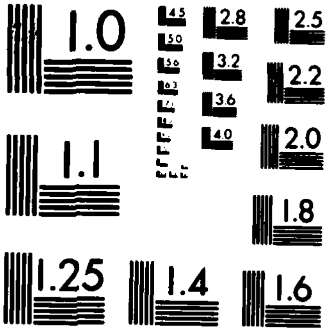
AD-A151 359 INTERACTIVE DATA ANALYSIS DEVELOPMENT OF AN INTERACTIVE 1/3
DATA MANIPULATION SYSTEM(U) NAVAL POSTGRADUATE SCHOOL
MONTEREY CA N G TOTOS JUN 84

UNCLASSIFIED F/G 9/2 NL


1.0
4.5
5.0
5.6
6.3
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

REPRODUCED AT GOVERNMENT EXPENSE

②

NAVAL POSTGRADUATE SCHOOL

Monterey, California

AD-A151 359

DTIC
ELECTE
MAR 14 1985
S D
B

THESIS

INTERACTIVE DATA ANALYSIS: DEVELOPMENT OF AN INTERACTIVE DATA MANIPULATION SYSTEM

by

Nicholas G. Totos

June 1984

Thesis Advisor: R. Weissinger-Baylon

DTIC FILE COPY

Approved for public release; distribution unlimited.

85 03 06 106

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Interactive Data Analysis: Development of an Interactive Data Manipulation System		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; June 1984
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Nicholas G. Totos		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		12. REPORT DATE June 1984
		13. NUMBER OF PAGES 222
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

SMYPNH	PATRIS
ELLADA	
NNGS	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Large amounts of data collected during experiments or produced as outputs of programs that simulate systems, usually need further treatment in order to complete the research task. One step of the data analysis process is the manipulation of the data.

The data, even when stored in a computer, can be considered as inert if it cannot be manipulated. Manipulation may be

#20 - ABSTRACT - (CONTINUED)

considered any appropriate arrangement or transformation of a logical data matrix composed by the data.

The inert data matrix is activated and becomes an "Active Matrix" by the developed system IDAMAN (Interactive DAta MANager). It is expected that this system, based on the idea of Dr. Daniel Guinier, fulfills the demands for such manipulations. Compared to existing similar alternative systems, this system possesses two particular merits: No specific language is required; and the separation of data manipulation task from acquisition and future calculus grants to the system a high power of expandability.

Accession For	
NTIS GRA&I	✓
[illegible]	☐
[illegible]	☐
[illegible]	
[illegible]	
Distribution/	
Availability Codes	

Dist	Avail and/or Special
A-1	

DTIC COPY INSPECTED

Approved for public release; distribution unlimited.

Interactive Data Analysis:
Development of an Interactive
Data Manipulation System

by

Nicholas G. Totos
Commander, Hellenic Navy
B.S., Hellenic Naval Academy, 1967

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1984

Author: ______________________

Approved by: ______________________
Thesis Advisor

____
Second Reader

____
Chairman, Department of Computer Science

____
Dean of Information and Policy Sciences

ABSTRACT

Large amounts of data collected during experiments or produced as outputs of programs that simulate systems, usually need further treatment in order to complete the research task. One step of the data analysis process is the manipulation of the data.

The data, even when stored in a computer, can be considered as inert if it cannot be manipulated. Manipulation may be considered any appropriate arrangement or transformation of a logical data matrix composed by the data.

The inert data matrix is activated and becomes an "Active Matrix" by the developed system IDAMAN (Interactive DAta MANager). It is expected that this system, based on the idea of Dr. Daniel Guinier, fulfills the demands for such manipulations. Compared to existing similar alternative systems, this system possesses two particular merits: No specific language is required; and the separation of data manipulation task from acquisition and future calculus grants to the system a high power of expandability.

Additional keywords: theses, Fortran, data processing, subroutines, sorting, transformations (mathematics), instructions.

TABLE OF CONTENTS

I. GENERAL DESCRIPTION OF IDAMAN ------------------- 10

A. INTRODUCTION ----------------------------------- 10

B. CONCEPT OF THE INTERACTIVE DATA MANAGER, IDAMAN -- 11

1. Creation of the Header ------------------ 11

2. Manipulation of the Data ---------------- 13

C. CONCEPTUAL PROPERTIES OF THE DEVELOPED SYSTEM IDAMAN ------------------------------- 16

D. EXISTING SIMILAR SOFTWARE PACKAGES FOR DATA ANALYSIS ------------------------------- 16

E. MATERIAL ------------------------------------- 18

F. OVERVIEW OF FUNCTIONAL CAPABILITIES OF IDAMAN --------------------------------------- 18

G. DESIGN --------------------------------------- 20

1. Modular Structure of IDAMAN ------------- 20

2. Brief Description of the Subroutines ---- 20

II. METHODS USED FOR THE IMPLEMENTATION ------------- 29

A. SORTING -------------------------------------- 29

1. Introduction ---------------------------- 29

2. Binary Tree Sorts (Monkey Puzzle, Hoare's Quicksort) ---------------------- 30

3. Shell's Method -------------------------- 31

4. Singleton's Method ---------------------- 31

5. Evaluation of Selected Sort Methods in Terms of Performance ----------------- 32

B. MULTIPLE SORTING ----------------------------- 34

1. Introduction ---------------------------- 34

2. Principle -------------------------------- 38

a. Phase One ------------------------------ 38

b. Phase Two ------------------------------ 42

C. TRANSFORMATIONS ---------------------------- 45

1. Introduction ---------------------------- 45

a. Fundamental Transformations --------- 46

(1) Logarithmic Transformation (Neperian and Common) ---------- 46

(2) The Inverse Sine or Arcsine Transformation ----------------- 46

(3) The Square Root Transformation - 46

(4) The Inverse Transformation ----- 47

(5) The Inverse Hyperbolic Sine or Arcsine Hyperbolic Transformation ----------------- 47

(6) Exponentiation at Tth Transformation ----------------- 47

b. Complex Transformations ------------- 47

2. Principle -------------------------------- 48

a. Breakdown of the Expression in Parts ------------------------------- 48

b. Conversion from Infix to Postfix Notation ---------------------------- 50

c. Evaluation of the Postfix Expression ------------------------- 55

D. RANDOMIZATION ------------------------------ 57

1. Introduction ---------------------------- 57

2. Principle -------------------------------- 58

III. DIRECTIONS FOR USE OF THE IDAMAN ---------------- 62

A. INTRODUCTION -------------------------------- 62

B. MANAGING DATA FILES ------------------------ 62

C. USER ERROR PREVENTION ---------------------- 63

D. SELECTING MODE OF OPERATION ---------------- 63

E. THE FUNCTION OF MODES AND HOW TO DEAL WITH THEM ---------------------------------- 64

1. Creating a New Header -------------------- 64

2. Assigning Information for the Columns --- 65

a. Number of Columns ------------------- 65

b. Mnemonic Names ---------------------- 66

c. Transformations --------------------- 66

d. Tracing Extrema --------------------- 69

e. Multi-sorting Column Guides --------- 70

f. Randomization of Data --------------- 72

g. Column Ranking ---------------------- 72

h. Display of Column Information ------- 73

i. Modification of Column Information -- 73

(1) Number of Columns -------------- 74

(2) Mnemonic Names ----------------- 74

(3) Tracing Extrema ---------------- 75

(4) Sorting Guides ----------------- 75

(5) Rank of Columns ---------------- 76

j. Data Retrieval ---------------------- 76

k. Display/Print of Data File ---------- 78

3. Assigning Information for the Rows ------ 78

a. Number of Rows ---------------------- 79
b. Mnemonic Names ---------------------- 79
c. Row Suppression --------------------- 80
d. Row Rejection ----------------------- 81
e. Row Ranking ------------------------- 82
f. Display Row Information ------------- 82
g. Modification of Row Information ----- 82
(1) Number of Rows ----------------- 83
(2) Mnemonics by Row --------------- 83
(3) Mnemonics by Series of Rows ---- 84
(4) Suppression -------------------- 85
(5) Rank of Rows ------------------- 86
h. Data Retrieval ---------------------- 86
i. Display/Print of Data File ---------- 86
2. Displaying an Existing Header ----------- 86
3. Modifying an Existing Header ------------ 87
4. Merging --------------------------------- 87
IV. CONCLUSION AND PERSPECTIVES --------------------- 90
APPENDIX A: FORTRAN 77 SOURCE CODE OF INTERACTIVE DATA MANAGER ------------------------------ 92
LIST OF REFERENCES ------------------------------------ 218
BIBLIOGRAPHY -- 220
INITIAL DISTRIBUTION LIST ----------------------------- 222

LIST OF FIGURES

1. Process of Data Analysis ------------------------- 12

2. Comparison of Several Concepts of Interactive Data Analysis Systems ---------------------------- 15

3. Modular Structure of IDAMAN ---------------------- 21

4. Modular Structure of IDAMAN Showing the "Overlay" Levels ---------------------------------- 22

5. Nested Sorting of Data Matrix -------------------- 35

6. Single Sorting ----------------------------------- 36

7. Set of Data Containing Two Classes and Three Subclasses ---------------------------------- 37

8. Virtual Process of Repeating Sorting ------------- 39

9. Real Process of Repeating Sorting ---------------- 40

10. Rearranging of the Rows of Data Matrix According to the Inverse Relative Addresses ------- 43

11. Infix to Postfix Conversion ---------------------- 53

12. Representation of the Postfix Expression ---------- 55

13. Evaluation of a Postfix Expression --------------- 56

I. GENERAL DESCRIPTION OF IDAMAN

A. INTRODUCTION

Research in many scientific fields is usually involved with such a large amount of data that manipulation becomes difficult, time consuming and error prone. A large amount of data requires auxilliary storage such as a disk for backup and data processing, while the efficiency required calls for direct access organization of the data file. The accomplishment of such a task requires several considerations.

The first consideration is availability of the data. The sources of data are either application programs in several areas such as language analysis, biology [Refs. 1 and 2], simulation, etc., or observations and measurements during experiments. The data are either readily available for manipulation or must be inserted into the system by the user via the keyboard.

The second consideration is the nature of manipulation of the data that has to be performed after it is available in the machine. The provided-by-the-system manipulation ability has to be such that the manipulated data will be ready for the next of the process, namely the mathematical model analysis, graphics, or statistical calculus. The system must also be capable of covering as many cases as possible. The data must be stored in an organized virtual structure, e.g., easily realized by the user logical table or matrix on random mass storage.

The above considerations call for a computer methodology that gives to the system DATA-USER-DATA manager a high level of interactivity, and at the end of the fourth step (Fig. 1), maximum services with minimum user activity. The data must be organized by the user so that it can be easily referred to, and manipulated. The creation of such an organization of the data can be called a "header" of the data matrix.

B. CONCEPT OF THE INTERACTIVE DATA MANAGER, IDAMAN

To perform a complete data processing cycle, the Interactive Data Manager, IDAMAN, supports two kinds of functions:

- Creation of a header.
- Data manipulation according to the user defined header.

The user is not involved with the manipulation of the data but rather only assigns what manipulation is desired, by creating the header of the data matrix. At the same time, the system supervises the creation of the appropriate files that are transparent to the user.

1. Creation of the Header

An interactive-informative communication between computer and user creates the header. The created header contains information regarding the columns and the rows of the data matrix. Such information causes the manipulation of the data. The header's data are integers (e.g., number of columns, number of rows), reals (e.g., actual or imposed

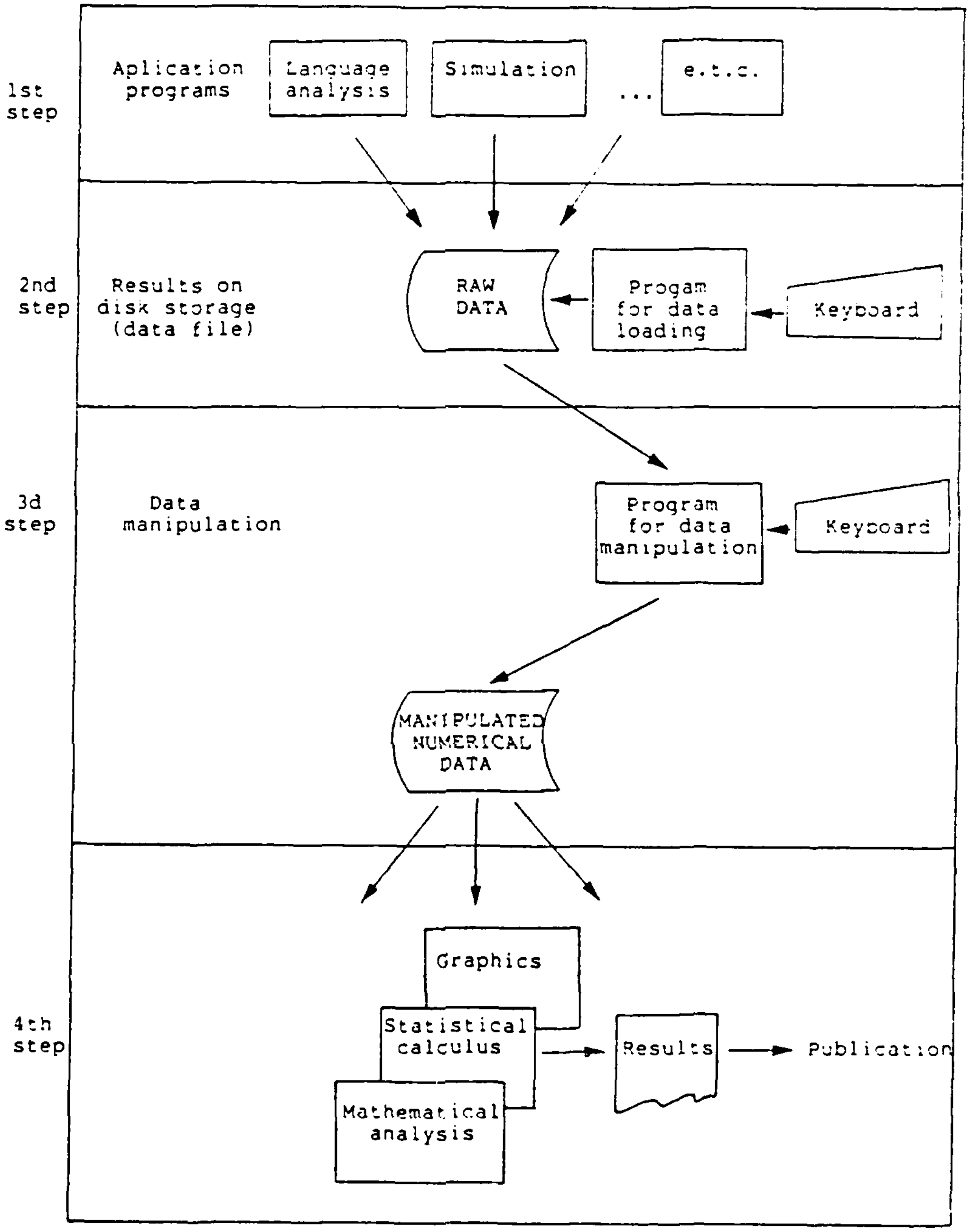

1st step
Aplication programs
Language analysis
Simulation
...
e.t.c.
2nd step
Results on disk storage (data file)
RAW DATA
Progam for data loading
Keyboard
3d step
Data manipulation
Program for data manipulation
Keyboard
MANIPULATED NUMERICAL DATA
4th step
Graphics
Statistical calculus
Mathematical analysis
Results
Publication

Fig. 1. Process of Data Analysis

by the user boundary values (extrema) for each column) and characters (e.g., names of the variables, names of sets of observations (rows)).

Several data of the header are stored in the mass storage in a sequential access file for future use.

2. Manipulation of the Data

The data to be manipulated are stored in mass storage in direct access organization files and in the logical form of a matrix with dimension NCOL*NROW, where NROW is the number of rows (or observations or lines) and NCOL is the number of columns (or parameters).

The manipulation, a transparent operation of the system, is executed during the creation of the header and at the same time that each manipulation is assigned. The manipulation of the data causes the creation of a new direct access file in the secondary storage whenever it is considered necessary. The same file is used in the opposite case for space economy reasons.

Manipulation functions executed by the IDAMAN are:

- Addition or suppression of lines or columns.
- Rearrangement of the order of lines or columns (ranking).
- Transformation, single or multipass, is performed by analysis of a full input character string (representing the transformation) conversion from infix to postfix notation, and evaluation using push-down stack. The assigned transformations may include functions with

arguments such as the number of columns (actually the value corresponding to column will be used as an argument), reals or integers, and the number of columns (in that case the value corresponding to the column number is taken to be constant).

- Sorting column guides, single or multiple, where successive sortings are executed as a function of successive columns.
- Searching for a particular value or a missing value involving the entire data matrix or columns and/or rows assigned by the user.
- Randomization of row order to eliminate problems created due to collection time effect.
- Merging of two files--either row-row (one over the other), or column-column (one beside the other).
- Display of the data matrix with user defined format via the keyboard.

As soon as the inert data are involved in a manipulation process, IDAMAN can be considered as activated. The data matrix with its assigned header can be called an "ACTIVE DATA MATRIX."

Fig. 2 illustrates the comparison of three different concepts: a) an interactive and conversational system for data analysis applied to biology [Refs. 1 and 2]; (b) the IDA from SPSS (Interactive Data Analysis from the Software Package for Social Sciences); and (c) the presented system, IDAMAN.

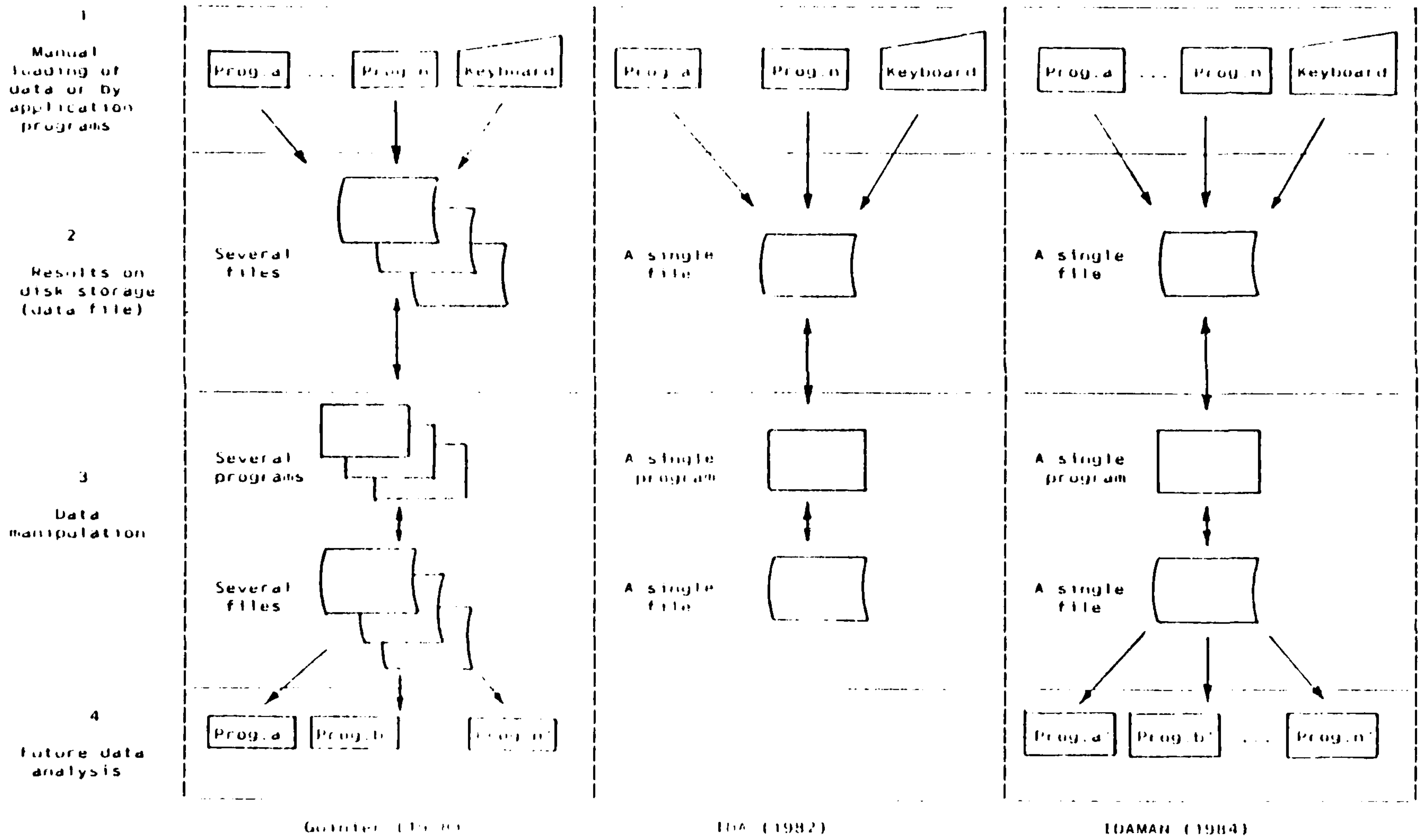

1
Manual loading of data or by application programs
2
Results on disk storage (data file)
3
Data manipulation
4
Future data analysis
Prog. a
Prog. n
Keyboard
Several files
Several programs
Several files
Prog. a'
Prog. b'
Prog. n'
A single file
A single program
A single file
Prog. a
Prog. n
Keyboard
A single file
A single program
A single file
Prog. a'
Prog. b'
Prog. n'
Gwinter ([illegible])
IDA (1982)
IDAMAN (1984)

Fig. 2. Comparison of Several Concepts of Interactive Data Analysis Systems

C. CONCEPTUAL PROPERTIES OF THE DEVELOPED SYSTEM IDAMAN

The developed system IDAMAN has the following properties:

- Universality of the source language: structured FORTRAN 77 is used for the implementation of the system.
- Portability: Granted by the use of FORTRAN 77 and the organization of the program. The required central memory size can even be satisfied by a microcomputer with module overlay.
- Simplicity of use: No special knowledge or language is required for the use of the system.
- All operations like creation and/or saving the data on files, are transparent to the user. Display or printing the data contained on these files (header or data file) is very simple and does not need any special consideration about formats.

D. EXISTING SIMILAR SOFTWARE PACKAGES FOR DATA ANALYSIS

Existing known packages providing the same task can be divided into two categories:

- Packages of subroutines for specific calculations without system data manipulation; such systems are the HARWELL Library (1981) from the United Kingdom Energy Authority, the IMSL Library (1982), and the System 360 Subroutines Library (1968).
- Packages of subroutines for calculations with the system of data manipulation; such systems are the BioMeDical Package (BMDP, 1981), and the Statistical Package for the Social Sciences (SPSS, 1983).

BMDP and SPSS are complete statistical packages. The first one is produced by UCLA (University of California at Los Angeles) and is oriented toward Bio-medical statistical applications. The second is oriented to social science statistical applications. These two packages have a wide range of applications in statistics but are not interactive and use their own conventional languages.

In the SPSS series in data analysis, IDA is presented as an interactive data analysis and forecasting system (1982).

The difference between the concepts of IDA and IDAMAN is that in IDA, the data are manipulated and elaborated by the same program (data summarization, regression, time-series analysis, etc.), while in IDAMAN, only manipulation is executed. IDAMAN is strictly a data manipulation system. The intention of the design was to separate the two tasks--considering them as different. In this way, modularization is obtained. The next step of the general task "data analysis" can be executed separately by another module. The data, after the manipulation, are used by independent or interdependent modules. Such modules can provide one, two or multivariable analysis (mathematical and/or statistical) involving one or several samples or/and series of samples. The IDAMAN provides the flexibility of executing the same data elaborations with the same data in different forms (manipulations).

The calculation subroutines, described by several similar packages, can be integrated by the process of a calling program that reads the data of the header created by the IDAMAN.

E. MATERIAL

IDAMAN was [illegible] at the Department of Computer Science at the Naval Postgraduate School, Monterey, California on the VAX11/780 under the VMS operating system. The disposable virtual memory is 32 Megabytes and the direct-access memory is the user's disk RM05 (256 Megabytes).

FORTRAN 77 on VAX11 permits the use of the facilities given to a 32 bit or to a 16 bit computer (PDP11 or LSI11 or any other 16 bit microcomputer precision for integer and real numbers, e.g., integers on 2 bytes or on 4 bytes).

The modularity of the program implementing the IDAMAN permits a transportability to a lower memory capacity microcomputer (16 bit or 16 bit-like microcomputer).

F. OVERVIEW OF FUNCTIONAL CAPABILITIES OF IDAMAN

- Column mnemonics: used to assign mnemonic names to columns (parameters or variables).
- Row mnemonics: used to assign mnemonic names to individual rows and/or sets (or subsets) or rows. In this way the data can be retrieved by index (the set number) or by the row or set mnemonic name. An example of the usefulness of this operation is the comparison of means of sets or the tracing of groups of data.
- Tracing extrema: used to permit the user to change the extrema values by giving a "window" for tracing. With this function, the system does not need additional information about scales if a tracing application program is required.

- Sorting and multi-sorting: used to reorganize the data according to column guides by sorting successively in increasing order to generate sets and subsets of rows (observations).
- Data retrieval: used to find one or several values inside specific fields (column(s) or row(s)) or a specific "gold" number that corresponds to missing values. The process of retrieval is applied in the full matrix or in particular parts of it, given by the user. The data to be retrieved can be:
  - Identified as erroneous data.
  - Missing data which are referenced as a "gold" value internal to the system or changed by the user to represent a pseudo-improbable value.
  - Listed, and the associated row, column and index numbers are given to the user. This information will be useful for eventual corrections or localization on the listing.
- Rearrangement: used to eliminate the column guides used for multisorting or to partition the data matrix for particular elaborations (e.g., multiple partial regression).
- Transformations: used to restore some statistical properties or to perform directly mathematical operations involving data of the matrix and/or numbers.
- Randomization: used to burst the original data to avoid effects of time or preordering.

- Compression: used to suppress some data for future calculations (logical suppression) without physical suppression of the data. This operation can be done using a special compressed binary file [Refs. 3 and 4].
- Input/Output: transparent to the user and can be directed to the desired device or file.
- Files generation: operations also transparent and executed by assignment of a single key (number of alphanumeric).
- Display of the header's information: used to facilitate the use of the program and the creation of the header.
- Display the data: used to display, on the terminal, the manipulated data being on direct access files in secondary memory, by a simple command and in format defined by the user, if any change of the normal format is required (e.g., nnnnnn.nn gives FORTRAN format F9.2).
- Merging: Files can be merged in a row-row (one over the other) or column-column (one aside the other) sense without any particular user involvement.

G. DESIGN

1. Modular Structure of IDAMAN

Figs. 3 and 4 present the structure of IDAMAN, showing all the subroutines used and the overlay form. The source code of the system is presented in Appendix A.

2. Brief Description of the Subroutines

ANADIS: Analyzes strings used to assign numbers of columns or rows that have to be displayed, retrieved or

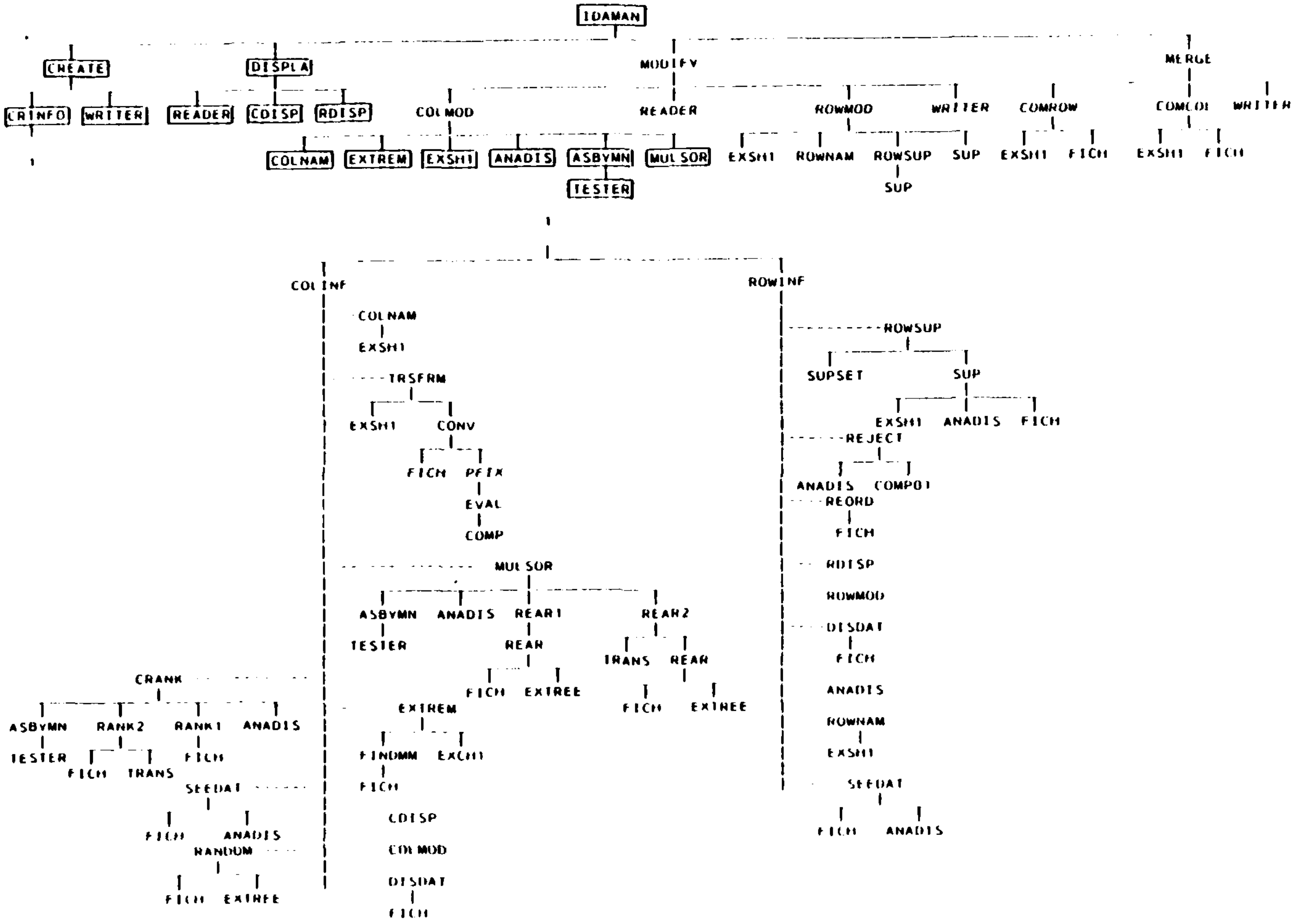

IDAMAN
CREATE
DISPLA
MODIFY
MERGE
CRINFO
WRITER
READER
CDISP
RDISP
COLMOD
READER
ROWMOD
WRITER
COMROW
COMCOL
WRITER
COLNAM
EXTREM
EXSH1
ANADIS
ASBYMN
MULSOR
TESTER
EXSH1
ROWNAM
ROWSUP
SUP
SUP
EXSH1
FICH
EXSH1
FICH
COLINF
ROWINF
COLNAM
EXSH1
TRSFRM
EXSH1
CONV
FICH
PFIX
EVAL
COMP
MULSOR
ASBYMN
ANADIS
REAR1
REAR2
TESTER
REAR
TRANS
REAR
FICH
EXTREE
FICH
EXTREE
EXTREM
FINDMM
EXCH
FICH
CDISP
COLMOD
DISDAT
FICH
CRANK
ASBYMN
RANK2
RANK1
ANADIS
TESTER
FICH
TRANS
FICH
SEEDAT
FICH
ANADIS
RANDOM
FICH
EXTREE
ROWSUP
SUPSET
SUP
EXSH1
ANADIS
FICH
REJECT
ANADIS
COMPO1
REORD
FICH
RDISP
ROWMOD
DISDAT
FICH
ANADIS
ROWNAM
EXSH1
SEEDAT
FICH
ANADIS

Fig. 3. Modular Structure of IDAMAN

```
-------------------------------------------------------------------------------------------------------------------
Level 0 : IDAMAN
-------------------------------------------------------------------------------------------------------------------
Level 1 : CREATE   DISPLA   MERGE    MODIFY
-------------------------------------------------------------------------------------------------------------------
Level 2 : CDISP    COLMOD   COMCOL   COMROW   CRINFO   RDISP    READER   ROWMOD   WRITER
-------------------------------------------------------------------------------------------------------------------
Level 3 : ANADIS   ASBYMN   COLINF   COLNAM   COLSOR   EXSH1    EXTREM   FICH     ROWINF   ROWNAM   ROWSUP
-------------------------------------------------------------------------------------------------------------------
Level 4 : CRANK    COLNAM   COLMOD   DISDAT   MULSOR   RANDOM   RDISP    REJECT   REORD    ROWMOD   SEEDAT   TESTER   TRSFR    FINDMM
-------------------------------------------------------------------------------------------------------------------
Level 5 : CONV     RANK1    RANK2    REAR1    REAR2    SUP      SUPSET   COMP01
-------------------------------------------------------------------------------------------------------------------
Level 6 : EVAL     PFIX     REAR     TRANS
-------------------------------------------------------------------------------------------------------------------
Level 7 : COMP     EXTREE
-------------------------------------------------------------------------------------------------------------------
(Overlay levels and module names in alphabetic order).
```

Fig. 4. Modular Structure of IDAMAN Showing the "Overlay" Levels

rearranged (e.g., 1,4,6:9,3 assigns successive numbers 1 4 6 7 8 9 3)

ASBYMN: Permits the assignment of columns or rows by their mnemonic names.

CDISP: Displays on the screen information about the columns of the header.

COLINF: Presents the selection table of the column information, and calls the appropriate subroutine regarding the information to be assigned by the user.

COLMOD: Modifies the information about the columns.

COLNAM: Assigns mnemonic names to columns.

COMCOL: Merges (combines) two data files and their headers in the column-column (one over the other) sense.

COMP: Computes the values of functions used in the user assigned expressions for transformations.

COMP01: Gets or sets the binary value 0 or 1 in a compressed binary matrix NROW*NCOL for any I's and J's.

COMROW: Merges (combines) two data files and their headers in the row-row (one over the other) sense.

CONV: Converts (analyzes) the expression string assigned for the transformation in an infix notation expression. This string is composed from character defined numbers, letters, delimiters, etc. It is decomposed to the several elements that constitute the infix expression. It calls the subroutines POSTF and EVAL for further processing of the expression. It stores the result of the evaluation on the next rightmost end of the data matrix--creating a new column.

CRANK: Assigns column ranking by column number or column mnemonic names. The assignment of the columns can be done as in the previous paragraph and the same subroutines are used. After the assignment, it calls the corresponding subroutine (RANK1 or RANK2) for ranking of columns.

CREATE: Creates a new header. It opens the sequential file on which information of the created header will be stored and passes control of the program to the subroutine CRINFO. At the end of the creation, it calls WRITER for recording of information, which resides in the main memory, to the opened file in mass storage.

CRINFO: Determines if either column or row information is to be assigned next. It calls the appropriate subroutine COLINF or ROWINF.

DISDAT: Displays any or all user assigned (columns, rows) parts of the data file.

DISPLA: Displays on the CRT the information related to the columns and rows of a previously defined header. It opens the sequential file on which information is stored, calls READER which reads the information and stores it in the main memory and then calls CDISP and RDISP which display information about columns and rows.

EXTREM: Assigns tracing extrema.

EXSH1: Executes internal and address calculation sorts using Shell's method to rank elements in increasing order, modified for address calculations.

FICH: Opens at run time, direct access unformatted or sequential files named 'FORnnn.DAT'.

FINDMM: Evaluates the minimum and maximum value of each column of the data matrix.

FORM: Permits the user to determine the format of the data that will be displayed.

ENSORT: Permits Input/Output modifications or assignments for physical Input and/or Output, and/or Print for Devices or Files within a FORTRAN Program or Subroutine.

EVAL: Evaluates (computes the result) the postfix expression.

MERGE: Merges two existing headers. It opens the two files to be merged and calls COMCOL or COMROW for the merging of the two files in the column-column (one aside the other) sense or row-row (one over the other) sense.

MODIFY: Modifies an existing header. It opens the file on which information is stored, calls READER to pass it from the secondary to the main memory and calls the appropriate subroutine for the modification (COLMOD for column or ROWMOD for row) which actually executes the modification. At the end, it restores the modified information by calling WRITER.

MULSOR: Assigns the guides (column numbers or column mnemonic names) that will be used for the multiple sorting. It calls the subroutine ANADIS which analyzes the string of column numbers and characters ("," and/or ":") assigned by the user via the terminal and isolates the column numbers when the assignment is done by column numbers. If the assignment is done by mnemonic names, it calls the subroutine ASBYMN which stores the names in the appropriate array. In both cases it calls the appropriate subroutine (REAR1 or REAR2) for further processing.

PFIX: Converts the infix expression to the corresponding postfix notation.

RANDOM: Randomizes the data matrix.

RANK: Rearranges the columns of the data matrix according to the column numbers assigned by the user.

RANK1: Rearranges the columns of the data matrix according to the column numbers assigned by the user.

RANK2: Rearranges the columns of the data matrix according to the column mnemonic names. It calls TRANS to transform the column name to the column number.

RDISP: Displays on the screen information about the rows of the header.

READER: Records information regarding the header on a sequential file.

REAR: Multisorts the data matrix. It opens the direct-access file of data, and consequently calls the subroutine EXTREE for the sorting. It rearranges the data according to the final inverted relative addresses without using extra file for temporary storage of the data.

REAR1: Used to continue the process of multi-sorting column numbers by calling the subroutine REAR.

REAR2: It functions like REAR1 for column assignment. It calls TRANS to transform column names to column numbers.

REJECT: Rejects rows not involved in particular computations ("logical suppression"); these rows are not physically suppressed but are just ignored when the calculation is executed. The row to be rejected is given the binary value of zero while the retained row is given the binary value of one. In this way, bit words are converted to decimal. The purpose of the method is to save space and time.

REORD: Reorders internally (without use of a temporary file) the rows of the data file after multi-sorting (via the produced invert relative addresses).

ROWINF: This corresponds to the previous subroutine for rows.

ROWMOD: Modifies information about the rows of the header.

ROWNAM: Assigns row mnemonics.

ROWSUP: Presents the prompts for the row suppression assignment, permits the assignment of the suppression row numbers which are stored in an array, and calls the subroutine SUP that executes the suppression.

SUP: It suppresses the rows of data file assigned by the previous subroutine.

SUPSET: Suppresses the rows of the data matrix that correspond to an assigned set of rows with a common mnemonic name.

SEEDAT: Retrieves user defined data or searches for missing values from the data matrix.

TESTER: Tests if a mnemonic name has been assigned as a column mnemonic name.

TRANS: Transforms a column mnemonic name to a column number.

TRSFRM: Presents the table of the available functions that can be used for transformations and calls the subroutine CONV that calculates the assigned expression.

WRITER: Writes the information regarding the header on a sequential file.

II. METHODS USED FOR THE IMPLEMENTATION

A. SORTING

1. Introduction

A large amount of data may need to be sorted. In order to avoid intermediate manipulations in mass memory, a study of the most efficient method, in terms of timing, has been carried out.

If the data matrix is stored on a direct-access file on a disk and not in central memory (which is the general case), a sort by address calculation avoids intermediate exchanges between central and secondary memory. This results in time burdening of the method [Ref. 5].

When data are preordered, methods using binary trees must be avoided because they tend to "bubble" [Refs. 6 and 7].

When multi-sorting is needed, methods requiring departure boundaries as the integer result of a division by two (dihotomic methods), are inappropriate--giving wrong results [Ref. 8].

Guinier (1980 [Ref. 5] has shown that the manipulation of numbers corresponding to records stored in a random access RK05 disk file under the RT11 operating system considerably increases the elapsed time by a factor of 40 if MacLarren's method is used and more than 80 if Singleton's method is used. It is necessary to transfer the records of the file in an image array of the column to sort and to have the ordered elements in the form of addresses in order to avoid manipulations of the full records during the sort.

2. Binary Tree Sorts (Monkey Puzzle, Hoare's Quicksort)

In the binary tree sorts, elements to be sorted and contained, in a corresponding array, are scanned and placed at the appropriate node of the created binary tree. The relative position of the node on which an element will be placed in the tree is dependent on a comparison of the size of each element with the elements already existing on the tree. The left or right branch is selected, respectively, if the element is larger or smaller.

If the original data are randomized:

- The number of nodes n at a given level 1 is n = 2**1.
- The depth of the resulting binary tree is d = log 2(n+1) - 1.
- The number of comparisons to place the node at level 1 is 1 + 1 and the total number of comparisons is c with d + c1 <= c <= c1 and c1 = SUM 1 = 1 to d-1 {(1 + 1)*2**1}. It can be shown that c is approximately equal to n*log 2(n).

If the original data are preordered:

- The resulting tree appears like a single branch tree and the total number of comparisons is c = 2 + 3 + 4 + ... + n. That is, c = n*(n+1)/2 - 1 which is approximately n*n/2.
- The method tends to "bubble", with the number of comparisons c = (n-1)**2 = n**2 - 2*n + 1--approximately equal to

n*n. An extra array is needed but the elements are not manipulated. Hibbard (1963) [Ref. 9] suggests the use of Shell's method (Shell (1959)) as an alternative method not sensitive to the preordering. In the case of dealing with preordered data, a good strategy is to randomize them with a single pass before sorting and without manipulation of the file records, using instead randomization of the addresses by which the data will be read. This is done by the randomization function of the IDAMAN.

3. Shell's Method

This method uses the principle of interchange by adjacent pairs. In contrast with the bubble sort, it moves list entries at most one position at a time, dividing the original list of n entries into two parts. This method can compare entries that are two positions apart.

This method is insensitive to preordering. But an ambiguity remains in the calculation of m = integer [m/2] and cannot be used for multi-sorting.

4. Singleton's Method

Singleton's method is an extension of Hoare's "Quicksort". Because a tree method gives bad results when the items are presorted or take a constant value, the problem has been changed to an exchange of elements when they are equal to or greater than a temporary value T in one set of values and less than T in the other set. This operation gives a better split of the original set of items.

The median set is generally missing. Therefore, comparison with the temporary value and the median of the values of X(i) is avoided. X([i+j]/2) and X(j) are used for T. This gives a better estimation of the median element than a single value.

In searching for two elements to exchange, the data-almost-sorted X(i) and X(j) are used as boundary values. X(i) <= T <= X(j) and the indexes are compared after performing the exchange.

The lower and the upper sets must have approximately the same size to result in efficient performance.

Speed of the method is greatest for less than 11 items when completing the sort by short sequences using Shell's method of sorting by interchange of adjacent pairs [Refs. 8 and 9].

For N elements, the dimension of the lower and upper sets (IL() and IU()) must be k with N = 2**(k-1) - 1.

5. Evaluation of Selected Sort Methods in Terms of Performance

Comparisons between the "monkey puzzle" binary tree, Shell's and Singleton's sorting methods can be obtained from the following data which represent a mean of ten tests, in order to reduce the effect of the activity of the VAX11.

For randomized data: including address calculation, on VAX11-VMS.

n	Shell	Tree	Singleton
1,000	0.4 s	0.2 s	0.2 s
2,000	1.1 s	0.6 s	0.4 s
3,000	2.1 s	1.0 s	0.6 s
5,000	3.8 s	2.0 s	0.9 s
10,000	7.2 s	4.2 s	1.9 s

For preordered data: including address calculation, on VAX11-VMS.

n	Shell	Tree	Singleton
1,000	0.2 s	15.1 s	0.1 s
2,000	0.5 s	44.8 s	0.2 s
3,000	1.1 s	117.8 s (1.9 mn)	0.3 s
5,000	1.9 s	310.1 s (5.1 mn)	0.5 s
10,000	4.8 s	1,072.7 s (17.9 mn)	1.0 s

When data are randomized between Tree's and Shell's methods, Berztiss (1975) [Ref. 10] gives a ratio of 1.3 in favor of Tree's on an Algol version running on a CDC 1604A. The ratio of the presented results is greater than this value if the data are randomized. Between Tree's and Singleton's methods, the ratio is 2 and the velocity is 5 times better on theVAX11/780 than the results given by Singleton on the Burroughs B5500.

When data are preordered, degeneration of Tree's method on a bubble appears clearly. The results of Shell's and Singleton's methods are stable and twice as good.

Singleton's method is 20 times faster on the VAX11/780 under VMS than on a PDP11/05 under RT11. Guinier (1980) gives 4 s. for 1000 items [Ref. 10].

The methods selected by IDAMAN are Shell's method for original single sorting and the binary tree sort for multi-sorting [Ref. 11].

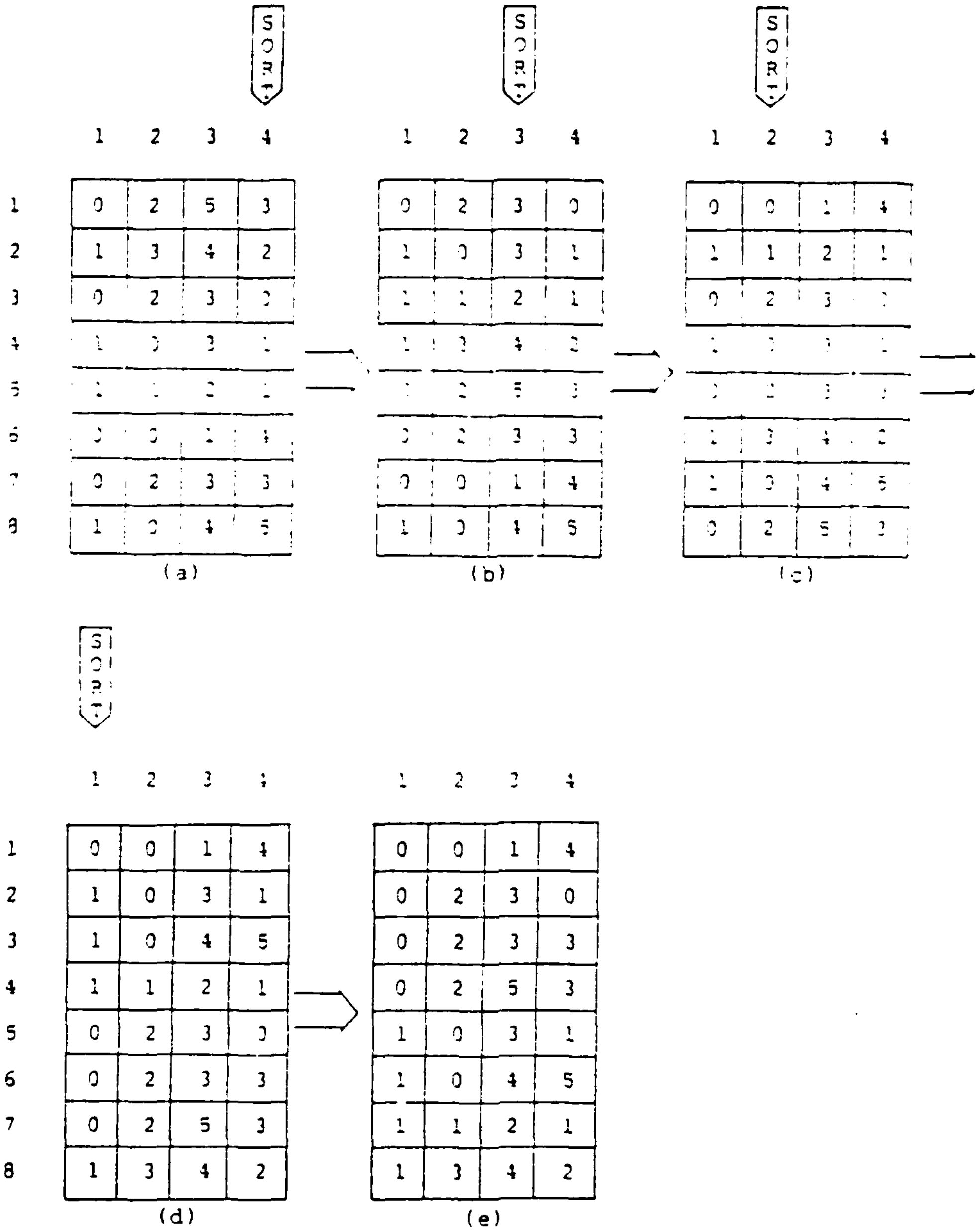
B. MULTIPLE SORTING

1. Introduction

A system, via multiple sorting, executes a repeated sorting of the data matrix according to successive column numbers, or column names assigned by the user via the terminal.

This function permits nested sorting of the data matrix. The concept of nested sorting is clearly illustrated by the following example. Suppose we have the data matrix as in Fig. 5(a) and we repeatedly sort it according to the columns 4, 3, 2 and 1. The resultant matrix is shown in Fig. 5(e). In this matrix the first four rows of column 1 have the same value. The result of the repeated sorting is that the corresponding first four rows of column 2 (which is nested to column 1 according to the assigned sorting sequence 4, 3, 2, 1) are sorted. Likewise, in colume 3 the rows 2, 3 and 4 have the same value and the result of the repeated sorting is that in column 4 the avlues in rows 2, 3 and 4 are sorted.

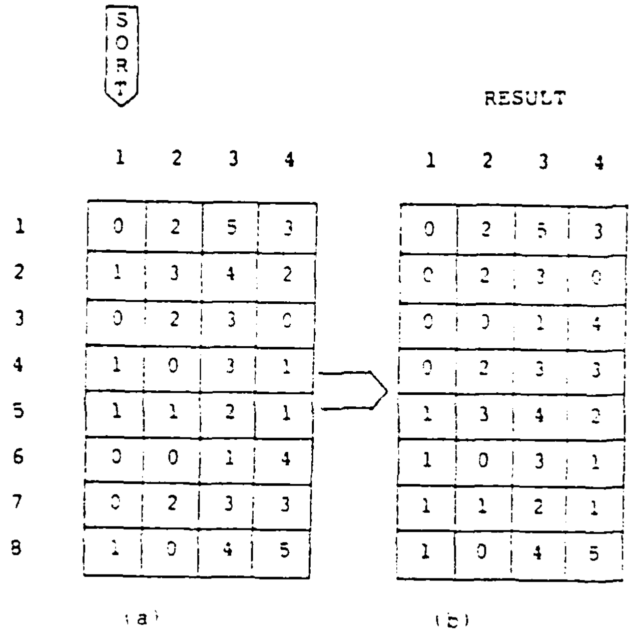
The concept is further clarified in Fig. 6 where direct sorting of the data matrix is presented based on column 1 (the last column of the sequence of the repeated sorting).


SORT
SORT
SORT
SORT
1 2 3 4
1 0 2 5 3
2 1 3 4 2
3 0 2 3 0
4 1 0 3 1
5 1 1 2 1
6 0 0 1 4
7 0 2 3 3
8 1 0 4 5
0 2 3 0
1 0 3 1
1 1 2 1
1 3 4 2
0 2 5 3
0 2 3 3
0 0 1 4
1 0 4 5
0 0 1 4
1 1 2 1
0 2 3 0
1 0 3 1
0 2 3 3
1 3 4 2
1 0 4 5
0 2 5 3
0 0 1 4
1 0 3 1
1 0 4 5
1 1 2 1
0 2 3 0
0 2 3 3
0 2 5 3
1 3 4 2
0 0 1 4
0 2 3 0
0 2 3 3
0 2 5 3
1 0 3 1
1 0 4 5
1 1 2 1
1 3 4 2

(a) (b) (c)

(d) (e)

Fig. 5. Nested Sorting of the Data Matrix



	1	2	3	4
1	0	2	5	3
2	1	3	4	2
3	0	2	3	0
4	1	0	3	1
5	1	1	2	1
6	0	0	1	4
7	0	2	3	3
8	1	0	4	5

(a)

1	2	3	4
0	2	5	3
0	2	3	0
0	0	1	4
0	2	3	3
1	3	4	2
1	0	3	1
1	1	2	1
1	0	4	5

(b)

Fig. 6. Single Sorting

The result is a matrix sorted in column 1. But the nested sorting of the previous case does not exist.

This type of sorting is useful in cases where some variables of the active data matrix are not quantitative (continuous) but descriptive (discrete) and are attached to classes or subclasses of data. Successive sorting involving such variables organizes the matrix in a corresponding order if the collection of data is not presorted by function of the classes or subclasses (Fig. 7).

1	1	3.2	0	0	8.1
0	0	8.1	0	0	12.0
0	0	12.0	0	1	13.4
1	1	3.1	0	2	1.5
0	2	1.5	0	2	2.6
1	0	0.7	1	0	0.7
0	1	13.4	1	1	3.1
0	2	2.6	1	1	3.2
1	2	13.2	1	2	13.2
"Descriptive" data		"Quantitative" data	"Descriptive" data		"Quantitative" data
Before sorting			After sorting		

Fig. 7. Set of Data Containing Two Classes and Three Subclasses

2. Principle

Operation of the repeated sorting of the data matrix is executed in two phases. The first phase sorts the data matrix consecutively according to the assigned sequence of column numbers or names, while the second rearranges the rows of the initial matrix according to the result of the repeated sortings.

a. Phase One

The virtual process by which the repeated sorting takes place is the following (Fig. 8):

(1) Read the first column according to which the data must be sorted. Store the data of this column in an array.

(2) Sort the array and get the invert relative addresses in an array (NO).

(3) Read the inverted relative addresses. (This means that the rows are rearranged before the next sorting.)

(4) Repeat until all the sorting guides have been used. In order to avoid rearranging and thus to save process time and space utilized, the following method is used (Fig. 9).

A one dimension array (NOR) is used to store the relative addresses in which the data file will be read and remains unchanged during the whole process.

(1) Initialize the NOR.

(2) Read the first column in which storing will take place.

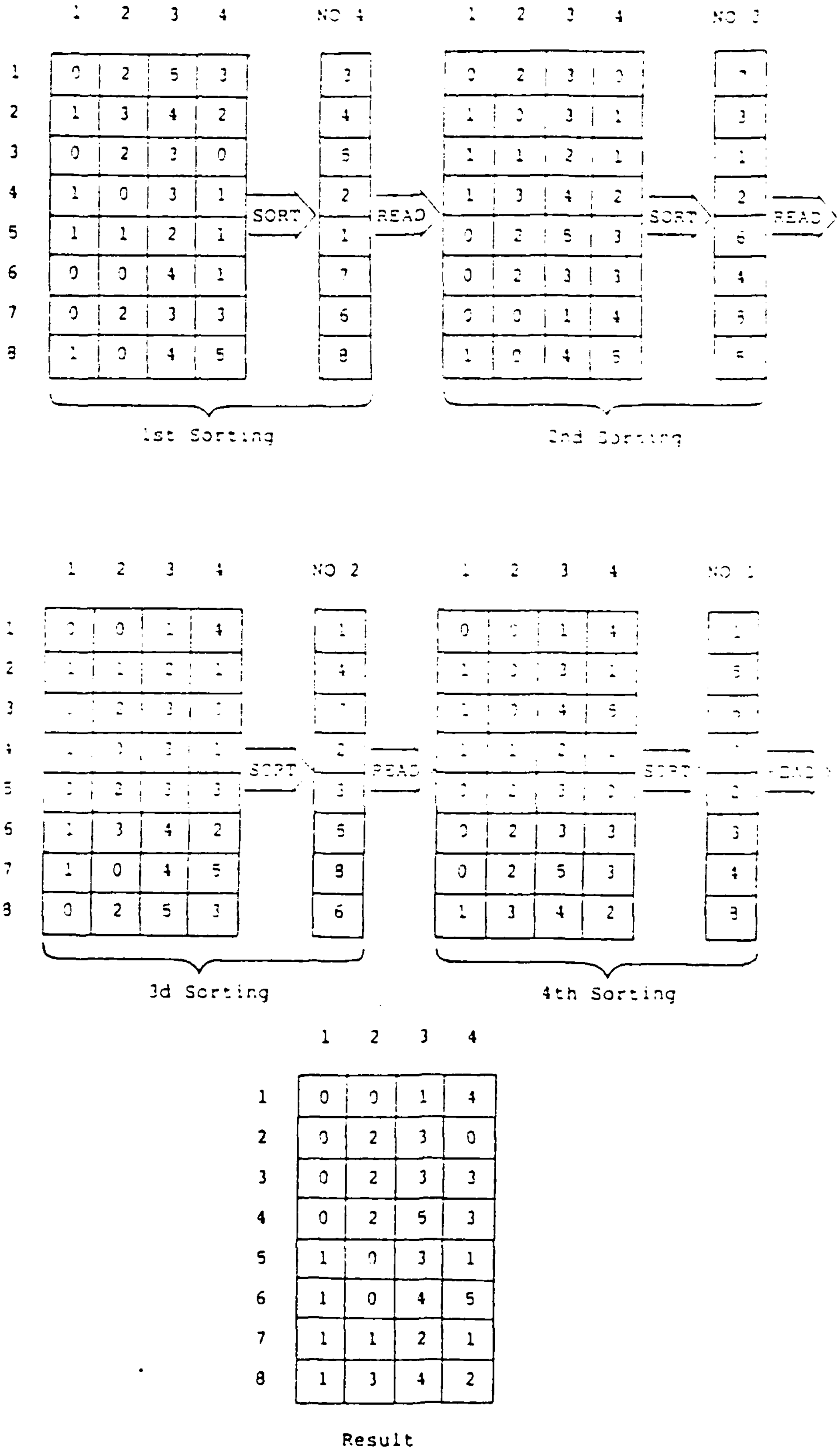
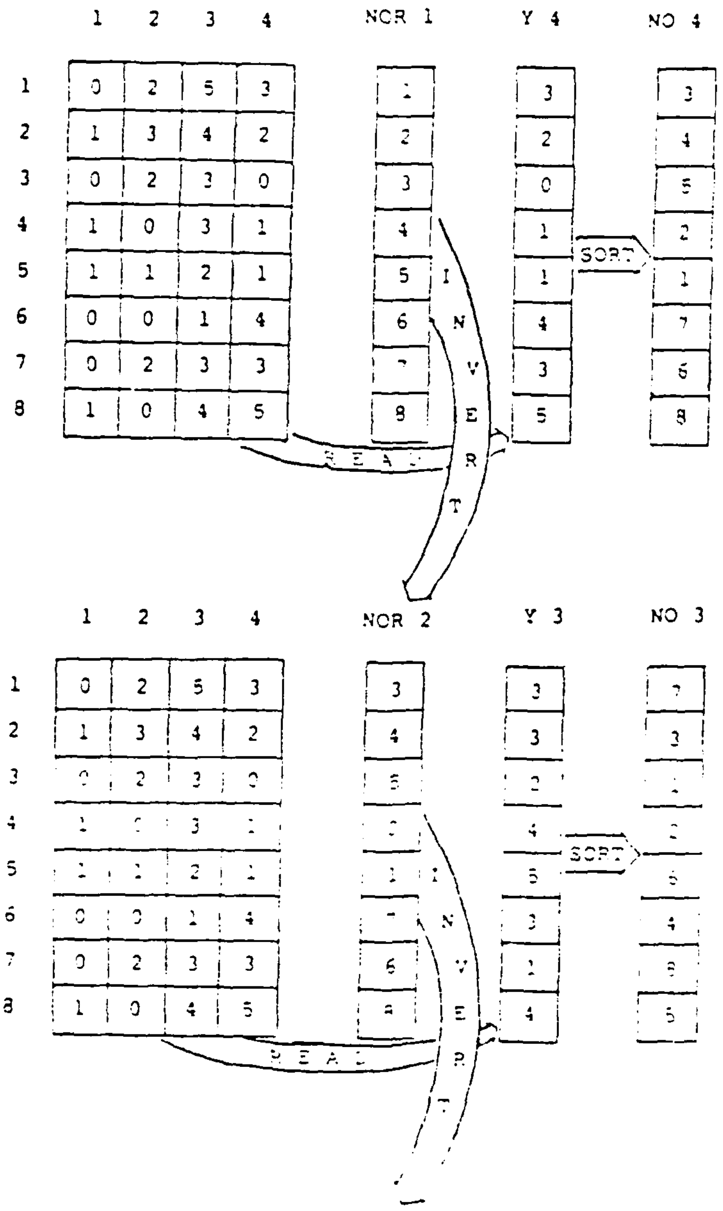

SORT
READ
NO 4
NO 3
NO 2
NO 1
1st Sorting
2nd Sorting
3d Sorting
4th Sorting
Result

Fig. 8. Virtual Process of Repeating Sorting


1 2 3 4
NOR 1
Y 4
NO 4
SORT
READ
INVERT
1 2 3 4
NOR 2
Y 3
NO 3
SORT
READ
INVERT

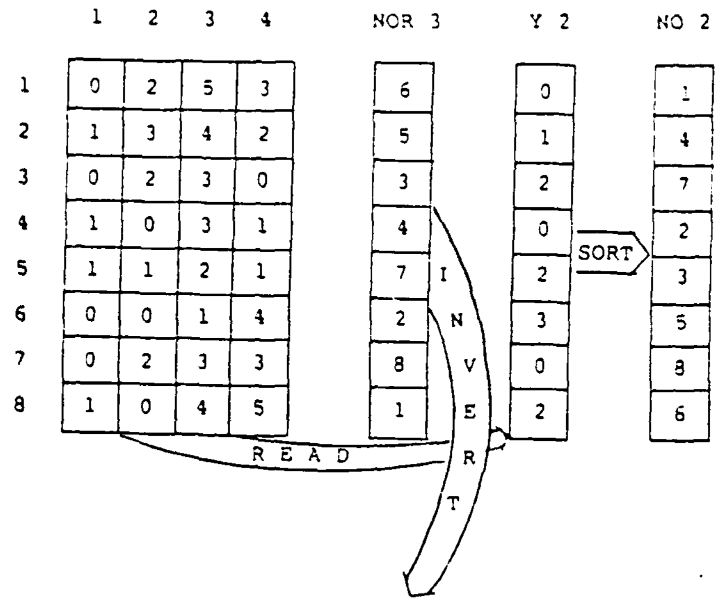
STEP 1

a. Initiate NOR 1.
b. Read column 4 according NOR 1.
c. Sort Y 4.

STEP 2

a. Invert NOR 1 according NO 4.
b. Read Y 3 according NOR 2.
c. Sort Y 3.

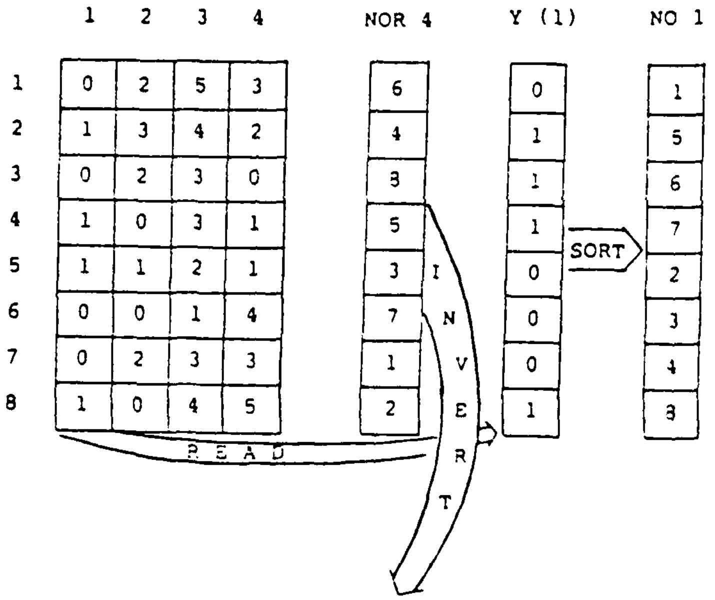
Fig. 9. Real Process of Repeating Sorting



	1	2	3	4	NOR 3	Y 2	NO 2
1	0	2	5	3	6	0	1
2	1	3	4	2	5	1	4
3	0	2	3	0	3	2	7
4	1	0	3	1	4	0	2
5	1	1	2	1	7	2	3
6	0	0	1	4	2	3	5
7	0	2	3	3	8	0	8
8	1	0	4	5	1	2	6

STEP 3

a. Invert NOR 2
according NO 3.
b. Read Y 2 according NOR3.
c. Sort Y 2.



	1	2	3	4	NOR 4	Y (1)	NO 1
1	0	2	5	3	6	0	1
2	1	3	4	2	4	1	5
3	0	2	3	0	8	1	6
4	1	0	3	1	5	1	7
5	1	1	2	1	3	0	2
6	0	0	1	4	7	0	3
7	0	2	3	3	1	0	4
8	1	0	4	5	2	1	8

STEP 4

a. Invert NOR 3
according NO 2.
b. Read Y 1 according NOR 4.
c. Sort Y 1.

Fig. 9. (CONTINUED)

(3) Sort the column and get the inverted relative addresses (NO).

(4) Invert the NOR according the NO.

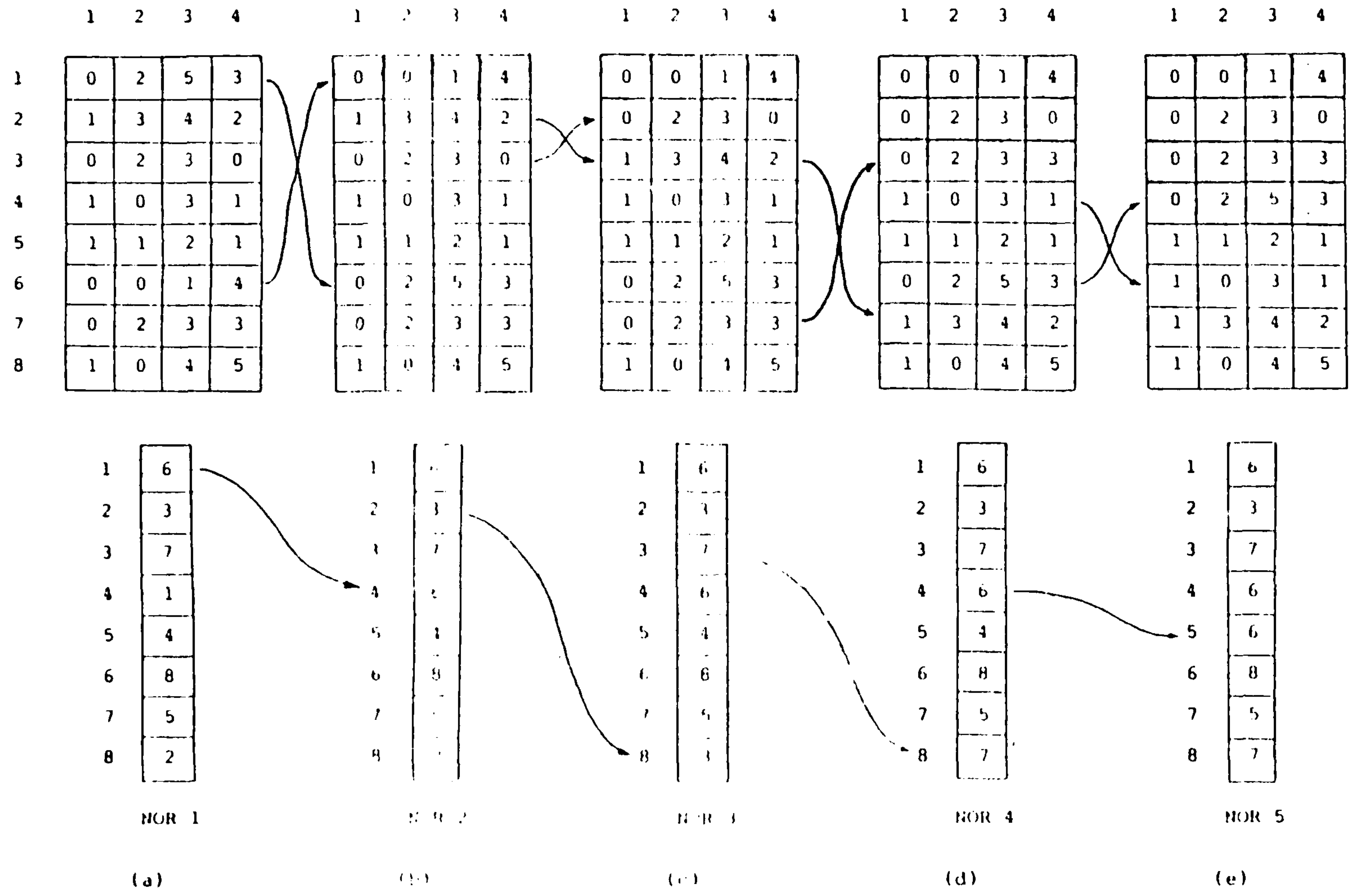
(5) Repeat until all sorting guides have been used.

b. Phase Two

The second phase is the rearranging of the data matrix according to the last inverted relative addresses, using the same external file. A pictorial presentation of the algorithm used is shown in Fig. 10.

The reasoning of this algorithm is the following: In Fig. 10 the array containing the inverted relative addresses is noted as NOR. The first pointer of NOR1 is pointing to row number 6. This means that row 6 must move to the first relative address of the matrix. The contents of this address however must be saved. For this reason a mutual exchange between the first and sixth row of the matrix is executed. In this way the first row already contains the appropriate values. A consequence of this exchange is that the first row which should move to the fourth row of the matrix, is now moved to the sixth position. The fourth relative address therefore of NOR must change from 1 to 6. The algorithm searches the NOR from 1 to 8 until it locates the relative address 1, which exchanges with the relative address 6.

Similarly, in the second step, column 3 of the matrix must move to the second relative address. Therefore a mutual exchange between rows 2 and 3 takes place. In the NOR now, a search takes place until the relative address 2



(a)

	1	2	3	4
1	0	2	5	3
2	1	3	4	2
3	0	2	3	0
4	1	0	3	1
5	1	1	2	1
6	0	0	1	4
7	0	2	3	3
8	1	0	4	5

	NOR 1
1	6
2	3
3	7
4	1
5	4
6	8
7	5
8	2

(b)

	1	2	3	4
	0	0	1	4
	1	3	4	2
	0	2	3	0
	1	0	3	1
	1	1	2	1
	0	2	5	3
	0	2	3	3
	1	0	4	5

	NOR 2
1	[illegible]
2	3
3	7
4	[illegible]
5	4
6	8
7	[illegible]
8	[illegible]

(c)

	1	2	3	4
	0	0	1	4
	0	2	3	0
	1	3	4	2
	1	0	3	1
	1	1	2	1
	0	2	5	3
	0	2	3	3
	1	0	4	5

	NOR 3
1	6
2	3
3	7
4	6
5	4
6	8
7	5
8	[illegible]

(d)

	1	2	3	4
	0	0	1	4
	0	2	3	0
	0	2	3	3
	1	0	3	1
	1	1	2	1
	0	2	5	3
	1	3	4	2
	1	0	4	5

	NOR 4
1	6
2	3
3	7
4	6
5	4
6	8
7	5
8	7

(e)

	1	2	3	4
	0	0	1	4
	0	2	3	0
	0	2	3	3
	0	2	5	3
	1	1	2	1
	1	0	3	1
	1	3	4	2
	1	0	4	5

	NOR 5
1	6
2	3
3	7
4	6
5	6
6	8
7	5
8	7

Fig. 10. Rearranging of the Rows of Data Matrix According to the Inverse Relative Addresses

is located. This occurs in the eighth position and therefore the number 3 is placed at the eighth position. The same process is repeated until all the rows of the data matrix are rearranged according to the inverted relative addresses (NOR).

The choice of sorting method must be made so that the expected result of the repeated sorting is achieved. In this case each set of data to be sorted (the data of a column) is not independent but rather is bound with the data of the other columns of the matrix. Whenever a duplication of values appears in the column to be sorted, these values must not be treated equally since their bound with the values of the neighboring columns differentiates them. In order to keep the order of the neighboring columns unaltered, the equivaled entries of the matrix must maintain their order after sorting. The following example clarifies this concept. Suppose that we have the matrix (a) and we repeatedly sort it according to

$$\begin{array}{ccc}
\begin{array}{ccc} 0 & 1 & 3 \\ 0 & 2 & 2 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \\ 1 & 0 & 2 \end{array} &
\begin{array}{ccc} 1 & 1 & 1 \\ 0 & 2 & 2 \\ 0 & 1 & 2 \\ 1 & 0 & 2 \\ 0 & 1 & 3 \end{array} &
\begin{array}{ccc} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 1 & 2 \\ 1 & 1 & 3 \\ 0 & 2 & 2 \end{array} &
\begin{array}{ccc} 0 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 2 & 2 \\ 1 & 0 & 2 \\ 1 & 1 & 3 \end{array} \\
\text{(a)} & \text{(b)} & \text{(c)} & \text{(d)}
\end{array}$$

columns 3, 2, 1; considering the sequence of the duplicate values in each column, The resultant matrix in Fig. 11 has the desired property. Repeating the same sorting but not considering the sequence of the duplicate values, the result is different, as illustrated below:

$$\begin{array}{ccccccccccccccc}
0 & 1 & 3 & & 1 & 1 & 1 & & 1 & 0 & 2 & & 0 & 1 & 3 \\
0 & 2 & 2 & & 1 & 0 & 2 & & 0 & 1 & 2 & & 0 & 2 & 2 \\
0 & 1 & 2 & & 0 & 2 & 2 & & 1 & 1 & 1 & & 0 & 1 & 2 \\
1 & 1 & 1 & & 0 & 1 & 2 & & 0 & 1 & 3 & & 1 & 1 & 1 \\
1 & 0 & 2 & & 0 & 1 & 3 & & 0 & 2 & 2 & & 1 & 0 & 2 \\
 & \text{(a)} & & & & \text{(b)} & & & & \text{(c)} & & & & \text{(d)} &
\end{array}$$

The above consideration imposes limitations in the selection of the sorting method to be used. Considering only the speed and the memory space consumed in the method selecting process is not sufficient. The method selected must also maintain the existing order of the equivaled elements. Such a method is the tree sort.

C. TRANSFORMATIONS

1. Introduction

A more valid analysis of row statistical data usually requires a series of transformations in order to change the scale of the measurements.

Initially, a test of normality of the data is executed to confirm if a parametric method of statistical analysis can be used. If the data to be analyzed are non-normal, normality can be obtained by applying appropriate transformations on them [Ref. 12].

A transformation also can be used to equalize the variances of the sample.

a. Fundamental Transformations

The most common transformations used for the purposes mentioned above are the following.

(1) Logarithmic Transformation (Neperian and Common).

$$X' = \mathrm{Log}(X) \quad \text{for} \quad X > 0 \quad \text{or}$$

$$X' = \mathrm{Log}\,(X+C) \quad \text{for} \quad (X+C) > 0$$

If considerable heterogeneity in numbers is present, the variance is often found to be correlated with the mean level on a square root scale, and may only be stabilized if transformation is made to the logarithmic scale [Ref. 13].

(2) The Inverse Sine or Arcsine Transformation.

$$X' = \mathrm{Sin}^{-1}(\mathrm{SQRT}(X)) = \mathrm{Arcsin}(\mathrm{SQRT}(X)) \quad \text{if} \quad 0. <= X <= 1.$$

This transformation permits an equalization of variances when the data are proportions or ratios.

(3) The Square Root Transformation.

$$X' = \mathrm{SQRT}(X) \quad \text{if} \quad X >= 0$$

This transformation is used when the variances of the samples are approximately proportional to their mean, or when the data follow the Poisson distribution.

(4) The Inverse Transformation.

$$X' = 1/X \quad \text{if} \quad X \text{ is different than zero}$$

This transformation is used to equalize the variances whenever they are approximately proportional to $\bar{x}^4$, $\bar{x}^5$, ..., ($\bar{x}$: mean).

(5) The Inverse Hyperbolic Sine or Arcsine Hyperbolic Transformation.

$$X' = \text{Sinh}^{-1}(\text{SQRT}(X)) = \text{Arcsine}(\text{SQRT}(X))$$

where

$$\text{Arcsine } (Z) = \text{Log } (|Z| + Z*Z + 1)$$

(6) Exponentiation at Tth Transformation.

$$X' = X^T$$

This transformation is suggested by Snedecor and Cohran (1967) for variance analysis when there is no additivity of variances [Ref. 14].

b. Complex Transformations

Although the above transformations are the most commonly used, the system provides an ability for any kind of transformation assigned by the user in the form of an expression via the keyboard. In this way, not only is a transformation of the column obtained, but calculations at run time involving intrinsic functions, arithmetic operators, constants (integers or reals) and data contained in the data matrix, can be performed according to user assigned mathematical expression. Examples of such expressions are:

- SQRT[7]/[5]
- (LOG[10]/2)^2
- [9] + 3.14
- SIN[5] - (-COS[3])^3

The number of the column must be enclosed in brackets in order to be identified, while the symbol "^" is used for the operation of exponentiation to avoid use of two characters as an operator.

The value of each assigned transformation is considered as a new column or parameter and is placed at the rightmost of the columns--increasing by one the number of columns of the data matrix.

2. Principle

a. Breakdown of the Expression in Parts

The inserted expression is read as a unique character string and is broken down into the following kinds of characters or character strings:

- Functions and corresponding column number (e.g., TAN[2])
- Column number (e.g., [5])

- Real number or integer
- All the above with negative signs
- Operators (+, -, *, /, ^)
- "("
- ")"

For example, the expression (COS[3]+2.7)/(-[5])^2-8 is broken down as follows:

```
(
COS[3]
+
2.7
)
/
(
-[5]
^
2
-
8
```

In this format the elements constitute an infix notation format. After the breakdown, the elements are stored in a two dimensional one character array INF. Three one dimension one character arrays FUN, NUM and CONS are used for temporary storage of the functions with their corresponding column number arguments, column numbers, and constants. Three flags--TEST, MARK and INDEX--are used to signal the existence of a negative value. This negative value is identified whenever a "-" character follows an "(" character. The variable BR is used to temporarily store a "[" or "]" character which, in combination with the flags, permits a column number to indicate that the datum of the value will be used in the expression or that the value is an argument of a function.

b. Conversion from Infix to Postfix Notation

The method used for evaluation of the expression requires the transformation of the infix expression (which is the normal form used by mathematics) into postfix notation.

A postfix notation is a rearrangement of the characters of the infix expression to remove the ambiguity of which operator is performed first--using parentheses to indicate the priority of the operators. For example, the expression a+b*c is ambiguous since it is not clear if the addition or the multiplication has to be performed first. This ambiguity requires the use of parentheses to specify the operation to be performed first.

In a postfix notation, all the information required for the evaluation of the expression is included in it. Moreover, the parentheses are not required any more.

Suppose that we have to evaluate the expression a+b*c+(d/e-f). For its evaluation two facts are used to determine the sequence of the operations. First is the existence of the parentheses which define which expression has to be performed first. The result has to be used as a new value for the remaining evaluation of the expression. Second, the precedence of the operators. Operation of the exponentiation is to be performed first, then the multiplication and division and finally the addition and subtraction. Therefore, in the expression in parentheses, the division has to be performed before the subtraction while in the part outside of the parentheses, multiplication is to be performed before addition.

The same expression in postfix notation is: abc*+de/f-+. This form does not need the conventions mentioned above, in order to be evaluated. Each operator simply uses the two operands preceding it, in a left to right sequence.

The method used for the conversion from infix to postfix is the push-down stack for the storage of the operators and parentheses of the expression.

In general terms, the algorithm used is as follows: Each character of the expression is examined. If it is an operand it is placed directly in the created postfix form. If it is an operand, then if the topmost character of the stack is an operator, priority comparison takes place. If the operator of the stack has equal or higher priority, it is popped, and placed in the postfix expression, while the other is pushed into the stack. If the priority of the stack operator is lower, the other is pushed into the stack. If the character of the infix expression is an "(", it is pushed into the stack, while if it is a ")", the topmost operator is popped and placed in the postfix and the two parentheses are ignored. Fig. 11 gives a pictorial view of the algorithm described.

In order for the procedure to be included in the transformations of the developed software package, the following considerations must be taken.

The operands can be positive or negative:

- Functions, e.g., SQRT[7] (square root of the column's seven value)
- Column number enclosed in brackets, e.g., [3] (the value of column three)
- Numerical values (real or integers).

The above considerations require the statement of the expression in character form. In this way, the several kinds of operands can be distinguished and treated appropriately. The produced postfix expression is stored in a two dimensional, one character array.

Suppose that we have the infix expression:

`(COS[3]+2.7)/(-[5]^2-8`

The resultant postfix expression is illustrated in Fig. 12 as it appears stored in the array POST.

Postfix expression	Stack	Infix expression
		a+b*c+(d/e-f)
a		+b*c+(d/e-f)
a	+	b*c+(d/e-f)
ab	+	*c+(d/e-f)
ab	+ *	c+(d/e-f)
abc	+ *	+(d/e-f)
abc*	+	+(d/e-f)
abc*+	+	(d/e-f)
abc*+	(+	d/e-f)

Fig. 11. Infix to Postfix Conversion

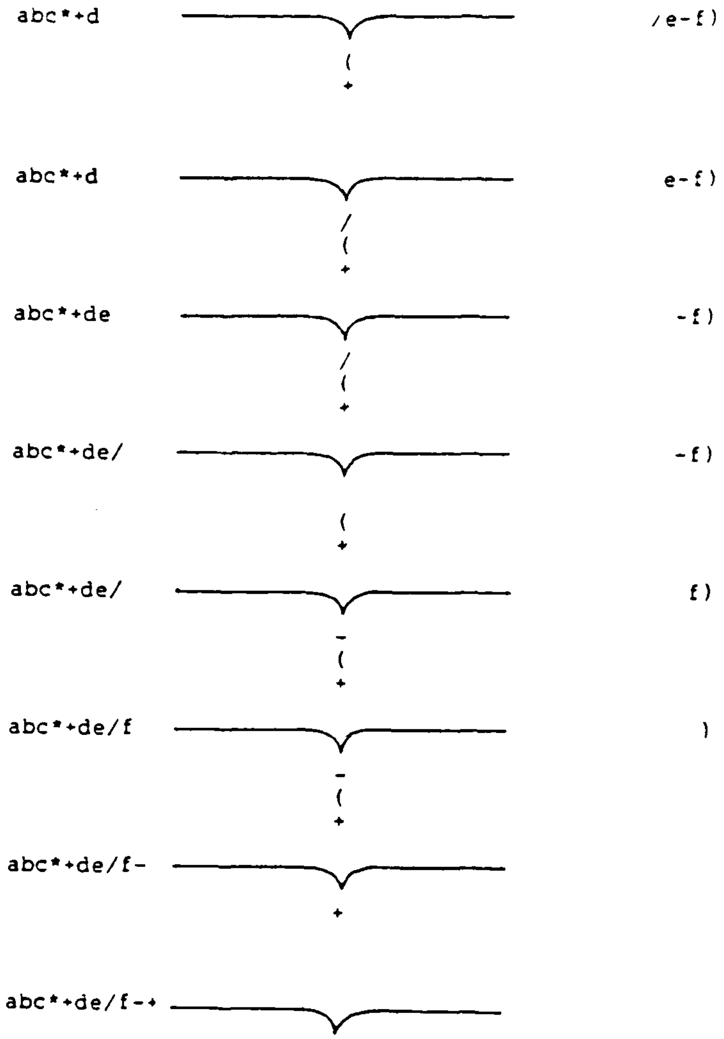

abc*+d
/e-f)
(
+
abc*+d
e-f)
/
(
+
abc*+de
-f)
/
(
+
abc*+de/
-f)
(
+
abc*+de/
f)
-
(
+
abc*+de/f
)
-
(
+
abc*+de/f-
+
abc*+de/f-+

Fig. 11. (CONTINUED)

```
C   O   S   [   3   ]
2   .   7
+
-   [   5   ]
2
^
/
8
-
```

Fig. 12. Representation of the Postfix Expression

c. Evaluation of the Postfix Expression

As previously mentioned, evaluation of the expression after its transformation into postfix notation has been simplified.

The procedure used is as follows:

a) Scan the postfix expression and push into the stack the encountered operands until an operator is encountered.

b) Pop the two topmost operands of the stack, apply the operator to them and push the result into the stack.

c) Repeat until the end of the postfix expression. The example given in Fig. 13 illustrates the concept.

To implement the above procedure, the "-" sign of a negative number has to be considered, and be distinguished from a "-" operator. This task is served by representation of the postfix expression as shown in Fig. 12.

STACK	OPERATION			POSTFIX
	OP1	OPERATOR	OP2	
				3,5,2,*,+,10,2,/,1,-,+
3				5,2,*,+,10,2,/,1,-,+
3,5				2,*,+,10,2,/,1,_,+
3,5,2				*,+,10,2,/,1,_,+
	5	*	2	
3,10				+,10,2,/,1,-,+
	3	+	10	
13				10,2,/,1,-,+
13,10				2,/,1,-,+
13,10,2				/,1,-,+
	10	/	2	
13,5				1,-,+
13,5,1				-,+
	5	-	1	
13,4				+
	13	+	4	
17				

Fig. 13. Evaluation of a Postfix Expression Using the Expression of the Previous Example: a+b*c+(d/e-f)

For each line of the array POST:

- If the first character is a letter, or number, or "[", the element is treated as an operand.
- If the first character is "+", "*", "/", "`", it is treated as an operator.
- If, however, the first character is "-", then the second character is examined.
  - If it is a letter, number or "[", the element is treated as a negative operand.
  - If it is a " ", then it is the minus operator.

Each of the three kinds of operands is identified, stored in a one dimension array and evaluated appropriately. The numeric characters must be decoded before an arithmetic operation is applied to them.

D. RANDOMIZATION

1. Introduction

The system provides a randomization ability of the data, needed in the following cases:

- Many observations (data) are collected as a function of time. In order to avoid any correlation influencing the way this data is collected, we can randomly burst the order of the rows of the data matrix.
- When multiple sorting is desired and some of the columns are already sorted. This fact will have negative results in the performance of the sorting method used by the

system (tree-sort) to obtain the multiple sorting (see Section II.B). In cases where the amount of data is large, it is preferable for randomization of the sorted columns to take place before multiple sorting.

2. Principle

The randomization of the data is obtained by reading it as they are ordered and rewriting it according to a sequence of random numbers produced by the function RANDAN.

The principle used is a congruential method of segmenting the modular arithmetic, described by K.D. Senne (1974) and the algorithms are given by D. Guinier (1984) [Ref. 3].

For all m bit generators, a sequence of integers L(n) in the interval {0,2**m-1} is calculated. The desired real variables Xn are obtained from the L(n)'s. The next sequence of the L(n+1)'s are calculated by dividing the L(n)'s by 2**m by the formula: L(n+1) = (A*L(n)+B) mod 2**m.

M. Abramowitz and I.A. Stegun (1965) have proven many successful combinations for A, B and m, e.g., A = R*2**S+/-1, where S is equal or greater than 2 and B = 0.

The multiplier R is selected so that the number of significant bits in R*2**S+/-1 is approximately m. L(n+1) is the next cycle. There is no correlation between adjacent numbers: L's, (L1,L2,L3,...,Lq-1,Lq).

If S = 3, let: A = 8*R-1; or any m bits A in which the last three bits equal 1.

The L(n)'s can be segmented into q integer parts by the arithmetic expression for any m bit number:

```
  L(n) = f(L1,L2,L3,...,Lq-1,Lq), that is:
  L(n) = L1*2**(m*(q-1)/q)
           + L2*2**(m*(q-2)/q)
           + ...
           + Lq*2**0
```

and A can be segmented in the same way:

```
  A = f(A1,A2,A3,...,Aq-1,Aq), that is:
  A = A1*2***m*(q-1)/q)
        + A2*2**(m*(q-2)/q)
        + ...
        + Aq*2**0
```

Set L(n+1) = A*L(n)+B, that is, if B = 0: L(n+1) = A*L(n):

```
  A*(n) = (A1    *L1          )*2**(2*m*(q-1)/q)
        + (A1    *L2+A2*L1    )*2**(2*m*(q-2)/q)
        +...
        + (Aq-1)*Lq+Aq*Lq-1)*2**(2*m       /q)
        + Aq     *Lq           *2** 0
```

The previous operations are performed in Part I of Algorithm I:

<u>Algorithm I</u>

```
  c=0
  k=q
  DO WHILE (k.GT.0)
     i=0
     p=0
```

PART 1: Calculation of products A*L(n):

```
     For any L(n)
     DO WHILE (i.LE.q-k)
        p = p+Ai+k*L(n)q-1
        i = i+1
     END DO
```

PART 2: Actualization of the L(n)'s in L(n+1)'s, by performing the operation modulo 2.**[m/q] :

```
     p         = p+c
     c         = p  /(2.**[m/q])
     L(n+1)i = p-c*(2.**[m/q])
     k = k-1
   END DO
```

After we take the modulus relative to 2**m, for the next cycle of Ln's, the expression is given by Part 2 of Algorithm I.

When we take the modulus 2.**m of the previous products A*L(n), it gives:

```
L(n+1)={(A1*Lq+A2*Lq-1+...+Aq*L1)mod2**(m/q)}2**((2*m/q)*[q/2])
      +(A2*Lq+A3*Lq-1+...+Aq*L2)*2**(((2*m/q)*([q/2-2))
      +...+Aq*Lq*2**0  and,
```

- the second half of the last term is: L(n+1)q,
- the remainder of the last term + the second half of the next to the last term is: L(n+1)q-1,
- the remainder of the next to the last + the second half of the term just before the next to the last term is: L(n+1)q-2, etc.

Ak and Lk are defined for k = 1 to q pieces at the beginning of the sequence.

The maximum precision requires 2*m/q bits for the update plus carry over bits to add q numbers of length 2*m/q and a number of length m/q, i.e.,

p = [log2{q**((2*m/q)-1**2)+2**(m/q)-1}]+1 bits.

The number q of possible pieces for a 36 bit pseudo-random number (m = 36) is: 2, 3, 6, 9, 12, 18 or 36. We take q = 6 for a 15 bit positive integer INTEGER*2, up to 2**15-1, i.e., 32767. The cycle length of the generator has been tested greater than 10**7.

The sequence of the x's is obtained from the following algorithm.

Algorithm II

```
i=0
x=0
DO WHILE (i.LT.q)
   i=i+1
   x=x+L(n+1)i*2.**(-[m/q]*i)
END DO
```

and the next L(n)'s will be the actual L(n+1)'s for the next sequence. The initial sequence is forced into a flag if it is different than zero (IND).

III. DIRECTIONS FOR USE OF THE IDAMAN

A. INTRODUCTION

The system IDAMAN is an interactive and conversational system intended to be used by any user regardless of his/her level of programming, and without any specific knowledge or training requirements. For this reason an effort was made for the system to be self-directed via meaningful screen displayed prompts.

The IDAMAN operates through five environments of operation. The transition from one environment to another is obtained by the carriage return (<CR>). In each environment appropriate prompts and menu-like tables are displayed directing the use of the system. The five environments are the following:

- MODES OF OPERATION
- COLUMN INFORMATION
- ROW INFORMATION
- COLUMN MODIFICATION
- ROW MODIFICATION

B. MANAGING DATA FILES

The data is stored in direct-access unformatted files in mass storage. These files have names of the form: FORxxx.DAT; where xxx is a key number for the particular file. During the manipulation of the data, the system requests the key number of an existing file on which the data to be manipulated is stored or a new one on which the manipulated data will be stored. This request is done by the prompt:

```
Assign the 'OLD' direct access file:
Key(xxx) For File FORxxx.DAT number y, (nnn=069):
```

and the user has to assign a numeric or alphanumeric three digit key. The number y indicates the logical unit number used for this file (in the example given by the prompt, the key number is 069).

C. USER ERROR PREVENTION

The program has been designed in such a way that the most common user errors can be prevented by an error message reaction of the program instead of a run time compiler error. After the appearance of the message, the program shifts to the appropriate position for reassignment of the erroneously given information.

D. SELECTING MODE OF OPERATION

Access of the first environment is obtained by the execution of the program and permits the selection of one of the available modes. The selection of the desired mode is obtained by assigning the corresponding number of the mode as it appears in the following table:

```
           AVAILABLE MODES

Creation of new header            :1
Display of existing header        :2
Modification of existing header   :3
Merging of two existing headers   :4
Answer                            :
```

E. THE FUNCTION OF MODES AND HOW TO DEAL WITH THEM

1. Creating a New Header

The creation of a new header is done by selecting mode 1. Upon entering this mode the following prompt appears on the screen:

```
Name of header's file :
```

and requests the assignment of any 24 character string which will be used as the file name of the sequential file in which information of the created header will be stored. After file name assignment, the prompt:

```
Column information: 1
Row information   : 2
Answer            :
```

requests the environment to be used next. By selecting 1, the program enters the environment in which information regarding the columns of the header will be assigned, while 2 permits assignment of information related to the rows of the header.

2. Assigning Information for the Columns

Upon entering the environment for column information, the following menu-like table appears on the screen:

```
SELECTION TABLE FOR COLUMN INFORMATION

   Number of columns (mandatory)   :1
   Mnemonic names                  :2
   Transformations                 :3
   Tracing extrema                 :4
   Multi-sorting column guides     :5
   Randomization of data           :6
   Column ranking                  :7
   Display column information      :8
   Modification of column info     :9
   Data retrieval                  :10
   Display/print of data file      :11
```

The selection of each entry of the table permits the assignment of the corresponding information. At the end of each assignment, the program returns for new column information. The exit of this environment is obtained by <CR>. The process for each assignment is as follows.

a. Number of Columns

Selecting this entry is mandatory for the operation of the program. The following prompt appears on the screen:

```
Assign number of columns :
```

The only action of the user is the assignment of the number of columns of the data matrix. The maximum number of columns permitted by the present design of the program is 128--which can easily be changed by a simple modification.

b. Mnemonic Names

The selection of this entry of the table permits the assignment of mnemonic names to columns of the data matrix. The prompt appearing on the screen this time is:

```
Assign column number (<CR> to RETURN) :
```

requesting the number of the column for the next mnemonic name to be assigned. After the assignment of the column number, the prompt appearing is:

```
Assign mnemonic name :
```

requesting the name for the assigned column number. The assignment of column names is not necessarily done in ascending order of column numbers; neither is it necessary to assign names to all columns. The program sorts the assigned column numbers in order, to appear in the display of the header's data in the proper order. <CR> terminates the assignment of names.

c. Transformations

IDAMAN uses this function to permit any number and kind of mathematical expression involving values of the data matrix to be assigned. In this way a transformation of the value according to the assigned expression is obtained. The value of each expression (calculated for each row) is considered as a new column of the header and of the data matrix. The result of the transformation is attached to the rightmost end of the columns as a new column. The total number of the original columns plus the number of columns created by transformations should not exceed the maximum permitted number of 128.

The function of the transformations is better explained by the following example. Suppose that the initial data matrix to be transformed is:

$$\begin{array}{rrrr}
2.2 & 5.3 & 9.0 & 0.3 \\
3.0 & 1.2 & 4.0 & 1.5 \\
1.1 & 6.7 & 16.0 & 23.5 \\
12.3 & 3.0 & 1.0 & 3.5 \\
0.0 & 1.0 & 0.0 & 1.0 \\
4.4 & 5.0 & 4.0 & 8.7
\end{array}$$

with NCOL=4 NROW=6 and the assigned transformation is SQRT[3]/2. This means that the column to be transformed is the third, and the transformation will be the square root of its value, and then dividing the result by 2. The new data matrix will have NCOL=5 and will be:

2.2	5.3	9.0	0.3	1.5
3.0	1.2	4.0	1.5	1.0
1.1	6.7	16.0	23.5	2.0
12.3	3.0	1.0	3.5	0.5
0.0	1.0	0.0	1.0	0.0
4.4	5.0	4.0	8.7	1.0

The form of the assigned transformation can be any algebraic expression involving values of the data matrix, any real or integer constant value, and values resulting from the application of the following functions, assigned by its number column:

· LOGC	Common logarithm.
· LOG	The natural logarithm.
· EXP	Exponential.
· ABS	Absolute value.
· SIN	Sine.
· COS	Cosine.
· TAN	Tangent.
· ASIN	Arc sine.
· ACOS	Arc cosine.
· ATAN	Arc tangent.
· SINH	Hyperbolic sine.
· COSH	Hyperbolic cosine.
· TANH	Hyperbolic tangent.
· SQRT	Square root.
· ASINH	Arc sine hyperbolic.

The column number for which the value will be substituted for in calculating the expression, must be enclosed in brackets ([]) in order to be recognized by the program. The notion of exponentiation is shown by the character, "`". All functions assigned in an expression must be in upper case letters in order to be recognized by the program. The following are some examples of expressions that can be assigned as transformations of columns:

- [3]+3.14
- SQRT[7]
- (SIN[5]+3.14)^2
- LOG[10]-([12]*(-SQRT[2])^3
- ((COS[15]+3.14)/[4])*5.23

At the end of the expression(s) assignment, the file name in which the data to be transformed are stored, must be assigned.

d. Tracing Extrema

By this function, IDAMAN permits the assignment of a range of values in which all existing values in the data matrix will be used for tracing. This area is defined by its lower and upper boundaries (extrema). Upon selecting the corresponding entry of the selection table, the program requests the name of the file on which the data are stored, in order to determine the existing maximum and minimum values of each column. These values are used for tracing in the case that not-user defined extrema are desired. The extrema are displayed during assignment to facilitate the work. The next assignment is the type of extrema that will be used. That is the purpose of the prompt:

```
Imposed extrema for tracing :1
Original extrema            :0
```

Selecting 0, the program automatically will use the minimum and maximum values of each column as tracing extrema. Selecting 1, the user is requested to assign the column number for the tracing extrema, by the following prompt:

```
Assign column number (<CR> to RETURN) :
```

The prompt for exrema assignment continues:

```
MIN (found):          Imposed:
MAX (found):          Imposed:
```

The program displays the existing minimum and maximum values of each column. The user thus assigns his/her different imposed extrema.

e. Multi-sorting Column Guides

Multi-sorting column guides allow the system to assign a series of columns according to which multiple sorting will take place. For details on multiple sorting, see Section II.B. The assignment of the columns that will be used as guides for the sorting can be done by the column numbers or by their mnemonic names. The following prompt requests the user to assign the desired mode of assignment:

```
By column numbers :1
By mnemonic names :2
Answer            :
```

By selecting 1, the following prompt will appear:

```
Assign no(s). of columns :
```

The assignment of sorting guides can be a series of column numbers separated by a "comma" or two numbers separated by a "colon" meaning that all the numbers contained therein will be used as multiple sorting guides. If, for example, 18,13,5 is assigned, the data will be sorted according to column 18, then according to column 13, and finally according to column 5. If 5:8 is assigned, the data will be sorted according to column 5, then according to column 6, then column 7, and finally column 8. If the assigned series of columns exceeds one screen line, the assignment can continue to the next screen line by using a "comma" as a continuation mark. In the case that assignment of sorting guides using mnemonic names is desired, the prompt:

```
Assign column mnemonic names :
```

will provide it. In this case the names must be assigned one in each screen line. After printing the name of the column, the carriage return (<CR>) will enter the name in the computer while the cursor will move to the next screen line for the next name assignment.

f. Randomization of Data

By this operation, the system permits randomization of the data for improvement of the sort method performance or for elimination of the time effect. The only assignment for this operation is the name of the file on which the data are stored. This assignment is done as soon as the prompt requesting the name of the file appears on the screen (see Section III.B).

g. Column Ranking

A new arrangement of the columns assigned by the user is permitted. This assignment can be done by column numbers or column names. Upon selection of the operation, the prompt:

```
By column numbers :1
By mnemonic names :2
Answer            :
```

requests the desired assignment mode. If the mode is the first one, the prompt:

Assign no(s) of columns:

requests the assignment of a series of column numbers which will be used for the new ranking. This assignment is done as described in Section III.B. The assignment of columns with mnemonic names is also done as in Section III.E.2.e, after the prompt:

Assign ranking by sequence of names :

appears on the screen.

The operation of ranking creates a new data file. The number of columns used for the ranking assignment is the number of columns of the new data matrix.

h. Display of Column Information

Display of column information shows assigned information related to the columns of the header.

i. Modification of Column Information

By this operation, the program passes to the environment for modification of information related to the columns. This is a modification of information assigned by the current run of the program. As mentioned above, the modification of a header created by another run and existing in a file, is done by another mode of the program. Upon selecting the modification operation, the table:

```
MODIFICATION SELECTION TABLE

 Number of columns   :1
 Mnemonic names      :2
 Tracing extrema     :3
 Sorting guides      :4
 Rank of columns     :5
 Answer              :
```

permits selection of the desired modification. The several modifications are executed with the same kind of conversational mode. The old information, which are candidates for modification, appear on the screen to facilitate the work. In case that modification is attempted for information that has not been assigned, the program reacts with a message informing the user to assign the information. The following prompts appear for the corresponding modification selections.

(1) <u>Number of Columns</u>.

```
Old number of columns :
Assign new number     :
```

(2) <u>Mnemonic Names</u>.

```
Assign column number :
```

Upon the assignment of column number for the mnemonic name which will be modified, the prompt:

```
Old name        :
Assign new name :
```

facilitates the modification.

(3) Tracing Extrema. The prompt:

```
Assign column number :
```

requests the number of the column for which the tracing extrema will be modified and in following the prompts:

```
Old MIN         :
Assigns new MIN :
Old MAX         :
Assign new MAX  :
```

permits the modification of the displayed old extrema.

(4) Sorting Guides. Selecting modification of the multi-sorting column guides, the old ones (column numbers or mnemonic names) are displayed initially, and then the prompts:

```
Assign no(s) of columns :
```

or

```
Assign mnemonic names :
```

requests new column assignment by numbers or mnemonic names.

(5) Rank of Columns. Modification of the ranking assignment is similar to modification of multi-sorting guides. The difference is that the number of columns must be modified before the modification of the ranking assignment in order to be consistent with the initial data matrix.

j. Data Retrieval

By this operation, the program searches the data file for a specific value assigned by the user or for a missing value represented by a "gold" value, and informs the user about them as explained below. The process for the assignment of this operation of the program is as follows:

```
Name of the existing data file :
```

requests the key number of the data file which is going to be searched. The next prompt:

```
Assign device (or file) :
```

permits the user to assign where he/she desires to receive information (terminal or file). The amount of file to be searched is also assigned by the user with determination of columns and rows of the data matrix. This is done after the appearance of prompts:

```
Assign no(s). of columns :
```

and

```
Assign no(s). of rows    :
```

The assignment can be done by the symbol "*" meaning "all", by any alphabetic character, by sequence of column or row numbers separated by a comma (",") (e.g., 2,5,36) which determines specific columns and rows, or by two numbers separated by a colon (":") which determines all columns or rows included between the two numbers. The prompt that follows:

```
Data to be retrieved (real or "gold") :
```

permits the assignment of the real value to be searched. The letter "g" or "G" determines that the program must search for missing values which are represented by the number 12345.678 or any number that can be assigned by the user with the next prompt:

```
Change or RETURN :
```

The data provided by the program after completion of the search is the Index (sequence number of the element in the matrix), the column number, the row number and the value.

k. Display/Print of Data File

For the data file to be displayed, information regarding its name, the device on which it will be printed or displayed, the numbers of columns and rows that will be printed or displayed are requested as in Section III.E.2.e. The format of the data values is also the object of user's specification. It follows the prompt:

```
The normal format     : nnnnnnnn.nnnnnn
To modify it assign your format (ex.nnn.nnn) <CR> :
```

which gives the normal format (F14.6 i.e., 14 digits--6 decimal) of the display or print. If modification is desired, an example-like assignment can be done as shown by the prompt's example using any keyboard character.

3. Assigning Information for the Rows

The function of the program in the row information environment is similar to that of column information. Upon entering this environment the following menu-like table appears on the screen:

```
SELECTION TABLE FOR ROW INFORMATION

   Number of rows (obligatory)   :1
   Mnemonic names                :2
   Row suppression               :3
   Row rejection                 :4
   Row ranking                   :5
   Display row information       :6
   Modification of row info      :7
   Data retrieval                :8
   Display/print of data file    :9
   Answer                        :
```

The process for the assignment of each information is as follows:

a. Number of Rows

The number of rows is assigned when the prompt:

```
Assign number of rows :
```

appears on the screen and can be up to 2048.

b. Mnemonic Names

For the rows of the data matrix, mnemonic names can be assigned for each individual row or for a user determined set of rows. The desired kind of assignment is determined as soon as the prompt:

```
Row by row      :
By set of rows  :
Answer          :
```

appears on the screen. For the assignment of row by row names, the number of rows has to be given, followed by the name just as requested by the prompts:

```
Assign row number (<CR> to RETURN) :
Assign mnemonic                    :
```

For assignment of names by sets of rows, the row numbers determining a set must be given by the user right after the set of the following prompts appears on the screen:

```
Common name set :
From row        :
To row          :
Name            :
```

The number of common name sets is automatically given by the program. Termination of assignment is obtained by hitting <CR> instead of a row number.

c. Row Suppression

The row suppression operation is a physical suppression of rows of the data matrix. Assignment of the rows to be suppressed can be done either by the row number or by the common name of the set. In this case, all the rows of the set will be suppressed. The selection of mode follows the prompt:

```
By number of rows :
By row set name   :
Answer            :
```

In both cases, assignment of the data file to be suppressed must be made as in Section III.B. Assignment of row numbers is done after:

```
Assign no(s) of rows :
```

by printing the individual row numbers separated by a comma (e.g., 4,6,7) or by a set of rows determined by two numbers separated by a colon (e.g., 5:10). Both sorts of assignments can be done consecutively. For example, the suppression assignment 1,2,3,4,8,11,16:20 will cause suppression of the rows 1,2,3,4,8,11,16,17,18,19,20. If, after this assignment, a display of row information is requested and the number of rows before suppression was 30, information appearing on the screen will be as follows:

```
          R O W   I N F O R M A T I O N
          *****************************

                  NUMBER OF ROWS
                  ______________

                        19

                   SUPPRESSIONS
                   ____________

    Suppression      From Row     To Row
    ____________________________________
         1              1            4
         2              8            8
         3             11           11
         4             16           20
```

The suppression assignment by a common name set of rows is done simply by assigning the name of the set after the prompt:

```
    _____________
    Suppression :
    Set name    :
    _____________
```

d. Row Rejection

By this operation rows of a data matrix can be assigned which are to be excluded from a calculation. The determination of those columns can be done by their numbers, which will be stored in a sequential file. The name is defined by the user as the following prompts appear on the screen:

```
Assign sequential file for rejected values
Key (xxx) For File FORxxx.DAT number n, (nnn=076) :
Assigned rejected rows for calculus :
Assign no(s). of rows :
```

e. Row Ranking

The row ranking, like column ranking, permits the assignment of the order by which the rows of the data matrix will be arranged. The assignment is done by the number of the rows after the prompt:

```
Assign no(s). of rows :
```

and the names of the old and new files (see Section III.B).

f. Display Row Information

The selection of this table of the entry simply displays the assigned row information without any other user intervention.

g. Modification of Row Information

This operation, as in the column information environment, permits modification of assigned information related to rows of the data matrix, by the current run of the program. In general the function of the program in this environment is similar to that for row modification. The menu of the available modifications is the following:

```
MODIFICATION SELECTION TABLE

Number of rows                 :1
Mnemonics by row numbers       :2
Mnemonics by series of rows    :3
Suppression                    :4
Rank of rows                   :5
Answer                         :
```

(1) Number of Rows. The modification of the number of rows is simply a new assignment of the desired rows following the prompts:

```
Old number of rows :
Assign new number  :
```

(2) Mnemonics by Rows. This operation permits modification of mnemonic names assigned individually for each row of the data matrix. The row number and name are required to be given after the prompt:

```
Assign column number :
```

and then the new name appears after the prompt:

```
Old name        :
Assign new name :
```

(3) Mnemonics by Series of Rows. In this operation, the program permits modification of the set of rows with a common name or modification of the name of the set. The desired modification is determined with the assistance of the prompt:

```
Series modification :1
Name modification   :2
Answer              :
```

If modification of the series of rows with a common name is desired, the number of the series (set) must be determined after the prompt:

```
Assign series number :
```

and the old series is displayed on the screen to facilitate the new assignment:

```
Old series :
From row   :
To row     :
```

while at the same time the prompt:

```
Assign new series :
From row          :
To row            :
```

permits assignment of the new specification of the series. If modification of the common name of the set is desired, the prompt:

```
Assign series number :
```

requests the number of the set to be modified. Then the old mnemonic appears on the screen together with the request for new mnemonic name assignment:

```
Old mnemonic          :
Assign new mnemonic :
```

(4) Suppression. As mentioned in Section III.E.3.c, the suppressions automatically have sequence numbers attached to them. Modification of the suppressions is done by first assigning the suppression sequence number requested by the prompt:

```
Assign suppression number :
```

Modification of the suppressed rows is facilitated by the display of the old suppression specification requesting the new assignment:

```
Old suppression        :
From row               :
To row                 :
Assign new suppression :
From row               :
To row                 :
```

(5) Rank of Rows. Modification of row ranking is executed by the following prompt:

```
Old ranking          :
Assign no(s) of rows :
```

h. Data Retrieval

This is simply another place in the program where the retrieval of data can be obtained in exactly the same way as in Section III.E.2.j.

i. Display/Print of Data File

This is an alternative position for display or print of the data file, executed the same way as in Section III.E.2.k.

2. Displaying an Existing Header

This is the second mode of operation of the program. It simply displays either the column or row information of a header created by a previous run of the program. As has been mentioned, information of a created header is stored in a sequential file in secondary storage. This file is automatically opened by the program. Data are read and displayed on the screen by the simple assignment of the name of the header's file requested by the prompt:

```
Name of header's file :
```

The next required assignment is the determination of column or row information display:

```
Column information :
Row information    :
Answer             :
```

Exit of the mode is obtained by <CR>.

3. Modifying an Existing Header

In this mode, the modification of a header created by a previous run of the program can be obtained by simply assigning the name of the header's file.

4. Merging

With this mode of operation, a merging of two data matrices and combination of their corresponding headers is obtained. This operation can be executed in the horizontal (column merging) or vertical (row merging) sense; in other words the matrices can be merged side by side or one over the other. Horizontal merging is only permitted by the program if the two matrices have the same number of rows while vertical merging is permitted if they have the same number of columns. The resultant headers in such an operation are as follows:

- Row merging.
  - Column names: If the same column has a mnemonic name in both headers, then the name of the first header (upper matrix) is kept as the column name. Otherwise the unique name in either header is kept as the column name.
  - Tracing extrema: If tracing extrema are assigned for the same column in both headers, then the minimum of the two minima and the maximum of the two maxima of the columns are kept as tracing extrema for the column. Otherwise the unique minimum and maximum values are kept as tracing extrema for the column.
  - Row names: The names of rows of the two headers whenever they exist.
- Column merging.

The naming of the rows of the combined header is done in the same way as the naming of columns. Since the first one of the two merged data matrices is adjusted to the left of the created common matrix, the naming of the sets of rows of this file is kept as the common name for the corresponding rows of the new matrix.

The operation of merging is executed without any extra intervention of the user, simply by assignment of the file names of the two headers to be combined:

```
Name of first Header's file  :
Name of second Header's file :
Name of new Header's file    :
```

and the sort of desired merging:

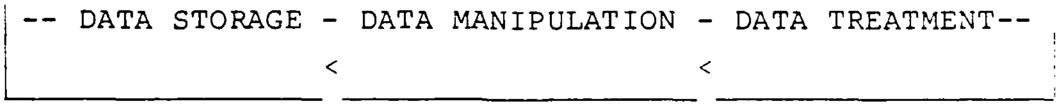
```
Row merging    :1
Column merging :2
```

The names of the files on which the two merged matrices are stored and the one on which the new one will be stored, uses the process described in Section III.B.

IV. CONCLUSION AND PERSPECTIVES

The interactive-conversational character makes the system easily usable and independent of knowledge requirements. The language used for implementation (FORTRAN 77) makes it transportable. The separation of the data manipulation process from the calculations also makes it highly expandable. Since the developed system composes only the module for manipulation of the data, development of the remaining modules constitute the perspectives of the concept in order for it to be complete. The cycle of data analysis is realized if the following process is completed:

(IDAMAN)



- Data can be obtained and stored automatically by a previous program or by the keyboard under control of a specific program--the Interactive DAta LOaDer, IDALOD, which is under development by Dr. Daniel Guinier.
- Data treatment (statistical or mathematical) can be done by modification of modules of existing packages by addition of the two subroutines READER, in relation with the "header file", and REJECT, in relation with a

"logical suppression file". This file has been previously created by IDAMAN. Application programs involve:

- one column (e.g., comparisons of means of sets of observations stored in one or several columns),
- two columns (e.g., graphs including observed and/or data fit with a given model),
- several columns (e.g., multivariate statistical analysis (discriminant, principal components, cluster, ... analysis), can be easily developed. There is no limitation to the expandability because these tasks are not included in the previous tasks of loading and management.

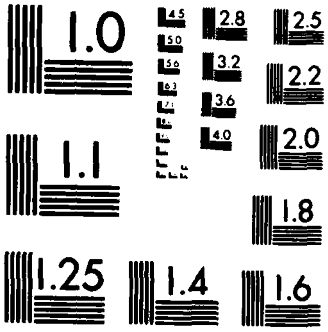
```
C                             APPENDIX A
C
C         FORTRAN 77 Source Code of Interactive Data Manager
C
C
C**********************************************************************
C
C            I n t e r a c t i v e   D A T a   M A N a g e r
C         -----------------------------------------------
C                             I D A M A N
C                             -----------
C      FUNCTION :
C      ----------
C
C      This system provides the means for  an  interactive  and
C      self explanatory manipulation of  large  collections  of
C      data  stored in direct-access files in secondary memory.
C      The way the system can manipulate the data is such  that
C      facilitates the future statistical analysis of them. The
C      system   can   execute   the   following   manipulations
C      (functions)
C
C      a. Sort the data matrix.
C      b. Multi-sort the data matrix.
C      c. Search the data matrix for a value.
C      d. Rearrangement the columns or rows of the data matrix.
C      e. Transformation of a column values.
C      f. Randomization   of   the   values   of   the   entire   data
C         matrix.
C      g. Logical suppression of data.
C      h. Display of the assigned HEADER.
C      i. Display of the data matrix.
C
C      By D.GUINIER and N.TOTOS (1984)
C      Naval Postgraduate School, Depatrment of Computer Science
C      Monterey, California 93940
C
C      VARIABLES :
C      ---------
C
C      NCOL       : The number of columns.
C      NROW       : The number of rows.
C      COLMNE     : Array storing the column mnemonic names.
C      NKEY       : Array storing the  numbers  of columns which
C                   will be used as sorting guides.
C      MNE        : Array  storing  the  names  of columns which
C                   will  be used as sorting guides.
C      ROWMNE1    : Array storing the row mnemonic names.
C      ROWMNE2    : Array storing the mnemonics of sets of rows.
C      NRANK      : Array storing the  numbers  of  columns that
C                   assign the ranking of columns.
```

```
C     MRANK      : Array storing the names of columns that
C                  assign the ranking of columns.
C     A,B        : Arrays storing the mumber of row on which
C                  starts a set of common name rows and the
C                  corresponding on which ends the set.
C     SUP1,SUP2  : Arrays storing the number of row on which
C                  starts suppression of rows and the
C                  corresponding on which ends the suppression.
C     RMIN,RMAX  : Arrays storing the minimum and maximum
C                  values between which tracing of the data is
C                  going to take place.
C     NCO        : The numbers of columns for which have been
C                  assigned names.
C     NRO        : The numbers of rows for which have been
C                  assigned names.
C     NTR        : The column numbers for which tracing extrema
C                  have been assigned.
C     NCIN       : The invert relative addresses of COLMNE.
C     NRIN       : The invert relative addresses of ROWMNE1.
C     NTIN       : The invert relative addresses of RMIN,RMAX.
C     SN         : The number of assigned suppressions.
C     FLAG1      : Integer array of six elements used as flag
C                  to indicate the existence or not of the
C                  several informations regarding the columns.
C     FLAG2      : The corresponding flag for rows.
C     FNAME      : The name of the sequential file assigned by
C                  the user for on which the headers's data
C                  will be stored.
C     H1FNAME    : The name of the first header's file which
C                  will be merged.
C     H2FNAME    : The name of the second file for merging.
C     NC         : The number of assigned column mnemonic names
C     NR         : The number of assigned row mnemonic names.
C     NT         : The number of columns for which tracing
C                  extrema have been assigned.
C     IJ         : The number of assigned tracing extrema sets.
C     NN         : The number of repeated sortings according
C                  column numbers.
C     NM         : The number of repeated sortings according
C                  column mnemonic names.
C     NNU        : The number of columns used for ranking.
C     KMN        : The number of column names used for ranking.
C     LEC        : The logical unit for writing on the terminal
C     IMP        : The logical unit for reading from the
C                  terminal.
C     LOGN       : The logical chanel for the file on which the
C                  header,s data will be stored.
C     NOROW      : Array the numbers of rows used as guides
C                  for the ranking of the rows.
C     NBROW      : The number of row numbers used for row
C                  ranking
C
C*************************************************************
```

```
C
C     DECLARATIONS :
C     --------------
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 SN,FLAG1(6),FLAG2(4),NKEY(128),NRANK(128),
      1         B(128),SUP1(128),SUP2(128),NCO(128),
      2         NRO(128),NTR(128),NTIN(128),NRIN(2048),
      3         NOROW(2048),A(128),NCIN(128)
      REAL*4 RMIN(128),RMAX(128)
      CHARACTER*24 FNAME,H1FNAME,H2FNAME,IND,COLMNE(128),
      1           MNE(128),MRANK(128),ROWMNE1(2048),
      2           ROWMNE2(128)
      LOGICAL*1 STAT,STAT1,STAT2
      DATA LEC,IMP,LOGN/5,6,0/

      IDUM=1
      DO WHILE (IDUM.EQ.1)
         IDUM=0
         WRITE(IMP,100)
         READ(LEC,200) IND
         IF (IND.EQ.'1') THEN
            CALL CREATE(LEC,IMP,LOGN,FLAG1,FLAG2,NCOL,NROW,
      1          COLMNE,NC,NCO,NCIN,ROWMNE1,NR,NRO,NRIN,
      2          ROWMNE2,RMIN,RMAX,NT,NTR,NTIN,NKEY,MNE,NN,
      3          NM,IJ,A,B,NRANK,MRANK,NNU,KMN,SUP1,SUP2,SN,
      4          NOROW,NBROW)
            IDUM=1
         ELSE IF (IND.EQ.'2') THEN
            CALL DISPLA(LEC,IMP,LOGN,FLAG1,FLAG2,NCOL,NROW,
      1          COLMNE,NC,NCO,NCIN,ROWMNE1,NR,NRO,NRIN,
      2          ROWMNE2,RMIN,RMAX,NT,NTR,NTIN,NKEY,MNE,NN,
      3          NM,IJ,A,B,NRANK,MRANK,NNU,KMN,SUP1,SUP2,SN,
      4          NOROW,NBROW)
            IDUM=1
         ELSE IF (IND.EQ.'3') THEN
            CALL MODIFY(LEC,IMP,LOGN,FLAG1,FLAG2,NCOL,NROW,
      1          COLMNE,NC,NCO,NCIN,ROWMNE1,NR,NRO,NRIN,
      2          ROWMNE2,RMIN,RMAX,NT,NTR,NTIN,NKEY,MNE,NN,
      3          NM,IJ,A,B,NRANK,MRANK,NNU,KMN,SUP1,SUP2,SN,
      4          NOROW,NBROW)
            IDUM=1
         ELSE IF (IND.EQ.'4') THEN
            CALL MERGE(LEC,IMP,LOGN,FLAG1,FLAG2,NCOL,NROW,
      1          COLMNE,NC,NCO,NCIN,ROWMNE1,NR,NRO,NRIN,
      2          ROWMNE2,RMIN,RMAX,NT,NTR,NTIN,NKEY,MNE,NN,
      3          NM,IJ,A,B,NRANK,MRANK,NNU,KMN,SUP1,SUP2,SN,
      4          NOROW,NBROW)
            IDUM=1
         ELSE IF (IND.EQ.' ') THEN
            IDUM=0
         ELSE
            WRITE(IMP,300)
            IDUM=1
         END IF
      END DO

C     FORMATS
C     -------
C
100   FORMAT(/////13X,'AVAILABLE MODES'/13X,
      1                '---------------'/
      2           4X,'Creation of new header           :1'/
      3           4X,'Display of existing header       :2'/
      4           4X,'Modification of existing header :3'/
      5           4X,'Merging of two existing headers :4'/
      6       '$',3X,'Answer                           :')
200   FORMAT(A24)
300   FORMAT(/3X,'INVALID CHARACTER!!')

      END

C*************************************************************
      SUBROUTINE CREATE(LEC,IMP,LOGN,FLAG1,FLAG2,NCOL,NROW,
      1          COLMNE,NC,NCO,NCIN,ROWMNE1,NR,NRO,NRIN,
      2          ROWMNE2,RMIN,RMAX,NT,NTR,NTIN,NKEY,MNE,NN,
      3          NM,IJ,A,B,NRANK,MRANK,NNU,KMN,SUP1,SUP2,SN,
      4          NOROW,NBROW)
C*************************************************************
C
C   This subroutine is used for the creation of a new header.
C
C     ARGUMENTS
C     ---------
C
C     LEC        : The logical unit number for  writing on  the
C                  terminal.
C     IMP        : The logical unit number for reading from the
C                  terminal.
C     LOGN       : The logical unit for the  file on  which the
C                  header,s data will be stored.
C     FLAG1      : Integer array of six  elements  used as flag
C                  to indicate  the  existence  or  not  of the
C                  several informations regarding the columns.
C     FLAG2      : The corresponding flag for rows.
C     NCOL       : The number of columns.
C     NROW       : The number of rows.
C     COLMNE     : Array storing the column mnemonic names.
C     NC         : The number of assigned column mnemonic names
C     NCO        : The  numbers of columns for which have  been
C                  assigned names.
C     NCIN       : The invert relative addresses of COLMNE.
C     ROWMNE1    : Array storing the row mnemonic names.
C     NR         : The number of assigned row mnemonic names.
C     NRO        : The numbers of rows  for  which  names  have
C                  been assigned.
C     NRIN       : The invert relative addresses of ROWMNE.
```

AD-A151 359 INTERACTIVE DATA ANALYSIS DEVELOPMENT OF AN INTERACTIVE 2/3
DATA MANIPULATION SYSTEM(U) NAVAL POSTGRADUATE SCHOOL
MONTEREY CA N G TOTOS JUN 84

UNCLASSIFIED F/G 9/2 NL


1.0
4.5
5.0
5.6
6.3
7.1
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
C     ROWMNE2   : Array storing the  mnemonics of sets of rows
C     RMIN,RMAX : Arrays  storing  the  minimum  and   maximum
C                 values between  which tracing of the data is
C                 going to take place.
C     NT        : The  number of  columns  for  which  tracing
C                 extrema have been assigned.
C     NTR       : The  column   numbers   for   which  tracing
C                 extrema have been assigned.
C     NTIN      : The invert relative addresses of RMIN,RMAX.
C     NKEY      : Array storing  the numbers  of columns which
C                 will be used as sorting guides.
C     MNE       : Array storing the  names  of  columns  which
C                 will be used as sorting guides.
C     NN        : The number of  repeated  sortings  according
C                 column numbers.
C     NM        : The  number of repeated  sortings  according
C                 column mnemonic names.
C     IJ        : The number of assigned  tracing extrema sets
C     A,B       : Arrays  storing the  mumber of  row on which
C                 starts  a  set of common  name rows  and the
C                 corresponding on which ends the set.
C     NRANK     : Array storing the  numbers  of  columns that
C                 assign the ranking of data
C     MRANK     : Array  storing  the  names  of  columns that
C                 assign the ranking of data
C     NNU       : The number of columns used for ranking.
C     KMN       : The number of column  names used for ranking
C     SUP1,SUP2 : Arrays storing the  number  of  row on which
C                 starts suppression  of   rows    and     the
C                 corresponding on  which ends the suppression
C     SN        : The number of assigned suppressions.
C     NOROW     : Array  the  numbers  of  rows used as guides
C                 for the ranking of the rows.
C     NBROW     : The number  of  row  numbers   used  for row
C                 ranking
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 SN,FLAG1(6),FLAG2(4),NKEY(128),NRANK(128),
      1         B(128),SUP1(128),SUP2(128),NCO(128),NCIN(128),
      2         NRO(128),NTR(128),NTIN(128),NRIN(2048),
      3          NOROW(2048),A(128)
      REAL*4 RMIN(128),RMAX(128)
      CHARACTER*24 FNAME,COLMNE(128),MNE(128),MRANK(128),
      1          ROWMNE1(2048),ROWMNE2(128),IND,H1FNAME,H2FNAME
      LOGICAL*1 STAT,STAT1,STAT2
C
C       Request a name for  the file on which the data  of the
C       header will be stored.

      WRITE(IMP,200)
      READ(LEC,100) FNAME

C       Examine if a file which this name already exists.

      INQUIRE(FILE=FNAME,EXIST=STAT)

C       If not, open a new sequential file.

      IF (STAT.EQ..FALSE.) THEN
         OPEN(UNIT=LOGN,FILE=FNAME,STATUS='NEW')

C           Call the subroutine which will  request  for column
C           or row informations.
         CALL CRINFO(LEC,IMP,FLAG1,FLAG2,NCOL,NROW,COLMNE,
     1               NC,NCO,NCIN,ROWMNE1,NR,NRO,NRIN,ROWMNE2,
     2               RMIN,RMAX,NT,NTR,NTIN,NKEY,MNE,NN,NM,IJ,
     3               A,B,NRANK,MRANK,NNU,KMN,SUP1,SUP2,SN,
     4               NOROW,NBROW)

C           Record the assigned data  in  the opened  file  for
C        future reference.

         CALL WRITER (LOGN,FLAG1,FLAG2,NCOL,NROW,KMN,NNU,NN,
     1                NM,IJ,A,B,COLMNE,NC,NCO,NCIN,RMIN,RMAX,
     2                NT,NTR,NTIN,NKEY,MNE,NRANK,MRANK,
     3                ROWMNE1,NR,NRO,NRIN,ROWMNE2,SUP1,SUP2,
     4                SN,NOROW,NBROW)

C       If the file already exists, respond with a message.

      ELSE
         WRITE(IMP,300)
         IDUM=1
      END IF

C     FORMATS
C     -------
C
100   FORMAT(A24)
200   FORMAT(///'$',3X,'Name of header''s file :')
300   FORMAT(///4X,'THE FILE ALREADY EXIST!!')
      RETURN
      END

C*************************************************************
      SUBROUTINE DISPLA(LEC,IMP,LOGN,FLAG1,FLAG2,NCOL,NROW,
     1                  COLMNE,NC,NCO,NCIN,ROWMNE1,NR,NRO,
     2                  NRIN,ROWMNE2,RMIN,RMAX,NT,NTR,NTIN,
     3                  NKEY,MNE,NN,NM,IJ,A,B,NRANK,MRANK,NNU,
     4                  KMN,SUP1,SUP2,SN,NOROW,NBROW)
C*************************************************************
C
C   This subroutine is used to display the data of a preveusly
```

```
C   defined header.
C
C     ARGUMENTS
C     ---------
C
C     LEC        : The logical unit number for  writing on  the
C                  terminal.
C     IMP        : The logical unit number for reading from the
C                  terminal.
C     LOGN       : The logical unit for the  file on  which the
C                  header,s data will be stored.
C     FLAG1      : Integer array of six  elements  used as flag
C                  to indicate  the  existence  or  not  of the
C                  several informations regarding the columns.
C     FLAG2      : The corresponding flag for rows.
C     NCOL       : The number of columns.
C     NROW       : The number of rows.
C     COLMNE     : Array storing the column mnemonic names.
C     NC         : The number of assigned column mnemonic names
C     NCO        : The  numbers of columns for which have  been
C                  assigned names.
C     NCIN       : The invert relative addresses of COLMNE.
C     ROWMNE1    : Array storing the row mnemonic names.
C     NR         : The number of assigned row mnemonic names.
C     NRO        : The numbers of rows  for  which  names  have
C                  been assigned.
C     NRIN       : The invert relative addresses of ROWMNE.
C     ROWMNE2    : Array storing the  mnemonics of sets of rows
C     RMIN,RMAX  : Arrays  storing  the  minimum  and  maximum
C                  values between  which tracing of the data is
C                  going to take place.
C     NT         : The  number of  columns  for  which  tracing
C                  extrema have been assigned.
C     NTR        : The  column   numbers   for   which  tracing
C                  extrema have been assigned.
C     NTIN       : The invert relative addresses of RMIN,RMAX.
C     NKEY       : Array storing  the numbers  of columns which
C                  will be used as sorting guides.
C     MNE        : Array storing the  names  of  columns  which
C                  will be used as sorting guides.
C     NN         : The number of  repeated  sortings  according
C                  column numbers.
C     NM         : The  number of repeated  sortings  according
C                  column mnemonic names.
C     IJ         : The number of assigned  tracing extrema sets
C     A,B        : Arrays  storing the  mumber of  row on which
C                  starts  a  set of common  name rows  and the
C                  corresponding on which ends the set.
C     NRANK      : Array storing the  numbers  of  columns that
C                  assign the ranking of data
C     MRANK      : Array  storing  the  names  of  columns that
C                  assign the ranking of data
```

```
C     NNU       : The number of columns used for ranking.
C     KMN       : The number of column  names used for ranking
C     SUP1,SUP2 : Arrays storing the  number  of  row on which
C                 starts  suppression  of   rows   and   the
C                 corresponding on  which ends the suppression
C     SN        : The number of assigned suppressions.
C     NOROW     : Array  the  numbers  of  rows used as guides
C                 for the ranking of the rows.
C     NBROW     : The number  of  row  numbers   used  for row
C                 ranking
C
C*************************************************************
C
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 SN,FLAG1(6),FLAG2(4),NKEY(128),NRANK(128),
      1         B(128),SUP1(128),SUP2(128),NCO(128),NCIN(128),
      2         NRO(128),NTR(128),NTIN(128),NRIN(2048),
                NOROW(2048),A(128)
      REAL*4 RMIN(128),RMAX(128)
      CHARACTER*24 FNAME,COLMNE(128),MNE(128),MRANK(128),
      1            ROWMNE1(2048),ROWMNE2(128),IND,H1FNAME,
                   H2FNAME
      LOGICAL*1 STAT,STAT1,STAT2

      WRITE(IMP,200)
C       Request the name of the file.

      READ(LEC,100) FNAME
      INQUIRE(FILE=FNAME,EXIST=STAT)

C       If the file exists open it.

      IF (STAT.EQ..TRUE.) THEN
         OPEN(UNIT=LOGN,FILE=FNAME,STATUS='OLD')
         REWIND LOGN

C      Call the subroutine READER to read the data of the file

         CALL READER(LOGN,FLAG1,FLAG2,NCOL,NROW,KMN,NNU,NN,NM,
      1              IJ,A,B,COLMNE,NC,NCO,NCIN,RMIN,RMAX,NT,
      2              NTR,NTIN,NKEY,MNE,NRANK,MRANK,ROWMNE1,NR,
      3              NRO,NRIN,ROWMNE2,SUP1,SUP2,SN,NOROW,NBROW)
         IDUM=1

C          Loop for succesive display capability.

         DO WHILE(IDUM.EQ.1)
            IDUM=0

C             Request for column or row display.

            WRITE(IMP,300)
            READ(LEC,100) IND
            IF (IND.EQ.'1') THEN

C                If column display is requested;

                CALL CDISP(LEC,IMP,FLAG1,NCOL,COLMNE,NC,NCO,
     1                   NCIN,RMIN,RMAX,NT,NTR,NTIN,NKEY,MNE,
     2                   NN,NM,NRANK,MRANK,NNU,KMN)
                IDUM=1
            ELSE IF (IND.EQ.'2') THEN

C             If row display is requested;

               CALL RDISP(LEC,IMP,FLAG2,NROW,ROWMNE1,NR,NRO,
     1                  NRIN,ROWMNE2,IJ,A,B,SUP1,SUP2,SN,
     2                  NOROW,NBROW)
               IDUM=1

C             If <CR> stop the loop.

            ELSE IF (IND.EQ.' ') THEN
                IDUM=0

C           If invalid character is hit give an error message.

            ELSE
               WRITE(IMP,400)
               IDUM=1
            END IF
         END DO
      END IF

C     FORMATS
C     -------
C
100   FORMAT(A24)
200   FORMAT(///'$',3X,'Name of header''s file :')
300   FORMAT(///4X,    'Column informations   :1'
     1        /4X,    'Row informations      :2'
     2        /'$',3X,'Answer                :')
400   FORMAT(/3X,'INVALID CHARACTER!!')
      RETURN
      END


C************************************************************
      SUBROUTINE MODIFY(LEC,IMP,LOGN,FLAG1,FLAG2,NCOL,NROW,
     1                COLMNE,NC,NCO,NCIN,ROWMNE1,NR,NRO,NRIN,
     2                ROWMNE2,RMIN,RMAX,NT,NTR,NTIN,NKEY,MNE,
     3                NN,NM,IJ,A,B,NRANK,MRANK,NNU,KMN,SUP1,
     4                SUP2,SN,NOROW,NBROW)
C************************************************************
C
C   This subroutine is  used  for  modification of an existing
```

```
C   header.
C
C     ARGUMENTS
C     ---------
C
C     LEC         : The logical unit number for  writing on  the
C                   terminal.
C     IMP         : The logical unit number for reading from the
C                   terminal.
C     LOGN        : The logical unit for the  file on  which the
C                   header,s data will be stored.
C     FLAG1       : Integer array of six  elements  used as flag
C                   to indicate  the  existence  or  not  of the
C                   several informations regarding the columns.
C     FLAG2       : The corresponding flag for rows.
C     NCOL        : The number of columns.
C     NROW        : The number of rows.
C     COLMNE      : Array storing the column mnemonic names.
C     NC          : The number of assigned column mnemonic names
C     NCO         : The  numbers of columns for which have  been
C                   assigned names.
C     NCIN        : The invert relative addresses of COLMNE.
C     ROWMNE1     : Array storing the row mnemonic names.
C     NR          : The number of assigned row mnemonic names.
C     NRO         : The numbers of rows  for  which  names  have
C                   been assigned.
C     NRIN        : The invert relative addresses of ROWMNE.
C     ROWMNE2     : Array storing the  mnemonics of sets of rows
C     RMIN,RMAX   : Arrays  storing  the  minimum  and   maximum
C                   values between  which tracing of the data is
C                   going to take place.
C     NT          : The  number of  columns  for  which  tracing
C                   extrema have been assigned.
C     NTR         : The  column   numbers   for   which  tracing
C                   extrema have been assigned.
C     NTIN        : The invert relative addresses of RMIN,RMAX.
C     NKEY        : Array storing  the numbers  of columns which
C                   will be used as sorting guides.
C     MNE         : Array storing the  names  of  columns  which
C                   will be used as sorting guides.
C     NN          : The number of  repeated  sortings  according
C                   column numbers.
C     NM          : The  number of repeated  sortings  according
C                   column mnemonic names.
C     IJ          : The number of assigned  tracing extrema sets
C     A,B         : Arrays  storing the  mumber of  row on which
C                   starts  a  set of common  name rows  and the
C                   corresponding on which ends the set.
C     NRANK       : Array storing the  numbers  of  columns that
C                   assign the ranking of data
C     MRANK       : Array  storing  the  names  of  columns that
C                   assign the ranking of data
C     NNU         : The number of columns used for ranking.
```

```
C     KMN       : The number of column names used for ranking
C     SUP1,SUP2 : Arrays storing the number of row on which
C                 starts suppression of rows and the
C                 corresponding on which ends the suppression
C     SN        : The number of assigned suppressions.
C     NOROW     : Array the numbers of rows used as guides
C                 for the ranking of the rows.
C     NBROW     : The number of row numbers used for row
C                 ranking
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 SN,FLAG1(6),FLAG2(4),NKEY(128),NRANK(128),
      1         B(128),SUP1(128),SUP2(128),NCO(128),NCIN(128),
      2         NRO(128),NTR(128),NTIN(128),NRIN(2048),
      3         NOROW(2048),A(128)
      REAL*4 RMIN(128),RMAX(128)
      CHARACTER*24 FNAME,COLMNE(128),MNE(128),MRANK(128),
      1            ROWMNE1(2048),ROWMNE2(128),IND,H1FNAME,
      2            H2FNAME
      LOGICAL*1 STAT,STAT1,STAT2

C       Request the name of the file where the data of the
C       existing header are stored.

      WRITE(IMP,200)
      READ(LEC,100) FNAME
      INQUIRE(FILE=FNAME,EXIST=STAT)

C       If the file exists open it.

      IF (STAT.EQ..TRUE.) THEN
         OPEN(UNIT=LOGN,FILE=FNAME,STATUS='OLD')
         REWIND LOGN

C          Read the data of the existing header.

         CALL READER(LOGN,FLAG1,FLAG2,NCOL,NROW,KMN,NNU,NN,NM,
      1            IJ,A,B,COLMNE,NC,NCO,NCIN,RMIN,RMAX,NT,NTR,
      2            NTIN,NKEY,MNE,NRANK,MRANK,ROWMNE1,NR,NRO,
      3            NRIN,ROWMNE2,SUP1,SUP2,SN,NOROW,NBROW)

C          Close the old file.

         CLOSE(UNIT=LOGN)

C          Open a new file with the same name.

         OPEN(UNIT=LOGN,FILE=FNAME,STATUS='NEW')
         IDUM=1

C          Loop for succesive modification capability.
```

```
         DO WHILE(IDUM.EQ.1)
            IDUM=0
C             Request for column or row modification.

            WRITE(IMP,400)
            READ(LEC,100) IND

C             If column modification is desired;

            IF (IND.EQ.'1') THEN

C               Call COLMOD for column modification.

                CALL COLMOD(LEC,IMP,FLAG1,NCOL,NROW,COLMNE,NC,
      1                    NCO,NCIN,RMIN,RMAX,NT,NTR,NTIN,
      2                    NKEY,MNE,NN,NM,NRANK,MRANK,NNU,KMN)

C               Call  WRITER to  record  the modified header's
C               data.

                CALL WRITER(LOGN,FLAG1,FLAG2,NCOL,NROW,KMN,
      1                     NNU,NN,NM,IJ,A,B,COLMNE,NC,NCO,
      2                     NCIN,RMIN,RMAX,NT,NTR,NTIN,NKEY,
      3                     MNE,NRANK,MRANK,ROWMNE1,NR,NRO,
      4                     NRIN,ROWMNE2,SUP1,SUP2,SN,NOROW,
      5                     NBROW)
                IDUM=1

C             If row modification is desired;

            ELSE IF (IND.EQ.'2') THEN

C              Call ROWMOD for row modification.

               CALL ROWMOD(LEC,IMP,FLAG2,NCOL,NROW,ROWMNE1,NR,
      1                    NRO,NRIN,ROWMNE2,IJ,A,B,SUP1,SUP2,
      2                    SN,NOROW,NBROW)

C              Call  WRITER  to record  the  modified header's
C              data.

               CALL WRITER(LOGN,FLAG1,FLAG2,NCOL,NROW,KMN,NNU,
      1                   NN,NM,IJ,A,B,COLMNE,NC,NCO,NCIN,
      2                   RMIN,RMAX,NT,NTR,NTIN,NKEY,NME,
      3                   NRANK,MRANK,ROWMNE1,NR,NRO,NRIN,
      4                   ROWMNE2,SUP1,SUP2,SN,NOROW,NBROW)
               IDUM=1

C             If <CR> stop the loop.

            ELSE IF (IND.EQ.' ') THEN
```

```
               IDUM=0
C           If invalid character is hit give an error message.
            ELSE
               WRITE(IMP,500)
               IDUM=1
            END IF
         END DO
C       If the file does not exist give an error message.
      ELSE
         WRITE(IMP,300)
         IDUM=1
      END IF
C     FORMATS
C     -------
C
100   FORMAT(A24)
200   FORMAT(///'$',3X,'Name of header''s file :')
300   FORMAT(///4X,'THE FILE DOES NOT EXIST!!')
400   FORMAT(///4X,'Column :1'/4X,'Row    :2'/'$',3X,
      1            'Answer :')
500   FORMAT(/3X,'INVALID CHARACTER!!')
      RETURN
      END


C************************************************************
      SUBROUTINE MERGE(LEC,IMP,LOGN,FLAG1,FLAG2,NCOL,NROW,
      1                COLMNE,NC,NCO,NCIN,ROWMNE1,NR,NRO,NRIN,
      2                ROWMNE2,RMIN,RMAX,NT,NTR,NTIN,NKEY,MNE,
      3                NN,NM,IJ,A,B,NRANK,MRANK,NNU,KMN,SUP1,
      4                SUP2,SN,NOROW,NBROW)
C************************************************************
C
C     This  subroutine is used to merge  two existing headers.
C     The  merging  can  be  executed  in  the  row-row  or
C     column-column sense.
C
C     ARGUMENTS
C     ---------
C     ARGUMENTS
C     ---------
C
C     LEC        : The logical unit number for  writing on  the
C                  terminal.
C     IMP        : The logical unit number for reading from the
C                  terminal.
C     LOGN       : The logical unit for the  file on  which the
C                  header,s data will be stored.
```

```
C     FLAG1       : Integer array of six elements used as flag
C                   to indicate the existence or not of the
C                   several informations regarding the columns.
C     FLAG2       : The corresponding flag for rows.
C     NCOL        : The number of columns.
C     NROW        : The number of rows.
C     COLMNE      : Array storing the column mnemonic names.
C     NC          : The number of assigned column mnemonic names
C     NCO         : The numbers of columns for which have been
C                   assigned names.
C     NCIN        : The invert relative addresses of COLMNE.
C     ROWMNE1     : Array storing the row mnemonic names.
C     NR          : The number of assigned row mnemonic names.
C     NRO         : The numbers of rows for which names have
C                   been assigned.
C     NRIN        : The invert relative addresses of ROWMNE.
C     ROWMNE2     : Array storing the mnemonics of sets of rows
C     RMIN,RMAX   : Arrays storing the minimum and maximum
C                   values between which tracing of the data is
C                   going to take place.
C     NT          : The number of columns for which tracing
C                   extrema have been assigned.
C     NTR         : The column numbers for which tracing
C                   extrema have been assigned.
C     NTIN        : The invert relative addresses of RMIN,RMAX.
C     NKEY        : Array storing the numbers of columns which
C                   will be used as sorting guides.
C     MNE         : Array storing the names of columns which
C                   will be used as sorting guides.
C     NN          : The number of repeated sortings according
C                   column numbers.
C     NM          : The number of repeated sortings according
C                   column mnemonic names.
C     IJ          : The number of assigned tracing extrema sets
C     A,B         : Arrays storing the mumber of row on which
C                   starts a set of common name rows and the
C                   corresponding on which ends the set.
C     NRANK       : Array storing the numbers of columns that
C                   assign the ranking of data
C     MRANK       : Array storing the names of columns that
C                   assign the ranking of data
C     NNU         : The number of columns used for ranking.
C     KMN         : The number of column names used for ranking
C     SUP1,SUP2   : Arrays storing the number of row on which
C                   starts suppression of rows and the
C                   corresponding on which ends the suppression
C     SN          : The number of assigned suppressions.
C     NOROW       : Array the numbers of rows used as guides
C                   for the ranking of the rows.
C     NBROW       : The number of row numbers used for row
C                   ranking
C
C*************************************************************
```

```
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 SN,FLAG1(6),FLAG2(4),NKEY(128),NRANK(128),
      1         B(128),SUP1(128),SUP2(128),NCO(128),NCIN(128),
      2         NRO(128),NTR(128),NTIN(128),NRIN(2048),
      3         NOROW(2048),A(128)
      REAL*4 RMIN(128),RMAX(128)
      CHARACTER*24 FNAME,COLMNE(128),MNE(128),MRANK(128),
      1            ROWMNE1(2048),ROWMNE2(128),IND,H1FNAME,
                   H2FNAME
      LOGICAL*1 STAT,STAT1,STAT2

      IDUM=1

C     Loop in case of user input errors.

      DO WHILE(IDUM.EQ.1)
         IDUM=0

C       Request  the  name  of  the  first  file to be merged;
C       examine if it exists.

         WRITE(IMP,400)
         READ(LEC,100) H1FNAME
         INQUIRE(FILE=H1FNAME,EXIST=STAT1)

C       If does  not  exist print an error message and  repeat
C        the request.

         IF (STAT1.EQ..FALSE.) THEN
            WRITE(IMP,300)
            IDUM=1
         ELSE
         END IF

C      Request  the  name  of  the  second  file to be merged;
C      examine if it exists.

         WRITE(IMP,500)
         READ(LEC,100) H2FNAME
         INQUIRE(FILE=H2FNAME,EXIST=STAT2)

C        If does  not  exist print an error message and repeat
C        the request.

         IF (STAT2.EQ..FALSE.) THEN
            WRITE(IMP,300)
            IDUM=1
         END IF

C        If both files exist, request name for the new file.

         IF (STAT1.EQ..TRUE..AND.STAT2.EQ..TRUE.) THEN
```

```
        WRITE(IMP,700)
        READ(LEC,100) FNAME

C       Examine if the file exists.

        INQUIRE(FILE=FNAME,EXIST=STAT)

C     If not, request if column or  row merging is going to
C     be executed.

        IF (STAT.EQ..FALSE.) THEN
          WRITE(IMP,600)
          READ(LEC,100) IND

C         If row merging call COMROW

          IF (IND.EQ.'1') THEN
            CALL COMROW(LEC,IMP,LOGN,NCOL,NROW,FLAG1,
    1               FLAG2,COLMNE,NC,NCO,NCIN,IJ,A,B,
    2               ROWMNE1,NR,NRO,NRIN,ROWMNE2,
    3               H1FNAME,H2FNAME,RMIN,RMAX,NT,
    4               NTR,NTIN)
            OPEN(UNIT=LOGN,FILE=FNAME,STATUS='NEW')
            CALL WRITER(LOGN,FLAG1,FLAG2,NCOL,NROW,KMN,
    1              NNU,NN,NM,IJ,A,B,COLMNE,NC,NCO,
    2              NCIN,RMIN,RMAX,NT,NTR,NTIN,NKEY,
    3              MNE,NRANK,MRANK,ROWMNE1,NR,NRO,
    4              NRIN,ROWMNE2,SUP1,SUP2,SN,NOROW,
    5              NBROW)
          ELSE IF (IND.EQ.'2') THEN
            CALL COMCOL(LEC,IMP,LOGN,NCOL,NROW,FLAG1,
    1               FLAG2,COLMNE,NC,NCO,NCIN,IJ,A,B,
    2               ROWMNE1,NR,NRO,NRIN,ROWMNE2,
    3               H1FNAME,H2FNAME,RMIN,RMAX,NT,
    4               NTR,NTIN)
            OPEN(UNIT=LOGN,FILE=FNAME,STATUS='NEW')
            CALL WRITER(LOGN,FLAG1,FLAG2,NCOL,NROW,KMN,
    1               NNU,NN,NM,IJ,A,B,COLMNE,NC,NCO,
    2               NCIN,RMIN,RMAX,NT,NTR,NTIN,NKEY,
    3               MNE,NRANK,MRANK,ROWMNE1,NR,NRO,
    4               NRIN,ROWMNE2,SUP1,SUP2,SN,NOROW,
    5               NBROW)
          END IF
        ELSE
          WRITE(IMP,200)
          IDUM=1
        END IF
      END IF
    END DO
100 FORMAT(A24)
200 FORMAT(///4X,'THE FILE ALREADY EXIST!!')
300 FORMAT(///4X,'THE FILE DOES NOT EXIST!!')
400 FORMAT(///'S',3X,'Name of  first Header''s file :')
```

```
500   FORMAT('$',3X,   'Name of second Header''s file :')
600   FORMAT(///4X,'Row merging    :1'/4X,
      1            'Column merging :2',
      2    /'$',3X,'Answer         :')
700   FORMAT('$',3X,   'Name  of  new  Header''s file :')
      RETURN
      END

C*********************************************************************
      SUBROUTINE CRINFO(LEC,IMP,FLAG1,FLAG2,NCOL,NROW,COLMNE,
      1                NC,NCO,NCIN,ROWMNE1,NR,NRO,NRIN,
      2                ROWMNE2,RMIN,RMAX,NT,NTR,NTIN,NKEY,
      3                MNE,NN,NM,IJ,A,B,NRANK,MRANK,NNU,KMN,
      4                SUP1,SUP2,SN,NOROW,NBROW)
C*********************************************************************
C
C     ARGUMENTS
C     ---------
C     ARGUMENTS
C     ---------
C
C     LEC        : The logical unit number for  writing on  the
C                  terminal.
C     IMP        : The logical unit number for reading from the
C                  terminal.
C     LOGN       : The logical unit for the  file on  which the
C                  header,s data will be stored.
C     FLAG1      : Integer array of six  elements  used as flag
C                  to indicate  the  existence  or  not  of the
C                  several informations regarding the columns.
C     FLAG2      : The corresponding flag for rows.
C     NCOL       : The number of columns.
C     NROW       : The number of rows.
C     COLMNE     : Array storing the column mnemonic names.
C     NC         : The number of assigned column mnemonic names
C     NCO        : The  numbers of columns for which have  been
C                  assigned names.
C     NCIN       : The invert relative addresses of COLMNE.
C     ROWMNE1    : Array storing the row mnemonic names.
C     NR         : The number of assigned row mnemonic names.
C     NRO        : The numbers of rows  for  which  names  have
C                  been assigned.
C     NRIN       : The invert relative addresses of ROWMNE.
C     ROWMNE2    : Array storing the  mnemonics of sets of rows
C     RMIN,RMAX  : Arrays  storing  the  minimum  and   maximum
C                  values between  which tracing of the data is
C                  going to take place.
C     NT         : The  number of  columns  for  which  tracing
C                  extrema have been assigned.
C     NTR        : The  column   numbers   for   which  tracing
C                  extrema have been assigned.
C     NTIN       : The invert relative addresses of RMIN,RMAX.
```

```
C     NKEY       : Array storing  the numbers  of columns which
C                  will be used as sorting guides.
C     MNE        : Array storing the  names  of  columns  which
C                  will be used as sorting guides.
C     NN         : The number of  repeated  sortings  according
C                  column numbers.
C     NM         : The  number of repeated  sortings  according
C                  column mnemonic names.
C     IJ         : The number of assigned  tracing extrema sets
C     A,B        : Arrays  storing the  mumber of  row on which
C                  starts  a  set of common  name rows  and the
C                  corresponding on which ends the set.
C     NRANK      : Array storing the  numbers  of  columns that
C                  assign the ranking of data
C     MRANK      : Array  storing  the  names  of  columns that
C                  assign the ranking of data
C     NNU        : The number of columns used for ranking.
C     KMN        : The number of column  names used for ranking
C     SUP1,SUP2  : Arrays storing the  number  of  row on which
C                  starts suppression  of   rows    and    the
C                  corresponding on  which ends the suppression
C     SN         : The number of assigned suppressions.
C     NOROW      : Array  the  numbers  of  rows used as guides
C                  for the ranking of the rows.
C     NBROW      : The number  of  row  numbers   used  for row
C                  ranking
C
C**************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 FLAG1(6),NCO(128),FLAG2(4),NKEY(128),NTR(128),
     1          SUP1(128),SUP2(128),NRO(128),NCIN(128),
     2          NTIN(128),NRIN(2048),NOROW(2048),SN,NRANK(128),
      REAL*4 RMIN(128),RMAX(128)
      CHARACTER*1 IND
      CHARACTER*24 COLMNE(128),MNE(128),MRANK(128),
     1             ROWMNE1(2048),ROWMNE2(128)

      NBYTES=NCOL*4
      IDUM=1
      DO WHILE (IDUM.EQ.1)
         IDUM=0
         WRITE (IMP,100)
         READ (LEC,200) IND
         IF (IND.EQ.'1') THEN
            CALL COLINF(LEC,IMP,FLAG1,NCOL,NROW,COLMNE,NC,
     1                  NCO,NCIN,RMIN,RMAX,NT,NTR,NTIN,NKEY,
     2                  MNE,NN,NM,NRANK,MRANK,NNU,KMN)
            IDUM=1
         ELSE IF (IND.EQ.'2') THEN
            CALL ROWINF(LEC,IMP,FLAG2,NCOL,NROW,ROWMNE1,
     1                 NR,NRO,NRIN,ROWMNE2,IJ,A,B,SUP1,SUP2,
     2                 SN,NOROW,NBROW)
```

```
              IDUM=1
           ELSE IF (IND.EQ.' ') THEN
              RETURN
           ELSE
              WRITE(IMP,300)
              IDUM=1
           END IF
        END DO
100     FORMAT (///4X,'Column information :1'/4X,
        1             'Row information    :2',/'S',3X,
        2             'Answer             :')
200     FORMAT (A24)
300     FORMAT(//6X,'INVALID CHARACTER!!')
        END

  C**************************************************************
        SUBROUTINE COLINF(LEC,IMP,FLAG1,NCOL,NROW,COLMNE,NC,NCO,
        1                 NCIN,RMIN,RMAX,NT,NTR,NTIN,NKEY,MNE,
        2                 NN,NM,NRANK,MRANK,NNU,KMN)
  C**************************************************************
  C
  C     ARGUMENTS
  C     ---------
  C
  C     LEC       : The logical unit number for  writing on  the
  C                 terminal.
  C     IMP       : The logical unit number for reading from the
  C                 terminal.
  C     FLAG1     : Integer array of  six  elements used as flag
  C                 to  indicate  the  existence  or  not of the
  C                 several informations regarding the columns.
  C     FLAG2     : The corresponding flag for rows.
  C     NCOL      : The number of columns.
  C     NROW      : The number of rows.
  C     COLMNE    : Array storing the column mnemonic names.
  C     NC        : The number of assigned column mnemonic names.
  C     NCO       : The numbers of  columns for which names have
  C                 been assigned.
  C     NCIN      : The invert relative addresses of COLMNE.
  C     RMIN,RMAX : Arrays  storing  the   minimum and   maximum
  C                 values between which  tracing of the data is
  C                 going to take place.
  C     NT        : The  number of  columns for   which  tracing
  C                 extrema have been assigned.
  C     NTR       : The column numbers for which tracing extrema
  C                 have been assigned.
  C     NTIN      : The invert relative addresses of RMIN,RMAX.
  C     NKEY      : Array storing the  numbers  of columns which
  C                 will be used as sorting guides.
  C     MNE       : Array  storing  the  names  of columns which
  C                 will be used as sorting guides.
  C     NN        : The number  of  repeated  sortings according
  C                 column numbers.
```

```
C      NM          : The number  of  repeated  sortings according
C                    column  mnemonic names.
C      NRANK       : Array storing  the  numbers  of columns that
C                    assign the ranking of data
C      MRANK       : Array storing  the  names  of  columns  that
C                    assign the ranking of data
C      NNU         : The number of columns used for ranking.
C      KMN         : The number of column names used for ranking.
C
C*********************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 FLAG1(6),NTIN(128),NKEY(128),NRANK(128),
     1          NCO(128),NCIN(128),NTR(128)
      REAL*4 RMIN(128),RMAX(128)
      CHARACTER*2 IND
      CHARACTER*24 COLMNE(128),MNE(128),MRANK(128)

      IDUM=1
      DO WHILE (IDUM.EQ.1)
         IDUM=0
         WRITE(IMP,100)
         READ (LEC,200) IND
         IF (IND.EQ.'1') THEN
            WRITE(IMP,500)
            READ(LEC,600) NCL
            IF (NCL.LE.128) THEN
               NCOL=NCL
            ELSE
               WRITE(IMP,800)
               IDUM=1
            END IF
            IDUM=1
         ELSE IF (IND.EQ.'2') THEN
            IF (NCOL.NE.0) THEN
               CALL COLNAM(LEC,IMP,FLAG1,NCOL,COLMNE,NC,NCO,
     1                     NCIN)
               IDUM=1
            ELSE
               WRITE(IMP,400)
               IDUM=1
            END IF
         ELSE IF (IND.EQ.'3') THEN
            IF (NCOL.NE.0) THEN
               CALL TRSFRM(LEC,IMP,FLAG1,NCOL,NROW,COLMNE,NC,
     1                     NCO,NCIN)
               IDUM=1
            ELSE
               WRITE(IMP,400)
               IDUM=1
            END IF
         ELSE IF (IND.EQ.'4') THEN
```

```
           IF (NCOL.NE.0) THEN
              CALL EXTREM(LEC,IMP,FLAG1,NCOL,NROW,RMIN,RMAX,
     1                    NT,NTR,NTIN)
              IDUM=1
           ELSE
              WRITE(IMP,400)
              IDUM=1
           END IF
        ELSE IF (IND.EQ.'5') THEN
           IF (NCOL.NE.0) THEN
              CALL MULSOR(LEC,IMP,NCOL,NROW,FLAG1,NKEY,MNE,
     1                    NN,NM,COLMNE)
              IDUM=1
           ELSE
              WRITE(IMP,400)
              IDUM=1
           END IF
        ELSE IF (IND.EQ.'6') THEN
           IF (NCOL.NE.0) THEN
              CALL RANDOM(LEC,IMP,NCOL,NROW)
              IDUM=1
           ELSE
              WRITE(IMP,400)
              IDUM=1
           END IF
        ELSE IF (IND.EQ.'7') THEN
           IF (NCOL.NE.0) THEN
              CALL CRANK(LEC,IMP,NCOL,NROW,FLAG1,NRANK,MRANK,
     1                     NNU,KMN,COLMNE)
              IDUM=1
           ELSE
              WRITE(IMP,400)
              IDUM=1
           END IF
        ELSE IF (IND.EQ.'8') THEN
           CALL CDISP (LEC,IMP,FLAG1,NCOL,COLMNE,NC,NCO,NCIN,
     1                      RMIN,RMAX,NT,NTR,NTIN,NKEY,MNE,
     2                      NN,NM,NRANK,MRANK,NNU,KMN)
           IDUM=1
        ELSE IF (IND.EQ.'9') THEN
           CALL COLMOD (LEC,IMP,FLAG1,NCOL,NROW,COLMNE,NC,
     1                  NCO,NCIN,RMIN,RMAX,NT,NTR,NTIN,NKEY,
     2                  MNE,NN,NM,NRANK,MRANK,NNU,KMN)
           IDUM=1
        ELSE IF (IND.EQ.'10') THEN
           IF (NCOL.NE.0.AND.NROW.NE.0) THEN
              CALL SEEDAT(LEC,IMP,NCOL,NROW,1,1)
              IDUM=1
           ELSE
              WRITE(IMP,700)
              IDUM=1
           END IF
           ELSE IF (IND.EQ.'11') THEN
              IF (NCOL.NE.0.AND.NROW.NE.0) THEN
                 CALL DISDAT(LEC,IMP,NCOL,NROW,1,1)
                 IDUM=1
              ELSE
                 WRITE(IMP,700)
                 IDUM=1
              END IF
           ELSE IF (IND.EQ.' ') THEN
              RETURN
           ELSE
              WRITE(IMP,300)
              IDUM=1
           END IF
        END DO
100     FORMAT(///4X,'SELECTION TABLE FOR COLUMN INFORMATIONS'/
        1         4X,'-----------------------------------------'/
        2            7X,'Number of columns (mandatory)  :1'/7X,
        3               'Mnemonic names                 :2'/7X,
        4               'Transformations                :3'/7X,
        5               'Tracing extrema                :4'/7X,
        6               'Multi-sorting column guides    :5'/7X,
        7               'Randomization of data          :6'/7X,
        8               'Column ranking                 :7'/7X,
        9               'Display column informations    :8'/7X,
        1               'Modification of column info    :9'/7X,
        2               'Data retrieval                 :10'/7X,
        3               'Display/print of data file     :11'//
        4        '$',3X,'Answer :')
200     FORMAT (A24)
300     FORMAT(//3X,'INVALID CHARACTER!!')
400     FORMAT(///4X,'NUMBER OF COLUMNS HAS NOT BEEN',
        1          /10X, ' ASSIGNED!!')
500     FORMAT(//'$',3X,'Assign number of columns :')
600     FORMAT(I5)
700     FORMAT(///4X,'NUMBER OF COLUMNS AND ROWS MUST BE'
        1          /12X,'ASSIGNED TO DISPLAY THE DATA FILE!!')
800     FORMAT(///4X,'THE ASSIGNED NUMBER OF COLUMNS EXEEDS'/
        1         4X,'      THE MAXIMUM PERMITTED 128')
        RETURN
        END

  C***************************************************************
        SUBROUTINE ROWINF(LEC,IMP,FLAG2,NCOL,NROW,ROWMNE1,NR,
        1                 NROW,NRIN,ROWMNE2,IJ,A,B,SUP1,SUP2,
        2                 SN,NOROW,NBROW)
  C***************************************************************
  C
  C      This subroutine is used for assignment of  the  several
  C     row informations
  C     ARGUMENTS
  C     ---------
```

```
C
C     LEC       : The logical unit number for  writing on  the
C                 terminal.
C     IMP       : The logical unit number for reading from the
C                 terminal.
C     FLAG2     : The corresponding flag for rows.
C     NCOL      : The number of columns.
C     NROW      : The number of rows.
C     ROWMNE1   : Array storing the row mnemonic names.
C     NR        : The number of assigned row mnemonic names.
C     NRO       : The  numbers  of  rows  for which names have
C                 been assigned.
C     NRIN      : The invert relative addresses of ROWMNE.
C     ROWMNE2   : Array storing the mnemonics of sets of rows.
C     IJ        : The number of assigned tracing extrema sets.
C     A,B       : Arrays storing the  mumber  of  row on which
C                 starts a  set of  common name rows  and the
C                 corresponding on which ends the set.
C     SUP1,SUP2 : Arrays  storing the  number of  row on which
C                 starts  suppression  of  rows  and   the
C                 corresponding  on which ends the suppression.
C     SN        : The number of assigned suppressions.
C     NOROW     : Array the numbers of rows used as guides for
C                 the ranking of the rows.
C     NBROW     : The  number  of  row  numbers  used  for row
C                 ranking.
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 SN,FLAG1(6),FLAG2(4),NKEY(128),NRANK(128),
      1         B(128),SUP1(128),SUP2(128),NRO(128),
      2         NRIN(2048),NOROW(2048),A(128),IJCOMP(128)
      REAL*4 RMIN(128),RMAX(128)
      CHARACTER*2 IND
      CHARACTER*7 MODROW
      CHARACTER*24 FNAME,COLMNE(128),MNE(128),MRANK(128),
      1            ROWMNE1(2048),ROWMNE2(128)
      DIMENSION NOCOL(256)
      DATA MODROW/'rows'/

      IDUM=1
      DO WHILE (IDUM.EQ.1)
         IDUM=0
         WRITE(IMP,100)
         READ(LEC,200) IND
         IF (IND.EQ.'1') THEN
            WRITE(IMP,500)
            READ(LEC,600) NROW
            IDUM=1
         ELSE IF (IND.EQ.'2') THEN
```

```
           IF (NROW.NE.0) THEN
              CALL ROWNAM(LEC,IMP,FLAG2,NROW,ROWMNE1,NR,NRO,
     1                      NRIN,ROWMNE2,IJ,A,B)
              IDUM=1
           ELSE
              WRITE(IMP,400)
              IDUM=1
           END IF
        ELSE IF (IND.EQ.'3')THEN
           IF (NROW.NE.0) THEN
              CALL ROWSUP(LEC,IMP,FLAG2,NCOL,NROW,SUP1,SUP2,
     1                    SN,ROWMNE2,A,B)
              IDUM=1
           ELSE
              WRITE(IMP,400)
              IDUM=1
           END IF
        ELSE IF (IND.EQ.'4') THEN
           CALL REJECT(LEC,IMP,NROW,IJCOMP,1,1,1,1,1)
           IDUM=1
        ELSE IF (IND.EQ.'5') THEN
           CALL ANADIS(LEC,IMP,NROW,NROW,NOROW,NBROW,MODROW)
           CALL REORD(LEC,IMP,NCOL,NROW,FLAG2,NOROW,NBROW)
           IDUM=1
        ELSE IF (IND.EQ.'6') THEN
           CALL RDISP (LEC,IMP,FLAG2,NROW,ROWMNE1,NR,NRO,
     1                 NRIN,ROWMNE2,IJ,A,B,SUP1,SUP2,SN,
     2                 NOROW,NBROW)
           IDUM=1
        ELSE IF (IND.EQ.'7') THEN
           CALL ROWMOD(LEC,IMP,FLAG2,NCOL,NROW,ROWMNE1,NR,NRO,
     1                 NRIN,ROWMNE2,IJ,A,B,SUP1,SUP2,SN,NOROW,
     2                 NBROW)
           IDUM=1
        ELSE IF (IND.EQ.'8') THEN
           IF (NCOL.NE.0.AND.NROW.NE.0) THEN
              CALL SEEDAT(LEC,IMP,NCOL,NROW,1,1)
              IDUM=1
           ELSE
              WRITE(IMP,700)
              IDUM=0
           END IF
        ELSE IF (IND.EQ.'9') THEN
           IF (NCOL.NE.0.AND.NROW.NE.0) THEN
              CALL DISDAT(LEC,IMP,NCOL,NROW,1,1)
              IDUM=1
           ELSE
              WRITE(IMP,700)
              IDUM=0
           END IF
        ELSE IF (IND.EQ.' ') THEN
```

```
            RETURN
         ELSE
            WRITE(IMP,300)
            IDUM=1
         END IF
      END DO
100      FORMAT(///4X,'SELECTION TABLE FOR ROW INFORMATIONS',
     1            4X,'-----------------------------------'/
     2            7X,    'Number of rows (mandatory)     :1'/7X,
     3                   'Mnemonic names                 :2'/7X,
     4                   'Row suppression                :3'/7X,
     5                   'Row rejection                  :4'/7X,
     6                   'Row ranking                    :5'/7X,
     7                   'Display row informations       :6'/7X,
     8                   'Modification of row info       :7'/7X,
     9                   'Data retrieval                 :8'/7X,
     1                   'Display/print of data file     :9'//
     2           '$',3X,'Answer :')
200   FORMAT(A24)
300   FORMAT(///4X,'INVALID CHARACTER!!')
400   FORMAT(///4X,'NUMBER OF ROWS HAS NOT BEEN ASSIGNED!!')
500   FORMAT(//'$',3X,'Assign number of rows :')
600   FORMAT(I3)
700   FORMAT(///4X,'NUMBER OF COLUMNS AND ROWS MUST BE'
     1        /4X,'ASSIGNED TO DISPLAY THE DATA FILE!!')
      RETURN
      END

C*****************************************************************
      SUBROUTINE COLNAM(LEC,IMP,FLAG1,NCOL,COLMNE,NC,NCO,NCIN)
C*****************************************************************
C
C      This subroutine is  used  for  assignment  of  names to
C      columns
C     ARGUMENTS
C     ---------
C
C     LEC       : The logical unit number  for  writing on the
C                 terminal.
C     IMP       : The logical unit number for reading from the
C                 terminal.
C     FLAG1     : Integer  array  of six elements used as flag
C                 to  indicate  the  existence  or  not of the
C                 several informations regarding the columns.
C     COLMNE    : Array storing the column mnemonic names.
C     NC        : The number of assigned column mnemonic names.
C     NCO       : The numbers of columns  for which names have
C                 been assigned.
C     NCIN      : The invert relative addresses of COLMNE.
C
C*****************************************************************
```

```
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 LEC,IMP,NCOL,FLAG1(6),NCO(128),NCIN(128)
      CHARACTER*24 COLMNE(128),NAME

      FLAG1(1)=1
      I=1
      NC=1
      DO WHILE (I.NE.0)
         WRITE(IMP,100)
         READ(LEC,200) I
         IF (I.LE.NCOL) THEN
            IF (I.EQ.0) THEN
               NC=NC-1
               CALL EXSH1(NCO,NCIN,NC,1)
               RETURN
            ELSE
               NCO(NC)=I
               WRITE(IMP,300)
               READ(LEC,400) NAME
               COLMNE(NC)=NAME
            END IF
         ELSE
            WRITE(IMP,500)
         END IF
         NC=NC+1
      END DO

100   FORMAT(/'$',3X,'Assign column number(<CR> to RETURN) :')
200   FORMAT(I5)
300   FORMAT('$',3X,'Assign mnemonic                       :')
400   FORMAT(A24)
500   FORMAT(//4X,'THE ASSIGNED COLUMN NUMBER EXEEDS ',
      1            /7X,'THE NUMBER OF COLUMNS!!')
      RETURN
      END

C*************************************************************
      SUBROUTINE TRSFRM(LEC,IMP,FLAG1,NCOL,NROW,COLMNE,NC,NCO,
      1                    NCIN)
C*************************************************************
C
C     ARGUMENTS
C     ---------
C
C     LEC       : The logical unit number for  writing  on the
C                 terminal.
C     IMP       : The logical unit number for reading from the
C                 terminal.
C     FLAG1     : Integer array  of six  elements used as flag
C                 to indicate  the  existence  or  not  of the
C                 several informations regarding the columns.
C     NCOL      : The number of columns.
```

```
  C     NROW        : The number of rows.
  C     COLMNE      : Array storing the column mnemonic names.
  C     NC          : The number of assigned column mnemonic names.
  C     NCO         : The numbers of  columns for which names have
  C                   been assigned.
  C     NCIN        : The invert relative addresses of COLMNE.
  C
  C*************************************************************

        IMPLICIT INTEGER*2 (I-N)
        INTEGER*2 FLAG1(6),NCO(128),NCIN(128)
        CHARACTER*24 COLMNE(128),TRANS
        FLAG1(1)=1
        WRITE(IMP,100)
        INDEX=NC
        TRANS='A'
        DO WHILE(TRANS.NE.' ')
           READ(LEC,200) TRANS
           IF (TRANS.NE.' ') THEN
              NC=NC+1
              NCO(NC)=NCOL+1
              COLMNE(NC)=TRANS
              NCOL=NCOL+1
           END IF
        END DO
        CALL EXSH1(NCO,NCIN,NC,1)
        CALL CONV(NCOL,NROW,COLMNE,NC,INDEX)

100     FORMAT(///6X,'AVAILABLE FUNCTIONS'/6X,
        1            '-------------------'/2X,
        2            '1.-LOGC[X] ',4X, ' 9.-ASIN[X]'/2X,
        3            '2.-LOG[X]  ',4X, '10.-ACOS[X]'/2X,
        4            '3.-EXP[X]  ',4X, '11.-ATAN[X]'/2X,
        5            '4.-ABS[X]  ',4X, '12.-SINH[X]'/2X,
        6            '5.-SIN[X]  ',4X, '13.-COSH[X]'/2X,
        7            '6.-COS[X]  ',4X, '14.-TANH[X]'/2X,
        8            '7.-TAN[X]  ',4X, '15.-SQRT[X]'/2X,
        9            '8.-ASINH[X]'//2X,
        1            'Assign transformations :'/)
200     FORMAT(A24)
        RETURN
        END

  C*************************************************************
        SUBROUTINE EXTREM(LEC,IMP,FLAG1,NCOL,NROW,RMIN,RMAX,NT,
        1                 NTR,NTIN)
  C*************************************************************
  C
  C      This subroutine is used for assignment of tracing
  C     extrema
  C
  C     ARGUMENTS
        ---------
```

```
C
C     LEC        : The logical unit number  for  writing on the
C                  terminal.
C     IMP        : The logical unit number for reading from the
C                  terminal.
C     FLAG1      : Integer  array of six  elements used as flag
C                  to  indicate  the  existence or  not  of the
C                  several informations regarding the columns.
C     NCOL       : The number of columns.
C     RMIN,RMAX  : Arrays  storing  the  minimum   and  maximum
C                  values between which tracing of  the data is
C                  going to take place.
C     NT         : The  number of columns  for  which   tracing
C                  extrema have been assigned.
C     NTR        : The column numbers for which tracing extrema
C                  have been assigned.
C     NTIN       : The invert relative addresses of RMIN,RMAX.
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 LEC,IMP,NCOL,FLAG1(6),NTR(128),NTIN(128)
      REAL*4 RMIN(128),RMAX(128),FMIN(128),FMAX(128)
      CHARACTER*1 IND

      FLAG1(2)=1
      CALL FINDMM(LEC,IMP,NCOL,NROW,FMIN,FMAX)
      IDUM=1
      DO WHILE (IDUM.EQ.1)
         IDUM=0
         WRITE(IMP,100)
         READ (LEC,200) IND
         IF (IND.EQ.'0') THEN
            RETURN
         ELSE IF (IND.EQ.'1') THEN
            I=1
            NT=1
            DO WHILE (I.NE.0)
               WRITE(IMP,700)
               READ(LEC,800) I
               IF (I.NE.0) THEN
                  NTR(NT)=I
                  WRITE(IMP,300) FMIN(I)
                  READ(LEC,400) RMIN(NT)
                  WRITE(IMP,500) FMAX(I)
                  READ(LEC,400) RMAX(NT)
                  NT=NT+1
               END IF
            END DO
            NT=NT-1
         ELSE IF (IND.EQ.' ') THEN
         RETURN
       ELSE
         WRITE(IMP,600)
         IDUM=1
       END IF
     END DO
     CALL EXSH1(NTR,NTIN,NT,1)
100  FORMAT(///4X,'Imposed extrema for tracing :1'/4X,
     1   'Original extrema            :0'/'$',3X,
     2   'Answer                      :')
200  FORMAT(A24)
300  FORMAT('$',3X,'MIN (found) :',F14.6,2X,'Imposed :')
400  FORMAT(F14.6)
500  FORMAT('$',3X,'MAX (found) :',F14.6,2X,'Imposed :')
600  FORMAT(//3X,'INVALID CHARACTER!!')
700  FORMAT(//'$',3X,'Assign column number(<CR> to RETURN':')
800  FORMAT(I5)
     RETURN
     END

 C*************************************************************
     SUBROUTINE MULSOR(LEC,IMP,NCOL,NROW,FLAG1,NKEY,MNE.NN,
     1                 NM,COLMNE)
 C*************************************************************
 C
 C     This subroutine is used for assignment of sorting
 C    column guides
 C
 C    ARGUMENTS
 C    ---------
 C
 C    LEC       : The logical unit number  for  writing on the
 C                terminal.
 C    IMP       : The logical unit number for reading from the
 C                terminal.
 C    NCOL      : The number of columns.
 C    NROW      : The number of rows.
 C    FLAG1     : Integer  array  of six  elements used as flag
 C                to  indicate  the  existence  or  not  of the
 C                several informations regarding the columns.
 C    NKEY      : Array storing  the  numbers  of columns which
 C                will be used as sorting guides.
 C    MNE       : Array storing the names of columns which will
 C                be used as sorting guides.
 C    NN        : The  number  of  repeated sortings  according
 C                column numbers.
 C    NM        : The  number  of  repeated sortings  according
 C                column mnemonic names.
 C    COLMNE    : Array storing the column mnemonic names.
 C
 C*************************************************************
```

```
        IMPLICIT INTEGER*2 (I-N)
        INTEGER*2 FLAG1(6),NKEY(128)
        CHARACTER*1 IND
        CHARACTER*7 MODCOL
        CHARACTER*24 MNE(128),COLMNE(128)
        DATA MODCOL/'columns'/

        IDUM=1
        DO WHILE (IDUM.EQ.1)
           IDUM=0
           WRITE(IMP,100)
           READ(LEC,200) IND
           IF (IND.EQ.'1') THEN
              IF (FLAG1(4).EQ.0) THEN
                 FLAG1(3)=1
                 CALL ANADIS(LEC,IMP,NCOL,NCOL,NKEY,NN,MODCOL)
                 CALL REAR1(LEC,IMP,NCOL,NROW,NKEY,NN)
              ELSE
                 WRITE(IMP,700)
                 RETURN
              END IF
           ELSE IF (IND.EQ.'2') THEN
              IF (FLAG1(1).EQ.1) THEN
                 IF (FLAG1(3).EQ.0) THEN
                    FLAG1(4)=1
                    WRITE(IMP,400)
                    CALL ASBYMN(LEC,IMP,NCOL,COLMNE,NM,MNE)
                    CALL REAR2(LEC,IMP,NCOL,NROW,COLMNE,MNE,NM)
                 ELSE
                    WRITE(IMP,800)
                    RETURN
                 END IF
              ELSE
                 WRITE(IMP,500)
              END IF
           ELSE
              WRITE(IMP,600)
              IDUM=1
           END IF
        END DO
        RETURN

100     FORMAT(///4X,'By column number :1'/4X,
        1            'By mnemonic name :2'
        2            /,'$',3X,'Answer           :')
200     FORMAT(A24)
400     FORMAT(///4X,'Assign column mnemonic names :')
500     FORMAT(//4X,'NO MNEMONIC NAMES HAVE BEEN ASSINGED!!')
600     FORMAT(//4X,'INVALID CHARACTER!!')
700     FORMAT(//4X,'MULTI-SORTING GUIDES HAVE ALREADY BEEN'/
        1            9X ' ASSIGNED BY MNEMONIC NAMES')
800     FORMAT(//4X,'MULTI-SORTING GUIDES HAVE ALREADY BEEN'/
      1             9X ' ASSIGNED BY COLUMN NUMBERS')
      END
C*************************************************************
      SUBROUTINE CRANK(LEC,IMP,NCOL,NROW,FLAG1,NRANK,MRANK,
      1                NNU,KMN,COLMNE)
C*************************************************************
C
C      This subroutine is used to assign column ranking by
C      column number or column mnemonic names.
C
C     ARGUMENTS
C     ---------
C
C     LEC         : The logical unit  number  for writing on the
C                   terminal.
C     IMP         : The logical unit number for reading from the
C                   terminal.
C     NCOL        : The number of columns.
C     NROW        : The number of rows.
C     FLAG1       : Integer  array of six  elements used as flag
C                   to  indicate  the  existence or  not  of the
C                   several informations regarding the columns.
C     NRANK       : Array storing the  numbers  of  columns that
C                   assign the ranking of data
C     MRANK       : Array  storing  the  names  of  columns that
C                   assign the ranking of data
C     NNU         : The number of columns used for ranking.
C     KMN         : The number of column names used for ranking.
C     COLMNE      : Array storing the column mnemonic names.
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 LEC,IMP,NCOL,NNU,KMN,FLAG1(6),NRANK(128)
      CHARACTER*1 IND
      CHARACTER*7 MODCOL
      CHARACTER*24 MNE(128),COLMNE(128),MRANK(128)
      DATA MODCOL/'columns'/

      IDUM=1
      DO WHILE (IDUM.EQ.1)
         IDUM=0
         WRITE(IMP,100)
         READ(LEC,400) IND
         IF (IND.EQ.'1') THEN
            IF (FLAG1(6).EQ.0) THEN
               FLAG1(5)=1
               CALL ANADIS(LEC,IMP,NCOL*2,NCOL,NRANK,NNU,
      1                    MODCOL)
               CALL RANK1(LEC,IMP,NCOL,NROW,NRANK,NNU)
```

```
              ELSE
                 WRITE(IMP,700)
                 RETURN
              END IF
           ELSE IF (IND.EQ.'2') THEN
              IF (FLAG1(5).EQ.0) THEN
                 IF (FLAG1(1).EQ.1) THEN
                    FLAG1(6)=1
                    WRITE(IMP,300)
                    CALL ASBYMN(LEC,IMP,NCOL,COLMNE,KMN,MRANK)
                    CALL RANK2(LEC,IMP,NCOL,NROW,MRANK,KMN,
        1                      COLMNE)
                 ELSE
                    WRITE(IMP,500)
                    RETURN
                 END IF
              ELSE
                 WRITE(IMP,800)
              END IF
           ELSE
              WRITE(IMP,600)
              IDUM=1
           END IF
        END DO

100     FORMAT(///4X,'By column mumber  :1'/4X,
        1            'By mnemonic names :2'/
        2     '$',3X,'Answer            :')
300     FORMAT(///3X,'Assign ranking by sequence of names :')
400     FORMAT(A24)
500     FORMAT(///4X,'NO MNEMONIC NAMES HAVE BEEN ASSINGED!!')
600     FORMAT(///4X,'INVALID CHARACTER!!')
700     FORMAT(///4X,'RANKING HAS ALREADY BE ASSIGNED'/6X,
        1            ' BY COLUMN MNEMONIC NAMES!!')
800     FORMAT(///4X,'RANKING HAS ALREADY BE ASSIGNED'/6X,
        1            ' BY COLUMN NUMBERS!!')
        RETURN
        END

  C****************************************************************
        SUBROUTINE ASBYMN (LEC,IMP,NCOL,COLMNE,N,AR)
  C****************************************************************
  C
  C     ARGUMENTS
  C     ---------
  C
  C     LEC    : The logical unit  number  for  writing  on  the
  C              terminal.
  C     IMP    : The logical unit number for  reading  from  the
  C              terminal.
  C     NCOL   : The number of columns.
  C     COLMNE : Array storing the column mnemonic names.
```

```
C     N      : The number of the assigned mnemonics (index).
C     AR     : Array storing the assigned mnemonics.
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 NCOL,N
      CHARACTER*24 COLMNE(128),AR(128),NAME
      LOGICAL*1 TEST

      N=0
      NAME='A'
      DO WHILE (NAME.NE.' ')
         IDUM=1
         DO WHILE(IDUM.EQ.1)
            IDUM=0
            READ(LEC,100) NAME
            CALL TESTER(NCOL,NAME,COLMNE,TEST)
            IF (N.LT.NCOL)THEN
               IF (TEST.EQ..TRUE.) THEN
                  N=N+1
                  AR(N)=NAME
               ELSE IF (TEST.EQ..FALSE..AND.NAME.NE.' ') THEN
                  WRITE(IMP,200)
                  IDUM=1
               END IF
            ELSE
               WRITE(IMP,300)
               RETURN
            END IF
         END DO
      END DO
100   FORMAT(A24)
200   FORMAT(///4X,'THE ASSIGNED MNEMONIC HAS NOT BEEN'/6X,
      1        ' ASSIGNED AS COLUMN MNEMONIC!!'//4X,
      2        'Assign next mnemonic')
300   FORMAT(///4X,'THE ASSIGNED NUMBER OF MNEMONICS IS '
      1        /4X, 'GREATER THAN THE NUMBER OF COLUMNS!!'/4X,
      2        'THE LAST ONE IS OMMITED')
      RETURN
      END

C*************************************************************
      SUBROUTINE ROWNAM(LEC,IMP,FLAG2,NROW,ROWMNE1,NR,NRO,
      1                 NRIN,ROWMNE2,IJ,A,B)
C*************************************************************
C
C      This subroutine is used for assignment of row mnemonics
C
C     ARGUMENTS
C     ---------
C
```

```
C     LEC     : The logical unit  number  for  writing  on  the
C               terminal.
C     IMP     : The logical unit number for  reading  from  the
C               terminal.
C     FLAG2      : The corresponding flag for rows.
C     NROW       : The number of rows.
C     ROWMNE1    : Array storing the row mnemonic names.
C     NR         : The number of assigned row mnemonic names.
C     NRO        : The  numbers  of  rows for  which names have
C                  been assigned.
C     NRIN       : The invert relative addresses of ROWMNE.
C     ROWMNE2    : Array storing the mnemonics of sets of rows.
C     IJ         : The number of assigned tracing extrema sets.
C     A,B        : Arrays storing the  mumber  of  row on which
C                  starts  a  set of common  name rows  and the
C                  corresponding on which ends the set.
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 LEC,IMP,NROW,IJ,FLAG2(4),A(128),B(128),
      1         NRO(128),NRIN(2048)
      CHARACTER*1 IND
      CHARACTER*24 ROWMNE1(2048),ROWMNE2(128),NAME

      IDUM=1
      DO WHILE (IDUM.EQ.1)
         IDUM=0
         WRITE(IMP,400)
         READ(LEC,200) IND
         IF (IND.EQ.'1') THEN
            FLAG2(1)=1
            I=1
            NR=1
            DO WHILE(I.NE.0)
               WRITE(IMP,510)
               READ(LEC,100) I
               IF (I.EQ.0) THEN
                  NR=NR-1
                  CALL EXSH1(NRO,NRIN,NR,1)
                  RETURN
               ELSE
                  NRO(NR)=I
                  WRITE(IMP,500)
                  READ(LEC,200) NAME
                  ROWMNE1(NR)=NAME
               END IF
               NR=NR+1
            END DO
         ELSE IF (IND.EQ.'2') THEN
            FLAG2(2)=1
            IJ=1
```

```
              I=1
              DO WHILE(I.NE.0)
                 IDUM=1
                 DO WHILE(IDUM.EQ.1)
                    IDUM=0
                    WRITE(IMP,600) IJ
                    READ(LEC,100) I
                    IF (I.NE.0) THEN
                       IF (I.LE.NROW) THEN
                          A(IJ)=I
                       ELSE
                          WRITE(IMP,800)
                          IDUM=1
                       END IF
                    ELSE
                       IJ=IJ-1
                       RETURN
                    END IF
                 END DO
                 JDUM=1
                 DO WHILE(JDUM.EQ.1)
                    JDUM=0
                    WRITE(IMP,700)
                    READ(LEC,100) J
                    IF (J.LE.NROW) THEN
                       B(IJ)=J
                       WRITE(IMP,900)
                       READ(LEC,200) ROWMNE2(IJ)
                       IJ=IJ+1
                    ELSE
                       WRITE(IMP,800)
                       JDUM=1
                    END IF
                 END DO
              END DO
              IDUM=1
           ELSE IF(IND.EQ.' ') THEN
              RETURN
           ELSE
              WRITE(IMP,1000)
              IDUM=1
           END IF
        END DO
        RETURN

100     FORMAT(I5)
200     FORMAT(A24)
400     FORMAT(///4X,'Row by row     :1'/4X,
        1            'By set of rows :2'/'$',3X,
        2            'Answer         :')
500     FORMAT('$',3X,'Assign mnemonic                   :')
510     FORMAT(/'$',3X'Assign row number(<CR> to RETURN) :')
```

```
600   FORMAT(///4X,'Common name set :',I5/'$',3X,
     1             'From row        :')
700   FORMAT('$',3X,'To row          :')
800   FORMAT(///3X,'THE ASSIGNED ROW NUMBER EXEEDS THE NUMBER',
     1       ' OF ROWS!!')
900   FORMAT('$',3X,'Name            :')
1000  FORMAT(///5X,'INVALID CHARACTER!!')
      END

C**************************************************************
      SUBROUTINE ROWSUP(LEC,IMP,FLAG2,NCOL,NROW,SUP1,SUP2,SN,
     1                  ROWMNE2,A,B)
C**************************************************************
C
C     Row suppression
C     ARGUMENTS
C     ---------
C
C     LEC    : The logical unit  number  for  writing  on  the
C              terminal.
C     IMP    : The logical unit number for  reading  from  the
C              terminal.
C     FLAG2     : The corresponding flag for rows.
C     NCOL      : The number of columns.
C     NROW      : The number of rows.
C     SUP1,SUP2 : Arrays storing the  number  of  row on which
C                 starts   suppression  of  rows   and    the
C                 corresponding on which ends the suppression.
C     SN        : The number of assigned suppressions.
C     ROWMNE2   : Array storing the mnemonics of sets of rows.
C     A,B       : Arrays storing the  mumber  of  row on which
C                 starts  a  set of common  name  rows and the
C                 corresponding on which ends the set.
C
C**************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 SN,FLAG2(4),SUP1(128),SUP2(128),A(128),B(128)
      CHARACTER*1 IND
      CHARACTER*24 NAME,ROWMNE2(128)
      FLAG2(3)=1
      WRITE(IMP,50)
      READ(LEC,60) IND
      IF (IND.EQ.'1') THEN
         CALL SUP(LEC,IMP,NCOL,NROW,SUP1,SUP2,SN)
      ELSE IF (IND.EQ.'2') THEN
         IF (FLAG2(2).EQ.1) THEN
            SN=1
            NAME='A'
            DO WHILE(NAME.NE.' ')
               WRITE(IMP,70) SN
               READ(LEC,80) NAME
               I=1
               DO WHILE(NAME.NE.ROWMNE2(I))
                  I=I+1
               END DO
               IF (NAME.NE.' ') THEN
                  SUP1(SN)=A(I)
                  SUP2(SN)=B(I)
               ELSE
                  SN=SN-1
                  CALL SUPSET(LEC,IMP,NCOL,NROW,SUP1,SUP2,SN)
                  RETURN
               END IF
               SN=SN+1
            END DO
         ELSE
            WRITE(IMP,500)
            RETURN
         END IF
      END IF
50    FORMAT(///4X,'By mumber of rows :1'/4X,
      1             'By row set name   :2'/'$',3X,
      2             'Answer            :')
60    FORMAT(A)
70    FORMAT(///4X,'Suppression :'I5,/'$',3X,'Set name    :')
80    FORMAT(A24)
100   FORMAT(///4X,'Suppression :',I5/'$',3X,'From row   :')
200   FORMAT(I5)
300   FORMAT(///4X,'THE ASSIGNED NUMBER IS GREATER THAN THE',
      1      /18X,              ' NUMBER OF ROWS')
400   FORMAT('$',3X,'To row     :')
500   FORMAT(///4X,'SETS WITH COMMON MNEMONICS HAVE'/
      1       10X,     ' NOT BEEN ASSIGNED!!')
      END

C***************************************************************
      SUBROUTINE SUP(LEC,IMP,NCOL,NROW,SUP1,SUP2,SN)
C***************************************************************
C
C     ARGUMENTS
C     ---------
C
C     LEC    : The logical unit  number  for  writing  on  the
C              terminal.
C     IMP    : The logical unit number for  reading  from  the
C              terminal.
C     NCOL      : The number of columns.
C     NROW      : The number of rows.
C     SUP1,SUP2 : Arrays storing the  number  of  row on which
C                 starts  suppression  of  rows    and    the
C                 corresponding  on which ends the suppression.
C     SN        : The number of assigned suppressions.
C
```

```
C***************************************************************

      IMPLICIT INTEGER*2 (I-N)
      REAL*4 Y(128)
      INTEGER*2 SN,SUP1(128),SUP2(128),NOROW(2048),NO(2048)
      CHARACTER* 7 MODROW
      CHARACTER*45 DEVDIR,NAME*10

      DATA MODROW/'rows'/DEVDIR,NAME/'DUA0 :',' '/
      1    LOGOLD,LOGNEW/2,3/

C     Assign files.
      WRITE(IMP,100)
      NBYTES=NCOL*4
      CALL FICH ('069',LOGOLD,1,DEVDIR,NAME,NROW,NBYTES,1,
      1          'DIRECT',LEC,IMP)
      WRITE(IMP,200)
      NBYTES=NCOL*4
      CALL FICH ('069',LOGNEW,1,DEVDIR,NAME,NROW,NBYTES,0,
      1          'DIRECT',LEC,IMP)

C     Input chain of numbers, ',' or '-' to assign
C     suppressions.

      CALL ANADIS(LEC,IMP,NROW,NROW,NOROW,NBROW,MODROW)

C     Sort NOROW()

      CALL EXSH1(NOROW,NO,NBROW,1)

C     Phase  I : Copy non-suppressed Data.
C     ------------------------------------
      J=0
      K=1
      DO I=1,NROW
         READ    (LOGOLD'I) (Y(L),L=1,NCOL)
         IF (I.EQ.NOROW(K)) THEN
            K=K+1
         ELSE
            J=J+1
            WRITE(LOGNEW'J) (Y(L),L=1,NCOL)
         END IF
      END DO

C     Phase II : SUP1, SUP2 and SN generation for Header.
C     ---------------------------------------------------
      SUP1(1)=NOROW(1)
      SN=0
      DO I=1,NBROW
         IF(NOROW(I+1).NE.NOROW(I)+1) THEN
            SN=SN+1
            SUP2(SN  )=NOROW(I  )
```

```
              SUP1(SN-1)=NOROW(I-1)
           END IF
        END DO
        NROW=NROW-NBROW
        CLOSE(LOGOLD)
        CLOSE(LOGNEW)

        RETURN

100     FORMAT(/'$',3X,'Assign no. of the ''OLD'' direct',
        1      ' access file :'/4X,44('-'))
200     FORMAT(/'$',3X,'Assign no. of the ''NEW'' direct',
        1      ' access file :'/4X,44('-'))

        END

  C*************************************************************
        SUBROUTINE SUPSET(LEC,IMP,NCOL,NROW,SUP1,SUP2,SN)
  C*************************************************************
  C
  C     Used to supress the rows that correspond to the assigned
  C     set with common name.
  C
  C     ARGUMENTS
  C     ---------
  C
  C     LEC    : The logical unit  number  for  writing  on  the
  C              terminal.
  C     IMP    : The logical unit number for  reading  from  the
  C              terminal.
  C     NCOL      : The number of columns.
  C     NROW      : The number of rows.
  C     SUP1,SUP2 : Arrays storing the  number  of  row on which
  C                 starts  suppression  of   rows   and    the
  C                 corresponding on which ends the suppression.
  C     SN        : The number of assigned suppressions.
  C
  C*************************************************************

        IMPLICIT INTEGER*2 (I-N)
        REAL*4 X(128),Y(128)
        INTEGER*2 SN,SUP1(128),SUP2(128)
        CHARACTER*45 DEVDIR,NAME*10

        DATA DEVDIR,NAME/'DUA0 :',' '/
        DATA LOGOLD,LOGNEW/2,3/

  C     Assign files.

        WRITE(IMP,100)
        NBYTES=NCOL*4
        CALL FICH ('069',LOGOLD,1,DEVDIR,NAME,NROW,NBYTES,1,
     1       'DIRECT',LEC,IMP)
     WRITE(IMP,200)
     NBYTES=NCOL*4
     CALL FICH ('069',LOGNEW,1,DEVDIR,NAME,NROW,NBYTES,0,
     1       'DIRECT',LEC,IMP)

 C   Supress the row between SUP1() and SUP2().

     J=1
     I=1
     L=1
     DO WHILE (I.LE.NROW)
       IF (J.LE.SN) THEN
         IF (SUP1(J).NE.I) THEN
           READ (LOGOLD'I) (X(K),K=1,NCOL)
           WRITE(LOGNEW'L) (X(K),K=1,NCOL)
           L=L+1
           I=I+1
         ELSE
           I=I+(SUP2(J)-SUP1(J)+1)
           IF (J.LE.SN) THEN
             J=J+1
           END IF
         END IF
       ELSE
         READ (LOGOLD'I) (X(K),K=1,NCOL)
         WRITE(LOGNEW'L) (X(K),K=1,NCOL)
         L=L+1
         I=I+1
       END IF
     END DO
     NROW=L-1
     CLOSE(LOGOLD)
     CLOSE(LOGNEW)

     RETURN

100  FORMAT(/'$',3X,'Assign no. of the ''OLD'' direct',
     1    ' access file :'/4X,44('-'))
200  FORMAT(/'$',3X,'Assign no. of the ''NEW'' direct',
     1    ' access file :'/4X,44('-'))

     END

 C*********************************************************************
     SUBROUTINE WRITER(LOGN,FLAG1,FLAG2,NCOL,NROW,KMN,NNU,NN,
     1           NM,IJ,A,B,COLMNE,NC,NCO,NCIN,RMIN,RMAX,
     2           NT,NTR,NTIN,NKEY,MNE,NRANK,MRANK,
     3           ROWMNE1,NR,NRO,NRIN,ROWMNE2,SUP1,SUP2,
     4           SN,NOROW,NBROW)
 C*********************************************************************
 C
```

```
C     This  subroutine is used  to  write  the   informations
C     regarding the header on a sequential file.
C
C     ARGUMENTS
C       ---------
C
C     LOGN       : The logical unit  for the file on  which the
C                  header,s data will be stored.
C     FLAG1      : Integer  array of six elements used as  flag
C                  to  indicate  the  existence  or not of  the
C                  several informations regarding the columns.
C     FLAG2      : The corresponding flag for rows.
C     NCOL       : The number of columns.
C     NROW       : The number of rows.
C     KMN        : The number of column names used for ranking.
C     NNU        : The number of columns used for ranking.
C     NN         : The  number of repeated   sortings according
C                  column numbers.
C     NM         : The  number  of repeated sortings  according
C                  column mnemonic names.
C     IJ         : The number of assigned tracing extrema sets.
C     A,B        : Arrays  storing  the  mumber of row on which
C                  starts  a  set  of common name rows  and the
C                  corresponding on which ends the set.
C     COLMNE     : Array storing the column mnemonic names.
C     NC         : The number of assigned column mnemonic names.
C     NCO        : The numbers of columns for which names  have
C                  been assigned.
C     NCIN       : The invert relative addresses of COLMNE.
C     RMIN,RMAX  : Arrays  storing  the  minimum  and   maximum
C                  values  between which tracing of the data is
C                  going to take place.
C     NT         : The  number of  columns for  which   tracing
C                  extrema  have been assigned.
C     NTR        : The column numbers for which tracing extrema
C                  have been assigned.
C     NTIN       : The invert relative addresses of RMIN,RMAX.
C     NKEY       : Array storing  the numbers of columns  which
C                  will be used as sorting guides.
C     MNE        : Array  storing  the  names  of columns which
C                  will be used as sorting guides
C     NRANK      : Array storing  the numbers  of  columns that
C                  assign the ranking of data
C     MRANK      : Array storing  the  names  of  columns  that
C                  assign the ranking of data.
C     ROWMNE1    : Array storing the row mnemonic names.
C     NR         : The number of assigned row mnemonic names.
C     NRO        : The  numbers  of  rows  for which names have
C                  been assigned.
C     NRIN       : The invert relative addresses of ROWMNE.
C     ROWMNE2    : Array storing the mnemonics of sets of rows.
C     SUP1,SUP2  : Arrays storing the  number  of  row on which
```

```
C                    starts  suppression  of   rows   and   the
C                    corresponding on which ends the suppression.
C     SN           : The number of assigned suppressions.
C     NOROW        : Array the numbers of rows used as guides for
C                    the ranking of the rows.
C     NBROW        : The  number  of  row  numbers  used  for row
C                    ranking.
C
C*****************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 SN,FLAG1(6),FLAG2(4),A(128),B(128),NKEY(128),
      1         NRANK(128),SUP1(128),SUP2(128),NCO(128),
      2         NCIN(128),NRO(128),NTR(128),NTIN(128),
      3         NRIN(2048),NOROW(2048)
      CHARACTER*24 COLMNE(128),MNE(128),MRANK(128),
      1             ROWMNE1(2048),ROWMNE2(128)
      REAL*4 RMIN(128),RMAX(128)

      WRITE(LOGN,200) NCOL,NROW,NC,NR,IJ,NT,KMN,NNU,NN,NM,SN,
      1               NBROW
      WRITE(LOGN,100) (FLAG1(I),I=1,6)
      IF (FLAG1(1).EQ.1) THEN
         WRITE(LOGN,300) (COLMNE(I),I=1,NC)
         WRITE(LOGN,100) (NCO(I),I=1,NC)
         WRITE(LOGN,100) (NCIN(I),I=1,NC)
      END IF
      IF (FLAG1(2).EQ.1) THEN
         WRITE(LOGN,400) (RMIN(I),I=1,NT)
         WRITE(LOGN,400) (RMAX(I),I=1,NT)
         WRITE(LOGN,100) (NTR(I),I=1,NT)
         WRITE(LOGN,100) (NTIN(I),I=1,NT)
      END IF
      WRITE(LOGN,100) (FLAG2(I),I=1,4)
      IF (FLAG2(1).EQ.1) THEN
         WRITE(LOGN,300) (ROWMNE1(I),I=1,NR)
         WRITE(LOGN,100) (NRO(I),I=1,NR)
         WRITE(LOGN,100) (NRIN(I),I=1,NR)
      END IF
      IF (FLAG2(2).EQ.1) THEN
         WRITE(LOGN,300) (ROWMNE2(I),I=1,IJ)
         WRITE(LOGN,100) (A(I),I=1,IJ)
         WRITE(LOGN,100) (B(I),I=1,IJ)
      END IF
      IF (FLAG1(3).EQ.1) THEN
         WRITE(LOGN,100) (NKEY(I),I=1,NN)
      END IF
      IF (FLAG1(4).EQ.1) THEN
         WRITE(LOGN,300) (MNE(I),I=1,NM)
      END IF
      IF (FLAG1(5).EQ.1) THEN
         WRITE(LOGN,100) (NRANK(I),I=1,NNU)
      END IF
      IF (FLAG1(6).EQ.1) THEN
         WRITE(LOGN,300) (MRANK(I),I=1,KMN)
      END IF
      IF (FLAG2(3).EQ.1) THEN
         WRITE(LOGN,100) (SUP1(I),I=1,SN)
         WRITE(LOGN,100) (SUP2(I),I=1,SN)
      END IF
      IF (FLAG2(4).EQ.1) THEN
         WRITE(LOGN,100) (NOROW(I),I=1,NBROW)
      END IF
100   FORMAT(I5)
200   FORMAT(12I5)
300   FORMAT(X,A24)
400   FORMAT(F14.6)
      RETURN
      END

C*************************************************************
      SUBROUTINE READER(LOGN,FLAG1,FLAG2,NCOL,NROW,KMN,NNU,NN,
      1                NM,IJ,A,B,COLMNE,NC,NCO,NCIN,RMIN,RMAX,
      2                NT,NTR,NTIN,NKEY,MNE,NRANK,MRANK,
      3                ROWMNE1,NR,NRO,NRIN,ROWMNE2,SUP1,SUP2,
      4                SN,NOROW,NBROW)
C*************************************************************
C
C     This subroutine is used to record the informations
C     regarding the header on a sequential file.
C
C     ARGUMENTS
C       ---------
C
C     LOGN       : The logical unit  for the file on  which the
C                  header,s data will be stored.
C     FLAG1      : Integer  array of six elements used as  flag
C                  to  indicate  the  existence  or not of  the
C                  several informations regarding the columns.
C     FLAG2      : The corresponding flag for rows.
C     NCOL       : The number of columns.
C     NROW       : The number of rows.
C     KMN        : The number of column names used for ranking.
C     NNU        : The number of columns used for ranking.
C     NN         : The  number of repeated   sortings according
C                  column numbers.
C     NM         : The  number  of repeated sortings  according
C                  column mnemonic names.
C     IJ         : The number of assigned tracing extrema sets.
C     A,B        : Arrays  storing  the  mumber of row on which
C                  starts  a  set  of common name rows  and the
C                  corresponding on which ends the set.
C     COLMNE     : Array storing the column mnemonic names.
C     NC         : The number of assigned column mnemonic names.
```

```
C     NCO       : The numbers of columns for which names  have
C                 been assigned.
C     NCIN      : The invert relative addresses of COLMNE.
C     RMIN,RMAX : Arrays  storing  the  minimum  and   maximum
C                 values  between which tracing of the data is
C                 going to take place.
C     NT        : The  number of  columns for  which   tracing
C                 extrema  have been assigned.
C     NTR       : The column numbers for which tracing extrema
C                 have been assigned.
C     NTIN      : The invert relative addresses of RMIN,RMAX.
C     NKEY      : Array storing  the numbers of columns  which
C                 will be used as sorting guides.
C     MNE       : Array  storing  the  names  of columns which
C                 will be used as sorting guides
C     NRANK     : Array storing  the numbers  of  columns that
C                 assign the ranking of data
C     MRANK     : Array storing  the  names  of  columns  that
C                 assign the ranking of data.
C     ROWMNE1   : Array storing the row mnemonic names.
C     NR        : The number of assigned row mnemonic names.
C     NRO       : The  numbers  of  rows  for which names have
C                 been assigned.
C     NRIN      : The invert relative addresses of ROWMNE.
C     ROWMNE2   : Array storing the mnemonics of sets of rows.
C     SUP1,SUP2 : Arrays storing the  number  of  row on which
C                 starts  suppression  of   rows   and    the
C                 corresponding on which ends the suppression.
C     SN        : The number of assigned suppressions.
C     NOROW     : Array the numbers of rows used as guides for
C                 the ranking of the rows.
C     NBROW     : The  number  of  row  numbers  used  for row
C                 ranking.
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 SN,FLAG1(6),FLAG2(4),A(128),B(128),NKEY(128),
      1         NRANK(128),SUP1(128),SUP2(128),NCO(128),
      2        NRO(128),NTR(128),NCIN(128),NTIN(128),
      3         NRIN(2048),NOROW(2048)
      CHARACTER*24 COLMNE(128),MNE(128),MRANK(128),
      1            ROWMNE1(2048),ROWMNE2(128)
      REAL*4 RMIN(128),RMAX(128)

      READ(LOGN,200) NCOL,NROW,NC,NR,IJ,NT,KMN,NNU,NN,NM,SN,
      1              NBROW
      READ(LOGN,100) (FLAG1(I),I=1,6)
      IF (FLAG1(1).EQ.1) THEN
         READ(LOGN,300) (COLMNE(I),I=1,NC)
         READ(LOGN,100) (NCO(I),I=1,NC)
         READ(LOGN,100) (NCIN(I),I=1,NC)
```

```
        END IF
        IF (FLAG1(2).EQ.1) THEN
           READ(LOGN,400) (RMIN(I),I=1,NT)
           READ(LOGN,400) (RMAX(I),I=1,NT)
           READ(LOGN,100) (NTR(I),I=1,NT)
           READ(LOGN,100) (NTIN(I),I=1,NT)
        END IF
        READ(LOGN,100) (FLAG2(I),I=1,4)
        IF (FLAG2(1).EQ.1) THEN
           READ(LOGN,300) (ROWMNE1(I),I=1,NR)
           READ(LOGN,100) (NRO(I),I=1,NR)
           READ(LOGN,100) (NRIN(I),I=1,NR)
        END IF
        IF (FLAG2(2).EQ.1) THEN
           READ(LOGN,300) (ROWMNE2(I),I=1,IJ)
           READ(LOGN,100) (A(I),I=1,IJ)
           READ(LOGN,100) (B(I),I=1,IJ)
        END IF
        IF (FLAG1(3).EQ.1) THEN
           READ(LOGN,100) (NKEY(I),I=1,NN)
        END IF
        IF (FLAG1(4).EQ.1) THEN
           READ(LOGN,300) (MNE(I),I=1,NM)
        END IF
        IF (FLAG1(5).EQ.1) THEN
           READ(LOGN,100) (NRANK(I),I=1,NNU)
        END IF
        IF (FLAG1(6).EQ.1) THEN
           READ(LOGN,300) (MRANK(I),I=1,KMN)
        END IF
        IF (FLAG2(3).EQ.1) THEN
           READ(LOGN,100) (SUP1(I),I=1,SN)
           READ(LOGN,100) (SUP2(I),I=1,SN)
        END IF
        IF (FLAG2(4).EQ.1) THEN
           READ(LOGN,100) (NOROW(I),I=1,NBROW)
        END IF
100     FORMAT(I5)
200     FORMAT(12I5)
300     FORMAT(X,A24)
400     FORMAT(F14.6)
        RETURN
        END

  C*************************************************************
        SUBROUTINE CDISP (LEC,IMP,FLAG1,NCOL,COLMNE,NC,NCO,NCIN,
        1                 RMIN,RMAX,NT,NTR,NTIN,NKEY,MNE,NN,NM,
        2                 NRANK,MRANK,NNU,KMN)
  C*************************************************************
  C     This  subroutine is used  to  display  on the screen the
  C     column informations of the header.
  C
```

```
C     ARGUMENTS:
C     ----------
C
C     LEC       : The logicalunit number for writing   on  the
C                 terminal.
C     IMP       : The logical unit number for reading from the
C                 terminal.
C     FLAG1     : Integer  array of six elements  used as flag
C                 to  indicate  the  existence or  not  of the
C                 several informations regarding the columns.
C     NCOL      : The number of columns.
C     COLMNE    : Array storing the column mnemonic names.
C     NC        : The number of assigned column mnemonic names.
C     NCO       : The  numbers of columns for which names have
C                 been assigned.
C     NCIN      : The invert relative addresses of COLMNE.
C     RMIN,RMAX : Arrays storing  the  minimum   and   maximum
C                 values between which  tracing of the data is
C                 going to take place.
C     NT        : The  number of columns  for  which   tracing
C                 extrema have been assigned.
C     NTR       : The column numbers for which tracing extrema
C                 have been assigned.
C     NTIN      : The invert relative addresses of RMIN,RMAX.
C     NKEY      : Array storing the  numbers  of columns which
C                 will be used as sorting guides.
C     MNE       : Array  storing  the  names  of columns which
C                 will be used as sorting guides.
C     NN        : The  number  of  repeated sortings according
C                 column numbers.
C     NM        : The  number  of repeated  sortings according
C                 column mnemonic names.
C     NRANK     : Array  storing the  numbers  of columns that
C                 assign the ranking of data
C     MRANK     : Array  storing  the  names  of  columns that
C                 assign the ranking of data.
C     NNU       : The number of columns used for ranking.
C     KMN       : The number of column names used for ranking.
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 LEC,IMP,LOGN,NCOL,NROW,NN,NM,NNU,KMN,IJ,
      1         NTR(128),FLAG2(4),NKEY(128),NRANK(128),A(128),
      2         B(128),NCO(128),NCIN(128),NTIN(128),FLAG1(6)
      REAL*4 RMIN(128),RMAX(128)
      CHARACTER*1 IND
      CHARACTER*24 FNAME,COLMNE(128),MNE(128),MRANK(128)

      WRITE(IMP,100)
      WRITE(IMP,200) NCOL
      IF (FLAG1(1).EQ.1) THEN
         WRITE(IMP,300)
```

```
           DO I=1,NC
              WRITE(IMP,400) NCO(I),COLMNE(NCIN(I))
           END DO
        END IF
        IF (FLAG1(2).EQ.1) THEN
           WRITE(IMP,500)
           DO I=1,NT
              WRITE(IMP,600) NTR(I),RMIN(NTIN(I)),RMAX(NTIN(I))
           END DO
        END IF
        IF (FLAG1(3).EQ.1) THEN
           WRITE(IMP,700)
           DO I=1,NN
              WRITE(IMP,800) NKEY(I)
           END DO
        END IF
        IF (FLAG1(4).EQ.1) THEN
           WRITE(IMP,900)
           DO I=1,NM
              WRITE(IMP,1000) MNE(I)
           END DO
        END IF
        IF (FLAG1(5).EQ.1) THEN
           WRITE(IMP,1100)
           DO I=1,NNU
              WRITE(IMP,1200) NRANK(I)
           END DO
        END IF
        IF (FLAG1(6).EQ.1) THEN
           WRITE(IMP,1300)
           DO I=1,KMN
              WRITE(IMP,1400) MRANK(I)
           END DO
        END IF
        READ(LEC,1500) IND
        IF (IND.EQ.' ') THEN
           WRITE(IMP,1600)
           RETURN
        END IF
100     FORMAT(20(/),18X,'C O L U M N   I N F O R M A T I O N S'
        1         /,18X,'*************************************')
200     FORMAT(//27X,'NUMBER OF COLUMNS'/27X,
        1            '-----------------'//31X,I5)
300     FORMAT(//27X,'COLUMN MNEMONICS'/27X,
        1            '----------------'/23X,
        2         'Column number',3X,'Mnemonic'/23X,
        3         '------------------------')
400     FORMAT(27X,I3,11X,A24)
500     FORMAT(//28X,'TRACING EXTREMA'/28X,
        1            '---------------'/13X,
        2     'Column number',6X,'Minimum',9X,'Maximum',/13X,
        3     '------------------------------------------')
```

```
600     FORMAT(16X,I3,6X,F14.6,3X,F14.6)
700     FORMAT(//18X,'COLUMN NUMBER GUIDES FOR MULTIPLE SORT'/
        2        18X,'--------------------------------------')
800     FORMAT(33X,I3)
900     FORMAT(//17X,'COLUMN NAME GUIDES FOR MULTIPLE SORT'/17X,
        1           '------------------------------------')
1000    FORMAT(41X,A24)
1100    FORMAT(/22X,'COLUMN NUMBERS FOR RANKING'/22X,
        1          '--------------------------')
1200    FORMAT(33X,I3)
1300    FORMAT(/23X,'COLUMN NAMES FOR RANKING'/23X,
        1          '------------------------')
1400    FORMAT(34X,A24)
1500    FORMAT(A)
1600    FORMAT(20(/))
        END

  C***************************************************************
        SUBROUTINE RDISP(LEC,IMP,FLAG2,NROW,ROWMNE1,NR,NRO,NRIN,
        1                ROWMNE2,IJ,A,B,SUP1,SUP2,SN,NOROW,NBROW)
  C***************************************************************
  C
  C     ARGUMENTS
  C     ---------
  C
  C     LEC       : The logical unit number for writing  on  the
  C                 terminal.
  C     IMP       : The logical unit number for reading from the
  C                 terminal.
  C     LOGN      : The logical unit for the file on  which  the
  C                 header,s data will be stored.
  C     FLAG1     : Integer array of six  elements  used as flag
  C                 to  indicate  the  existence or  not  of the
  C                 several informations regarding the columns.
  C     FLAG2     : The corresponding flag for rows.
  C     NCOL      : The number of columns.
  C     NROW      : The number of rows.
  C     COLMNE    : Array storing the column mnemonic names.
  C     NC        : The number of assigned column mnemonic names.
  C     NCO       : The numbers of  columns for which names have
  C                 been assigned.
  C     NCIN      : The invert relative addresses of COLMNE.
  C     ROWMNE1   : Array storing the row mnemonic names.
  C     NR        : The number of assigned row mnemonic names.
  C     NRO       : The numbers  of  rows  for  which names have
  C                 been assigned.
  C     NRIN      : The invert relative addresses of ROWMNE.
  C     ROWMNE2   : Array storing the mnemonics of sets of rows.
  C     RMIN,RMAX : Arrays  storing  the  minimum  and  maximum
  C                 values between which  tracing of the data is
  C                 going to take place.
  C     NT        : The  number of columns  for  which  tracing
```

```
C                    extrema have been assigned.
C      NTR         : The column numbers for which tracing extrema
C                    have been assigned.
C      NTIN        : The invert relative addresses of RMIN,RMAX.
C      NKEY        : Array storing the  numbers  of columns which
C                    will be used as sorting guides.
C      MNE         : Array storing the  names  of  columns  which
C                    will be used as sorting guides.
C      NN          : The number of  repeated  sortings  according
C                    column  numbers.
C      NM          : The number of  repeated  sortings  according
C                    column mnemonic names.
C      IJ          : The number of assigned tracing extrema sets.
C      A,B         : Arrays storing the  mumber  of  row on which
C                    starts  a  set of common name rows  and the
C                    corresponding on which ends the set.
C      NRANK       : Array  storing the numbers  of  columns that
C                    assign the ranking of data
C      MRANK       : Array storing  the  names  of  columns  that
C                    assign the ranking of data
C      NNU         : The number of columns used for ranking.
C      KMN         : The number of column names used for ranking.
C      SUP1,SUP2   : Arrays storing the  number of row  on  which
C                    starts  suppression  of  rows  and  the
C                    corresponding  on which ends the suppression.
C      SN          : The number of assigned suppressions.
C      NOROW       : Array the numbers of rows used as guides for
C                    the ranking of the rows.
C      NBROW       : The number of  row  numbers  used  for  row
C                    ranking
C
C*************************************************************

       IMPLICIT INTEGER*2 (I-N)
       INTEGER*2 LEC,IMP,LOGN,NCOL,NROW,NN,NM,NNU,KMN,IJ,SN,
       1         FLAG1(6),FLAG2(4),NKEY(128),NRANK(128),A(128),
       2         B(128),NOROW(2048),SUP1(128),SUP2(128),
       3         NRO(128),NRIN(2048)
       REAL*4 RMIN(128),RMAX(128)
       CHARACTER*1 IND
       CHARACTER*24 FNAME,COLMNE(128),MNE(128),MRANK(128),
       1            ROWMNE1(2048),ROWMNE2(128),NAME

       WRITE(IMP,100)
       WRITE(IMP,200) NROW
       IF (FLAG2(1).EQ.1) THEN
          WRITE(IMP,300)
          DO I=1,NR
             WRITE(IMP,400) NRO(I),ROWMNE1(NRIN(I))
          END DO
       END IF
       IF (FLAG2(2).EQ.1) THEN
```

```
           WRITE(IMP,500)
           DO I=1,IJ
              WRITE(IMP,600) I,ROWMNE2(I)
              DO J=A(I),B(I)
                 NAME=' '
                 K=1
                 DO WHILE(K.LE.NR.AND.NAME.EQ.' ')
                    IF (J.EQ.NRO(K)) THEN
                       NAME=ROWMNE1(K)
                    END IF
                    K=K+1
                 END DO
                 WRITE(IMP,610) J,NAME
              END DO
              IF (I.LT.IJ) THEN
                 WRITE(IMP,620)
              END IF
           END DO
        END IF
        IF (FLAG2(3).EQ.1) THEN
           WRITE(IMP,700)
           DO I=1,SN
              IF (SUP1(I).NE.0.AND.SUP2(I).NE.0) THEN
                 WRITE(IMP,800) I,SUP1(I),SUP2(I)
              END IF
           END DO
        END IF
        IF (FLAG2(4).EQ.1) THEN
           WRITE(IMP,1100)
           DO I=1,NBROW
              WRITE(IMP,1200) NOROW(I)
           END DO
        END IF
        READ(LEC,900) IND
        IF (IND.EQ.' ') THEN
           WRITE(IMP,1000)
           RETURN
        END IF
100     FORMAT(20(/),20X,'R O W   I N F O R M A T I O N S'
        1           /20X,'*******************************')
200     FORMAT(//29X,'NUMBER OF ROWS'/29X,
        1            '---------------'//32X,I5)
300     FORMAT(//30X,'ROW MNEMONICS'/30X,
        1            '--------------'//23X,
        2      'Row number',6X,'Mnemonics'/23X,
        3      '--------------------------')
400     FORMAT(25X,I5,11X,A24)
500     FORMAT(//20X,'SETS OF ROWS WITH COMMON MNEMONICS'/20X,
        1            '----------------------------------'//20X,
        2      'Set',7X,'Set',7X,'Row',7X,'Row',7X/19X,'Number',
        3     3X,'mnemonic'3X,'number',3X,'mnemonic'/19X,
        4      '---------------------------------------')
```

```
600     FORMAT(17X,I5,7X,A24)
610     FORMAT(36X,I5,7X,A24)
620     FORMAT(19X,'----------------------------------------')
700     FORMAT(//30X,'SUPPRESSIONS'/30X,'------------'//20X,
        1      'Suppression',
        2       3X,'From row',3X,'To row'/20X,
        3       '--------------------------------')
800     FORMAT(19X,I5,8X,I5,5X,I5)
900     FORMAT(A)
1000    FORMAT(20(/))
1100    FORMAT(//30X,'ROW RANKING'/30X,'-----------')
1200    FORMAT(32X,I5)
        END

  C************************************************************
        SUBROUTINE COLMOD(LEC,IMP,FLAG1,NCOL,NROW,COLMNE,NC,NCO,
        1                NCIN,RMIN,RMAX,NT,NTR,NTIN,NKEY,MNE,NN,
        2                NM,NRANK,MRANK,NNU,KMN)
  C************************************************************
  C     ARGUMENTS
  C     ---------
  C
  C     LEC       : The logical unit number for writing  on  the
  C                 terminal.
  C     IMP       : The logical unit number for reading from the
  C                 terminal.
  C     LOGN      : The logical unit for the  file  on which the
  C                 header,s data will be stored.
  C     FLAG1     : Integer  array of six  elements used as flag
  C                 to  indicate  the  existence or  not  of the
  C                 several informations regarding the columns.
  C     NCOL      : The number of columns.
  C     NROW      : The number of rows.
  C     COLMNE    : Array storing the column mnemonic names.
  C     NC        : The number of assigned column mnemonic names.
  C     NCO       : The numbers of  columns for which names have
  C                 been assigned.
  C     NCIN      : The invert relative addresses of COLMNE.
  C     ROWMNE1   : Array storing the row mnemonic names.
  C     NR        : The number of assigned row mnemonic names.
  C     NRO       : The  numbers  of  rows  for which names have
  C                 been assigned.
  C     NRIN      : The invert relative addresses of ROWMNE.
  C     ROWMNE2   : Array storing the mnemonics of sets of rows.
  C     RMIN,RMAX : Arrays  storing  the  minimum  and   maximum
  C                 valueS between which tracing of  the data is
  C                 going to take place.
  C     NT        : The  number of columns  for  which   tracing
  C                 extrema have been assigned.
  C    ·NTR       : The column numbers for which tracing extrema
  C                 have been assigned.
  C     NTIN      : The invert relative addresses of RMIN,RMAX.
```

```
C     NKEY      : Array storing the  numbers  of columns which
C                 will be used as sorting guides.
C     MNE       : Array storing  the  names  of  columns which
C                 will be used as sorting guides.
C     NN        : The number of  repeated  sortings  according
C                 column  numbers.
C     NM        : The  number of repeated  sortings  according
C                 column mnemonic names.
C     IJ        : The number of assigned tracing extrema sets.
C     A,B       : Arrays  storing the mumber  of  row on which
C                 starts a  set  of  common name rows  and the
C                 corresponding on which ends the set.
C     NRANK     : Array  storing  the  numbers of columns that
C                 assign the ranking of data
C     MRANK     : Array  storing  the  names  of  columns that
C                 assign the ranking of data
C     NNU       : The number of columns used for ranking.
C     KMN       : The number of column names used for ranking.
C     SUP1,SUP2 : Arrays storing the  number  of  row on which
C                 starts  suppression  of  rows  and  the
C                 corresponding  on which ends the suppression.
C     SN        : The number of assigned suppressions.
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 LEC,IMP,LOGN,NCOL,NROW,NN,NM,NNU,KMN,IJ,
      1         FLAG1(6),NTR(128),FLAG2(4),NKEY(128),
      2         NRANK(128),A(128),B(128),NCO(128),NCIN(128),
      3         NTIN(128)
      REAL*4 RMIN(128),RMAX(128),RMIND(128),RMAXD(128)
      CHARACTER*1 IND
      CHARACTER*7 MODCOL
      CHARACTER*24 FNAME,COLMNE(128),MNE(128),MRANK(128),NAME,
      1            ROWMNE1(2048),ROWMNE2(128),COLMNED(128),
      DATA MODCOL/'columns'/

      IDUM=1
      DO WHILE (IDUM.EQ.1)
         IDUM=0
         WRITE(IMP,100)
         READ(LEC,200) IND
         IF (IND.EQ.'1') THEN

C     Column number modification

            WRITE(IMP,300) NCOL
            READ(LEC,310) NCOL
            IDUM=1

         ELSE IF (IND.EQ.'2') THEN
```

```
C     Column mnemonic names modification
            IF (FLAG1(1).EQ.1) THEN
               I=1
               DO WHILE(I.NE.0)
                  WRITE(IMP,500)
                  READ(LEC,400) I
                  IF (I.NE.0)THEN
                     J=1
                     DO WHILE(NCO(J).NE.I.AND.J.LE.NC)
                        J=J+1
                     END DO
                     IF (J-1.EQ.NC.AND.NCO(J).NE.I) THEN
                        DO K=1,NC
                           COLMNED(K)=COLMNE(NCIN(K))
                        END DO
                        DO K=1,NC
                           COLMNE(K)=COLMNED(K)
                        END DO
                        NC=NC+1
                        NCO(NC)=I
                        WRITE(IMP,600) COLMNE(NC)
                        READ(LEC,700) COLMNE(NC)
                        CALL EXSH1(NCO,NCIN,NC,1)
                     ELSE
                        WRITE(IMP,600) COLMNE(NCIN(J))
                        READ(LEC,700) COLMNE(NCIN(J))
                     END IF
                  END IF
               END DO
               IDUM=1
            ELSE
               WRITE(IMP,800)
               CALL COLNAM(LEC,IMP,FLAG1,NCOL,COLMNE,NC,NCO,
     1                     NCIN)
            END IF
        ELSE IF (IND.EQ.'3') THEN

C     Tracing extrema modification

           IF (FLAG1(2).EQ.1) THEN
              I=1
              DO WHILE (I.NE.0)
                 WRITE(IMP,500)
                 READ(LEC,400) I
                 IF (I.NE.0) THEN
                    J=1
                    DO WHILE(NTR(J).NE.I.AND.J.LE.NT)
                       J=J+1
                    END DO
                    IF (J-1.EQ.NT.AND.NTR(J).NE.I) THEN
                       DO K=1,NT
```

```
                          RMIND(K)=RMIN(NTIN(K))
                          RMAXD(K)=RMAX(NTIN(K))
                       END DO
                       DO K=1,NT
                          RMIN(K)=RMIND(K)
                          RMAX(K)=RMAXD(K)
                       END DO
                       NT=NT+1
                       NTR(NT)=I
                       WRITE(IMP,900) RMIN(NT)
                       READ(LEC,1700) RMIN(NT)
                       WRITE(IMP,1800) RMAX(NT)
                       READ(LEC,1700) RMAX(NT)
                       CALL EXSH1(NTR,NTIN,NT,1)
                    ELSE
                       WRITE(IMP,900) RMIN(NTIN(J))
                       READ(LEC,1700) RMIN(NTIN(J))
                       WRITE(IMP,1800) RMAX(NTIN(J))
                       READ(LEC,1700) RMAX(NTIN(J))
                    END IF
                 END IF
              END DO
              IDUM=1
           ELSE
              WRITE(IMP,1000)
              CALL EXTREM(LEC,IMP,FLAG1,NCOL,NROW,RMIN,RMAX,
      1                   NT,NTR)
           END IF
        ELSE IF (IND.EQ.'4') THEN

C     Sorting guides modification

           IF (FLAG1(3).EQ.1) THEN

C     Sorting guides by column number modification

              WRITE(IMP,1100)
              WRITE(LEC,400) (NKEY(I),I=1,NN)
              CALL ANADIS(LEC,IMP,NCOL,NCOL,NKEY,NN,MODCOL)
              CALL REAR1(LEC,IMP,NCOL,NROW,NKEY,NN)
           ELSE IF (FLAG1(4).EQ.1) THEN
              WRITE(IMP,1100)
              DO I=1,NM
                 WRITE(LEC,1300) MNE(I)
              END DO
              WRITE(IMP,1400)
              CALL ASBYMN(LEC,IMP,NCOL,COLMNE,NM,MNE)
              CALL REAR2(LEC,IMP,NCOL,NROW,COLMNE,MNE,NM)
           ELSE
              WRITE(IMP,1500)
              CALL MULSOR(LEC,IMP,NCOL,FLAG1,NRANK,MRANK,NNU,
              1          KMN,COLMNE)
```

```
             END IF
             IDUM=1
          ELSE IF (IND.EQ.'5') THEN

  C     Ranking modification

             IF (FLAG1(5).EQ.1) THEN
                WRITE(IMP,1900)
                WRITE(LEC,400) (NRANK(I),I=1,NNU)
                CALL ANADIS(LEC,IMP,NCOL,NCOL,NRANK,NNU,MODCOL)
                CALL RANK1(LEC,IMP,NCOL,NROW,NRANK,NNU)
             ELSE IF (FLAG1(6).EQ.1) THEN
                WRITE(IMP,1900)
                DO I=1,KMN
                   WRITE(IMP,1300) MRANK(I)
                END DO
                WRITE(IMP,2000)
                CALL ASBYMN(LEC,IMP,NCOL,COLMNE,KMN,MRANK)
                CALL RANK2(LEC,IMP,NCOL,NROW,MRANK,KMN,COLMNE)
             ELSE
                WRITE(IMP,2200)
                CALL CRANK(LEC,IMP,NCOL,NROW,FLAG1,NRANK,MRANK,
                1         NNU,KMN,COLMNE)
             END IF
          ELSE IF (IND.EQ.' ') THEN
             RETURN
          END IF
        END DO
        RETURN
100     FORMAT(///8X,'MODIFICATION SELECTION TABLE'/8X,
        1            '----------------------------'/9X,
        2              'Number of columns       1'/9X,
        3              'Mnemonic names          2'/9X,
        4              'Tracing extrema         3'/9X,
        5              'Sorting guides          4'/9X,
        6              'Rank of columns         5'//'$',3X,
        7              'Answer :')
200     FORMAT(A)
300     FORMAT(///,4X,'Old number of columns :',I3,/'$',3X,
        1             'Assign new number      :')
310     FORMAT(I3)
400     FORMAT(/4X,100(20(I3,',')/))
500     FORMAT(/,'$',3X,'Assign column number :')
600     FORMAT(///4X,'Old name         :',A24/'$',3X,
        1       'Assign new name :')
700     FORMAT(A24)
800     FORMAT(///4X,'MNEMONIC NAMES HAVE NOT BEEN ASSIGNED!!')
900     FORMAT(///4X,'Old MIN          :',F14.6/'$',3X,
        1       'Assign new MIN :')
1000    FORMAT(///4X,'TRACING EXTREMA HAVE NOT BEEN ASSIGNED!!')
1100    FORMAT(///4X,'Old sorting guides     :')
1200    FORMAT(///4X,'Assign column numbers :')
```

```
1300    FORMAT(/2X,A24)
1400    FORMAT(///4X,'Assign mnemonic names :')
1500    FORMAT(///4X,'SORTING GUIDES DO NOT EXIST!!')
1600    FORMAT(///4X,'THE ASSIGNED MNEMONICHAS NOT BEEN'/6X,
        1      ' ASSIGNED AS COLUMN MNEMONIC!!'//4X,
        2       'Assign next mnemonic')
1700    FORMAT(F14.6)
1800    FORMAT(///4X,'Old MAX         :',F14.6/'$',3X,
        1       'Assign new MAX :')
1900    FORMAT(///4X,'Old ranking :')
2000    FORMAT(///4X,'Assign new ranking guides :')
2200    FORMAT(///4X,'RANKING HAS NOT BEEN ASSIGNED!!')
        END

     C**************************************************************
            SUBROUTINE ROWMOD(LEC,IMP,FLAG2,NCOL,NROW,ROWMNE1,NR,
            1                 NRO,NRIN,ROWMNE2,IJ,A,B,SUP1,SUP2,SN,
            2                 NOROW,NBROW)
     C**************************************************************
     C
     C      ARGUMENTS
     C      ---------
     C
     C      LEC       : The logical unit number for writing  on  the
     C                  terminal.
     C      IMP       : The logical unit number for reading from the
     C                  terminal.
     C      FLAG2     : The corresponding flag for rows.
     C      NCOL      : The number of columns.
     C      NROW      : The number of rows.
     C      ROWMNE1   : Array storing the row mnemonic names.
     C      NR        : The number of assigned row mnemonic names.
     C      NRO       : The numbers of rows for  which   names  have
     C                  been assigned.
     C      NRIN      : The invert relative addresses of ROWMNE.
     C      ROWMNE2   : Array storing the mnemonics of sets of rows.
     C      IJ        : The number of assigned tracing extrema sets.
     C      A,B       : Arrays storing  the  mumber of  row on which
     C                  starts  a  set  of common name  rows and the
     C                  corresponding on which ends the set.
     C      SUP1,SUP2 : Arrays  storing  the number of  row on which
     C                  starts   suppression  of   rows    and   the
     C                  corresponding on which ends the suppression.
     C      SN        : The number of assigned suppressions.
     C      NOROW     : Array the numbers of rows used as guides for
     C                  the ranking of the rows.
     C      NBROW     : The  number  of  row  numbers  used  for row
     C                  ranking
     C
     C**************************************************************

            IMPLICIT INTEGER*2 (I-N)
```

```
INTEGER*2 FLAG2(4),A(128),B(128),SUP1(128),SUP2(128),
1         NRO(128),NRIN(2048),SN,NOROW(2048)
CHARACTER*1 IND
CHARACTER*7 MODROW
CHARACTER*24 ROWMNE1(2048),ROWMNE2(128),ROWMNE1D(128)
DATA MODROW/'rows'/

IDUM=1
DO WHILE (IDUM.EQ.1)
   IDUM=0
   WRITE(IMP,100)
   READ(LEC,200) IND
   IF (IND.EQ.'1') THEN
      WRITE(IMP,300) NROW
      READ(LEC,400) NROW
      IDUM=1
   ELSE IF (IND.EQ.'2') THEN
      IF (FLAG2(1).EQ.1) THEN
         I=1
         DO WHILE (I.NE.0)
            WRITE(IMP,500)
            READ(LEC,400) I
            IF (I.NE.0) THEN
               J=1
               DO WHILE(NRO(J).NE.I.AND.J.LE.NR)
                  J=J+1
               END DO
               IF (J-1.EQ.NR.AND.NRO(J).NE.I) THEN
                  DO K=1,NR
                     ROWMNE1D(K)=ROWMNE1(NRIN(K))
                  END DO
                  DO K=1,NR
                     ROWMNE1(K)=ROWMNE1D(K)
                  END DO
                  NR=NR+1
                  NRO(NR)=I
                  WRITE(IMP,600) ROWMNE1(NR)
                  READ(LEC,700) ROWMNE1(NR)
                  CALL EXSH1(NRO,NRIN,NR,1)
               ELSE
                  WRITE(IMP,600) ROWMNE1(NRIN(J))
                  READ(LEC,700) ROWMNE1(NRIN(J))
               END IF
            END IF
         END DO
         IDUM=1
      ELSE
         WRITE(IMP,800)
         CALL ROWNAM(LEC,IMP,FLAG2,NROW,ROWMNE1,NR,NRO,
         1          NRIN,ROWMNE2,IJ,A,B)
      END IF
   ELSE IF (IND.EQ.'3') THEN
   IF (FLAG2(2).EQ.1) THEN
      WRITE(IMP,900)
      READ(LEC,200) IND
      IF (IND.EQ.'1') THEN
         I=1
         DO WHILE(I.NE.0)
            WRITE(IMP,1000)
            READ(LEC,400) I
            IF (I.NE.0) THEN
               WRITE(IMP,1100) I,A(I),B(I)
               READ(LEC,400) A(I)
               WRITE(IMP,1200)
               READ(LEC,400) B(I)
            END IF
         END DO
      ELSE IF (IND.EQ.'2') THEN
         I=1
         DO WHILE(I.NE.0)
            WRITE(IMP,1100)
            READ(LEC,400) I
            IF (I.NE.0) THEN
               WRITE(IMP,1400) ROWMNE2(I)
               READ(LEC,700) ROWMNE2(I)
            END IF
         END DO
         IDUM=1
      ELSE
         WRITE(IMP,1500)
         IDUM=1
      END IF
   ELSE
      WRITE(IMP,800)
      CALL ROWNAM(LEC,IMP,FLAG2,NROW,ROWMNE1,ROWMNE2,
      1          IJ,A,B)
   END IF

ELSE IF (IND.EQ.'4') THEN
   IF (FLAG2(3).EQ.1) THEN
      I=1
      DO WHILE (I.NE.0)
         WRITE(IMP,1600)
         READ(LEC,400) I
         IF (I.NE.0) THEN
            IF (SUP1(I).EQ.0.AND.SUP2(I).EQ.0) THEN
               SN=SN+1
               WRITE(IMP,1700) SUP1(I),SUP2(I)
               READ(LEC,400) SUP1(I)
               WRITE(IMP,1800)
               READ(LEC,400) SUP2(I)
            ELSE IF (SUP1(I).NE.0.AND.SUP2(I).NE.0)
            1     THEN
               WRITE(IMP,1700) I,SUP1(I),SUP2(I),I
```

```
                          READ(LEC,400) SUP1(I)
                          WRITE(IMP,1800)
                          READ(LEC,400) SUP2(I)
                          IF (SUP1(I).EQ.0.AND.SUP2(I).EQ.0)
        1                       THEN
                             NROW=NROW-1
                          END IF
                       END IF
                    END IF
                 END DO
                 CALL SUPSET(LEC,IMP,NCOL,NROW,SUP1,SUP2,SN)
                 IDUM=1
              ELSE
                 WRITE(IMP,1900)
                 CALL ROWSUP(LEC,IMP,FLAG2,NROW,SUP1,SUP2,SN)
              END IF
           ELSE IF (IND.EQ.'5') THEN
              IF (FLAG2(4).EQ.1) THEN
                 WRITE(IMP,2000)
                 WRITE(LEC,2100) (NOROW(I),I=1,NBROW)
                 CALL ANADIS(LEC,IMP,NROW,NROW,NOROW,NBROW,
        1                    MODROW)
                 CALL REORD(LEC,IMP,NCOL,NROW,FLAG2,NOROW,
        1                   NBROW)
                 IDUM=1
              ELSE
                 WRITE(IMP,2200)
                 CALL ANADIS(LEC,IMP,NROW,NROW,NOROW,NBROW,
        1                    MODROW)
                 CALL REORD(LEC,IMP,NCOL,NROW,FLAG2,NOROW,
        1                   NBROW)
              END IF
           ELSE IF (IND.EQ.' ') THEN
              RETURN
           ELSE
              WRITE(IMP,1500)
              IDUM=1
           END IF
        END DO
100     FORMAT(10(/),4X,'MODIFICATION SELECTION TABLE'/4X,
        1               '----------------------------'/4X,
        2              'Number of rows               :1'/4X,
        3              'Mnemonics by row             :2'/4X,
        4              'Mnemonics by series of rows :3'/4X,
        5              'Suppression                  :4'/4X,
        6              'Rank of rows                 :5'
        7      /'$',3X,'Answer                       :')
200     FORMAT(A)
300     FORMAT(///4X,'Old number of rows :',I5/'$',3X,
        1           'Assign new number  :')
400     FORMAT(I5)
500     FORMAT(///'$',3X,'Assign column number :')
```

```
600     FORMAT(///4X,'Old name        :',A24 '$',3X,
        1       'Assign new name :')
700     FORMAT(A24)
800     FORMAT(///3X,'MNEMONIC NAMES DO NOT EXIST!!')
900     FORMAT(///4X,'Serie modification  :1'/4X,
        1            'Name modification   :2'/'$',3X,
        2            'Answer              :')
1000    FORMAT(///'$',3X,'Assign serie number :')
1100    FORMAT(///4X, 'Old serie            :',I5/4X,
        1             'From row             :',I5/4X,
        2             'To row               :',I5//4X,
        3             'Assign new serie     :'/'$',3X,
        4             'From row             :')
1200    FORMAT('$',3X,'To row               :')
1400    FORMAT(///4X, 'Old mnemonic         :',X,A24/'$',3X,
        1             'Assign new mnemonic :')
1500    FORMAT(///5X,'INVALID CHARACTER!!')
1600    FORMAT(///'$',3X,'Assign suppression number :')
1700    FORMAT(///4X,    'Old suppression           :',I5,/4X,
        1                'From row                  :',I5,/4X,
        2                'To row                    :',I5//4X,
        3                'Assign new suppression    :',I5,/'$',
        4             3X,'From row                  :')
1800    FORMAT('$',3X,   'To row                    :')
1900    FORMAT(///4X,'NO SUPPRESSION HAS BEEN ASSINGED!!')
2000    FORMAT(///4X,'Old ranking :')
2100    FORMAT(/4X,100(20(I3,',')/))
2200    FORMAT(///4X,'RANKING HAS NOT BEEN ASSIGNED!!')
        RETURN
        END

  C*************************************************************
        SUBROUTINE TESTER (NCOL,NAME,COLMNE,TEST)
  C*************************************************************
  C
  C      This subroutine is used to  test if a mnemonic name has
  C      been assigned as column mnemonic name
  C
  C     ARGUMENTS
  C     ---------
  C
  C     NCOL   : The number of columns
  C     NAME   : The mnemonic to be tested.
  C     COLMNE : Array storing the column mnemonic names
  C              data will be stored.
  C     TEST   : Flag  indicating  that  the  tested name is  an
  C              assigned mnemonic or not.
  C
  C*************************************************************

        IMPLICIT INTEGER*2 (I-N)
```

```
      INTEGER*2 NCOL
      CHARACTER*24 COLMNE(128),NAME
      LOGICAL*1 TEST
      I=1
      TEST=.FALSE.
      DO WHILE(TEST.EQ..FALSE..AND.I.LE.128)
         IF (COLMNE(I).EQ.NAME) THEN
            TEST=.TRUE.
         END IF
         I=I+1
      END DO
      RETURN
      END

C*************************************************************
      SUBROUTINE REORD(LEC,IMP,NCOL,NROW,FLAG2,NOROW,NBROW)
C*************************************************************
C
C     ARGUMENTS
C     ---------
C
C     LEC   : The logical unit number for writing on the
C             terminal.
C     IMP   : The logical unit number for reading from the
C             terminal.
C     NCOL  : The number of columns
C     NROW  : The number of rows
C     NOROW : Array the numbers of rows used as guides for the
C             ranking of the rows.
C     NBROW : The number of row numbers used for row ranking.
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 AR(128),NO(128),NOR(128),DUM(128),FLAG2(4),
      1         NOROW(2048)
      CHARACTER*45 DEVDIR,NAME*10
      REAL*4 X(128),Y(128)
      DIMENSION NOCOL(256)

      DATA DEVDIR,NAME/'DUA0 :',' '/
      DATA LOGOLD,LOGNEW/2,2/

      FLAG2(4)=1
      NBYTES=NCOL*4
      CALL FICH   ('069',LOGOLD,1,DEVDIR,NAME,NROW,NBYTES,1,
      1             'DIRECT', LEC,IMP)
      IF (NBROW.NE.NROW) THEN
         NBYTES=NCOL*4
         LOGNEW=3
         CALL FICH('069',LOGNEW,1,DEVDIR,NAME,NROW,NBYTES,0,
         1         'DIRECT'LEC,IMP)
         DO I=1,NBROW
            READ (LOGOLD'NOROW(I)) (Y(J),J=1,NCOL)
            WRITE(LOGNEW'I       ) (Y(J),J=1,NCOL)
         END DO
         NROW=NBROW
      ELSE
         DO I=1,NBROW
            READ (LOGOLD'I       ) (X(J),J=1,NCOL)
            READ (LOGOLD'NOROW(I)) (Y(J),J=1,NCOL)
            WRITE(LOGOLD'I       ) (Y(J),J=1,NCOL)
            WRITE(LOGOLD'NOROW(I)) (X(J),J=1,NCOL)
            J=I
            DO WHILE (NOROW(J).NE.I)
               J=J+1
            END DO
            NOROW(J)=NOROW(I)
         END DO
      END IF
      CLOSE(LOGOLD)
      CLOSE(LOGNEW)
      RETURN
      END


C************************************************************
      SUBROUTINE RANK1 (LEC,IMP,NCOL,NROW,NRANK,NNU)
C************************************************************
C
C     Used to rearrange the columns of the data matrix
C     according the column numbers assigned by the user.
C
C     ARGUMENTS:
C
C     LEC   : The logical unit number for writing on the
C             terminal.
C     IMP   : The logical unit number for reading from the
C             terminal.
C     NCOL  : The number of columns.
C     NROW  : The number of rows.
C     NRANK : Array storing the column numbers that determine
C             the reordering of the columns.
C     NNU   : The number of columns that will be ranked.
C
C************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 NRANK(128)
      CHARACTER*24 A(128),B(128)
      CHARACTER*45 DEVDIR,NAME*10
      REAL*4 X(128),Y(128)
      DATA DEVDIR,NAME/'DUA0 :',' '/
      DATA LOGOLD,LOGNEW/2,2/
```

```
C     For the 'old' file.

      WRITE(IMP,100)
      NBYTES=NCOL*4
      CALL FICH   ('069',LOGOLD,1,DEVDIR,NAME,NROW,NBYTES,1,
      1                'DIRECT',LEC,IMP)

C     For the eventual 'new' file.

      IF (NNU.NE.NCOL) THEN
         LOGNEW=3
         WRITE(IMP,200)
         NBYTES=NNU*4
         CALL FICH('069',LOGNEW,1,DEVDIR,NAME,NROW,NBYTES,0,
      1               'DIRECT',LEC,IMP)
      END IF

      DO I=1,NROW
         READ (LOGOLD'I) (X(J),J=1,NCOL)
         WRITE(LOGNEW'I) (X(NRANK(J)),J=1,NNU)
      END DO
      NCOL=NNU
      CLOSE(LOGOLD)
      CLOSE(LOGNEW)
      RETURN

100   FORMAT(/'$',3X,'Assign the ''OLD'' direct access file :
      1        '/4X,37('-'))
200   FORMAT(/'$',3X,'Assign the ''NEW'' direct access file :
      1        '/4X,37('-'))

      END


C*****************************************************************
      SUBROUTINE RANK2 (LEC,IMP,NCOL,NROW,MRANK,KMN,COLMNE)
C*****************************************************************
C
C     Used to rearrange the columns of the data matrix
C     according the column mnemonic names. It calls TRANS to
C     transform the column name to column number.
C
C     ARGUMENTS :
C
C     LEC    : The logical unit number for writing on the
C              terminal.
C     IMP    : The logical unit number for reading from the
C              terminal.
C     NCOL   : The number of columns.
C     NROW   : The number of rows.
C     MRANK  : Array storing the column mnemonic names that
```

```
C               determine the reordering of the columns.
C      KMN    : The number of columns that will be ranked.
C      COLMNE: Array storing the column mnemonic names.
C
C*****************************************************************

       IMPLICIT INTEGER*2 (I-N)
       INTEGER*2 N(128)
       REAL*4 X(128),Y(128)
       CHARACTER*45 DEVDIR,NAME*10
       CHARACTER*24 COLMNE(128),MRANK(128)

       DATA DEVDIR,NAME/'DUA0 :',' '/
       DATA LOGOLD/2/

       NBYTES=NCOL*4
       WRITE(IMP,100)
       CALL FICH('069',LOGOLD,1,DEVDIR,NAME,NROW,NBYTES,1,
       1          'DIRECT',LEC,IMP)
       CALL TRANS(NCOL,COLMNE,KMN,MRANK,N)
       DO I=1,NROW
          READ(LOGOLD'I) (X(J),J=1,NCOL)
          WRITE(LOGOLD'I) (X(N(J)),J=1,KMN)
       END DO
       CLOSE(LOGOLD)
       RETURN
100    FORMAT(/'$',3X,'Assign the ''OLD'' direct access file :
       1       '/4X,37('-'))
       END

C*****************************************************************
       SUBROUTINE REAR1 (LEC,IMP,NCOL,NROW,NKEY,NN)
C*****************************************************************
C
C      Used to  continue the process of multi-sorting according
C      column numbers by calling the subroutine REAR .
C
C      ARGUMENTS :
C
C      LEC    : The logical  unit  number  for  writing  on  the
C               terminal.
C      IMP    : The  logical  unit  number  for reading from the
C               terminal.
C      NCOL   : The number of columns.
C      NROW   : The number of rows.
C      NKEY   : Array storing the column numbers used as sorting
C               column guides.
C      NN     : The  number  of  repeating  sortings   according
C               column number.
C
C*****************************************************************
```

```
      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 NKEY(128)
      DATA LOGOLD/2/
      CALL REAR(LEC,IMP,NCOL,NROW,NKEY,NN)
      RETURN
      END

C**************************************************************
      SUBROUTINE REAR2(LEC,IMP,NCOL,NROW,COLMNE,MNE,NM)
C**************************************************************
C
C     It functions like  REAR1 for column assignment according
C     column names. It calls TRANS to transform  column  names
C     to colun numbers.
C
C     ARGUMENTS :
C
C     LEC   : The logical  unit  number  for  writing  on  the
C             terminal.
C     IMP   : The  logical  unit  number  for reading from the
C             terminal.
C     NCOL  : The number of columns.
C     NROW  : The number of rows.
C     COLMNE: Array storing the column mnemonic names.
C     MNE   : Array storing the names of columns which will be
C             used as sorting column guides.
C     NM    : The  number  of   repeating  sortings  according
C             column mnemonic names.
C
C**************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 N(128)
      CHARACTER*24 COLMNE(128),MNE(128)
      DATA LOGNEW/2/
      CALL TRANS(NCOL,COLMNE,NM,MNE,N)
      CALL REAR(LEC,IMP,NCOL,NROW,N,NM)
      RETURN
      END


C**************************************************************
      SUBROUTINE REAR(LEC,IMP,NCOL,NROW,AR,NSORT)
C**************************************************************
C
C     Used for the multisorting of the data matrix.  It  opens
C     the direct-access file of  data,  and  consequenty calls
C     the subroutine EXTREE for the sorting. It rearranges the
C     data according  the  filnal  inverted relative addresses
C     without  using  extra  file for temporary storage of the
C     data.
C
```

```
C
C     ARGUMENTS :
C
C     LEC   : The logical unit number for writing on the
C             terminal.
C     IMP   : The logical unit number for reading from the
C             terminal.
C     NCOL  : The number of columns.
C     NROW  : The number of rows.
C     AR    : Array storing the numbers of columns used as
C             guides for the multisorting.
C     NSORT : The number of guides.
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 AR(128),NO(128),NOR(128),DUM(128)
      CHARACTER*45 DEVDIR,NAME*10
      REAL*4 X(128),Y(128)

      DATA DEVDIR,NAME/'DUA0 :',' '/
      DATA LOGOLD/2/

      NBYTES=NCOL*4
      WRITE(IMP,100)
      CALL FICH('069',LOGOLD,1,DEVDIR,NAME,NROW,NBYTES,1,
      1         'DIRECT',LEC,IMP)
C
C  PART 1 : Successive actualization of NO() according to the
C           address calculations after each sort over.
C
C     For each sorting colum guide.
      DO I=1,NROW
         NOR(I)=I
      END DO
      DO J=1,NSORT
         DO I=1,NROW

C        For next column guides, permits actualization of NO().
            READ(LOGOLD'NOR(I)) (X(K),K=1,NCOL)
            Y(I)=X(AR(J))
         END DO

C        Tree sort is needed for multi sorts, if the data are
C        in previous order, the user will "burst" them by a
C        dummy sort according to any unsorted column to
C        restore the performances of this method.

         CALL EXTREE(Y,NO,NROW,1)
C
         DO L=1,NROW
            K=NOR(NO(L))
            DUM(L)=K
         END DO
         DO N=1,NROW
            NOR(N)=DUM(N)
         END DO
      END DO
C
C  PART 2 : Rearranging lines  according to last NO(), without
C           extra storage.
C
      DO I=1,NROW
         READ(LOGOLD'I) (X(J),J=1,NCOL)
         READ(LOGOLD'NOR(I)) (Y(J),J=1,NCOL)
         WRITE(LOGOLD'I) (Y(J),J=1,NCOL)
         WRITE(LOGOLD'NOR(I)) (X(J),J=1,NCOL)
         J=I
         DO WHILE(NOR(J).NE.I)
            J=J+1
         END DO
         NOR(J)=NOR(I)
      END DO
      CLOSE(LOGOLD)
      RETURN
100   FORMAT(/'$',3X,'Assign the ''OLD'' direct access file :
1            '/4X,37('-'))
      END

C**************************************************************
      SUBROUTINE TRANS(NCOL,COLMNE,N1,N2,N)
C**************************************************************
C
C     Used to transform column mnemonic names to corresponding
C     column numbers.
C
C     ARGUMENTS :
C
C     NCOL   : The number of columns.
C     COLMNE: Array storing the column mnemonic names.
C     N1     : The number of guides.
C     N2     : Array  storing  the column names used as  guides
C              for the multisorting.
C     N      : Array  storing   the   column   numbers    after
C              transformation.
C
C**************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 N(128)
      CHARACTER*24 COLMNE(128),N2(128)
      LOGICAL*1 TEST

      DO I=1,N1
```

```
         DO J=1,NCOL
            IF (COLMNE(J).EQ.N2(I)) THEN
               TEST=.TRUE.
            ELSE
               TEST=.FALSE.
            END IF
            IF (TEST.EQ..TRUE.) THEN
               N(I)=J
            END IF
         END DO
      END DO
      RETURN
      END

C*****************************************************************
      SUBROUTINE COMROW(LEC,IMP,LOGN,NCOL,NROW,FLAG1,FLAG2,
     1                  COLMNE,NC,NCO,NCIN,IJ,A,B,ROWMNE1,NR,
     2                  NRO,NRIN,ROWMNE2,H1FNAME,H2FNAME,RMIN,
     3                  RMAX,NT,NTR,NTIN)
C*****************************************************************
C
C     Used to merge (combine) two data files and their headers
C     in the row-row (one over the other) sense.
C
C     ARGUMENTS :
C
C     LEC        : The logical unit number for writing on the
C                  terminal.
C     IMP        : The logical unit number for reading from the
C                  terminal.
C     LOGN       : The logical chanel for the file on which
C                  will be stored the header's data.
C     NCOL       : The number of columns.
C     NROW       : The number of rows.
C     FLAG1      : Integer array of six elements used as flag
C                  to indicate the existence or not of the
C                  several informations regarding the columns.
C     FLAG2      : The corresponding flag for rows.
C     COLMNE     : Array storing the column mnemonic names.
C     NC         : The number of assigned column mnemonic names.
C     NCO        : The numbers of columns for which have been
C                  assigned names.
C     NCIN       : The invert relative addresses for COLMNE.
C     IJ         : The number of assigned tracing extrema sets.
C     A,B        : Arrays storing the number of row on which
C                  starts a set of common name rows and the
C                  corresponding on which it terminates.
C     ROWMNE1    : Array storing the row mnemonic names.
C     NR         : The number of assigned row mnemonic names.
C     NRO        : The numbers of rows for which have been
C                  assigned names.
C     NRIN       : The invert relative addresses of ROWMNE.
```

```
C     ROWMNE2   : Array storing the mnemonics of sets of rows.
C     H1FNAME   : The  name of the first  header's file  which
C                 will be merged.
C     H2FNAME   : The name of the second file for merging.
C     RMIN,RMAX : Arrays  storing  the minimum and the maximum
C                 values between  which tracing of the data is
C                 going to take place.
C     NT        : The number  of  columns  for  which  tracing
C                 extrema have been assigned.
C     NTR       : The  column  numbers  for  which  for  which
C                 tracing extrema have been assigned.
C     NTIN      : The invert relative addresses for RMI,RMAX.
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 FLAG1(6),FLAG2(3),NCO(128),NRO(128),A(128),
      1         B(128),AD,BD,NTR(128),NCOD(128),FLAG12(6),
      2         FLAG22(3),NTRD(128),NTIN(128),NTIND(128),
      3         NRIN(128),NCIN(128),NCIND(128)
      CHARACTER*24 COLMNE(128),H1FNAME,H2FNAME,COLMNED(128),
      1         ROWMNE1(128),ROWMNE2(128),ROWMNED
      REAL*4 RMIN(128),RMAX(128),RMIND(128),RMAXD(128),X(128)
      CHARACTER*45 DEVDIR,NAME*10

      DATA DEVDIR,NAME/'DUA0 :',' '/
      DATA LOGNEW,LOGOLD1,LOGOLD2/3,1,2/

C     Open the first header's file.

      OPEN(UNIT=LOGN,FILE=H1FNAME,STATUS='OLD')
      REWIND LOGN

C     Read the data to be combined.

      READ(LOGN,100) NC1,NR1,NCA,NRA,IJA,NTA
      READ(LOGN,200) (FLAG1(I),I=1,6)
      IF (FLAG1(1).EQ.1) THEN
         READ(LOGN,300) (COLMNE(I),I=1,NCA)
         READ(LOGN,200) (NCOD(I),I=1,NCA)
         READ(LOGN,200) (NCIN(I),I=1,NCA)
         DO I=1,NCA
            COLMNED(I)=COLMNE(NCIN(I))
         END DO
      END IF
      IF (FLAG1(2).EQ.1) THEN
         READ(LOGN,500) (RMIN(I),I=1,NTA)
         READ(LOGN,500) (RMAX(I),I=1,NTA)
         READ(LOGN,200) (NTRD(I),I=1,NTA)
         READ(LOGN,200) (NTIN(I),I=1,NTA)
         DO I=1,NTA
            RMIND(I)=RMIN(NTIN(I))
```

```
                RMAXD(I)=RMAX(NTIN(I))
             END DO
          END IF
          READ(LOGN,200) (FLAG2(I),I=1,3)
          IF (FLAG2(1).EQ.1) THEN
             READ(LOGN,300) (ROWMNE1(I),I=1,NRA)
             READ(LOGN,200) (NRO(I),I=1,NRA)
             READ(LOGN,200) (NRIN(I),I=1,NRA)
          END IF
          IF (FLAG2(2).EQ.1) THEN
             READ(LOGN,300) (ROWMNE2(I),I=1,IJA)
             READ(LOGN,200) (A(I),I=1,IJA)
             READ(LOGN,200) (B(I),I=1,IJA)
          END IF
          CLOSE(UNIT=LOGN)

    C     Open the second header's file.

          OPEN(UNIT=LOGN,FILE=H2FNAME,STATUS='OLD')
          REWIND LOGN
          READ(LOGN,100) NC2,NR2,NCB,NRB,IJB,NTB

    C      If the two headers have the same number of columns
    C     the process of combination is continued.

          IF (NC1.EQ.NC2.AND.(NR1+NR2).LE.128) THEN
             NCOL=NC1
             NROW=NR1+NR2
             READ(LOGN,200) (FLAG12(I),I=1,6)
             IF (FLAG12(1).EQ.1) THEN
                FLAG1(1)=1
                NC=NCA+NCB
                READ(LOGN,300) (COLMNE(I),I=1,NCB)
                READ(LOGN,200) (NCOD(I),I=1+NCA,NC)
                READ(LOGN,200) (NCIN(I),I=1,NCB)
                DO I=1,NCB
                   COLMNED(I+NCA)=COLMNE(NCIN(I))
                END DO
                CALL EXSH1(NCOD,NCIND,NC,1)

    C      Combination of mnemonic names.

                J=1
                K=1
                L=0
                DO WHILE(J.LE.NC)
                   IF (NCOD(J).EQ.NCOD(J+1)) THEN
                      NCO(K)=NCOD(J)
                      COLMNE(K)=COLMNED(NCIND(J))
                      NCIN(K)=K
                      J=J+2
                      K=K+1
                L=L+1
             ELSE
                NCO(K)=NCOD(J)
                COLMNE(K)=COLMNED(NCIND(J))
                NCIN(K)=K
                J=J+1
                K=K+1
             END IF
          END DO
       END IF
       NC=NC-L
       IF (FLAG12(2).EQ.1) THEN
          FLAG1(2)=1
          NT=NTA+NTB
          READ(LOGN,500) (RMIN(I),I=1,NTB)
          READ(LOGN,500) (RMAX(I),I=1,NTB)
          READ(LOGN,200) (NTRD(I),I=1+NTA,NT)
          READ(LOGN,200) (NTIN(I),I=1,NTB)
          DO I=1,NTB
             RMIND(I+NTA)=RMIN(NTIN(I))
             RMAXD(I+NTA)=RMAX(NTIN(I))
          END DO
          CALL EXSH1(NTRD,NTIND,NT,1)
          J=1
          K=1
          L=0
          DO WHILE(J.LE.NT)
             IF (NTRD(J).EQ.NTRD(J+1)) THEN
                NTR(K)=NTRD(J)
                RMIN(K)=MIN(RMIND(NTIND(J)),
     1                          RMIND(NTIND(J+1)))
                RMAX(K)=MAX(RMAXD(NTIND(J)),
     1                          RMAXD(NTIND(J+1)))
                NTIN(K)=K
                J=J+2
                K=K+1
                L=L+1
             ELSE
                NTR(K)=NTRD(J)
                RMIN(K)=RMIND(NTIND(J))
                RMAX(K)=RMAXD(NTIND(J))
                NTIN(K)=K
                J=J+1
                K=K+1
             END IF
          END DO
       END IF
       NT=NT-L
       READ(LOGN,200) (FLAG22(I),I=1,3)
       IF (FLAG22(1).EQ.1) THEN
          NR=NRA+NRB
          FLAG2(1)=1
```

```
      READ(LOGN,300) (ROWMNE1(I),I=1-NRA,NR)
      DO J=1,NRB
         READ(LOGN,200) NROD
         NRO(J+NRA)=NROD+NR1
      END DO
      DO J=1,NRB
         READ(LOGN,200) NRIND
         NRIN(J+NRA)=NRIND+NRA
      END DO
   END IF
   IF (FLAG22(2).EQ.1) THEN
      FLAG2(2)=1
      IJ=IJA+IJB
      DO I=1,IJB
         READ(LOGN,300) ROWMNED
         ROWMNE2(I+IJA)=ROWMNED
      END DO
      DO I=1,IJB
         READ(LOGN,200) AD
         A(I+IJA)=AD+NR1
      END DO
      DO I=1,IJB
         READ(LOGN,200) BD
         B(I+IJA)=BD+NR1
      END DO
   END IF
   CLOSE(UNIT=LOGN)
   NBYTES=NCOL*4
   WRITE(IMP,600)
   CALL FICH('069',LOGOLD1,1,DEVDIR,NAME,NR1,NBYTES,1,
1           'DIRECT',LEC,IMP)
   WRITE(IMP,700)
   CALL FICH('069',LOGOLD2,1,DEVDIR,NAME,NR2,NBYTES,1,
1           'DIRECT',LEC,IMP)
   WRITE(IMP,800)
   CALL FICH('069',LOGNEW,1,DEVDIR,NAME,NR1+NR2,NBYTES,
1            0,'DIRECT',LEC,IMP)
   DO I=1,NR1
      READ(LOGOLD1'I) (X(J),J=1,NCOL)
      WRITE(LOGNEW'I) (X(J),J=1,NCOL)
   END DO
   DO I=1,NR2
      READ(LOGOLD2'I) (X(J),J=1,NCOL)
      WRITE(LOGNEW'I+NR1) (X(J),J=1,NCOL)
   END DO
ELSE
   WRITE(IMP,400)
   CLOSE(LOGOLD1)
   CLOSE(LOGOLD2)
   CLOSE(LOGNEW)
   RETURN
END IF
```

```
100   FORMAT(6I5)
200   FORMAT(I5)
300   FORMAT(A24)
400   FORMAT(///4X,'COMBINATION IS NOT POSSIBLE DUE TO',
      1              7X,'DIFFERENT NUMBER OF COLUMNS!!')
500   FORMAT(F14.6)
600   FORMAT(/'$',3X,'Assign the first ''OLD'' direct ',
      1       'access file :'/4X,37('-'))
700   FORMAT(/'$',3X,'Assign the second ''NEW'' direct',
      1       ' access file :'/4X,37('-'))
800   FORMAT(/'$',3X,'Assign the ''NEW'' direct access file :
      1      '/4X,37('-'))
      CLOSE(LOGOLD1)
      CLOSE(LOGOLD2)
      CLOSE(LOGNEW)
      RETURN
      END

C**************************************************************
      SUBROUTINE COMCOL (LEC,IMP,LOGN,NCOL,NROW,FLAG1,FLAG2,
      1                  COLMNE,NC,NCO,NCIN,IJ,A,B,ROWMNE1,NR,
      2                  NRO,NRIN,ROWMNE2,H1FNAME,H2FNAME,
      3                  RMIN,RMAX,NT,NTR,NTIN)
C**************************************************************
C
C     Used to  merge (combine) two data files and their headrs
C     in the column-column (one aside the other) sense.
C
C     ARGUMENTS :
C
C     LEC        : The logical unit number for writing  on  the
C                  terminal.
C     IMP        : The logical unit number for reading from the
C                  terminal.
C     LOGN       : The  logical  chanel  for  the file on which
C                  will be stored the header's data.
C     NCOL       : The number of columns.
C     NROW       : The number of rows.
C     FLAG1      : Integer array of six elements  used as  flag
C                  to indicate  the existence  or  not  of  the
C                  several informations regarding the columns.
C     FLAG2      : The corresponding flag for rows.
C     COLMNE     : Array storing the column mnemonic names.
C     NC         : The number of assigned column mnemonic names.
C     NCO        : The numbers of columns for which have   been
C                  assigned names.
C     NCIN       : The invert relative addresses for COLMNE.
C     IJ         : The number of assigned tracing extrema sets.
C     A,B        : Arrays storing the number  of  row on  which
C                  starts a  set of common  name  rows  and the
C                  corresponding on which it terminates.
C     ROWMNE1    : Array storing the row mnemonic names.
```

```
C     NR         : The number of assigned row mnemonic names.
C     NRO        : The  numbers  of  rows  for  which have been
C                  assigned names.
C     NRIN       : The invert relative addresses of ROWMNE.
C     ROWMNE2    : Array storing the mnemonics of sets of rows.
C     H1FNAME    : The  name of the first  header's file  which
C                  will be merged.
C     H2FNAME    : The name of the second file for merging.
C     RMIN,RMAX  : Arrays  storing  the minimum and the maximum
C                  values between  which tracing of the data is
C                  going to take place.
C     NT         : The number  of  columns  for  which  tracing
C                  extrema have been assigned.
C     NTR        : The  column  numbers  for  which  for  which
C                  tracing extrema have been assigned.
C     NTIN       : The invert relative addresses for RMI,RMAX.
C
C**************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 FLAG1(6),FLAG2(3),NCO(128),NRO(128),A(128),
     1          B(128),NTR(128),NROD(128),FLAG12(6),FLAG22(3),
     2          NTRD,NTIN(128),NTIND,NRIN(128),NCIN(128),
     3          NRIND(128),NCIND
      CHARACTER*24 COLMNE(128),H1FNAME,H2FNAME,ROWMNE1D(128),
     1          ROWMNE1(128),ROWMNE2(128),ROWMNED
      REAL*4 RMIN(128),RMAX(128),RMIND(128),RMAXD(128),X(128)
      CHARACTER*45 DEVDIR,NAME*10

      DATA DEVDIR,NAME/'DUA0 :',' '/
      DATA LOGNEW,LOGOLD1,LOGOLD2/3,1,2/

      OPEN(UNIT=LOGN,FILE=H1FNAME,STATUS='OLD')
      REWIND LOGN
      READ(LOGN,100) NC1,NR1,NCA,NRA,IJA,NTA
      READ(LOGN,200) (FLAG1(I),I=1,6)
      IF (FLAG1(1).EQ.1) THEN
         READ(LOGN,300) (COLMNE(I),I=1,NCA)
         READ(LOGN,200) (NCO(I),I=1,NCA)
         READ(LOGN,200) (NCIN(I),I=1,NCA)
      END IF
      IF (FLAG1(2).EQ.1) THEN
         READ(LOGN,500) (RMIN(I),I=1,NTA)
         READ(LOGN,500) (RMAX(I),I=1,NTA)
         READ(LOGN,200) (NTR(I),I=1,NTA)
         READ(LOGN,200) (NTIN(I),I=1,NTA)
      END IF
      READ(LOGN,200) (FLAG2(I),I=1,3)
      IF (FLAG2(1).EQ.1) THEN
         READ(LOGN,300) (ROWMNE1(I),I=1,NRA)
         READ(LOGN,200) (NROD(I),I=1,NRA)
         READ(LOGN,200) (NRIN(I),I=1,NRA)
         DO I=1,NRA
            ROWMNE1D(I)=ROWMNE1(NRIN(I))
         END DO
      END IF
      IF (FLAG2(2).EQ.1) THEN
         READ(LOGN,300) (ROWMNE2(I),I=1,IJA)
         READ(LOGN,200) (A(I),I=1,IJA)
         READ(LOGN,200) (B(I),I=1,IJA)
      END IF
      CLOSE(UNIT=LOGN)
      OPEN(UNIT=LOGN,FILE=H2FNAME,STATUS='OLD')
      REWIND LOGN
      READ(LOGN,100) NC2,NR2,NCB,NRB,IJB,NTB
      IF (NR1.EQ.NR2.AND.NC1+NC2.LE.128) THEN
         NROW=NR1
         NCOL=NC1+NC2
         READ(LOGN,200) (FLAG12(I),I=1,6)
         IF (FLAG12(1).EQ.1) THEN
            NC=NCA+NCB
            FLAG1(1)=1
            READ(LOGN,300) (COLMNE(I),I=1+NCA,NC)
            DO I=1,NCB
               READ(LOGN,200) NCOD
               NCO(I+NCA)=NCOD+NC1
            END DO
            DO J=1,NCB
               READ(LOGN,200) NCIND
               NCIN(J+NCA)=NCIND+NCA
            END DO
         END IF
         IF (FLAG12(2).EQ.1) THEN
            NT=NTA+NTB
            FLAG1(2)=1
            READ(LOGN,500) (RMIN(I),I=NTA+1,NT)
            READ(LOGN,500) (RMAX(I),I=NTA+1,NT)
            DO J=1,NTB
               READ(LOGN,200) NTRD
               NTR(J+NTA)=NTRD+NC1
            END DO
            DO J=1,NTB
               READ(LOGN,200) NTIND
               NTIN(J+NTA)=NTIND+NTA
            END DO
         END IF
         READ(LOGN,200) (FLAG22(I),I=1,3)
         IF (FLAG22(1).EQ.1) THEN
            FLAG2(1)=1
            NR=NRA+NRB
            READ(LOGN,300) (ROWMNE1(I),I=1,NRB)
            READ(LOGN,200) (NROD(I),I=1+NRA,NR)
            READ(LOGN,200) (NRIN(I),I=1,NRB)
            DO I=1,NRB
```

```
               ROWMNE1D(I-NRA)=ROWMNE1(NRIN(I))
            END DO
            CALL EXSH1(NROD,NRIND,NR,1)
            J=1
            K=1
            L=0
            DO WHILE (J.LE.NR)
               IF (NROD(J).EQ.NROD(J+1)) THEN
                  NRO(K)=NROD(J)
                  ROWMNE1(K)=ROWMNE1D(NRIND(J))
                  NRIN(K)=K
                  J=J+2
                  K=K+1
                  L=L+1
               ELSE
                  NRO(K)=NROD(J)
                  ROWMNE1(K)=ROWMNE1D(NRIND(J))
                  NRIN(K)=K
                  J=J+1
                  K=K+1
               END IF
            END DO
         END IF
         NR=NR-L
         CLOSE(UNIT=LOGN)
         NBYTES=NC1*4
         WRITE(IMP,600)
         CALL FICH('069',LOGOLD1,1,DEVDIR,NAME,NR1,NBYTES,
      1            1,'DIRECT',LEC,IMP)
         NBYTES=NC2*4
         WRITE(IMP,700)
         CALL FICH('069',LOGOLD2,1,DEVDIR,NAME,NR2,NBYTES,
      1            1,'DIRECT',LEC,IMP)
         NBYTES=(NC1+NC2)*4
         WRITE(IMP,800)
         CALL FICH('069',LOGNEW,1,DEVDIR,NAME,NR1+NR2,
      1              NBYTES,0,'DIRECT',LEC,IMP)
         DO I=1,NROW
            READ(LOGOLD1'I) (X(J),J=1,NC1)
            READ(LOGOLD2'I) (X(J),J=NC+1,NC2)
            WRITE(LOGNEW'I) (X(J),J=1,NC2)
         END DO
      END IF
100   FORMAT(6I5)
200   FORMAT(I5)
300   FORMAT(A24)
400   FORMAT(///4X,'COMBINATION IS NOT POSSIBLE DUE TO',
      1             7X,'DIFFERENT NUMBER OF ROWS!!')
500   FORMAT(F14.6)
600   FORMAT(/'$',3X,'Assign the first ''OLD'' direct',
      1       ' access file :'/4X,37('-'))
700   FORMAT(/'$',3X,'Assign the second ''NEW'' direct ',
     1       'access file : '/4X,37('-'))
800  FORMAT(/'$',3X,'Assign the ''NEW'' direct access file :
     1      '/4X,37('-'))
     CLOSE(LOGOLD1)
     CLOSE(LOGOLD2)
     CLOSE(LOGNEW)
     RETURN
     END

C***************************************************************
     SUBROUTINE EXTREE(A,NO,N,IND)
C***************************************************************
C
C  Subroutine  for internal and address calculation sort using
C  the  tree sort ("Monkey-puzzle sort") to obtain the rank of
C  the X's, in  this  case  the  original  array  X()  is  not
C  modified.
C
C  If the X() are in random order the number of comparisons is
C  of  the  order of  N log2(N); if the X() are already in the
C  required order  or  reverse order the number of comparisons
C  is of the order of N*N/2.
C
C  ARGUMENTS:
C
C  X    : Elements to sort.
C  NO   : Address calculation for the rank.
C  N    : Number of elements to sort.
C  IND  : If >0, elements  are  sorted  in ascending order, if
C         not, in descending order.
C         2048 items maximum.
C
C  Reference :
C  Windley P.F. (1960) : "Trees, Forests and Rearranging",
C  Computer J, vol.3, no.2, July, p.84-88.
C
C***************************************************************
C
     IMPLICIT INTEGER*2 (I-N)
C
     INTEGER*2 ILB(2048),IRB(2048),NO(2048)
C
     REAL*4 A(2048),AI,AJ
C
C    Place at first A(1) on the root.
C
     ILB(1)=0
     IRB(1)=0
     DO I=2,N
        ILB(I)=0
        IRB(I)=0
```

```
C     Select the root of the tree as the item for comparison.

         J=1
         K=1
         DO WHILE(K.EQ.1)
            K=0
C
C           Permutation in function of IND.
C
            IF(IND.GT.0) THEN
C
C              Sort by  ascending order.
C
               AI=A(I)
               AJ=A(J)
            ELSE
C
C              Sort by descending order.
C
               AI=A(J)
               AJ=A(I)
            END IF
C
C     Compare the new item with the item for comparison. If
C     the new item should follow,
C
            IF(AI.GE.AJ) THEN
C     If the item for comparison has a right brance, take the
C     item to which that branch points as the new item for
C     comparison,
C
               IF (IRB(J).LE.0) THEN
C
C     Copy the backtrack of the item for comparison into the
C     backtrack of the new item, and set the right pointer of
C     the item for comparison to point to the new item.
C
                  IRB(I)=IRB(J)
                  IRB(J)=I
               ELSE
                  J=IRB(J)
C
C     Make the backtracks of the new item point to the item
C     for comparison, and set the left pointer of the item for
C     comparison to point to the new item.

                  K=1
               END IF
            ELSE
C
C     If the item for comparison has a left branch, take the
C     item to which that branch points as the new item for
```

```
C     comparison.
C
               IF (ILB(J).EQ.0) THEN
                  IRB(I)=-J
                  ILB(J)=I
               ELSE
                  J=ILB(J)
                  K=1
               END IF
            END IF
         END DO
      END DO
C
C     Start with the item at the root of the tree.
C
      M=0
C
      J=1
      K=1
      DO WHILE (K.EQ.1)
         K=0
C
C     If  this  item has a left branch, take the item to which
C     that branch points.
C
         IF(ILB(J).GT.0) THEN
            J=ILB(J)
            K=1
         ELSE
            L=1
C
C     This is the next item in order.
C
            DO WHILE (L.EQ.1)
               L=0
C
C                 Ordered item found, adress NO().
C
               M=M+1
               NO(M)=J
C
C     If this  item has a right branch, take the item to which
C     that branch points.
C
                  IF(IRB(J).NE.0) THEN
                     IF(IRB(J).GT.0) THEN
                        J= IRB(J)
                        K=1
                     ELSE
                        J=-IRB(J)
C
C     Take the item to which the backtrack points.
```

```
C
                        L=1
                     END IF
                  ELSE
C                    End of process.
                     RETURN
                  END IF
               END DO
            END IF
         END DO
      END

C***************************************************************
      SUBROUTINE RANDOM(LEC,IMP,NCOL,NROW)
C***************************************************************
C
C     Used  to  randomize  the  data  matrix  of  an  assigned
C     mass-storage file.
C
C     ARGUMENTS :
C
C     LEC  : The  logical  unit  number  for  writing  on   the
C            terminal.
C     IMP  : The  logical  unit  number  for  reading from the
C            terminal.
C     NCOL : The number of columns.
C     NROW : The number of rows.
C
C***************************************************************

      IMPLICIT INTEGER*2 (I-N)
      REAL*4 XRAN(2048)
      INTEGER*2 AR(128),NO(128),INIT(6)
      CHARACTER*45 DEVDIR,NAME*10
      REAL*4 X(128),Y(128)

      DATA DEVDIR,NAME/'DUA0 :',' '/
      DATA LOGOLD/2/
      DATA INIT/59,47,62,38,45,23/
      DATA INDRAN/1/

      NBYTES=NCOL*4
      WRITE(IMP,100)
      CALL FICH('069',LOGOLD,1,DEVDIR,NAME,NROW,NBYTES,1,
      1         'DIRECT',LEC,IMP)

      DO I=1,NROW
         XRAN(I)=RANDAN(INIT,INDRAN)
      END DO
C     Use tree sort because data are randomized.
```

```
      CALL EXTREE(XRAN,NO,NROW,1)
      DO I=1,NROW
         READ(LOGOLD'I) (X(J),J=1,NCOL)
         READ(LOGOLD'NO(I)) (Y(J),J=1,NCOL)
         WRITE(LOGOLD'I) (Y(J),J=1,NCOL)
         WRITE(LOGOLD'NO(I)) (X(J),J=1,NCOL)
         J=I
         DO WHILE(NO(J).NE.I)
            J=J+1
         END DO
         NO(J)=NO(I)
      END DO
      CLOSE(LOGOLD)
      RETURN
100   FORMAT(/'$',3X,'Assign the ''OLD'' direct access file :
     1      '/4X,37('-'))
      END

C**************************************************************
      FUNCTION RANDAN(INIT,IND)
C**************************************************************
C
      IMPLICIT INTEGER*2 (I-N)
C
      DOUBLE PRECISION X
C
      INTEGER*2 A(6),INIT(6),LN(6),LNP1(6),C,P,Q
C
      DATA M/36/Q/6/A/59,47,62,38,45,23/
C
      IF(IND.NE.0)THEN
C
         I=0
         DO WHILE (I.LT.Q)
            I=I+1
            LN(I)=INIT(I)
         END DO
C
         IND=0
C
      ELSE
      END IF
C
      C=0
      K=Q
      MQ=M/Q
      IMQ=2.**MQ
C
      DO WHILE (K.GT.0)
         IQMK=Q-K
         I=0
         P=0
         DO WHILE (I.LE.IQMK)
            P=P+A(I+K)*LN(Q-I)
            I=I+1
         END DO
         P=P+C
         C=P/IMQ
         LNP1(K)=P-C*IMQ
         K=K-1
      END DO
C
      I=0
      X=0.D0
C
      DO WHILE (I.LT.Q)
         I=I+1
         X=X+LNP1(I)*2.**(-MQ*I)
         LN(I)=LNP1(I)
      END DO
C
      RANDAN=X
C
      RETURN
C
      END

C************************************************************
      SUBROUTINE FINDMM (LEC,IMP,NCOL,NROW,FMIN,FMAX)
C************************************************************

      IMPLICIT INTEGER*2 (I-N)
      REAL*4 X(128),FMIN(128),FMAX(128)
      CHARACTER*45 DEVDIR,NAME*10
      DATA DEVDIR,NAME/'DUA0 :',' '/
      DATA LOGOLD/2/

      DO J=1,NCOL
         FMIN(J)=+1.E+38
         FMAX(J)=-1.E+38
      END DO

      WRITE(IMP,100)
      NBYTES=NCOL*4
      CALL FICH   ('069',LOGOLD,1,DEVDIR,NAME,NROW,NBYTES,1,
      1               'DIRECT',LEC,IMP)
      DO I=1,NROW
         READ(LOGOLD'I) (X(K),K=1,NCOL)
         DO J=1,NCOL
            IF (FMIN(J).GT.X(J)) THEN
               FMIN(J)=X(J)
            END IF
            IF (FMAX(J).LT.X(J)) THEN
               FMAX(J)=X(J)
            END IF
         END DO
      END DO
      CLOSE(LOGOLD)
      RETURN
100   FORMAT(/'$',3X,'Assign the ''OLD'' direct access file :
      1      '/4X,37('-'))
      END

C*************************************************************
      SUBROUTINE CONV(NCOL,NROW,COLMNE,NC,INDEX)
C*************************************************************
C
C     This subroutine analyzes the expression  inserted by the
C     keyboard in the form of string in the following kinds of
C     elements:
C        . Functions (eg. SQRT[7])
C        . Row assignment (eg. [8])
C        . Reals or integers
C        . Delimiters (eg."(",")")
C        . Operators : '+','-','*','/','^'
C     The result  of  this  analysis  is  an  infix expression
C     consisted  of  the  above  kind  of  elements,  but   in
C     characters.
C
C     ARGUMENTS:
C
C       NCOL   : The number of columns.
C     NROW   : The number of rows.
C     COLMNE : Array storing the column mnemonic names and the
C              assigned transformations.
C     NC     : The index of the array COLMNE.
C     INDEX  : Pointer marking the index of  COLMNE  on  which
C              the transformations start.
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 CN,PP
      REAL*4 X(128),ARG,VAL,R,RES
      CHARACTER*24 COLMNE(128),STRING
      CHARACTER*1 SYMB(24),INF(18,14),POST(18,14),F(7),NUM(3),
      1           CONS(14),BR
      CHARACTER*5 FUN
      CHARACTER*45 NAME*10,DEVDIR
      LOGICAL*1 MARK,TEST,NEG,CHECK

      EQUIVALENCE(SYMB(1),STRING)

      DATA DEVDIR,NAME/'DUA0 :',' '/
      DATA LOGOLD,LOGNEW/2,3/
      NBYTES=(NCOL-(NC-INDEX))*4
C     Open the file which contains the data to be transformed.
      WRITE (6,300)
      CALL FICH('069',LOGOLD,1,DEVDIR,NAME,NROW,NBYTES,1,
     1          'DIRECT',5,6)
      NBYTES=NCOL*4
C     Open   a   new   file   with   the   same   name,   on which the
C     transformed data will be stored.
      WRITE (6,400)
      CALL FICH('069',LOGNEW,1,DEVDIR,NAME,NROW,NBYTES,0,
     1          'DIRECT',5,6)
C     For each row of the  data  matrix,  execute the assigned
C     transformation.
      DO JK=1,NROW
C        Read the entire row of data.
      READ(LOGOLD'JK) (X(K),K=1,NCOL-(NC-INDEX))
C        Calculate the number of transformations (IND).
      IND=NC-INDEX
C        For each transformation,
      DO IK=1,NC-INDEX
C     Store  the  expression  representing  the transformation
C     in the variable STRING. In following this string will be
C     analyzed.
         STRING=COLMNE(IK+INDEX)
C        Initialize the pointers.
         J=1
         N=1
         L=1
         M=1
         IP=1
         BR=' '
C        Initialize the flags.
         TEST=.FALSE.
         MARK=.FALSE.
         CHECK=.FALSE.
         NEG=.FALSE.

C        For each character of the string,

         DO I=1,24

C        If the character is a letter, store it in the one
C        character array F and signal the encountering of a
C        function in the transformation by seting the flag
C        TEST to true.

            IF (SYMB(I).GE.'A'.AND.SYMB(I).LE.'Z') THEN
               F(N)=SYMB(I)
               TEST=.TRUE.
               N=N+1

C              If the character is number or ".", then

               ELSE IF ((SYMB(I).GE.'0'.AND.SYMB(I).LE.'9').OR.
      1                  SYMB(I).EQ.'.') THEN

C             If the bracket is closed or no bracket has still
C             been encountered, the number is a constant used
C             by the transformation. Store the number in the
C             array CONS and mark the encountering of a
C             constant by seting the flag MARK to true.

                  IF (BR.EQ.']'.OR.BR.EQ.' ') THEN
                     CONS(L)=SYMB(I)
                     MARK=.TRUE.
                     L=L+1

C                 Else if the bracket is opened, then

                  ELSE IF (BR.EQ.'[') THEN

C                if function has been encountered, the number
C                is the representation of the column number
C                the value of wich is going to be used as
C                argument of the function.

                     IF (TEST.EQ..TRUE.) THEN
                        F(N)=SYMB(I)
                        N=N+1

C                    Else, the number is the representation of
C                    column but the coresponding value is
C                    going to be used as a constant.

                     ELSE
                        NUM(M)=SYMB(I)
```

```
                CHECK=.TRUE.
                M=M-1
              END IF
            END IF
C         If the character is "[", set it to variable BR

          ELSE IF (SYMB(I).EQ.'[') THEN
            BR=SYMB(I)

C         if function has been encounterd, store  the "["
C         in the same array with the function (F)

            IF (TEST.EQ..TRUE.) THEN
              F(N)=SYMB(I)
              N=N+1

C           else  store  it  in  the same array with the
C           numbers representing columns.

            ELSE
              NUM(M)=SYMB(I)
              M=M+1
            END IF

C       Else if the symbol is "]" then set it to variable
C       BR and,

        ELSE IF (SYMB(I).EQ.']') THEN
          BR=SYMB(I)

C         if function has been encountered,  store it in
C         the same array

          IF (TEST.EQ..TRUE.) THEN
            F(N)=SYMB(I)

C         else  store  it  with  the mumber representing
C         column.

          ELSE
            NUM(M)=SYMB(I)
          END IF

C       Else if the symbol is "-", then

        ELSE IF (SYMB(I).EQ.'-') THEN

C       if  the  previous  character  was "(", means that
C       follows negative number and mark it by seting the
C       flag NEG to true.
```

```
                        IF (SYMB(I-1).EQ.'(') THEN
                           NEG=.TRUE.

C                       Else the character  "-"  represents the symbol
C                       of the subtraction.

                        ELSE

C                          If function has been encountered,

                           IF (TEST.EQ..TRUE.) THEN

C                          If  the  NEG flag is true the "-" is placed
C                          at the begining  of the function storing in
C                          the generated infix expression.

                              IF (NEG.EQ..TRUE.) THEN
                                 INF(IP,1)='-'

C                             and in followng the function name stored
C                             in the array F is  also  placed  in  the
C                             infix expression.

                                 DO J=1,N
                                    INF(IP,J+1)=F(J)
                                 END DO

C                             Since  the  "-"  is  used  to negate the
C                             function the flag is reset to FALSE,

                                 NEG=.FALSE.

C                             and the  pointer  of the generated infix
C                             expression IP, is increased.

                                 IP=IP+1

C                             The "(" is also placed as element of the
C                             infix,

                                 INF(IP,1)=SYMB(I)
                                 IP=IP+1

C                             and the flag  TEST  is reset to FALSE to
C                             indicate that no function is active.

                                 TEST=.FALSE.

C                             The  index  of  array  F  is set to one,
C                             ready to accept the next fuction name.

                                 N=1
```

```
C             Else (the function is not negative)
C             repeat the same procces without negating
C             the function.

                ELSE
                  DO J=1,N
                    INF(IP,J)=F(J)
                  END DO
                  IP=IP+1
                  INF(IP,1)=SYMB(I)
                  IP=IP+1
                  TEST=.FALSE.
                  N=1
                END IF

C             Else if constant has been encountered,

              ELSE IF (MARK.EQ..TRUE.) THEN

C           if  the NEG flag is set, place the constant
C           stored in array CONS in the generated
C           infix expression but negeting it first.

                IF (NEG.EQ..TRUE.) THEN
                INF(IP,1)='-'
                DO J=1,L-1
                  INF(IP,J+1)=CONS(J)
                END DO
                NEG=.FALSE.
                IP=IP+1
                INF(IP,1)=SYMB(I)
                IP=IP+1
                MARK=.FALSE.
                L=1

C           Otherwise do the same without negation.

              ELSE
                DO J=1,L-1
                  INF(IP,J)=CONS(J)
                END DO
                IP=IP+1
                INF(IP,1)=SYMB(I)
                IP=IP+1
                MARK=.FALSE.
                L=1
              END IF

C         Repeat the same process if number representing
C         column has been encountered.
```

```
                   ELSE IF (CHECK.EQ..TRUE.) THEN
                      IF (NEG.EQ..TRUE.) THEN
                         INF(IP,1)='-'
                         DO J=1,M
                            INF(IP,J+1)=NUM(J)
                         END DO
                         NEG=.FALSE.
                         IP=IP+1
                         INF(IP,1)=SYMB(I)
                         IP=IP+1
                         CHECK=.FALSE.
                         M=1
                      ELSE
                         DO J=1,M
                            INF(IP,J)=NUM(J)
                         END DO
                         IP=IP+1
                         INF(IP,1)=SYMB(I)
                         IP=IP+1
                         CHECK=.FALSE.
                         M=1
                      END IF
                   END IF
                END IF

C         If  the character is an operator or end of string is
C         encountered, the current active function,constant or
C         column value must be  placed  in the generated infix
C         expression. The same process used  above for the "-"
C         is going to be used herein also.

             ELSE IF ((SYMB(I).EQ.'('.OR.SYMB(I).EQ.')'.OR.
     1                  SYMB(I).EQ.'+'.OR.
     2                  SYMB(I).EQ.'*'.OR.SYMB(I).EQ.'/'.OR.
     3                  SYMB(I).EQ.'^'.OR.SYMB(I).EQ.' ').AND.
     4                  SYMB(I-1).NE.' ') THEN

                IF (TEST.EQ..TRUE.) THEN
                   IF (NEG.EQ..TRUE.) THEN
                      INF(IP,1)='-'
                      DO J=1,N
                         INF(IP,J+1)=F(J)
                      END DO
                      NEG=.FALSE.
                      IP=IP+1
                      INF(IP,1)=SYMB(I)
                      IP=IP+1
                      TEST=.FALSE.
                      N=1
                   ELSE
                      DO J=1,N
                         INF(IP,J)=F(J)
```

```
      END DO
      IP=IP+1
      INF(IP,1)=SYMB(I)
      IP=IP+1
      TEST=.FALSE.
      N=1
   END IF
ELSE IF (MARK.EQ..TRUE.) THEN
   IF (NEG.EQ..TRUE.) THEN
      INF(IP,1)='-'
      DO J=1,L-1
         INF(IP,J+1)=CONS(J)
      END DO
      NEG=.FALSE.
      IP=IP+1
      INF(IP,1)=SYMB(I)
      IP=IP+1
      MARK=.FALSE.
      L=1
   ELSE
      DO J=1,L-1
         INF(IP,J)=CONS(J)
      END DO
      IP=IP+1
      INF(IP,1)=SYMB(I)
      IP=IP+1
      MARK=.FALSE.
      L=1
   END IF
ELSE IF (CHECK.EQ..TRUE.) THEN
   IF (NEG.EQ..TRUE.) THEN
      INF(IP,1)='-'
      DO J=1,M
         INF(IP,J+1)=NUM(J)
      END DO
      NEG=.FALSE.
      IP=IP+1
      INF(IP,1)=SYMB(I)
      IP=IP+1
      CHECK=.FALSE.
      M=1
   ELSE
      DO J=1,M
         INF(IP,J)=NUM(J)
      END DO
      IP=IP+1
      INF(IP,1)=SYMB(I)
      IP=IP+1
      CHECK=.FALSE.
      M=1
   END IF
ELSE IF (TEST.EQ..FALSE..AND.MARK.EQ..FALSE.
      2               .AND.CHECK.EQ..FALSE.) THEN
                   INF(IP,1)=SYMB(I)
                   IP=IP+1
                END IF
             END IF
          END DO
          IP=IP-2

C      As  soon  an infix expression is generated, the process
C      for  its   evaluation  is  continued,  by  calling  the
C      appropriate subroutines.

          CALL PFIX(INF,POST,IP,PP)
          CALL EVAL(LOGOLD,POST,RES,PP,INDEX,NC,JK,NCOL,
      1                    X,FUN,ARG,IERR)

C      The  result  is  placed  in one position after the last
C      column increasing the numbers of column by one.

          X(NCOL-IND+1)=RES

C      One  transformation has been evaluated and their number
C      is decreased.

          IND=IND-1
       END DO

C      If   the flag IERR used to signal that invalid argument
C      for a function has been encountered (eg. Square root of
C      negative  number) is not set then the new row (contanig
C      the transformed columns) is written to the new file.

       IF (IERR.NE.1.AND.IERR.NE.2) THEN
          WRITE(LOGNEW'JK) (X(K),K=1,NCOL)

C      Else a message is displayed and the program returns for
C      new assignement.

       ELSE IF (IERR.EQ.1) THEN
           WRITE(6,100) FUN,ARG
           CLOSE(LOGNEW)
           RETURN
       ELSE
           WRITE(6,200)
           CLOSE(LOGNEW)
           RETURN
         END IF
      END DO
      CLOSE(LOGNEW)
      RETURN
100   FORMAT(//4X,'INVALID FUNCTION ARGUMENT : ',A5,'(',
      1       F14.6,')')
```

```
200    FORMAT(//4X,'INVALID DIVISION BY 0 !!')
300    FORMAT(//4X,'Assign the ''OLD'' direct access file :'
       1       /4X     ,37('-'))
400    FORMAT(//4X,'Assign the ''NEW'' direct access file :'
       1       /4X     ,37('-'))
       END

C***************************************************************
       SUBROUTINE COMP(FUN,ARG,AP,IERR)
C***************************************************************
C
C      This  subroutine  is  used  to compute the values of the
C      functions that may be included in a transformation using
C      the FORTRAN 77 build-in functions.
C
C
C      ARGUMENTS:
C
C      FUN  : Name of the function.
C      ARG  : Argument of the function.
C      AP   : Result of the function.
C      IERR : Flag  indicating  that  an  invalid  argument  is
C             assigned.
C
C***************************************************************

       IMPLICIT INTEGER*2 (I-P)
       CHARACTER*5 FUN
       REAL*4 ARG,AP,AX

       IERR=0
       IF      (FUN.EQ.' SQRT') THEN
          IF (ARG.GE.0) THEN
             AP=SQRT(ARG)
          ELSE
             IERR=1
          END IF

C      Use  the build in  intrincic functions of FORTRAN 77 for
C      the  computation  of  the   several   assigned  by   the
C      expression functions.

       ELSE IF (FUN.EQ.'  LOG') THEN
          IF (ARG.LE.0) THEN
             AP=ALOG(ARG)
          ELSE
             IERR=1
          END IF
       ELSE IF (FUN.EQ.' LOGC') THEN
          IF (ARG.LE.0) THEN
             AP=LOG10(ARG)
          ELSE
```

```
              IERR=1
           END IF
        ELSE IF (FUN.EQ.'  EXP') THEN
           AP=EXP(ARG)
        ELSE IF (FUN.EQ.'  SIN') THEN
           AP=SIN(ARG)
        ELSE IF (FUN.EQ.'  COS') THEN
           AP=COS(ARG)
        ELSE IF (FUN.EQ.'  TAN') THEN
           AP=TAN(ARG)
        ELSE IF (FUN.EQ.' ASIN') THEN
           AP=ASIN(ARG)
        ELSE IF (FUN.EQ.' ACOS') THEN
           AP=ACOS(ARG)
        ELSE IF (FUN.EQ.' ATAN') THEN
           AP=ATAN(ARG)
        ELSE IF (FUN.EQ.'  ABS') THEN
           AP=ABS(ARG)
        ELSE IF (FUN.EQ.' SINH') THEN
           AP=SINH(ARG)
        ELSE IF (FUN.EQ.' COSH') THEN
           AP=COSH(ARG)
        ELSE IF (FUN.EQ.' TANH') THEN
           AP=TANH(ARG)

C       Compute the Arc Hyperbolic Sin function not  included in
C       the build in functions of the language.

        ELSE IF (FUN.EQ.'ASINH') THEN
           ARG=ABS(ARG)
           AP=ALOG(ARG+SQRT(ARG*ARG+1))
           IF (AP.LT.0) THEN
              AP=-AP
           END IF
        END IF

        RETURN
        END

  C*************************************************************
        SUBROUTINE PFIX(INF,POST,IP,PP)
  C*************************************************************
  C
  C     This subroutine transforms the infix expression produced
  C     by  the subroutine CONVERT to postfix notation using the
  C     push-down stack method.
  C
        IMPLICIT INTEGER*2 (I-N)
        INTEGER*2 PP
        CHARACTER*1 POST(18,14),INF(18,14),STACK(30)
  C
  C     Initialize the stack pointer K and the pointer PP of the
```

```
C     array holding the created postfix expression.
C
      K=0
      PP=0
C
C     For each element of the infix expression,
C
      DO I=1,IP
C
C     If  the  first character of the element is letter or "[",
C     place it on the generated postfix expression.
C
         IF ((INF(I,1).GE.'A'.AND.INF(I,1).LE.'Z').OR.INF(I,1).
     1        EQ.'[') THEN
            PP=PP+1
            DO J=1,14
               POST(PP,J)=INF(I,J)
            END DO

C     Else if  the  first  character  is  a number or "." then
C     place it on the generated postfix expresion.
C
         ELSE IF ((INF(I,1).GE.'0'.AND.INF(I,1).LE.'9').OR.
     1                INF(I,1).EQ.'.') THEN
            PP=PP+1
            DO J=1,14
               POST(PP,J)=INF(I,J)
            END DO

C     Else  if  the  element  is  "+",  which  has  the lowest
C     priority of operators,
C
         ELSE IF (INF(I,1).EQ.'+') THEN

C     if the stack is not empty and the topmost element is not
C     "(", pop the  stack and place the element on the postfix
C     expression, push the  next  element  of  the  infix
C     expression into the stack.

            IF (K.NE.0.AND.STACK(K).NE.'(') THEN
               PP=PP+1
               POST(PP,1)=STACK(K)
               K=K-1
            END IF
            K=K+1
            STACK(K)=INF(I,1)

C     Else if the element is "-",

         ELSE IF (INF(I,1).EQ.'-') THEN

C     if the next character of the same element is a number or
```

```
C     "[", which  means  that  the  "-"  is  the symbol of the
C     subtraction, place the element in the postfix expression.

             IF ((INF(I,2).GE.'0'.AND.INF(I,2).LE.'9').OR.
     1           INF(I,2).EQ.'[') THEN
                PP=PP+1
                DO J=1,14
                   POST(PP,J)=INF(I,J)
                END DO

C     Else  the  second element  is a letter, also place it on
C     the postfix.

             ELSE IF (INF(I,2).GE.'A'.AND.INF(I,2).LE.'Z') THEN
                PP=PP+1
                DO J=1,14
                   POST(PP,J)=INF(I,J)
                END DO

C     Else, if  the stack is not empty and the topmost element
C     of the  stack  is  not "(", place the topmost element of
C     the  stack  on  the  postfix and push into the stack the
C     next element.

             ELSE
                IF (K.NE.0.AND.STACK(K).NE.'(') THEN
                   PP=PP+1
                   POST(PP,1)=STACK(K)
                   K=K-1
                END IF
                K=K+1
                STACK(K)=INF(I,1)
             END IF

C     If the next element is "*" or "/",

         ELSE IF (INF(I,1).EQ.'*'.OR.INF(I,1).EQ.'/') THEN

C     if the topmost element of the stack is an operator with
C     lower priority ("+","-") or "(", push it into the stack,

            IF (STACK(K).EQ.'+'.OR.STACK(K).EQ.'-'.OR.STACK(K).
     1            EQ.'(') THEN
               K=K+1
               STACK(K)=INF(I,1)

C     else  until the stack is empty or  the "("  delimiter is
C     encountered, pop the stack and place  the  operators  on
C     the postfix expression.

            ELSE
               DO WHILE(K.NE.0.AND.STACK(K).NE.'(')
                  PP=PP+1
                  POST(PP,1)=STACK(K)
                  K=K-1
               END DO
               K=K+1
               STACK(K)=INF(I,1)
            END IF

C     If the next element is of hihger priority ("^"), push it
C     into the stack.

         ELSE IF (INF(I,1).EQ.'^') THEN
            K=K+1
            STACK(K)=INF(I,1)

C     If it is "(", also push it into the stack.

         ELSE IF (INF(I,1).EQ.'(') THEN
            K=K+1
            STACK(K)=INF(I,1)

C     But if it is a ")", pop the stack and place the
C     operators on the postfix until the "(" is encountered.

         ELSE IF (INF(I,1).EQ.')') THEN
            DO WHILE(STACK(K).NE.'(')
               PP=PP+1
               POST(PP,1)=STACK(K)
               K=K-1
            END DO
            K=K-1
         END IF
      END DO

C     At the end if the stack is not empty, pop it until is
C     empty.

      IF (K.NE.0) THEN
         DO WHILE(K.NE.0)
            PP=PP+1
            POST(PP,1)=STACK(K)
            K=K-1
         END DO
      END IF
      RETURN
      END

C**************************************************************
      SUBROUTINE EVAL(LOGOLD,POST,RES,PP,INDEX,NC,JK,NCOL,
      1               X,FUN,ARG,IERR)
C**************************************************************
```

```
C
C     This subroutine is used to evaluate a postfix expression
C     using the push-down stack technique .
C
C     ARGUMENTS:
C     ----------
C     LOGOLD : The  direct  access  file  containing  the data
C              before  any  transformation.  The  same file is
C              used to store  the  data  after transformations
C              have been performed to them.
C     POST   : Two  dimention  array   storing   the   postfix
C              expresison.
C     RES    : The result of the expression.
C     PP     : The  number  of   elements   in   the   postfix
C              expression.
C     INDEX  : Pointer  marking  the  end of mnemonic names in
C              the array (COLMNE) that stores column mnemonics
C              and transformations.
C     NC     : The index of the array COLMNE.
C     JK     :
C     NCOL   : The number of columns.
C     X      :
C     FUN    : Array  storing  the  name  of  the   functions
C              contained in an expression.
C     ARG    : The argument of the function.
C     IERR   : Flag indicating invalid function arguments or
C              division by zero.
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 NC,PP,INDEX,POINT,CN
      REAL*4 STACK(24),X(128),OP1,OP2,OP,RES,R,ARG,AP
      CHARACTER*1 POST(18,14),VAL(14),DUM(14),FUN(5),COL(3),
      1           CONS(14),FUN1(5)

      CHARACTER*5 FUN5
      EQUIVALENCE (FUN5,FUN1(1))

      IERR=0
      L=1

C     For each element of the postfix expression.

      DO I=1,PP

C     If the element is a letter,

         IF (POST(I,1).GE.'A'.AND.POST(I,1).LE.'Z') THEN
            J=1

C     Store until the "[" is encountered the characters of the
C     function name in FUN.
```

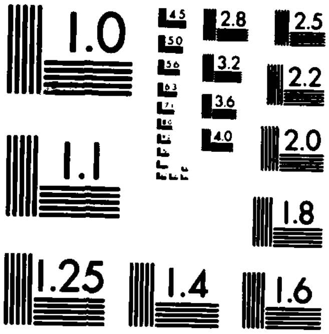
```
            DO WHILE(POST(I,J).NE.'[')
               FUN(J)=POST(I,J)
               J=J+1
            END DO
            JA=J-1
            IJ=0
            DO WHILE (JA.LT.5)
               IJ=IJ+1
               JA=JA+1
               FUN1(IJ)=' '
            END DO
            JA=0
            DO WHILE(IJ.LT.5)
               IJ=IJ+1
               JA=JA+1
               FUN1(IJ)=FUN(JA)
            END DO
            J=J+1
            KA=1

C           Store until "]" is encountered the numbers
C           contained between the brackets and representing
C           column number in array COL.

            DO WHILE(POST(I,J).NE.']')
                COL(KA)=POST(I,J)
                KA=KA+1
                J=J+1
            END DO

C           Decode the characters of COL to obtain the number
C           of column.

            DECODE(3,100,COL) CN
100         FORMAT(I<KA-1>)

C           Using the uncoded column number CN assign the value
C           of  the  corresponding  column   as argument to the
C           function.

            ARG=X(CN)

C           Compute the value of the function.

            CALL COMP(FUN5,ARG,AP,IERR)

C           Push the result into the stack.

            STACK(L)=AP
            L=L+1
```

```
C        Else if the current element of the postfix is "[",

         ELSE IF (POST(I,1).EQ.'[') THEN
            J=2
            KB=1

C           Place the numbers (characters) contained between
C           the brackets in array COL.

            DO WHILE(POST(I,J).NE.']')
               COL(KB)=POST(I,J)
               J=J+1
               KB=KB+1
            END DO

C           Decode  the  content  of COL, obtain the number of
C           colun CN and push into the stack the corresponding
C           column value.

            DECODE(3,200,COL) CN
200         FORMAT(I<KB-1>)
            STACK(L)=X(CN)
            L=L+1

C        Else  if the  current element of the post is a number
C        or ".",

            ELSE IF ((POST(I,1).GE.'0'.AND.POST(I,1).LE.'9').
      1                OR.POST(I,1).EQ.'.') THEN
               J=1
               KC=1

C           store the  characters of this element in the array
C           CONS.

               DO WHILE((POST(I,J).GE.'0'.AND.POST(I,J).LE.
      1                 '9').OR.POST(I,J).EQ.'.')
                  CONS(KC)=POST(I,J)
                  J=J+1
                  KC=KC+1
               END DO
               POINT=0

C           Find the position of ".",

               DO IK=1,KC-1
                  IF (CONS(IK).EQ.'.') THEN
                     POINT=IK
                  END IF
               END DO

C           If no point exists then add a point
```

```
                 IF (POINT.EQ.0) THEN
                    CONS(KC)='.'

C                decode the number and push it into the stack

                    DECODE(KC,500,CONS) OP
500                 FORMAT(F<KC-1>.0)
                    STACK(L)=OP
                    L=L+1

C              else decode  the representation of a real and push
C              it into the stack.

                  ELSE
                     DECODE(KC-1,300,CONS) OP
300                  FORMAT(F<KC-2>.<KC-1-POINT>)
                     STACK(L)=OP
                     L=L+1
                  END IF

C            If  the  element  is  "-"  then  examine  the second
C            character to determine if represents subtraction or a
C            negetive   value  is present. In the first case treat
C            the "-" as the  other  operators  while in the second
C            negate the following the sign value.

               ELSE IF (POST(I,1).EQ.'-') THEN
                  IF (POST(I,2).GE.'0'.AND.POST(I,2).LE.'9') THEN
                     J=2
                     KD=1
                     DO WHILE((POST(I,J).GE.'0'.AND.POST(I,J).LE.
     1                  '9').OR.POST(I,J).EQ.'.')
                        CONS(KD)=POST(I,J)
                        J=J+1
                        KD=KD+1
                     END DO
                     POINT=0
                     DO IK=2,KD-1
                        IF (CONS(IK).EQ.'.') THEN
                           POINT=IK
                        END IF
                     END DO
                     IF (POINT.EQ.0) THEN
                        CONS(KC)='.'
                        DECODE(KD,600,CONS) OP
600                     FORMAT(F<KD-1>.0)
                        STACK(L)=OP
                        L=L+1
                     ELSE
                        DECODE(KD-1,400,CONS) OP
400                     FORMAT(F<KD-2>.<KD-1-POINT>)
```

AD-A151 359 INTERACTIVE DATA ANALYSIS DEVELOPMENT OF AN INTERACTIVE 3/3
DATA MANIPULATION SYSTEM(U) NAVAL POSTGRADUATE SCHOOL
MONTEREY CA N G TOTOS JUN 84
UNCLASSIFIED F/G 9/2 NL

END
FILMED
DTIC


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
                     STACK(L)=-OP
                     L=L+1
                  END IF
               ELSE IF (POST(I,2).EQ.'[') THEN
                  J=3
                  KE=1
                  DO WHILE(POST(I,J).NE.']')
                     COL(KE)=POST(I,J)
                     J=J+1
                     KE=KE+1
                  END DO
                  DECODE(3,800,COL) CN
800               FORMAT(I<KE-1>)
                  STACK(L)=-X(CN)
                  L=L+1
               ELSE IF (POST(I,2).GE.'A'.AND.POST(I,2).LE.'Z')
      1                THEN
                  J=2
                  DO WHILE(POST(I,J).NE.'[')
                     FUN(J)=POST(I,J)
                     J=J+1
                  END DO
                  JA=J-1
                  IJ=0
                  DO WHILE (JA.LT.5)
                     IJ=IJ+1
                     JA=JA+1
                     FUN1(IJ)=' '
                  END DO
                  JA=0
                  DO WHILE(IJ.LT.5)
                     IJ=IJ+1
                     JA=JA+1
                     FUN1(IJ)=FUN(JA)
                  END DO
                  J=J+1
                  KF=1
                  DO WHILE(POST(I,J).NE.']')
                     COL(KF)=POST(I,J)
                     KF=KF+1
                     J=J+1
                  END DO
                  DECODE(3,700,COL) CN
 700              FORMAT(I<KF-1>)
                  ARG=X(CN)
                  CALL COMP(FUN5,ARG,AP,IERR)
                  STACK(L)=-AP
                  L=L+1
               ELSE
                  L=L-1
                  OP2=STACK(L)
                  L=L-1
                  OP1=STACK(L)
                  R=OP1-OP2
                  STACK(L)=R
                  L=L+1
               END IF

C        If the element is an operator, pop the two topmost
C        elements of the stack, apply the operator to them
C        and push the result into the stack.

            ELSE IF (POST(I,1).EQ.'+'.OR.POST(I,1).EQ.'*'.OR.
      1           POST(I,1).EQ.'/'.OR.POST(I,1).EQ.'^') THEN
               L=L-1
               OP2=STACK(L)
               L=L-1
               OP1=STACK(L)
            IF (POST(I,1).EQ.'+') THEN
               R=OP1+OP2
            ELSE IF (POST(I,1).EQ.'*') THEN
               R=OP1*OP2
            ELSE IF (POST(I,1).EQ.'/') THEN
               IF (OP2.NE.0) THEN
                  R=OP1/OP2
               ELSE
                  IERR=2
                  RETURN
               END IF
            ELSE IF (POST(I,1).EQ.'^') THEN
               R=OP1**OP2
            END IF
            STACK(L)=R
            L=L+1
         END IF
      END DO
      L=L-1
      RES=STACK(L)
      RETURN
      END


C*************************************************************
      SUBROUTINE DISDAT(LEC,IMP,NCOL,NROW,IOPEN,ICLOSE)
C*************************************************************
C
C     This  subroutine  is  used for the display of all or any
C     user assigned part of the data file.
C
C  ARGUMENTS :
C
C  LEC, IMP : Input/Output logical numbers for terminal.
C  NCOL     : Number of columns.
C  NROW     : Number of rows.
C  IOPEN    : If equal to 1, assign and open the direct-access
C             data file and output display.
C  ICLOSE   : If equal to 1, close the direct-access data file.
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)
      BYTE STRING(72),FMT1(72),FMT2(72)
      CHARACTER*7 MODCOL,MODROW
      CHARACTER*72 FMTC1,FMTC2
      CHARACTER*45 DEVDIR,NAME*10

      REAL*4 X(128)
      DIMENSION NOCOL(256),NOROW(2048)

      EQUIVALENCE (FMTC1,FMT1(1)),(FMTC2,FMT2(1))

      DATA LOGOLD,IMP2/2,7/DEVDIR,NAME/'DUA0 :',' '/
      DATA MODCOL/'columns'/MODROW/'rows'/
      DATA FMTC2/'(I6,  F14. 6)'/

C     Declare LOGOLD unit data file.

      IF (IOPEN.EQ.1) THEN
         NBYTES=NCOL*4
         WRITE(IMP,100)
         CALL FICH('069',LOGOLD,1,DEVDIR,NAME,NROW,NBYTES,1,
     1             'DIRECT',LEC,IMP)

C        Display output unit attribute.

         CALL ENSORT(LEC,IMP,IMP2)
      END IF
```

```
C     Display conditions for columns.

      CALL ANADIS(LEC,IMP,NCOL*2,NCOL,NOCOL,NBCOL,MODCOL)
      IF(NBCOL.EQ.0) RETURN

C     Display conditions for rows.

      CALL ANADIS(LEC,IMP,NROW  ,NROW,NOROW,NBROW,MODROW)
      IF(NBROW.EQ.0) RETURN

      NOUT=0

C     Change FORMATs for output.
      CALL FORM(LEC,IMP,FMTC1,FMT2,N,I1)

      DO WHILE (NOUT.LT.NBCOL)
         IBEG=NOUT+1
         IEND=IBEG+N-1
         IF(IEND.GT.NBCOL) THEN
            IEND=NBCOL
         END IF
C           For columns.
         M=IEND-IBEG+1
         ENCODE(2,200,FMT1(07)) M
C           For underlining.
         M=5+M*I1
         ENCODE(2,200,FMT1(25)) M
         WRITE(IMP2,FMT1) (NOCOL(J),J=IBEG,IEND)
         I=1
         DO WHILE(I.LE.NBROW)
            READ(LOGOLD'NOROW(I))(X(J),J=1,NCOL)
            WRITE(IMP2,FMT2) NOROW(I),(X(NOCOL(J)),J=IBEG,IEND)
            I=I+1
         END DO
         NOUT=NOUT+N
      END DO

      IF (ICLOSE.EQ.1) CLOSE (LOGOLD)
      RETURN

C     Formats.

100   FORMAT(//4X,'Name of the existing data file :')
200     FORMAT(I2)

      END

C**************************************************************
      SUBROUTINE FORM(LEC,IMP,FMTC1,FMT2,N,I1)
C**************************************************************
C
```

```
C     Permits  the  user  to  determine the format of the data
Cthat will be displayed.
C
      IMPLICIT INTEGER*2 (I-N)

      BYTE FMT(72),FMT2(72),STRING(72)
      CHARACTER*72 FMTC,FMTC1

      EQUIVALENCE (FMTC,FMT(1))

      DATA FMTC /'(//7X,  (  X,I3,  X)/1X,75(\-\)/)'/

C     Because compiler does no admit quotes inside the chain,
C     they are changed by '\' and substituted after.

      FMT(28)=039
      FMT(30)=039

      WRITE(IMP,100)
100   FORMAT(/4X,'The normal format is      : nnnnnnn.nnnnnn'
      1       //'$' 3X,
      2'To modify it assign your format (ex. nnn.nn) <CR> :')

C     Read the user assigned FORMAT.

      READ(LEC,200) STRING
200   FORMAT(72A1)
      IPOINT=0
      I=1
      DO WHILE(STRING(I).NE.' ')
         IF (STRING(I).EQ.'.') IPOINT=I
            I=I+1
      END DO

C     Data format.

      IF (I.EQ.1) THEN
C        For FORMAT(5F14.6) for Data.
         I1=14
         IPOINT=8
      ELSE
C        For general format for Data.
         IF (IPOINT.EQ.0) THEN
            IPOINT=I
            I1=I
         ELSE
            I1=I-1
         END IF
      END IF

      ENCODE(2,300,FMT2(08)) I1
300   FORMAT(I2)
```

```
      N=70/I1
      ENCODE(2,300,FMT2(05)) N
      FMT2(07)='F'
      FMT2(10)='.'
      FMT2(13)=')'

      I=I1-IPOINT
      ENCODE(2,300,FMT2(11)) I
      DO I=14,72
         FMT2(I)=' '
      END DO

C        Format for Title.

      I3=(I1-3)/2
      I4=I1-(I3+3)

      IF (I3.NE.0) THEN
         ENCODE(2,300,FMT(10)) I3
      ELSE
         FMT(12)=' '
         FMT(13)=' '
      END IF
      IF (I4.NE.0) THEN
         ENCODE(2,300,FMT(17)) I4
      ELSE
         FMT(16)=' '
         FMT(19)=' '
      END IF

      FMTC1=FMTC

      RETURN

      END


C***************************************************************
      SUBROUTINE SEEDAT(LEC,IMP,NCOL,NROW,IOPEN,ICLOSE)
C***************************************************************
C
C  LEC, IMP : Input/Output logical numbers for terminal.
C  NCOL     : Number of columns.
C  NROW     : Number of rows.
C  IOPEN    : If equal to 1, assign and open the direct-access
C             data file and output display.
C  ICLOSE   : If equal to 1, close the direct-access data file.
C
C***************************************************************

      IMPLICIT INTEGER*2 (I-N)
      CHARACTER*45 DEVDIR,NAME*10
      CHARACTER*7 MODCOL,MODROW

      BYTE STRING (80)
      REAL*4 X(128),Y(32)
      DIMENSION NOCOL(256),NOROW(2048)

      DATA LOGOLD,IMP2/2,7/DEVDIR,NAME/'DUA0 :',' '/
      1GOLD/12345.678/IFOU/0/
      DATA MODCOL/'columns'/MODROW/'rows'/

C     Declare LOGOLD unit data file.

      IF (IOPEN.EQ.1) THEN
         NBYTES=NCOL*4
         WRITE(IMP,100)
         CALL FICH('069',LOGOLD,1,DEVDIR,NAME,NROW,NBYTES,1,
      1             'DIRECT',LEC,IMP)

C        Display output unit attribute.

         CALL ENSORT(LEC,IMP,IMP2)
      END IF

C     Display conditions for columns.

      CALL ANADIS(LEC,IMP,NCOL*2,NCOL,NOCOL,NBCOL,MODCOL)
      IF(NBCOL.EQ.0) RETURN

C     Display conditions for rows.

         CALL ANADIS(LEC,IMP,NROW  ,NROW,NOROW,NBROW,MODROW)
            IF(NBROW.EQ.0) RETURN

               NB=0

C     Flags  for  evental  continuation  lines  if  last
C     character=','.
               IWRIT=1
               ICONT=1
C     ---------------------
      DO WHILE (ICONT.EQ.1)
C     ---------------------
      ICONT=0

C     Input string
      IF (IWRIT.NE.0) WRITE(IMP,600)
         READ (LEC,700) STRING
         IWRIT=0

C     Analysis of the string.
         NCAR=80
         DO WHILE (STRING(NCAR).EQ.' ')
            NCAR=NCAR-1
         END DO
         IF (STRING(NCAR).EQ.',') THEN
            ICONT=1
            STRING(NCAR)=' '
         ELSE
            NCAR=NCAR+1
         END IF
         IGOLD=0
         IBEG =1

         I=0
         DO WHILE (I.LT.NCAR)
            I=I+1
C           'GOLD' can be any letter.
            IF ((STRING(I).GE.'A'.AND.STRING(I).LE.'Z').OR.
      1         (STRING(I).GE.'a'.AND.STRING(I).LE.'z')) THEN
               IF (IGOLD.EQ.0) THEN
                  WRITE(IMP,200) GOLD
                  READ (LEC,300) GOLD2
                  IF (GOLD2.NE.0.) GOLD=GOLD2
               END IF
               NB=NB+1
               Y(NB)=GOLD

               DO WHILE (STRING(I).NE.','.AND.I.LT.NCAR.AND.
      1                  STRING(I).NE.' ')
                  I=I+1
               END DO
            ELSE

               IPOINT=0
               DO WHILE (STRING(I).NE.','.AND.I.LT.NCAR.AND.
      1                  STRING(I).NE.' ')
                  IF(STRING(I).EQ.'.') IPOINT=I
                     I=I+1
               END DO
               IEND=I-1
C              Number : nnn
               IF (IPOINT.EQ.0) THEN
                  STRING(I)='.'
                  IPOINT   = I
                  IEND     = I
               END IF
               I1=IEND-IBEG+1
               I2=I1-IPOINT+IBEG-1
               NB=NB+1
               DECODE(I1,400,STRING(IBEG)) Y(NB)
            END IF
            IBEG=I+1
         END DO
```

```
C      ------
       END DO
C      ------

       K=0
       DO WHILE(K.LT.NBROW)
          K=K+1
          READ(LOGOLD'NOROW(K))(X(L),L=1,NCOL)
          DO L=1,NBCOL
             DO N=1,NB
                IF(X(NOCOL(L)).EQ.Y(N)) THEN
                   INDEX=NCOL*(NOROW(K)-1)+NOCOL(L)
                   IFOU=IFOU+1
                   WRITE(IMP2,500)IFOU,INDEX,NOCOL(L),NOROW(K),
       1                Y(N)
                END IF
             END DO
          END DO
       END DO

       IF (ICLOSE.EQ.1) CLOSE (LOGOLD)
       RETURN

C      Formats.

100    FORMAT(//4X,'Name of the existing data file :')
200    FORMAT(/4X,'Missing value "gold" number is : ',F9.3//
       1'$',3X,'Change or RETURN : ')
300    FORMAT(F18.8)
400    FORMAT(F<I1>.<I2>)
500    FORMAT(//I6,3X,'Index :',I6,' ,Column : ',I4,' ,Line :',
       1I6,' -- Value :',F14.6)
600    FORMAT(/'$',3X,'Data to be retrieved (real or "gold") : ')
700    FORMAT(80A1)

       END

C*************************************************************
       SUBROUTINE ANADIS(LEC,IMP,MAXDIM,N,NOCORO,NELEM,MODE)
C*************************************************************

C   This subroutine ANADIS analyses the assignement for columns
C   or rows through a string given by the user in four modes.

C      Examples :                          Comments :

C   1).    * or all or ALL        : take all columns or rows.
C   2).    3:12                   : take  all  columns   or  rows
C                                   between  the  lower  and upper
C                                   boundaries.
```

```
C  3).   1,3,8,2,1              : take  successively  the  named
C                                 columns  or rows, even repeted
C                                 in the given order.
C  4).   4                      : take a single column or row.

C  5).   1,3,12:15,20:27,31     : mixing modes 2). and 3).


C     ARGUMENTS :

C  LEC and IMP : Logical numbers for input/output keyboard.
C  MAXDIM      : Maximum dimension for NOCORO().
C  N           : Maximum number   of   columns   (NCOL)   or   rows
C                (NROW).
C  NCORO       : Nos. of the displayed columns or rows.
C  NELEM       : Total  number of elements to take (columns or
C                rows).
C  MODE        : Represents  a chain of characters : 'columns'
C                or 'rows   '.

C*************************************************************

      IMPLICIT INTEGER*2 (I-N)

      DIMENSION NOCORO(MAXDIM)

      CHARACTER*7 MODE

      BYTE STRING(80)

C     Initialization.

      NELEM=1

C     Flags  for  evental  continuation  lines  if  last
C     character=','.
      IWRIT=1
      ICONT=1
C     ---------------------
      DO WHILE (ICONT.EQ.1)
C     ---------------------
      ICONT=0

      NBEG=0
      NCAR=0
      IBEG=1
      ISER=0

C     Input string.
      IF (IWRIT.NE.0) WRITE(IMP,100) MODE
         READ (LEC,200) STRING
         IWRIT=0
```

```
C     If empty string.
C     ----------------
         IF (STRING(1).EQ.' ') THEN
            NELEM=0
            RETURN
         END IF

C     If all colums or lines are displayed. (* or any letters.
C     ------------------------------------------------------------
         IF (STRING(1).EQ.'*'.OR.
      1  (STRING(1).GE.'A'.AND.STRING(1).LE.'Z').OR.
      2  (STRING(1).GE.'a'.AND.STRING(1).LE.'z')) THEN
            DO I=1,N
               NOCORO(I)=I
            END DO
            NELEM=N
            RETURN
         END IF

C     String analysis.

         I=1
         DO WHILE (STRING(I).NE.' ')

C        Control of string validity.

            IF((STRING(I).GE.'0'.AND.STRING(I).LE.'9').OR.
      1     (STRING(I).EQ.':'.OR .STRING(I).EQ.',')) THEN
            ELSE
               NELEM=0
               WRITE(IMP,400)
               RETURN
            END IF

C        If between lower and upper boundaries (ex: 3:8).
C        ----------------------------------------------------
C        or/and if list of numbers (ex: 2,5,8,3).
C        ----------------------------------------------
            IF (STRING(I).EQ.','.OR.STRING(I).EQ.':') THEN
               IEND=I-1
               DECODE(NCAR,300,STRING(IBEG)) NOCORO(NELEM)
               IF (NOCORO(NELEM).GT.N) THEN
                  WRITE(IMP,500) N
                  NELEM=0
                  RETURN
               END IF
               IF (STRING(I).EQ.':') THEN
                  ISER=1
               ELSE
               IF (STRING(I).EQ.',') THEN
                  IF (ISER.EQ.1) THEN
```

```
                     NBEG=NOCORO(NELEM-1)+1
                     NEND=NOCORO(NELEM)
                     NELEM=NELEM-1
                     DO J=NBEG,NEND
                        NELEM=NELEM+1
                        NOCORO(NELEM)=J
                     END DO
                     ISER=0
                  END IF
               END IF
            END IF
               NCAR=0
               NELEM=NELEM+1
               IBEG=I   +1
         ELSE
            NCAR=NCAR+1
         END IF
         I=I+1
      END DO

C     If the number of delimiters is zero
C     Just one column or row is displayed (ex : 4).
C     ---------------------------------------------
      IF (NELEM.EQ.1) THEN
         I=I-1
         DECODE(   I,300,STRING(IBEG)) NOCORO(1)
         IF (NOCORO(1).GT.N) THEN
         WRITE(IMP,500) N
         NELEM=0
         RETURN
      ELSE
      END IF
   ELSE

C        Analyse string residual.
C        ------------------------
      DECODE(NCAR,300,STRING(IBEG)) NOCORO(NELEM)
      IF (NOCORO(NELEM).GT.N) THEN
         WRITE(IMP,500) N
         NELEM=0
         RETURN
      ELSE
C           For continuation line, if last character is ','.
         IF (NOCORO(NELEM).EQ.0) THEN
            ICONT=1
         ELSE
            IF (ISER.EQ.1) THEN
               NBEG=NOCORO(NELEM-1)+1
               NEND=NOCORO(NELEM)
               NELEM=NELEM-1
               DO J=NBEG,NEND
                  NELEM=NELEM+1
                  NOCORO(NELEM)=J
               END DO
            END IF
         END IF
      END IF
   END IF

C     ------
      END DO
C     ------

      RETURN

C     Formats.

100   FORMAT(/'$',3X,'Assign no(s). of ',A7,' : ')
200   FORMAT(80A1)
300   FORMAT(I<I>)
400   FORMAT(4X//' INVALID INPUT !!
      1 (must be numerical AND/OR  "," or ":")')
500   FORMAT(4X//' NUMBER EXCEEDS',I6,' !!')

      END


C*************************************************************
      SUBROUTINE FICH(NNN,NLOGIC,IAUT,DEVDIR,NAME,NENR,NBYTES,
      1               ISTAT,ACCESS,LEC,IMP)
C*************************************************************
C
C  This  Subroutine  permits to Open, within a FORTRAN Program
C  at  Run  Time,   specific   Files   named   'FORnnn.DAT'for
C  calculation   and   deletion   or   general   Files   named
C  'file name.ext'.  These  Files  may be used for Unformatted
C  Direct Fixed Access or Formatted Sequential Variable Access
C
C     ARGUMENTS :
C
C  'NNN'  Can  be  3  {0,9}  numerical  equivalent  characters
C  otherwise, a Key (ex: '012', '326', a key : 'G13', 'TT2')
C  If NNN=' ', the File will be named as the content of NAME.
C  If  not,   the  File  will  be  named  as 'FORnnn.DAT', ex:
C  'FOR012.DAT'.
C
C  NLOGIC : Logical Unit number.
C
C  IAUT   : If IAUT.NE.0, Input the 3 characters Key, the File
C           is named as 'FORkey.DAT', or Input NAME. If IAUT=0,
C           the File is named as 'FORNNN.DAT', or 'name'
C
C  DEVDIR : If ' ',  ask for Device and Directory-Subdirectory
C           names. If 'DUA0:' or 'DUA0:[DIRECTORY.SUBdir]' for
```

```
C             ex.,no change.
C  NAME     : If NNN= ' ', asks by Keyboard the name of the File,
C             when IAUT,NE.0. NAME contents up to 10 characters.
C
C  NENR     : Maximum records number.
C
C  NBYTES   : Number of BYTES per Record, If UNFORMATED Records,
C             transformation in full 32 bits  Words, MUST  be  a
C             multiple of 4 BYTES (If not, BELL rings, a message
C             and a complementation occur)
C
C  ACCESS   : If 'DIRECT' :  DIRECT, UNFORMATTED, FIXED  records,
C             otherwise : SEQUENTIAL,FORMATTED,VARIABLE records.
C
C  ISTAT    : If ISTA=0 : 'NEW', otherwise : 'OLD'
C
C  INDX     : Associated Variable for Unformatted Direct Access,
C             not  a  dummy  argument,  it  will  be auto-post-
C             Incremented,COMMON /INDEX/INDX must be present in
C             the calling program.  Be CAREFULL, INDX is common
C             for all the opened files.If no COMMON /INDEX/INDX
C             or Using another integer variable,  no auto-post-
C             Incrementation, when READ or WRITE.
C
C**************************************************************
C
C  Examples :
C
C     NNN     IAUT    NAME       QUESTION     RESULT
C
C     'nnn'   0       ' '          None       FORnnn.DAT
C     'key'   0       ' '          None       FORkey.DAT
C     ' '     0       'name.dat'   None       name.ext
C     ' '     0       ' '          NAME ?     NAME
C
C     'nnn'   1       ' '          KEY  ?     FORkey.DAT
C     ' '     1       ' '          NAME ?     NAME
C
C  Remark : Priority  is  always  given  to  'nnn' or 'key' if
C           present.
C
C**************************************************************
C
      IMPLICIT INTEGER*2 (I-N)
C
      CHARACTER*3 FOR,NNN,KEY,STAT,EXT*4,DEV*5,NAME*10,
      1ACCESS*6,DIR*40,DEVDIR*45,DEDINA*55,BELL*1
C
      COMMON /INDEX/INDX
C
      DATA FOR,EXT/'FOR','.DAT'/BELL/007/
C
```

```
      IF(DEVDIR.EQ.' ')THEN
C
      WRITE(IMP,100)NLOGIC
100   FORMAT('$Device for File',I3,' (If Default : <CR> : ')
      READ(LEC,200)DEV
200   FORMAT(A5)
      WRITE(IMP,300)NLOGIC
300   FORMAT('$Directory for File',I3,' (If Default : <CR>) :')
      READ(LEC,400)DIR
400   FORMAT(A40)
      DEVDIR=DEV//DIR
C
      ELSE
C
      IF(IAUT.EQ.0)THEN
C
         IF(NNN.EQ.' ')THEN
C
            IF(NAME.EQ.' ')THEN
C

               WRITE(IMP,500)NLOGIC
500            FORMAT('$Name for File',I3,' : ')
               READ(LEC,600)NAME
600            FORMAT(A10)
C
            ELSE
C
            END IF
C
            DEDINA=DEVDIR//NAME
            GO TO 1
C
         ELSE
C
         END IF
C
         DEDINA=DEVDIR//FOR//NNN//EXT
         GO TO 1
C
      ELSE
C
         IF(NNN.EQ.' ')THEN
C
            WRITE(IMP,500)NLOGIC
            READ(LEC,600)NAME
            DEDINA=DEVDIR//NAME
            GO TO 1
C
         ELSE
C
            WRITE(IMP,700)NLOGIC,NNN
```

```
700     FORMAT(/'$',3X,
    1       'Key (xxx) For File FORxxx.DAT number',I3,
    2       ', (nnn=',A3,') : ')
        READ(LEC,800)KEY
800     FORMAT(A3)
        DEDINA=DEVDIR//FOR//KEY//EXT
        GO TO 1
C
      END IF
C
    END IF
C
  END IF
C
C Assign File to a logical Unit.
C
1   IF(ISTAT.EQ.0) THEN
C
      STAT='NEW'
C
    ELSE
C
      STAT='OLD'
C
    END IF
C
C Control the Access Mode :
C

    IF(ACCESS.EQ.'DIRECT')THEN
C
      NMOT32=INT(NBYTES/4)
C
      IF(MOD(NBYTES,4).NE.0)THEN
C
C   If a message wanted.
C       WRITE(IMP,900)BELL
900     FORMAT(/1X,A1,' ERROR IN RECORDSIZE ARGUMENT, '
    1    'MUST BE A MULTIPLE OF 4'/)
        NMOT32=NMOT32+1
        NBYTES=NMOT32*4
C
      ELSE
C
      END IF
C
      OPEN(unit=NLOGIC,name=DEDINA,status=STAT,
    1access='DIRECT',recordsize=NMOT32,recordtype='FIXED',
    2initialsize=NENR,form='UNFORMATTED',
    3associatevariable=INDX)
C
    ELSE
```

```
C
         OPEN(unit=NLOGIC,name=DEDINA,status=STAT,
      1access='SEQUENTIAL',recordsize=NBYTES,
      2recordtype='VARIABLE',initialsize=NENR,form='FORMATTED')
C
      END IF
C
      RETURN
C
      END


C*************************************************************
      SUBROUTINE EXSHEL(X,NO,N,IND)
C*************************************************************
C
C  Subroutine  for internal and address calculation sort using
C  the Shell's  method  to rank elements X in increasing order
C  and modified for address calculations.
C
C  ARGUMENTS :
C
C  X     : Elements to sort.
C  NO    : Address calculation for the rank.
C  N     : Number of elements to sort.
C  IND   : If >0 elements are  ranged in increasing order, if
C          not, in decreasing order.
C
C  Reference : Shell Donald L. (1959) :  "A High-Speed Sorting
C              Procedure", Comm  of  the  ACM,  vol.2,   July,
C              p.30-32.
C
C*************************************************************
C
      IMPLICIT INTEGER*2 (I-N)
C
C  Elements X and TEMP can be declared as :
C  INTEGER *2 or *4, REAL *4, *8, *16 ,CHARACTER
C
      REAL*4 X(N),TEMP
      DIMENSION NO(N)
C
1     NO(I)=I
C
      M=N
      M=M/2
C
      DO WHILE(M.GT.0)
         DO 5 K=1,M
            NMM=N-M
            DO 4 I=K,NMM,M
               J=I
               IPM=I+M
               TEMP   = X(IPM)
               NOTEMP=NO(IPM)
2              IF(IND.GE.0)THEN
                  IF(TEMP.GT.X(J))GO TO 3
               ELSE
                  IF(TEMP.LE.X(J))GO TO 3
               END IF
               JPM=J+M
               X (J+M)=X (J)
               NO(JPM)=NO(J)
               J=J-M
               IF(J.GE.1)GO TO 2
3                 JPM=J+M
                  X( JPM)=TEMP
                  NO(JPM)=NOTEMP
4                 CONTINUE
5                 CONTINUE
                  M=M/2
      END DO
C
      RETURN
C
      END


C**************************************************************
      SUBROUTINE EXSH1(X,NO,N,IND)
C**************************************************************

      IMPLICIT INTEGER*2 (I-N)
      INTEGER*2 X(N),TEMP
      DIMENSION NO(N)

C
1     NO(I)=I
C
      M=N
      M=M/2
C
      DO WHILE(M.GT.0)
         DO 5 K=1,M
            NMM=N-M
            DO 4 I=K,NMM,M
               J=I
               IPM=I+M
               TEMP   = X(IPM)
               NOTEMP=NO(IPM)
2              IF(IND.GE.0)THEN
                  IF(TEMP.GT.X(J))GO TO 3
               ELSE
                  IF(TEMP.LE.X(J))GO TO 3
               END IF
               JPM=J+M
               X (J+M)=X (J)
               NO(JPM)=NO(J)
               J=J-M
               IF(J.GE.1)GO TO 2
3                 JPM=J+M
                  X( JPM)=TEMP
                  NO(JPM)=NOTEMP
4                 CONTINUE
5                 CONTINUE
                  M=M/2
      END DO
C
      RETURN
C
      END

C*************************************************************
      SUBROUTINE ENSORT(NIN,NOUT,NPRINT)
C*************************************************************
C
C  This subroutine permits Input/Output modifications or
C  assignments for physical Input or/and Output, or/and Print
C  for Devices or Files within a FORTRAN Program or Subroutine.
C
C  ARGUMENTS :
C
C             NIN and NOUT are the respective Logical FORTRAN
C             numbers for Input and Output (for Terminal
C             in general) and NPRINT for Print (for Printer
C             in general, but may be used for Terminal Output
C             or File output).
C
C  SUBROUTINE called :
C
C             INOUT(NLOGIC,I,MODE)
C
      IMPLICIT INTEGER*2 (I-N)
C
C

C  Call for Input.
      CALL INOUT(NIN,1,1)
C  Call for Output.
      CALL INOUT(NOUT,2,0)
C  Call for Output-Print.
      CALL INOUT(NPRINT,3,0)
C
      RETURN
      END
```

```
C***************************************************************
      SUBROUTINE INOUT(NLOGIC,I,MODE)
C***************************************************************
C
C  This  Subroutine  permits  to  OPEN Devices or/and Files to
C  associate within any  FORTRAN  Program or Subroutine at RUN
C  time.
C
C  ARGUMENTS :
C
C     NLOGIC : FORTRAN logical number associate to the unit to
C              OPEN, (NIN, NOUT,NPRINT in the calling module).
C     I      : Index (1): Input, (2): OUTput (for  terminal in
C              general), (3) : Print-Output  (for  Printer  or
C              File in general).
C     MODE   : (0): 'NEW', (1): 'OLD', otherwise: 'unknown'.
C
C
C  SUBROUTINE Called : None.
C
C***************************************************************

      IMPLICIT INTEGER*2 (I-N)
C
      CHARACTER*40 KEY,NUL
C  KEY is Accept via the Keyboard at RUN time.
      CHARACTER*28 DEV(3)
C  DEV() is Typed on the Terminal in fonction of I.
      CHARACTER* 3 STA(3)
C  STA() is referenced as 'NEW','OLD' or 'unknown' in fonction
C     of MODE.
      DIMENSION NSYS(3),INCOM(3)
C  NSYS()  represents  the  general  used  logical numbers for
C     Input,  Output  and  Print;  INCOM(),  the  relative
C     incompatibility for the NSYS()'s.
C
      DATA NUL/' '/
      DATA DEV/' Input Device (or File)   : ','Output Device
      1 (or File)   : ',' Print Device (or File)   : '/
      DATA STA/'NEW','OLD','   '/
      DATA NSYS/5,6,6/INCOM/6,5,5/
C  Incompatibilities Input with Output in general.
C
      IF(NLOGIC.EQ.NSYS(I))RETURN
      IF(NLOGIC.GT.0.AND.NLOGIC.NE.INCOM(I))GO TO 1
      NLOGIC=NSYS(I)
      RETURN
C  On System Terminal.
1     TYPE 100,DEV(I)
100   FORMAT(/'$ ',A28)
      ACCEPT 200,KEY
```

```
200    FORMAT(A40)
       IF(KEY.EQ.NUL)RETURN
C  If Change.
          CLOSE(NLOGIC)
          IF(MODE.LT.0.OR.MODE.GT.2)MODE=2
             OPEN(unit=NLOGIC,file=KEY,status=STA(MODE+1))
       RETURN
       END

C**************************************************************
       SUBROUTINE REJECT(LEC,IMP,NROW,IJ,IND,I,IBIT,IOPEN,
       1                  ICLOSE)
C**************************************************************
C
C  Subroutine  for  rejection of rows ("logical suppression");
C  these  rows  are  not  physically  suppressed  but are just
C  elliminated  from  future  calculus  if  the  Ith.  row  is
C  associated to the binary value  zero  store in the assigned
C  rejected values support "binary" sequential  file  given by
C  the user.
C
C  ARGUMENTS :

C  LEC, IMP : Input/Output logical numbers for Terminal.
C  NROW     : Total  number  of  rows  for  the  Data   matrix
C             NCOL*NROW.
C  IJ       : Integer  array  that  contains compressed binary
C             information for rejection.
C  IND      : If IND=0 : to examine bits level.
C            If IND=1 : to load bits level.
C  I        : Ith examination             if IND=0.
C  IBIT     : Bit value 0 or 1 to return if IND=0.
C  IOPEN    : If  equal  to  1,  to  open a sequential file to
C             store [0,1].
C  ICLOSE   : If equal to 1, to close the sequential file.
C
C  SUBROUTINES called : COMP01 for binary compression,
C                       BIT01  for bits manipulation.
C
C**************************************************************

       IMPLICIT INTEGER*2 (I-N)

C  Compression  :  dimension  of  IJ()  = 2048/NBITW to read a
C  single record.

       INTEGER*2 NOROW(2048),NO(2048),IJ(128)

       CHARACTER*45 DEVDIR,NAME*10
       CHARACTER*7 MODROW
```

```
      DATA DEVDIR,NAME/'DUA0 :',' '/
      DATA MODROW/'rows'/LOGSUP/8/NCOLS,J/2*1/NBITW/16/
      MAXDIM=(NROW*NCOLS-1)/NBITW-1

      IF (IND.EQ.1) THEN

         IF (IOPEN.EQ.1) THEN
C           Assign  rejected  rows support "binary" sequential
C           file according to the WRITE FORMAT 300.
            WRITE(IMP,100)
            CALL FICH ('076',LOGSUP,1,DEVDIR,NAME,1,1024,0,
      1               'SEQUENTIAL',LEC,IMP)
         END IF
C        Select rejected rows ("logical suppression").
         WRITE(IMP,200)
         CALL ANADIS(LEC,IMP,NROW,NROW,NOROW,NBROW,MODROW)

C        Initialization : Set all values to 1.
         CALL COMP01(IJ,MAXDIM,NCOLS,NROW,I,J,1,NBITW,1,1)

C        Change selected rejectable values to 0.
         DO L=1,NBROW
            CALL COMP01(IJ,MAXDIM,NCOLS,NROW,NOROW(L),J,0,
      1                 NBITW,2,1)
         END DO
         WRITE(LOGSUP,300)IJ
      ELSE
         IF (IOPEN.EQ.1) THEN
C           Assign  rejected  rows support "binary" sequential
C           file according to the WRITE FORMAT.
            WRITE(IMP,100)
            CALL FICH('076',LOGSUP,1,DEVDIR,NAME,1,1024,1,
      1               'SEQUENTIAL',LEC,IMP)
            READ(LOGSUP'300)IJ
         END IF
         CALL COMP01(IJ,MAXDIM,NCOLS,NROW,I,J,IBIT,NBITW,2,0)
      END IF

      IF (ICLOSE.EQ.1) CLOSE(LOGSUP)


      RETURN

100   FORMAT(/4X,'Assign sequential file for rejected values'
      1/4X,42('-'))
200   FORMAT(/'$',3X,'Assign rejected rows for calculus : ')
300   FORMAT(16(8I8/))

      END

C*****************************************************************
      SUBROUTINE COMP01(IJ,MAXDIM,NCOL,NROW,I,J,IBIT,NBITW,
      1                 INI,IND)
C*************************************************************
C
C  This subroutine is used to get or set the binary value 0 or
C  1 in a  compressed  binary matrix NROW*NCOL for any I's and
C  J's.
C
C  ARGUMENTS :
C
C  IJ     : Integer   one   machine  word   by   element  that
C           represents the stored binary values (0,1).
C  MAXDIM : Dimension of the compressed table IJ().
C  NCOL   : Maximum value for the J's, (number of columns).
C  NROW   : Maximum value for the I's, (number of rows).
C  I, J   : Actual value for I and J.
C  IBIT   : Set of return the binary value.
C  NBITW  : Number of bits per word (16, 32, 36).
C  INI    : If 0 or 1, initialize to one of these values,
C           otherwise, no action.
C  IND    : If IND=0 : examine bits level and return the value
C           IBIT,
C  If IND=1 : Set bit level to the current IBIT.
C
C  SUBROUTINE called : BIT01 for bits manipulations.
C
C*************************************************************

      IMPLICIT INTEGER*2 (I-N)

      INTEGER*2 IJ(MAXDIM),BITSET(36),BIT(36)

C     Initialization to 0 or 1.
C     ------------------------
      INIT=0
      IF (INI.EQ.0.OR.INI.EQ.1) THEN
         IF (INI.EQ.1) INIT=-1
            DO K=1,MAXDIM
               IJ(K)=INIT
            END DO

         ELSE

C  Process to examine or set bits for the actual I's and J's.
C  ----------------------------------------------------------

C  PHASE  I : Calculate bit no. : NOBIT, index  NOW of the one
C  ---------- word array  IJ()  and position NOPOS in the word
C             (1 to NBITW).

            NOBIT=J+NCOL*(I-1)
            NOW  =(NOBIT-1)/NBITW+1
            NOPOS=NOBIT-(NOW-1)*NBITW
```

```
C  PHASE II : Examine and return IBIT or set to IBIT for any I
C  ---------- and J.

C  Set the correct bit to the value IND, without effect on the
C  other bits of IJ(NOW).

         IF (IND.EQ.1) THEN
           DO K=1,NBITW
             BITSET(K)=2
           END DO
           BITSET(NOPOS)=IBIT
           CALL BIT01 (IJ(NOW),IND,BIT,BITSET,NBITW)
         ELSE

C          Examine the correct bit value of IJ(NOW).

           CALL BIT01 (IJ(NOW),IND,BIT,BITSET,NBITW)
           IBIT=BIT(NOPOS)
         END IF
       END IF

     RETURN

     END

C*************************************************************************
     SUBROUTINE BIT01(I,IND,BIT,BITSET,NBITW)
C*************************************************************************

C  Machine independent subroutine to examine or set to a value
C  0 or 1  the  bits of any machine word equivalent to a given
C  integer I.
C
C  ARGUMENTS :
C
C  NBITW    : Number of bits per machine word or considered by
C             the system.  For  the  VAX11,  NBITW=16  and
C             INTEGER*2.  (NBITWth... ...9th,8th,7th,6th,5th,
C             4th,3th,2nd,1nd)
C  I        : INTEGER  (NBITW bits)  (ex.: NBITW=16, 32 OR 36)
C  IND      : If  IND=0  :  examine  the  NBITW bits level and
C             store the result in BIT().
C             If IND=0  :  set  the  NBITW  bits  level  of  I
C             according the elements of BITSET() equal to 0 or
C             1.
C  BIT ()   : Give the content of the NBITW bits of I.
C  BITSET() : If the content is 0, set the corresponding bit of
C             I to 0,
C             If the content is 1, set the corresponding bit of
C             I to 1,
C             otherwise, no effect on the integer I.
```

```
C
C  REFERENCE :
C
C  GUINIER D.  (1983)  :  High  level  multilanguage   machine-
C              independent programmation  (16,  32, 36 bits) : A
C              subroutine  for  bits  manipulations in BASIC and
C              FORTRAN IV.  DECUS,  RT11 SIG, Mini-Tasker, vol9,
C              no.4, 10-1983.
C
C***************************************************************

      IMPLICIT INTEGER*2 (I-N)

      INTEGER*2 BITSET(36),BIT(36)


C  PHASE I : Search of the bits'level for the field integer I.
C  -------------------------------------------------------------
C     Examination of the bits level of the integer I.

      NBR=NBITW-1
      IP=I
      BIT(NBITW)=0
      IF(IP.LT.0) THEN
C        IP is previously a negative integer.
         BIT(NBITW)=1
         IP=IP+2.**(NBITW-1)
      END IF
C          IP is a positive integer.
      DO J=1,NBR
         BIT(J)=MOD(IP,2)
         IP=IP/2
      END DO

      IF (IND.EQ.0) RETURN

C  PHASE II : Possible changes of the actual level of the bits.
C  -------------------------------------------------------------
C     Set bits of integer I to 0 or 1 if required.

      DO J=1,NBITW
         IF (BITSET(J).NE.0) THEN
            IF (BITSET(J).EQ.1) THEN
               BIT   (J)= 1
               BITSET(J)=-1
            END IF
         ELSE
            BIT   (J)= 0
            BITSET(J)=-1
         END IF
      END DO
```

```
C  PHASE III : Releasing of a new field integer I after change
C  -----------------------------------------------------------
C                        of levels.
C                        ----------
C  Integer I reconstitution using Horner Scheme.

      I=BIT(NBR)
      N=NBR-1
      DO J=N,1,-1
         I=I*2+BIT(J)
      END DO

C     If I is a negative number.
      IF (BIT(NBITW).EQ.1) I=I-2.**(NBITW-1)

      RETURN

      END
```

LIST OF REFERENCES

1. Guinier, D. (1979): Bioinformatique; realisation d'un sous-systeme interactif et conversationnel, son application dans l'exploitation des donnees experimentales en Biologie, These d'Universite es Sciences, Dec. 29, no. 265, pp. 1-165.

2. Guinier, D. (1982): System informatique integre pour le traitement et l'exploitation des donnees en Physiologie animale. Communication, poster: Journees d'etude sur les applications de l'informatique a la Physiologie, CFDBM, Intern Fed for Bio-Med Engineering, 10-1982. Publication: Applications de l'Informatique a la Physiologie, Prof. d'Alche Ed., pp. 67-81.

3. Guinier, D. (1984): Simulation; Methods and structured FORTRAN 77 subroutines for compression of decision binary matrices, DECUS VAX1- SIG and ACM SIGIR (submitted).

4. Guinier, D. (1983): High level multilanguage machine-independent programmation (16, 32, 36 bits): A subroutine for bit manipulations in BASIC and FORTRAN IV. DECUS, RT11 SIG, Mini-Tasker, Vol. 9, No. 4, 10-1983.

5. Guinier, D. (1980): Comparison of several sorting methods for numbers of characters running on a PDP11/05 under RT11 in FORTRAN IV language, DECUS RT11 SIG, Mini-Tasker, Vol. 6, No. 2, 4-1980.

6. Singleton, R.C. (1969): Algorithm 347, SORT, an efficient algorithm for sorting with minimal storage, Comm. ACM, Vol. 12, pp. 185-186.

7. Tenenbaum, A.M., Augenstein, M.J. (1981): Data Structures Using Pascal, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, pp. 1-545.

8. Shell, D.L. (1959): A high speed sorting procedure, Comm. ACM 2, July, pp. 30-32.

9. Hibbard, T.N. (1963): An empirical study of minimal storage sorting, Comm. ACM, 6, May, pp. 206-213.

10. Berztiss, A.T. (1975): Data Structures, Theory and Practice, Academic Press, Second Edition, pp. 1-586.

11. Guinier, D., Totos, N. (1984): Sorting methods and evaluation, Developmentof structured FORTRAN 77 subroutines for sorting by address calculation using deviations of the "Tree monkey puzzle"'s, Shell's and Singleton's methods and performances evaluation, DECUS VAX11 SIG and ACM SIGIR (submitted).

12. Bartlett, M.S. (1947): The use of transformations, Biometrics, 3, pp. 39-52.

13. Airchison, J., Brown, J.A.C. (1957): The Lognormal Transformation, Cambridge University Press, Cambridge, Mass.

14. Snedecor, G.W., Cochran, W.G. (1967): Statistical Methods, Iowa University Press.

BIBLIOGRAPHY

Brillinger, P.C., Cohen, D.J. (1972): Introduction to Data Structures and Non-Numeric Computation, Prentice-Hall, pp. 1-629.

Floyd, R.W. (1964): Algorithm 245: Treesort 3, Comm. of the ACM, Vol. 7, No. 12, December, p. 701.

Griffin, R., Redish, K.A. (1969): Remark on algorithm 347 [M1], An efficient algorithm for sorting with minimal storage, Collected algorithms from ACM, Vol. 1, 347-p 3- 0.

Guinier, D., Kirsch, R. (1979): An overlay interactive and conversational program to load and verify real paris of Y, X, from any keyboard information. DECUS, RT11 SIG, Mini-Tasker, Vol. 5, No. 1, 1-1979.

Guinier, D., Totos, N. (1984): Interactive data analysis, development of a structured FORTRAN 77 interactive data manager, ACM SIGBIO and DECUS, VAX11 SIG (submitted).

IDA-SPSS (1982): Conversational Statistics with IDA. An Introduction to Data Analysis and Regression. The SPSS Series in Data Analysis, The Scientific Press, McGraw-Hill, 19 chap., 2 appendix, 1 index.

Knuth, E. (1981): The Art of Programming--Vol. 3: Sorting and Searching, Addison-Wesley, pp. 1-388.

Ling, R.F., Roberts, H.W. (1982): User's Guide to the Interactive Data Analysis and Forecasting System.

Mac-Larren, D. (1966): Internal sorting by radix plus shifting, Journal of the ACM.

Page, E.S., Wilson, L.B. (1978): Information representation and Manipulation in a Computer, Cambridge University Press, Second Edition, pp. 1-271.

Peto, R. (1970): Remark on Algorithm 347, Collected Algorithms from ACM, Vol. 2, 347-p 4- 0.

Schaub, R. (1973): Etude de quelques methodes de tri (Note d'etude AN-No. 35, C.C.S.A.

Scowen, R.S. (1965): Algorithm 271, Quickersort, Comm. ACM, Vol. 8, November, pp. 669-670.

Windley, P.F. (1960): Trees, Forests and Rearranging, Computer J., Vol. 3, No. 2, July, pp. 84-88.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Department Chairman, Code 54Hq Department of Computer Science Naval Postgraduate School Monterey, California 93943	1
4.	Hellenic Navy General Staff Stratopedon Papagou Holargos, Athens, Greece	2
5.	Adjunct Professor Daniel L. Guinier Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93943	5
6.	Adjunct Professor Daniel L. Guinier Groupe de Laboratoires du C.N.R.S. BP 20 CR F 67037 Strasbourg Cedex, France	5
7.	Associate Professor Roger Weissinger-Baylon 80 Cuesta Vista Dr. Monterey, California 93940	5
8.	Nicholas G. Totos Sevastias 35 N. Smirni Athens, Greece	5

END

FILMED

4-85

DTIC