                       ((upcase(Crit_Code[Crit_Index]))
                        < value) then Print_Labels;
                else Display_Error
              end;      (*    case ','      *)
        '/': case operator of
                '=': if ((upcase(Crit_Code[Crit_Index1]))
                        (value1) or
                       ((upcase(Crit_Code[Crit_Index]))
                        = value) then Print_Labels;
                '>': if ((upcase(Crit_Code[Crit_Index1]))
                        (value1) or
                       ((upcase(Crit_Code[Crit_Index]))
                        > value) then Print_Labels;
                '<': if ((upcase(Crit_Code[Crit_Index1]))
                        (value1) or
                       ((upcase(Crit_Code[Crit_Index]))
                        < value) then Print_Labels;
                else Display_Error
              end      (*    case '/'      *)
        else Display_Error
      end;     (*    case '('     *)
')': case relation of
        ',': case operator of
                '=': if ((upcase(Crit_Code[Crit_Index1]))
                        )value1) and
                       ((upcase(Crit_Code[Crit_Index]))
                        = value) then Print_Labels;
                '>': if ((upcase(Crit_Code[Crit_Index1]))
                        )value1) and
                       ((upcase(Crit_Code[Crit_Index]))
                        > value) then Print_Labels;
                '<': if ((upcase(Crit_Code[Crit_Index1]))
                        )value1) and
                       ((upcase(Crit_Code[Crit_Index]))
                        < value) then Print_Labels;
                else Display_Error
              end;      (*    case ','      *)
        '/': case operator of
                '=': if ((upcase(Crit_Code[Crit_Index1]))
                        )value1) or
                       ((upcase(Crit_Code[Crit_Index]))
                        = value) then Print_Labels;
                '>': if ((upcase(Crit_Code[Crit_Index1]))
                        )value1) or
                       ((upcase(Crit_Code[Crit_Index]))
                        > value) then Print_Labels;
```

```
                        '(': if ((upcase(Crit_Code[Crit_Index1]))
                                  )value1) or
                                 ((upcase(Crit_Code[Crit_Index]))
                                  ( value) then Print_Labels;
                              else Display_Error
                            end     (*    case '/'     *)
                      else Display_Error
                    end     (*    case ')'     *)
                  else Display_Error
              end     (*    Case     *)
          end;    (*    do while    *)
          close(AddsFile)
end;    (*    Do_Two_Criteria     *)



(*****************************************************************)
(* This procedure is the same as the procedure Do_Two_Criteria *)
(* described above except it only deals with one criterion.    *)
(*****************************************************************)


Procedure Do_One_Criterion;
var I : integer;
begin
   assign(AddsFile,Filename);
   reset(AddsFile);
   while not (EOF(AddsFile)) do
   begin
      read(AddsFile,TempAdds);
      with TempAdds do
      case operator1 of
        '=' : if (upcase(Crit_Code[Crit_Index1])) = value1 then
                  Print_Labels;
        '(' : if (upcase(Crit_Code[Crit_Index1])) ( value1 then
                  Print_Labels;
        ')' : if (upcase(Crit_Code[Crit_Index1])) ) value1 then
                   Print_Labels;
        else Display_Error
     end     (*    operator1    *)
   end;    (*    do while    *)
   close(AddsFile)
end;    (*    Do_One_Criterion    *)


(*****************************************************************)
(*  This procedure prints the  labels for the  entire address  *)
(*  file.                                                        *)
(*****************************************************************)


Procedure Do_Select_All;
begin
```

```
   assign(AddsFile,Filename);
   reset(AddsFile);
   while not (EOF(AddsFile)) do
   begin
     read(AddsFile,TempAdds);
     if TempAdds.Id_No <> '          ' then Print_Labels
   end;
   close(AddsFile)
end;     (*     Do_Select_All     *)

Procedure Do_Mail_Labels;
var
    Done : boolean;
begin
    Select_All := false;
    Done := false;
    Error := false;
    Index := 0;
    while (not (Done)) and (not (Error)) do
    begin
       Do_Mail;
       If not Error then
       begin
          if Select_All then Do_Select_All
          else
          if One_Criterion then Do_One_Criterion
          else Do_Two_Criteria;
          Done := true
       end      (*     if     *)
    end;  (*    do while     *)
    if not Error then
    begin
       GoToXY(23,19);
       write('Total ',Index,' label(s) printed');
       Delay(2500)
    end
end;     (*     Do_Mail_Labels     *)


(*****************************************************************)
(*  This procedure allows a user to return to the main menu,     *)
(*  ask for help or print the labels.                            *)
(*****************************************************************)


Procedure Mail_Labels;
var
  Display_M_Label : boolean;
  Leng : integer;
  File_Type : string[3];
begin
   Leng := Length(Filename);
   File_Type := Copy(Filename,Leng-2,3);
   if (File_Type = 'ads') or (File_Type = 'ADS') then
   begin
```

```
      Display_M_Label := true;
      ClrScr;
      while Display_M_Label do
      begin
         Display_Mail_Labels_Menu(Filename);
         GoToXY(4,22); read(Trm,Ch);
         if (upcase(Ch) in ['H','R','L']) then
            case Upcase(Ch) of
                 'H' : Help_Mail_Label;
                 'L' : Do_Mail_Labels;
                 'R' : Display_M_Label := false;
            end      (*     case     *)
         else
         begin
            GoToXY(4,22); ClrEol;
            repeat
              read(kbd,Ch);
            until upcase(Ch) in ['H','L','R']
         end      (*     else     *)
    end       (*   do while     *)
  end      (*     if     *)
  else begin
         GoToXY(25,23);
         write('File is not an address file (ads)!!');
         write(Bell);
         delay(2500)
       end
end;     (*     Mail_Labels     *)

Procedure Open_Message;
begin
    ClrScr;
    gotoxy(25,11);
    writeln('WELLCOME TO BULK MAILING SYSTEM');
    Delay(2500);
    ClrScr;
end;     (*  Open_message  *)

Procedure Mail_Letter(var File_Id : Anystring);
begin
end;

Procedure Print_Letter_Form(var File_Id : Anystring);
begin
end;


(*****************************************************************)
(*  This procedure allows a user to delete the address record  *)
(*  from the address file or delete the existing file.         *)
(*****************************************************************)


Procedure Delete_Menu;
var
```

```
    Sel_Type : set of char;
    D : char;
    Display_Delete : boolean;
    Delfile : file;
begin
    Sel_Type := ['a','A','f','F','h','H','r','R'];
    Display_Delete := True;
    while Display_Delete do
    begin
      ClrScr;
      Display_D_Menu;
      GoToXY(38,17);
      read(Trm,Ch);
      Ch := Upcase(Ch);
      if not (Ch in Sel_Type) then
      begin
           GoToXY(18,22); write('PLEASE TRY AGAIN !!');
           write(Bell); Delay(1200);
           GoToXY(38,17); ClrEol; GoToXY(18,22); ClrEol;
      end    (*  if    *)
      else
          case Ch of
            'h','H' : Help_Delete;
            'r','R' : Display_Delete := false;
          else
          begin
             GoToXY(33,19);
             Get_File_Name(Filename);
             if Filename <> '' then if (not Exist(Filename)) then
                begin repeat
                        GoToXY(18,22); write('FILE NOT FOUND !!');
                        write(Bell);
                        Delay(1200); GoToXY(33,19);
                        ClrEol; GoToXY(18,22);
                        ClrEol; GoToXY(33,19);
                        Get_File_Name(Filename);
                      until Exist(Filename) or (Filename = '');
                end;
             if Filename <> '' then
                  case Ch of
                     'a','A' : begin
                                 assign(Addsfile,Filename);
                                 Delete_Address(Addsfile);
                                 Display_Delete := false
                               end;
                     'f','F' : begin
                                 assign(Delfile,Filename);
                                 Delete_File(Filename);
                                 Display_Delete := false
                               end
                  end;   (*    case    *)
        end    (*    else    *)
      end      (*    case    *)
    end;     (*   do while   *)
  ClrScr
```

```
end;    (*     Delete_Menu     *)


(*****************************************************************)
(*  This procedure allows a user to create a criterion file,     *)
(*  an address file or a letter file.                            *)
(*****************************************************************)


Procedure Create_Menu;
var
   Sel_Type : set of char;
   C : char;
   Display_Create : boolean;
begin
   Sel_Type := ['a','A','c','C','I','L','h','H','r','R'];
   Display_Create := True;
   while Display_Create do
   begin
     ClrScr;
     Display_C_Menu;
     GoToXY(37,18);
     read(Trm,Ch);
     if not (Ch in Sel_Type) then
     begin
         GoToXY(21,24); write('PLEASE TRY AGAIN !!');
         write(Bell); Delay(1200);
         GoToXY(37,18); ClrEol; GoToXY(21,24); ClrEol;
     end       (*    if    *)
     else
         case Ch of
           'h','H' : Help_Create;
           'r','R' : Display_Create := false;
           else
           begin
            GotoXY(18,22); ClrEol;
            GoToXY(32,20);
            Ch := upcase(Ch);
            Get_File_Name(Filename);
            if Filename <> '' then
                  case Ch of
                    'A' : begin
                            Filename := Filename + '.' + 'ADS';
                            assign(Addsfile,Filename);
                            Add_Address(AddsFile);
                            Display_Create := False
                          end;
                    'L' : begin
                                Filename := Filename + '.' + 'LTR';
                                assign(Lettfile,Filename);
                                Add_Letter;
                                Display_Create:= false
                          end;
                    'C' : begin
                                Filename := Filename + '.' + 'CRI';
```

```
                                    assign(Critfile,Filename);
                                    Create_Criteria(Critfile);
                                    Display_Create := false;
                                end
                            end     (*    case    *)
                        end    (*   else   *)
                    end    (*    case    *)
                end;   (*   do while    *)
            ClrScr;
        end;    (*    Create_Menu    *)


(*****************************************************************)
(*  This procedure will either display the file on the screen    *)
(*  or print a hardcopy on the line printer.                     *)
(*****************************************************************)


Procedure Print_Menu;
var
    P : char;
    Sel_Type : set of char;
    Display_Print : boolean;
begin
    Sel_Type := ['d','D','p','P','h','H','r','R'];
    Display_Print := True;
    while Display_Print do
    begin
      ClrScr;
      Display_P_Menu;
      GoToXY(38,18);
      read(Trm,Ch);
      if not (Ch in Sel_Type) then
      begin
          GoToXY(18,23); write('PLEASE TRY AGAIN !!');
          write(Bell); Delay(1200);
          GoToXY(38,18); ClrEol; GoToXY(18,23); ClrEol;
      end       (*      if       *)
      else
          case Ch of
            'h','H' : Help_Print;
            'r','R' : Display_Print := false;
          else
          begin
             GoToXY(33,20);
             Get_File_Name(Filename);
             if Filename <> '' then if (not Exist(Filename)) then
                begin repeat
                        GoToXY(18,23); write('FILE NOT FOUND !!');
                        write(Bell); Delay(1000);
                        GoToXY(33,20); ClrEol; GoToXY(18,23);
                        ClrEol; GoToXY(33,20);
                        Get_File_Name(Filename);
                      until Exist(Filename) or (Filename = '');
                end;
```

```
            if Filename <> '' then
                 case Ch of
                    'd','D' : begin
                                 Display_On_Screen(Filename);
                                 Display_Print := false
                              end;
                    'p','P' : Print_Letter_Form(Filename);
                 end     (*    case     *)
          end     (*     else     *)
       end     (*     case     *)
   end;    (*    do while     *)
  ClrScr
end;    (*     Print_Menu     *)


(*****************************************************************)
(*  This procedure allows a user to either perform the Mail_     *)
(*  Label or the Mail_Letter operation.                          *)
(*****************************************************************)


Procedure Mailing_Menu;
var
   M : char;
   Sel_Type : set of char;
   Display_Mailing : boolean;
begin
   Sel_Type := ['a','A','h','H','r','R','l','L'];
   Display_Mailing := True;
   while Display_Mailing do
   begin
     ClrScr;
     Display_M_Menu;
     GoToXY(39,16);
     read(Trm,Ch);
     if not (Ch in Sel_Type) then
     begin
         GoToXY(23,20); write('PLEASE TRY AGAIN !!');
         write(Bell); Delay(1200);
         GoToXY(39,16); ClrEol; GoToXY(23,20); ClrEol;
     end     (*    if      *)
     else
         case Ch of
           'h','H' : Help_Mail;
           'r','R' : Display_Mailing := false;
           else
           begin
            GoToXY(35,18);
            Get_File_Name(Filename);
            if Filename <> '' then if (not Exist(Filename)) then
               begin repeat
                        write(Bell);
                        GoToXY(25,23); write('FILE NOT FOUND !!');
                        Delay(1200); GoToXY(35,18); ClrEol;
                        GoToXY(35,18); ClrEol;
```

```
                        GoToXY(25,23); ClrEol; GoToXY(35,18) ;
                        Get_File_Name(Filename);
                      until Exist(Filename) or (Filename = '');
                end;
             if Filename <> '' then
                  case Ch of
                     'a','A' : begin
                                 Mail_Labels;
                                 Display_Mailing := false
                               end;
                     'l','L' : begin
                                 Mail_Letter(Filename);
                                 Display_Mailing := false
                               end
                end    (*    case    *)
             end   (*    else    *)
       end    (*    case    *)
   end;   (*   do while    *)
  ClrScr;
end;   (*    Mailing_Menu    *)


(*************************************************************)
(*   This procedure allows a user to enter create menu, delete *)
(*   menu, mailing menu, print menu, help menu or return to    *)
(*   the DOS system.                                           *)
(*************************************************************)


Procedure Main_Menu;
var
   Display_Main : boolean;
begin
   Display_Main := true;
   Display_Main_Menu;
   while Display_Main do
      begin
         GoToXY(44,19);
         read(Trm,ch);
         case ch of
            'c','C'  : begin Create_Menu;  Display_Main_Menu end;
            'd','D'  : begin Delete_Menu;  Display_Main_Menu end;
            'm','M'  : begin Mailing_Menu; Display_Main_Menu end;
            'p','P'  : begin Print_Menu;   Display_Main_Menu end;
            'h','H'  : begin Help_Main;    Display_Main_Menu end;
            'x','X'  : Display_Main := False;
            else begin
                    GoToXY(29,23);
                    writeln(output,'PLEASE TRY AGAIN!!');
                    write(Bell); Delay(1200);
                    GoToXY(44,19);
                    ClrEol; GoToXY(29,23); ClrEol;
                 end  (*  else  *)
         end    (* case  *)
      end;    (*  do while  *)
```

```
   ClrScr;
end;    (*    Main_Menu   *)

begin   (*   Main    *)
   ClrScr;
   Open_Message;
   Main_Menu;
   write('back to DOS system')
end.    (*   Main   *)
```

A FORM BASED PERSONALIZED BULK MAILING SYSTEM

by

LIN, HUI-LING

B.A., Kansas State University, 1982

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1984

ABSTRACT

The purpose of this paper is to describe the design of a personalized bulk mailing system and its implementation on the Columbia Data Product microcomputer (an IBM-PC look-alike). The objective of this design was to provide a bulk mailing system which may be used by small businesses and companies on a microcomputer system. The mailing, optionally, may be flexibly and efficiently personalized on a document by document basis. The recipients may include the entire customer mailing list or may be restricted to only a certain subset of customers on that list through logical operations on up to ten characteristics of these individuals. The mailing system is designed to be user-friendly, to promote understanding, and to allay fears of the computer among the general public.

The implementation was done in TURBO Pascal using the MS-DOS operating system version 2.0 and contains no machine specific code.