AD-A042 213 DIRECTORATE OF AEROSPACE STUDIES KIRTLAND AFB N MEX F/G 5/2
FORTRAN SOFTWARE FOR CREATING AND MAINTAINING A LIBRARY CATALOG--ETC(U)
JUL 77 C A FEUCHTER

UNCLASSIFIED DAS-TR-77-3 NL

| OF |
AD
A042213

END

DATE
FILMED
8-77

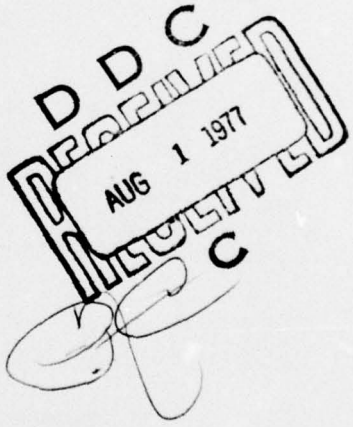
12

DAS-TR-77-3

FORTRAN SOFTWARE FOR CREATING AND MAINTAINING A LIBRARY CATALOGING SYSTEM ON SCIENTIFICALLY ORIENTED COMPUTERS:

VOLUME II, PROGRAM LIBLIST - CATALOG LISTINGS

JULY 1977



DIRECTORATE OF AEROSPACE STUDIES
DCS/DEVELOPMENT PLANS HQ AFSC
KIRTLAND AFB, NEW MEXICO 87117

The usefulness of a collection of reference documents is highly dependent upon the ease of identifying which documents are of potential interest in any given instance. Increasing the scope and flexibility of the document cataloging system is one method of making this identification easier. With this as motivation, the Directorate of Aerospace Studies undertook a computerization of the cataloging system for its document center in 1975. The principles of the system and the required software were developed inhouse over a period of months. They have since been refined and have proved themselves in use for two years. This documentation was written to describe the software and its use in enough detail to permit its adoption by other groups seeking the benefits coming from computerized cataloging; be it of documents, books, records - anything which can be categorized by various types of information.

The author would like to acknowledge the assistance of Mary C. Kennedy in designing the principles of the system. I am only sorry that it has not yet been possible to implement all her suggestions.

Christopher A Feuchter

CHRISTOPHER A. FEUCHTER
Study Director

This report has been reviewed and is approved for publication.

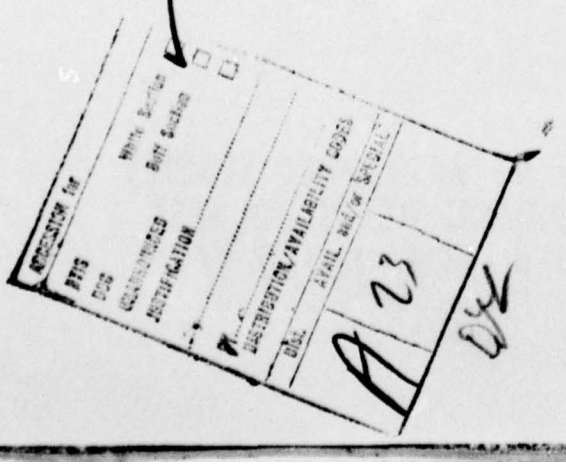
Harry L. Gogan

HARRY L. GOGAN
Technical Director

James P. Ditz

JAMES P. DITZ, Lt Colonel, USAF
Director of Aerospace Studies

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

(14) 1. REPORT NUMBER: DAS-TR-77-3

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

(6) 4. TITLE (and Subtitle): FORTRAN SOFTWARE FOR CREATING AND MAINTAINING A Library CATALOGING SYSTEM ON SCIENTIFICALLY ORIENTED COMPUTERS. VOLUME II. PROGRAM LIBLIST - CATALOG LISTINGS.

5. TYPE OF REPORT & PERIOD COVERED

6. PERFORMING ORG. REPORT NUMBER

(10) 7. AUTHOR(s): Christopher A. Feuchter

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Directorate of Aerospace Studies
Kirtland AFB NM 87117

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

(11) 12. REPORT DATE: July 1977

13. NUMBER OF PAGES: 42

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

(12) 41 p.

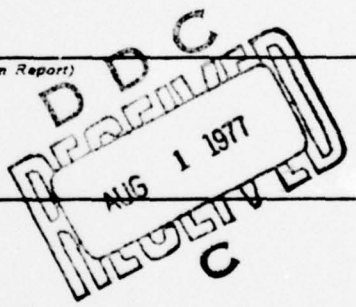
15. SECURITY CLASS. (of this report): Unclassified

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)

Automation
Computers
Libraries
Computer Programs
Computer Program Documentation
FORTRAN

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

The cataloging system of the document center of the Directorate of Aerospace Studies has been computerized on the CDC 6600 computers of the Air Force Weapons Laboratory. The system is unusual in that catalog listings of the holdings may be arbitrary in informational content, i.e., with regard to what types of information appear and their order of appearance. Although its initial application was to library cataloging it may be used to catalog any collection of objects which can be described by at most 20 categories of information.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409237

(Cont'd p 1473 A) → This volume

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

Block 20. This document describes LIBLIST, the FORTRAN program which produces printed catalogs from data bases maintained by its companion program ADM. The volume is a user's manual and a programmer's manual. LIBLIST is described in sufficient detail to permit its adoption to other high level computer languages and/or other large, scientifically oriented computers.

A

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

Program LIBLIST is one of two FORTRAN programs which is used to maintain the computerized cataloging system of the document center of the Directorate of Aerospace Studies. It produces catalog listings from any data base which is maintained by its companion program, ADM, discussed in volume I of this report. LIBLIST is written specifically for the CDC 6600 computer. Hence, it is designed about a 60-bit computer word, six bit characters, and a particular 64 member printer character set. These program peculiarities will not represent much difficulty to an experienced programmer desiring to adapt the program to another large scale machine. Additionally, the program should be easily converted from FORTRAN to other high level languages provided suitable bit manipulation capabilities can be provided.

This volume is designed to provide the degree of understanding required to run and modify the program and to make computer or language conversions possible. Section 2 provides a general discussion of the program, including an introduction to a vocabulary which is helpful in explaining the overall techniques of manipulating the data. Section 3 describes program input. Appendix A provides definitions for major program variables, while the program listing, interspersed liberally with comments, is presented in appendix B.

2. GENERAL PROGRAM DISCUSSION

The data base is the total of all information available to the computer about library holdings. It is maintained on a mass storage file which is referred to in LIBLIST as NEWFIL. The information on NEWFIL is organized by document, each document being defined by at most 20 descriptors. The sum total of information for a document is called a data set. The individual descriptors are called data items. The data items in use at present[1] are given in table 1 with descriptive remarks. Data items may contain multiple distinct entries. This results in that document appearing under each entry in a listing ordered by that data item.

Through inputs the data items to appear in a listing are selected and the order of their appearance specified. There are no restrictions on which data items may appear in a listing or their order. When the program receives this information it begins an exhaustive examination of the data base, document by document, to construct the listing. As an example, suppose that the listing is to be by keyword followed by several other data items. As each document data set is read from the data base it is examined for the presence of a keyword entry. If none exists, no further processing of that document occurs. If a keyword entry is present, it is examined to determine if it is single or multiple. For each keyword (up to a maximum of ten) an entry for the listing is prepared. Each of these entries differs only in the keyword, hence its eventual placement in the catalog listing. These entries are called character strings.

Groups of these character strings are prepared, with approximately 500 per group. As each group is completed, it is ordered alphanumerically. Alphanumeric ordering is essentially a well defined extension of alphabetic ordering. It is discussed in detail in section 2.2. After ordering a group, it is written on mass storage to await further processing. When all data sets from the data base have been processed in this manner, there exists one or more of these ordered groups of character strings which must be combined to form the one ordered group which comprises the catalog listing. This combining is done through a merging process

---

1. Other categories of information can easily be substituted for those given in this report. See volume I, section 1.

Table 1

DATA ITEMS

	ITEM	REMARKS
1.	Accession Number	A unique alphanumeric document identifier assigned by the Directorate of Aerospace Studies document librarian to all documents, except microfiche.
2.	AD Number	An alphanumeric identifier assigned by DDC to all copies of any document they process.
3.	Title	Document title.
4.	Copy Number	Copies of classified documents are numbered sequentially at printing.
5.	Author	Personal Author(s)
6.	Corporate Author	The organization(s) responsible for authoring the document.
7.	Report Number	A number assigned to a document by the corporate author.
8.	Document Date	The nominal date of document publication.
9.	Date Received	The date the document was received at the Directorate of Aerospace Studies.
10.	Requestor	The individual initially requesting the document.
11.	Requesting Study	The study for which the document was acquired.
12.	Classification	The security classification of the document.
13.	Special Access	Special restrictions on distribution of document.
14.	Downgrading Information	The date(s) of downgrading a classified document to a lower classification.
15.	Document Status	e.g., on loan, loaned, etc.
16.	Comments	Anything not covered by other categories.
17.	Format	Document format, e.g., book, technical report, vugraph, etc.
18.	Key Words	Words, mnemonics, or expressions characterizing the document.
19.	Abstract	This category is not currently used.
20.	Not used	

similar to that used to order the subgroups. The process is described in section 2.3. When the final ordered list is created it must be further processed (section 2.4) to alter its appearance for the actual printout. An example of the final result is presented in figure 1.

2.1 DATA BASE

The collection of all document information available to LIBLIST on file NEWFIL is the data base. This information has been organized by document into data sets, and in turn the data sets have been organized into the data items listed in table 1. In arriving at this structure, many points had to be considered. The discussion which follows will examine those considerations from the point of view of the data item, the data set, and the data base, respectively.

a. Data Items. Data items represent categories of information, e.g., title, personal author, keywords. For a given document some of these categories may not be represented. There may be no personal author, for example. Some categories may have multiple entries which should be treated independently when forming a catalog listing. An obvious example is keywords. And, whereas the title for one document may occupy a single computer word, that for another document may require 15 computer words.

To handle these variations the data item was conceived as having an indefinite length, and as having a substructure. A data item has zero length if there is no information about a document in that category. The upper limit for its length is indefinite. (The sum of the lengths of all data items in a data set can not exceed 54 computer words when stored in the data base.)[2] Notice that this applies equally to all data items. The substructure of a data item is achieved by linking multiple independent components together with plus (+) signs. This eliminates the plus sign from the otherwise usable character set.

The CDC printer characters are given in table 2 with their display codes (octal representation in the computer). Note that the colon is not available for

---

2. ADM and LIBLIST could be modified to increase this limit. Tradeoffs of program lengths and running times are involved.

00509 - SYSTEM APPROACH TO BASE STOCKAGE OF RECOVERABLE ITEMS - FEENEY, G. J. - RAND CORPORATION - RAND-RM-4720-PR - DEC 1965 - UNCL - ME -

00510 - DESCRIPTION AND APPLICATION OF A COMPUTER PROGRAM FOR AREA INSPECTION BY SATELLITE - ROWELL, L. N. - RAND CORPORATION - RAND-RM-4761-PR - DEC 1965 - UNCL - ME -

00511 - EVOLVING NATURE OF THE WARSAW PACT - WOLFE, T. W. - RAND CORPORATION - RAND-RM-4835-PR - DEC 1956 - UNCL - ME -

00537 - FEASIBILITY AND DESIRABILITY OF MOBILE AREA DEFENSE AGAINST THE SLBM - 126 - BLUMENTHAL, I. S. - RAND CORPORATION - RAND-RM-4675-ARPA - OCT 1965 - SECRET/RD - ME - SLBM -

00563 - MANPOWER PLANNING FACTORS FOR AIR FORCE SPACE SYSTEMS IN THE CONCEPTUAL STAGES OF DEVELOPMENT - HEUSTON, M. C. - RAND CORPORATION - RAND-RM-2823-PR - FEB 1962 - UNCL - ME -

00592 - GREAT CIRCLEDOM REVISITED-A SIMPLE-ACCURATE METHOD FOR DETERMINING GREAT CIRCLE DISTANCES AND ROUTES ON POLAR STEREOGRAPHIC PROJECTIONS - MURROW, R. B. - RAND CORPORATION - RAND-RM-4081-PR - NOV 1965 - UNCL - ME -

00593 - DETECTION OF SONAR SINUSOIDS OF UNKNOWN FREQUENCY AND KNOWN OR UNKNOWN PHASE - HILL, F. S., JR. - RAND CORPORATION - RAND-RM-4908-ARPA - DEC 1965 - UNCL - ME -

00647 - STUDIES ON PREDICATION IN RUSSIAN-2 ON THE PREDICATIVE USE OF THE RUSSIAN INFINITIVE - BIRNBAUM, H. - RAND CORPORATION - RAND-RM-4477-PR - DEC 1965 - UNCL - ME -

00648 - SYSTEM OF DENUMERABLY MANY TRANSIENT MARKOV CHAINS - PORT, S. C. - RAND CORPORATION - RAND-RM-4650-PR - JAN 1966 - UNCL - ME -

00649 - MUILTIPLE SCATTERING IN HOMOGENEOUS PLANE-PARALLEL ATMOSPHERES - MULLIKIN, T. W. - RAND CORPORATION - RAND-RM-4846-PR - DEC 1965 - UNCL - ME -

00697 - PRACTICABILITY OF EXTENDED-RANGE BALLISTIC MISSILES - 1 - GORDON, G. - RAND CORPORATION - RAND-RM-2378 - 12 FEB 1959 - UNCL - ME -

00760 - OFFSET CIRCLE PROBABILITIES - RAND CORPORATION - RAND-R-234 - 14 MAR 1952 - UNCL - PAM -

00761 - INTEGRAL OF THE GAUSSIAN DISTRIBUTION OVER AN OFFSET ELLIPSE - GERMOND, H. H. - RAND CORPORATION - RAND-P-94 - 28 JUL 1949 - UNCL -

00762 - EXPECTED COVERAGE OF A SMALL CIRCULAR TRAGET BY A NUMBER OF CIRCULAR BOMBS - DISHINGTON, R. H. - RAND CORPORATION - RAND-RM-413 - 26 JUN 1950 - UNCL - ME -

00831 - STUDIES IN INTER-SENTENCE CONNECTION - HARPER, K. E. - RAND CORPORATION - RAND-RM-4828-PR - DEC 1965 - UNCL - ME -

00970 - ECONOMICS DEPARTMENT PUBLICATIONS-1960-1965-AN AUTHOR INDEX OF THE OPEN LITERATURE WITH ABSTRACTS - PORCH, M. - RAND CORPORATION - RAND-RM-2800-1 (SUPPLEMENT) - JAN 1966 - UNCL - ME -

00971 - EVALUATION OF THE DETERMINANTAL POLYNOMIAL OF A MATRIX OF POLYNOMIAL ELEMENTS AND AN APPLICATION TO AMTI RADAR - LIECHENSTEIN, M. I. - RAND CORPORATION - RAND-RM-4837-PR - JAN 1966 - UNCL - ME -

00972 - NUMERICAL INVERSION OF LAPLACE TRANSFORMS AND SOME INVERSE PROBLEMS IN RADIATIVE TRANSFER - BELLMAN, R. E. - RAND CORPORATION - RAND-RM-4856-PR - DEC 1965 - UNCL - ME -

01030 - 92288 - GRAPHICAL DETERMINATION OF BALLISTIC TRAJECTORIES THROUGH OUTER SPACE WITH COMPASS AND STRAIGHTEDGE - FRICK, R. H. - RAND CORPORATION - RAND-RM-1641 - 24 FEB 1956 - UNCL - RM -

01049 - BASIC MEASURES FOR COMPARING THE EFFECTIVENESS OF CONVENTIONAL WEAPONS - SCHAFFER, M. B. - RAND CORPORATION - RAND-RM-4647-PR - JAN 1966 - UNCL - ME -

Figure 1. A Sample Listing by Accession Number

Table 2

THE PRINTER CHARACTER SET

CHARACTER	DISPLAY CODE*	CHARACTER	DISPLAY CODE	CHARACTER	DISPLAY CODE
:	00	0	33	≠	64
A	01	1	34	→	65
B	02	2	35	∨	66
C	03	3	36	∧	67
D	04	4	37	↑	70
E	05	5	40	↓	71
F	06	6	41	<	72
G	07	7	42	>	73
H	10	8	43	≤	74
I	11	9	44	≥	75
J	12	+	45	¬	76
K	13	-	46	;	77
L	14	*	47		
M	15	/	50		
N	16	(	51		
O	17	)	52		
P	20	$	53		
Q	21	=	54		
R	22	Blank	55		
S	23	,	56		
T	24	.	57		
U	25	≡	60		
V	26	[	61		
W	27	]	62		
X	30	%	63		
Y	31				
Z	32				

*The display code is the octal representation in the machine.

use even though it is a legitimate character. This is because the display code of the colon is 00, which means that in the alphanumeric ordering process the colon plays the role of a blank. More is said of this in section 2.4. The plus (+) sign is also reserved as previously indicated. Two consecutive colons (blank substitutes) may not appear imbedded in a data item. Two consecutive colons are the indication of the end of a data item. Hence, they must appear at the end of every data item, even if an extra computer word must be added to provide one or both of them. The remaining 61 characters are available to be used in any manner in a data item with the additional exception that all punctuation characters are automatically spaced at the time ADM creates the data base. Thus, for example, commas are always followed by a colon (blank substitute), but never preceded by one. This information is given in table 3.

Table 3

PROGRAMED PUNCTUATION SPACING

No blank precedes

+ - * / = %)] , . ; $

No blank follows

([+ - * / =

A blank precedes

([

A blank follows

)] % , . ; $

Should two characters with conflicting rules be adjacent to one another, that applying to the second character will prevail, e.g.) and - is spaced)-.

Data items in the data base are stored in display code and each data item occupies an integral number of computer words. Thus, for example, when Rand Corporation is used as a corporate author it would be stored in the data base as the two computer words

2201160400031722201 7|2201241117160000000 0

which when converted to characters is

RAND:CORPO|RATION::::

Notice that this data item is terminated by at least two consecutive "blanks," i.e., colons. Were the corporate author Bigdome Corporation, then it would appear in character representation as:

BIGDOME:CO|RPORATION: ::::::::::

where the extra word of "blanks" has been added to provide the second "blank" of the pair.

Since the rules discussed above apply to all data items, all data items have essentially the same status. This equality is exactly what allows catalog listings to be arbitrary with respect to which data items appear and the order of their appearance.

b. Data Set. To form a data set, the data items are assembled in consecutive locations in the data base in the order given in table 1.

Some method must be used for specifying the locations of each of the data items. The method selected is to include in the data set a table giving the location of the first word of each data item (numbering the first word of the data set 1) and the number of words comprising it. This has been done by packing this information for the 20 data items into five computer words, four data items to a word. Since a CDC 6600 word has 60 bits, this means that the length and location information for a data item must be presented in 15 bits. The first six are used to give the length; the last nine the location. This is shown in figure 2.

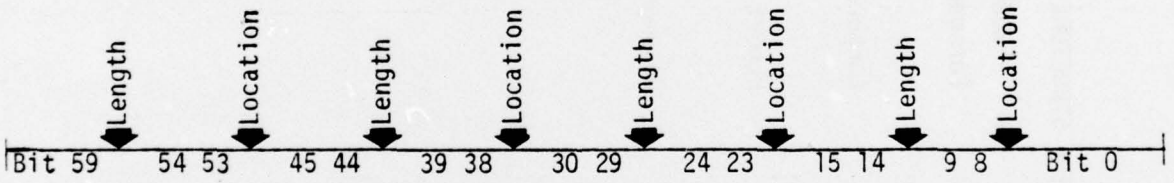


Figure 2. Format of a Word of the Length-Location Table

The correspondence between data item and computer words and bit numbers is given in table 4. The maximum expressable length of a data item is 77 octal (63 decimal) computer words. The maximum location is 777 octal (511 decimal) words. These limits have proved more than adequate.

The data set is completed by preceding the length-location table with a single computer word giving the total length of the data set.

The final data set, thus, has the form shown in figure 3.

Table 4

THE STRUCTURE OF THE LENGTH-LOCATION TABLE

BITS	WORD 2	WORD 3	WORD 4	WORD 5	WORD 6
Length 59-54 Location 53-45	Accession Number	Personal Author	Date Received	Special Access	Format
Length 44-39 Location 38-30	AD Number	Corporate Author	Requestor	Downgrading Information	Key Words
Length 29-24 Location 23-15	Title	Report Number	Requesting Study	Document Status	Abstract
Length 14-9 Location 8-0	Copy Number	Document Date	Classification	Comments	Not Used

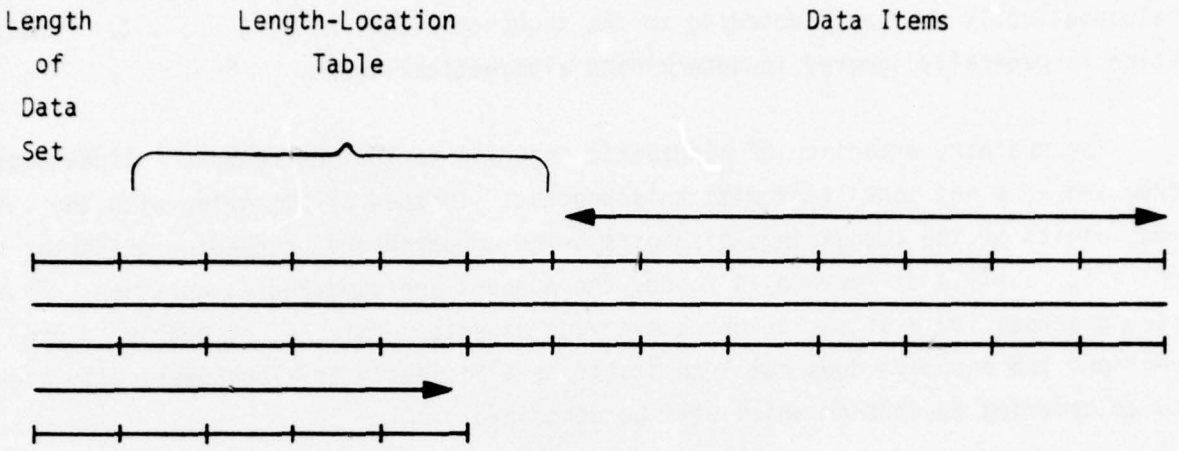


Figure 3. Format of a Data Set

c. Data Base. The data base is composed of the collection of all data sets. For reasons which will become obvious, it is necessary to have these data sets ordered in some manner in the data base. This is brought about not by any requirement of LIBLIST, but by the need to be able to delete or modify data base data sets with Program ADM. Suppose, for example, that 100 data sets are to be deleted from a data base of 1000 data sets because the corresponding documents are no longer part of the collection. If there is no order to the data sets in the data base, the list of data sets to be removed will have to be searched over and over again as the data sets in the data base are examined one by one. If, however, both lists are in order, each list will need to be examined only once to delete the 100 data sets.

The ability to order the data sets depends upon each data set having a unique identifier. For microfiche with AD numbers, this identifier is the AD number. For all other documents it is the accession number assigned to it at the Directorate of Aerospace Studies.

2.2 THE CONCEPT OF ALPHANUMERIC ORDERING

Ordinary alphabetical ordering is accomplished by comparing two words (or word groups) beginning with the left-most characters of each and proceeding character by character to the right until a difference is found. The two corresponding but different characters then determine which word or word group comes first by which is

"alphabetically smaller" according to the sequence blank, A, B, C . . . Z. Punctuation is generally ignored in determining alphabetical order.

The ordinary extension of alphabetic ordering to include numbers - alphanumeric ordering - is not consistent with this process. Instead of beginning with the leftmost digits of the two strings of digits being compared and proceeding character by character until a difference is found, the numbers are compared as entities. Thus nine precedes 10, a situation which everyone expects. This is, of course, a convention, and one that does not lend itself as efficiently to alphanumerically computer ordering as another which will be described.

For efficiency in computer ordering it is desirable to allow the ordering to proceed in the nonconventional way of comparing numbers exactly as letters are compared - character by character from the left. Using this scheme, very nonconventional ordering takes place, e.g., 10 precedes 9; 1000 precedes 11, etc. This too is a convention, but not one everyone will readily adapt to. A partial solution is to preface numbers with zero. Thus 09 will indeed precede 10.[3] This works as long as the two numbers being compared have the same number of digits. In most cases this is a practical way to obtain the traditional ordering.

There is one further difference between traditional alphanumeric ordering and the computer ordering appearing in ADM. Punctuation marks are considered in ADM as characters just as are the blank, letters, and the digits 0-9. The alphanumeric order of all the characters is that given previously in table 2. The order corresponds to the numerical order of the display codes. This consideration of punctuation in the ordering process makes consistency of punctuation essential if listings are to have the desired order at all times. Below are some pairs of expressions ordered traditionally and as the ADM/LIBLIST programs order them.

---

3. Roman numerals provide another problem in computer ordering. For example C(100) will order before I (1), IX (9) before V(5), etc. It will generally pay to replace Roman numerals with Arabic numbers.

	Traditional	ADM/LIBLIST
1.	BASE	BASE
	BASES	BASES
2.	ALPHA9 ALPHA09	ALPHA10 ALPHA09
	or	or
	ALPHA10 ALPHA10	ALPHA9 ALPHA10
3.	1812	18027
	18027	1812
4.	01812	01812
	18027	18027
5.	STATISTICS, VOL. 1	STATISTICS, VOL 2
	STATISTICS, VOL 2	STATISTICS, VOL. 1

These ordering characteristics should be understood and planned for before the first input data set is ever keypunched for program ADM.

2.3 ALPHANUMERIC ORDERING OF A SET OF CHARACTER STRINGS

There are many methods for computer ordering of a set of alphanumeric data. They vary widely in efficiency and ease of implementation, often as a function of the nature of the problem (for example, the initial amount of order present in the data). The character strings ordered in LIBLIST usually have very little intrinsic order.[4] A very efficient method for ordering in such a case is based on merging as suggested by the "tournament chart" of figure 4. In this context merging is the process of combining two ordered lists into a single ordered list by repeatedly comparing the smallest remaining element of each list and choosing the smaller of these as the next element of the combined list.

At the left of figure 4 are represented N (= 16 for this example) subsets of character strings, each subset containing a single character string. For convenience N is an integral power of 2. At the first step these subsets are merged by

4. An exception occurs when the ordering is being done on accession number or AD number. In these cases, the data are perfectly ordered as extracted from the data base.

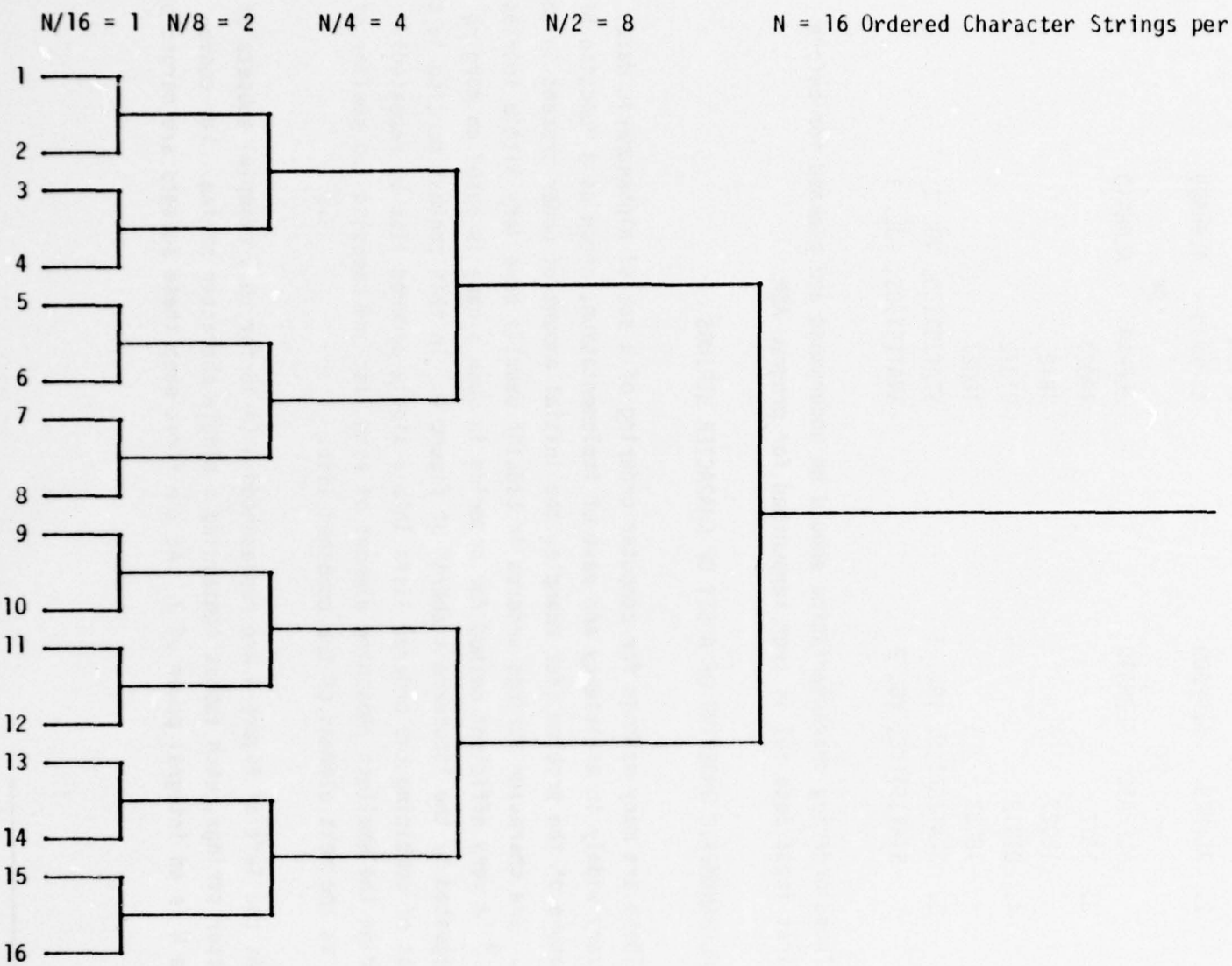


Figure 4. An Illustration of the Ordering Process Used in LIBLIST

pairs (1 and 2, 3, and 4, . . . (N-1) and N) to produce N/2 ordered subsets each with two character strings. At the next step these subsets are again merged by pairs, resulting in N/4 ordered subsets each with four character strings. At each step of the ordering process the number of character string subsets decreases by one-half, while the size of the individual subsets increases by a factor of two. Since $N = 2^n$ (n an integer), after n steps in the process, there remains one ordered subset of N character strings.

In actual practice, each subset is formed only in terms of where in the array of character strings its members reside. This technique is very efficient when compared with actually moving the character strings to form the subsets in adjacent locations in the computer.

In LIBLIST N was chosen as 512. Since each of the n steps of the process requires handling N strings and $512=2^9=2^n$, a maximum of Nxn or 512 x 9 = 4608, comparisons are necessary to order 512 character strings.[5] A comparison takes only a few microseconds on a CDC 6600 making the process quite fast.

In the event that there are fewer than N character strings to be ordered, the ordering must still proceed as though there were N character strings. In this event, dummy one-word character strings are used to complete the N entries for ordering. The dummy character strings always order at the end of the final ordered subset. Thus, it takes as long to order one character string by itself as N character strings. This drawback is of little practical consequence, given the overall speed of the scheme.

The above method as programmed in LIBLIST requires that every character string in the set being ordered reside in the computer's central memory during the ordering process. This generates a storage requirement of 512 x 54 = 27648_{10} = 66000_8 computer words for the character strings. The size of the set being ordered could be doubled and still fit in the CDC 6600. It is obvious however that there is a limit

---

5. Nxn is an upper bound to the number of comparisons which must be made, but not the least upper bound. In merging two lists comparisons stop as soon as the elements of one list are exhausted.

to how large a set may be ordered at one time in central memory. This problem is circumvented by introducing a merging procedure using four mass storage files. These files are referred to in the program as TAPE1, TAPE2, TAPE3, and TAPE4. At the beginning of the merging procedure the character strings to be ordered reside on TAPE1 and TAPE2 as indicated in figure 5. Each of the indicated subsets is a final ordered list of the previously described process for ordering 512 character strings.

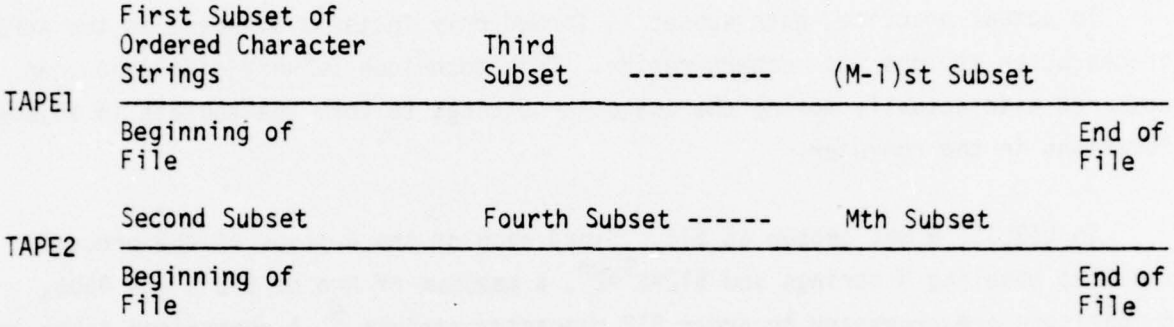


Figure 5. File Structure at Beginning of Merging Process

If there are an odd number of subsets of ordered character strings, the last one will be on TAPE1. If there are an even number, the last one will appear on TAPE2. The minimum number of such subsets is one; the maximum is currently 20. The alphanumeric order within a subset is from left to right in figure 5, i.e., A is to the left of Z.

The first merging operation involves the first subsets on TAPE1 and TAPE2. A character string from each is read and they are compared. The alphanumerically smaller is written on TAPE3. A new character string is read from the appropriate file to replace the string written on TAPE3, and the process of comparison is repeated. This is done until both subsets of character strings have been read from TAPE1 and TAPE2 and written on TAPE3. (When one of the two subsets has had all its entries read and merged, the remainder of the other subset is copied to TAPE3.) The subset on TAPE3 contains all entries of the subsets from TAPE1 and TAPE2 and the merging has properly ordered them. The next merging operation repeats the procedure with the second ordered subsets on TAPE1 and TAPE2. The only difference is that the merged subset is written on TAPE4.

This overall scheme is pursued until all subsets from TAPE1 and TAPE2 have been merged by pairs, alternately, on TAPE3 and TAPE4. Odd numbered merging operations are written on TAPE3; even on TAPE4. In the event that the total number of subsets in TAPE1 and TAPE2 is odd, the final subset on TAPE1 will not have a corresponding subset on TAPE2 with which to be merged. The merging process then consists simply of copying the last subset on TAPE1 to TAPE3 or TAPE4 as appropriate. When all data from TAPE1 and TAPE2 have been processed, the roles of TAPE1 and TAPE2, and TAPE3 and TAPE4 are reversed. Character strings are read from TAPE3 and TAPE4 and merged alternately on TAPE1 and TAPE2.

Obviously, the merging of two subsets reduces the total number of subsets by one. The logical conclusion of the above process is one ordered subset containing all character strings for the listing. A graphic example is given below.

ORIGINAL SITUATION

TAPE1 Subset 1 Subset 3 Subset 5

TAPE2 Subset 2 Subset 4

AFTER FIRST MERGING OPERATION

TAPE1 Subset 3 Subset 5

TAPE2 Subset 4

TAPE3 Subset 1 + Subset 2

AFTER SECOND MERGING OPERATION

TAPE1 Subset 5

TAPE3 Subset 1 + Subset 2

TAPE4 Subset 3 + Subset 4

AFTER THIRD MERGING OPERATION

TAPE3 Subset 1 + Subset 2 Subset 5

TAPE4 Subset 3 + Subset 4

AFTER FIRST MERGING OPERATION AFTER REVERSING FILE ROLES

TAPE1 Subset 1 + Subset 2 + Subset 3 + Subset 4

TAPE3 Subset 5

AFTER SECOND MERGING OPERATION AFTER REVERSING ROLES

TAPE1 Subset 1 + Subset 2 + Subset 3 + Subset 4

TAPE2 Subset 5

The roles of the files are reversed once again, and each file reverts to its original role. The final merging operation results in TAPE3 containing the final ordered collection of character strings. If there are 1, 2, 5, 6, 9, 10 . . . subsets, the final ordered collection of character strings will reside on TAPE3. Otherwise, it will reside on TAPE1. The final phase of the program converts the character strings to a form suitable for output and then does the actual output.

2.4 OUTPUT PROCESSING OF THE ALPHANUMERICALLY ORDERED CHARACTER STRINGS

At this point in the program, the character strings are unsuitable for printing. First, blanks are represented as colons. These must be converted to blanks. Second, each data item is terminated by at least two (at most 11) consecutive colons. These must be converted to a suitable "end of data item" marker to serve as a separator of adjacent data items in the printout. These two procedures require a character-by-character examination of each character string. This is the most time consuming portion of the program.

A typical character string if printed without these modifications might look like:[6]

SOC:::::::ROCKET-BOOSTED:GLIDE:BOMB:DISPERSION:ANALYSIS:(U):
::::::::::SANDIA:LABORATORIES::::::::::34707:::::

When processed it appears as:

SOC ↵ ROCKET-BOOSTED GLIDE BOMB DISPERSION ANALYSIS (U) ↵ SANDIA LABORATORIES 34707 ↵

6. The CDC 6600 can not print strings of colons as shown using the logic of LIBLIST as 12 zero bits at the end of a word are taken as an end-of-line indicator.

The actual processing involves constructing a new character string from the old. As each character is examined from the old character string it is either copied into the new character string, replaced (blank for colon), inserted (↱), or discarded (excess colons). When the conversion is complete, the new character string is printed.

3. INPUT TO PROGRAM LIBLIST

Card input to Program LIBLIST is used to specify the structure of data base printouts. This is done through use of the ITEM card which contains three fields. The first field is read under format A10 and contains ITEM beginning in column 1. The second field is read under format I5 (columns 11-15). The number in this field specifies the position in the final printout of the data item specified by the number read from the third field under format F10.0 (columns 16-25). The data item is identified by a number from 1-20 as given in table 1.

To request the program to produce a list consisting of document title, corporate author, and accession number would require the three cards

```
ITEM    1    3.0
ITEM    2    6.0
ITEM    3    1.0
```

plus a card to signify that the input for that case has been defined. This last card contains ENDCASE beginning in column 1. It is read by the same format as the ITEM cards, the numeric fields being ignored. The order of the ITEM cards defining a case is immaterial. However, if N ($1 \leq N \leq 20$) data items are being requested in a printout, then each of the N numbers specifying positions in the printout should be represented exactly once (with none greater than N). Multiple cases may be stacked in any run, each terminated by an ENDCASE card. The input deck should always be terminated by an ENDJOB card (ENDJOB beginning in column 1).

In addition to the card input, a file containing the data base must be supplied. This file is produced by Program ADM. Its structure is discussed earlier in this volume as well as in the volume I of this report, which describes Program ADM. This file has the logical file name NEWFIL. It is equated to TAPE5 in the program.

APPENDIX A
DEFINITIONS OF IMPORTANT PROGRAM VARIABLES

ENDATA	ENDATA=1 indicates that all data sets have been read from the data base.
ICA(I)	An array which contains the character string resulting from a read from unit NFRD1.
ICB(I)	An array which contains the character string resulting from a read from unit NFRD2.
ICT	A word containing one six-bit character with all other bits zero. The position of the character within the word is dependent upon usage.
IDIR	IDIR = -1 causes data being merged to be read from TAPE1 and TAPE2, written on TAPE3 or TAPE4. IDIR = 1 causes data being merged to be read from TAPE3 and TAPE4, written on TAPE1 and TAPE2.
IMASK(I)	For I = 1, . . ., 10, a mask of six consecutive bits set to one (111111 or 77 octal), occupying bit positions (59 - (I-1)6) to (54-(I-1)6). For I = 11, IMASK(11) = IMASK(1).
INAD(I)	An array containing the indirect addresses of the character strings in the ISORT array which are being ordered.
INADS(I)	An array which saves the INAD array for the next step of the ordering process. After the ordering process is complete INADS(I)=J gives the value of J such that ISORT(J,K) is the I-th ordered character string.
IND(I)	An array used to indicate the order and composition of the listing. Order is given by I. Composition is given by IND(I) according to the implicit numbering of data items specified in table 1.

IODATA(I)	Contains the most recent data set read from file NEWFIL.
IPOS	A counter which gives the number of characters contained in the result of a character-by-character examination of the first data item of the list or a character string.
IPOSP	A position indicator used to direct the placement of characters coming from a character-by-character examination of the first data item of the list or a character string. For IPOSP=1, . . ., 10 bit positions (59-(IPOSP-1)6) to 54-(IPOSP-1)6) are indicated.
IPRINT(I)	An array containing the version of a character set which is printed as part of the catalog listing.
ISORT(I,J)	The array where the character strings to be ordered are assembled. ISORT(I,1) gives the length of the character string.
ISPACE	A counter used to count the consecutive colons (blanks) appearing in a character string. ISPACE=2 indicates the end of a data item.
ISUM	A running count of the number of computer words in the character string being assembled in the ISORT array.
IWORD	A counter giving the number of the word of the IPRINT array being filled as a result of the character-by-character examination of the raw character string.
I1	The location in the INAD array of the indirect address of the element of the first of the two adjacent subsets of the ISORT array being merged, which is to be compared with an element from the second subset.
I1P	The location in the INAD array of the indirect address (in the ISORT array) of the last element of the first of the two adjacent subsets being merged.
I1S	Saves the value of the previous I1 for the calculation of a new I1.

I2	The location in the INAD array of the indirect address of the element of the second of the two adjacent subsets of the ISORT array being merged, which is to be compared with an element from the first subset.
I2S	Saves the value of I2 for later use.
LENLIM	The maximum number of words allowed in a character string. At present LENLIM = 50.
LENLOC (I,J)	LENLOC (1,J) contains the number of words in the J-th data item. LENLOC (2,J) indicates the location of the first word of the data item in the IODATA array.
LF (I,J)	For TAPEI, I=1,2,3,4, and the J-th subset of character strings on TAPEI; LF gives the length of that subset.
LIMUP	The maximum number of character strings which may be ordered at any one time. At present LIMUP = 512.
LIMUPH	LIMUP/2
LISTTOT	A running count of the number of character strings in the catalog listing being created.
NENTRY	The number of character strings to be formed from a given data set.
NF(I)	Maintains the count of the number of subsets of character strings on TAPEI, I = 1,2,3,4.
NFRD1	Equals 1 or 3 to indicate that READ(NFRD1) specified reading from TAPE1 or TAPE3, respectively.
NFRD2	Equals 2 or 4 to indicate that READ(NFRD2) specifies reading from TAPE2 or TAPE4, respectively.

NFTEST	An indicator used in selecting the value of NFWRT. For NFTEST = -1, NFWRT = 1 or 3, as appropriate. For NFTEST = 1, NFWRT = 2 or 4, as appropriate.
NFWRT	Equals 1,2,3, or 4, to indicate that WRITE(NFWRT) specifies writing on TAPE1, TAPE2, TAPE3, or TAPE4, respectively.
NITEM	The number of data items to be included in the listing.
NLINE	The number of lines needed to print the character string just processed.
NLIST	The number of character strings currently in the ISORT array.
NOREAD1	Equals 0 or 1 to indicate that the subset of character strings being read from the file specified by NFRD1 has or has not been exhausted, respectively.
NOREAD2	Similar to NOREAD1, except application is to NFRD2.
NR1	Counts the number of character strings read from the subset of character strings being read from unit NFRD1.
NR2	Similar to NR1, except application is to NFRD2.
NTF	The combined number of subsets of character strings residing on TAPE1, TAPE2, TAPE3, and TAPE4.
NWTEST1	NWTEST1 = 1 when ICA array contains a character set to be merged.
NWTEST2	NWTEST2 = 1 when ICB array contains a character set to be merged.
N2	At the I-th stage (I=1, . . ., 9) of the ordering process, N2 gives the length of the subsets being merged, viz., $2^{(I-1)}$.

APPENDIX B

PROGRAM LISTING


```
      PROGRAM LIBLIST(INPUT,OUTPUT,NEWFIL,TAPE1,TAPE2,TAPE3,TAPE4,TAPE5=
     1NEWFIL)
C
C     + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
C
C     COMMENTS SET OFF BY ASTERISKS ARE IN FORCE FOR ALL STATEMENTS
C     FOLLOWING THE COMMENTS UP TO THE NEXT GROUP OF SUCH COMMENTS. A
C     COMMENT FOLLOWED BY A NUMBER IN PARENTHESES APPLIES TO THE
C     FOLLOWING STATEMENTS THROUGH THE STATEMENT NUMBERED WITH THAT
C     NUMBER.
C
C     PROGRAM LIBLIST USES THE FORTRAN FUNCTIONS MASK(I), SHIFT(I,J),
C     AND MOD(I,J). MASK(I) FORMS A MASK OF I ONES IN BIT POSITIONS 0
C     THROUGH (I-1). BIT POSITIONS ARE NUMBERED LEFT TO RIGHT FROM 59 TO
C     0. SHIFT(I,J) AS USED IN THIS PROGRAM (J POSITIVE) RESULTS IN A
C     LEFT END-AROUND SHIFT OF THE WORD I BY J BITS. MOD(I,J) CALCULATES
C     I MODULO J. THE LOGICAL .AND. AND. OR. OPERATORS ARE ALSO USED.
C     THEY DO A BIT-BY-BIT LOGICAL SUM OR PRODUCT OF THE QUANTITIES
C     THEY CONNECT.
C
C     + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
C
      COMMON /A/ IND(20),NITE
      DIMENSION INAD(512), IODATA(100), ISORT(512,50), LENLOC(2,20), LIS
     1TT(10,25), INADS(512), IBIT(11), IMASK(10), IPRINT(100), LF(4,20),
     2 NF(4), ICA(100), ICB(100), NFP(4)
C
      INTEGER ENDATA
      DATA IBIT/60,54,48,42,36,30,24,18,12,6,0/
C
      DATA LENLIM/50/,LIMUP/512/,MULTLST/1/
C
C
      DO 10 I=1,10
      IMASK(I)=SHIFT(MASK(6),IBIT(I))
   10 CONTINUE
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     INITIALIZATION REQUIRED FOR EACH CATALOG LISTING
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
   20 CONTINUE
      CALL INPUT
      REWIND 1
      REWIND 2
      REWIND 5
      LIMUPH=LIMUP/2
      LISTTOT=0
```

```
      ENDATA=0
      IDIR=-1
      DO 30 I=1,4
      DO 30 J=1,20
      LF(I,J)=0
   30 CONTINUE
      NF(1)=NF(2)=NF(3)=NF(4)=0
      NTF=0
C
C     . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C     INITIALIZATION REQUIRED FOR EACH SUBSET OF CHARACTER STRINGS
C
C     . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
   40 CONTINUE
      NLIST=0
      DO 50 I=1,LIMUP
      DO 50 J=1,LENLIM
      ISORT(I,J)=10H;;
   50 CONTINUE
C
C
C     . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C     READ A DATA SET FROM FILE NEWFIL.
C
C     . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
   60 CONTINUE
      READ (5) N,(IODATA(K),K=2,N)
      IF (EOF(5)) 210,70
   70 CONTINUE
C
C     . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C     FOR THE DATA SET JUST READ, EXTRACT FROM IODATA(2) THROUGH
C     IODATA(6) THE LOCATIONS IN IODATA AND THE CORRESPONDING LENGTHS OF
C     THE DATA ITEMS TO BE USED IN THE CATALOG LISTING. K IS USED TO
C     SPECIFY WHICH WORD OF THE IODATA ARRAY CONTAINS THIS INFORMATION
C     FOR THE I-TH DATA ITEM. L SPECIFIES WHICH 15 BITS WITHIN THAT WORD
C     CONTAIN THE INFORMATION. THE SHIFT, MASK, AND .AND. FUNCTIONS ARE
C     USED TO ISOLATE THE INFORMATION WHICH IS STORED IN LENLOC IN
C     INTEGER FORMAT.
C
C     . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      DO 80 K=1,40
      LENLOC(K)=0
   80 CONTINUE
```

```
      DO 30 I=1,NITEM
      KL=IND(I)
      L=MOD(KL,4)
      IF (L.EQ.0) L=4
      K=(KL-L)/4+1
      LEN=SHIFT(IODATA(K+1),(L-1)*15).AND.MASK(6)
      LOC=SHIFT(IODATA(K+1),6+(L-1)*15).AND.MASK(9)
      LENLOC(1,KL)=SHIFT(LEN,6)
      LENLOC(2,KL)=SHIFT(LOC,9)
   30 CONTINUE
C          DO NOT PROCESS DATA SET IF FIRST ITEM DOES NOT EXIST
      IF (LENLOC(1,IND(1)).EQ.0) GO TO 60
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C     IF THE MOST RECENTLY READ DATA SET IS TO BE REPRESENTED IN THE
C     CATALOG LISTING, EXAMINE THE FIRST DATA ITEM FOR MULTIPLE ENTRIES,
C     AND IF PRESENT, BREAK OUT THE NENTRY ENTRIES AND STORE THEM IN
C     LISTT. THE PLUS SIGN SEPARATES THE MULTIPLE ENTRIES, IF ANY.
C
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C          INITIALIZE
      INITIAL=LENLOC(2,IND(1))
      LAST=INITIAL+LENLOC(1,IND(1))-1
      IPOS=IPOSP=0
      DO 100 K=1,250
      LISTT(K)=0
  100 CONTINUE
      NENTRY=1
C          ANALYSE THE FIRST DATA ITEM CHARACTER BY CHARACTER FOR + SIGNS
      DO 130 K=INITIAL,LAST
      DO 120 L=1,10
      ICT=SHIFT(IODATA(K),IBIT(12-L)).AND.MASK(6)
      IF (ICT.NE.1L+) GO TO 110
      LEN=(IPOS-1)/10+1
      IF (IPOSP.GT.8) LEN=LEN+1
      LISTT(NENTRY,1)=LEN
      NENTRY=NENTRY+1
      IF (NENTRY.GT.10) GO TO 140
      IPOS=IPOSP=0
      GO TO 120
  110 CONTINUE
      IPOS=IPOS+1
      IPOSP=IPOSP+1
      IF (IPOSP.EQ.11) IPOSP=1
      IT=(IPOS-1)/10+2
      LISTT(NENTRY,IT)=LISTT(NENTRY,IT).OR.SHIFT(ICT,IBIT(IPOSP))
  120 CONTINUE
  130 CONTINUE
```

```
C          INSURE PROPER TERMINATION OF FINAL ENTRY
      LEN=(IPOS-1)/10+1
      LISTT(NENTRY,1)=LEN
      IF (LISTT(NENTRY,LEN+1).EQ.0.AND.(SHIFT(MASK(12),12).AND.LISTT(NEN
     1TRY,LEN)).EQ.0) LISTT(NENTRY,1)=LISTT(NENTRY,1)-1
  140 CONTINUE
      IF (NENTRY.GT.10) NENTRY=10
C
C     . . . . . .
C
C     FORM NENTRY CHARACTER STRINGS FOR THE DATA SET BEING PROCESSED,
C     AND STORE THEM IN ISORT.  THIS IS DONE BY INSERTING THE NENTRY
C     ENTRIES IN LISTT BACK INTO IODATA ONE BY ONE, AND THEN LOOPING ON
C     NITEM TO ASSEMBLE THE DATA ITEMS IN THE PROPER ORDER.
C
C     . . . . . .
C
      DO 200 K=1,NENTRY
      NLIST=NLIST+1
C          FIRST DATA ITEM HAD ONLY ONE ENTRY, NO NEED TO PUT INTO IODATA
      IF (NENTRY.EQ.1) GO TO 160
C          PLACE AN ENTRY FROM LISTT IN IODATA
      LAST=LISTT(K,1)
      LENLOC(1,IND(1))=LAST
      DO 150 L=1,LAST
      IODATA(LENLOC(2,IND(1))+L-1)=LISTT(K,L+1)
  150 CONTINUE
  160 CONTINUE
C          ASSEMBLE THE CHARACTER STRING
      ISUM=1
      DO 190 J=1,NITEM
      LEN=LENLOC(1,IND(J))
      LOC=LENLOC(2,IND(J))
      IF (LOC.EQ.0) LEN=1
      IF (LOC.NE.0) GO TO 170
C          DO NOT ALLOW THE CHARACTER STRING TO EXCEDE LENLIM WORDS
      IF (ISUM+1.GT.LENLIM-1) GO TO 190
      ISORT(NLIST,ISUM+1)=10H;;
      ISUM=ISUM+1
      ISORT(NLIST,1)=ISUM
      GO TO 190
  170 CONTINUE
C          DO NOT ALLOW THE CHARACTER STRING TO EXCEDE LENLIM WORDS
      IF (ISUM+LEN.GT.LENLIM-1) LEN=LENLIM-ISUM-1
      IF (LEN.LE.0) GO TO 190
      DO 180 M=1,LEN
      ISORT(NLIST,ISUM+M)=IODATA(LOC+M-1)
  180 CONTINUE
      ISUM=ISUM+LEN
      ISORT(NLIST,1)=ISUM
```

```
  180 CONTINUE
  200 CONTINUE
C         DO NOT ALLOW ISORT ARRAY TO OVERFLOW
      IF (MULTLST.EQ.1.AND.NLIST.GE.LIMUP-10) GO TO 220
C         CONTINUE WITH NEXT DATA SET
      GO TO 60
C         DATA BASE EXHAUSTED
  210 CONTINUE
      ENDATA=1
C         SUBSET OF CHARACTER STRINGS READY TO BE ORDERED
  220 CONTINUE
      LISTTOT=LISTTOT+NLIST
C
C     . . . . . .
C
C     ORDER THE ISORT ARRAY ALPHANUMERICALLY. THE PROCEDURE DEPENDS ON 9
C     MERGING CYCLES. IN THE FIRST CYCLE CHARACTER STRINGS ARE MERGED
C     BY PAIRS - 1 AND 2, 3 AND 4, ETC. TO FORM ORDERED PAIRS OF
C     CHARACTER STRINGS. IN THE SECOND CYCLE THE FIRST PAIR IS MERGED
C     WITH THE SECOND, THE THIRD WITH THE FOURTH, ETC. TO FORM ORDERED
C     QUADS OF CHARACTER STRINGS. THE RESULT OF THE 9-TH CYCLE IS A
C     COMPLETELY ORDERED SET OF CHARACTER STRINGS. THE ORDER OF THE
C     CHARACTER STRINGS IS MAINTAINED IN THE INAD AND INADS ARRAYS.
C
C     . . . . . .
C
C         INITIALIZE INADS
      DO 230 I=1,LIMUP
      INADS(I)=I
  230 CONTINUE
C         BEGIN THE 9 CYCLE PROCESS
      DO 370 I=1,9
      N2=2**(I-1)
      IIS=-N2*2+1
      DO 350 J=1,LIMUPH,N2
      II=IIS=IIS+N2*2
      I2=I2S=II+N2
      INITIAL=II
      LAST=I2+N2-1
      L=2
      DO 340 K=INITIAL,LAST
  240 CONTINUE
      IF (K.GT.12) GO TO 250
  250 CONTINUE
      IF (ISORT(INADS(II),L).GE.0.AND.ISORT(INADS(I2),L).GE.0) GO TO 260
      IF (ISORT(INADS(II),L).LT.0.AND.ISORT(INADS(I2),L).LT.0) GO TO 260
      IF (ISORT(INADS(II),L).GE.0) GO TO 270
      GO TO 290
  260 CONTINUE
      IF (ISORT(INADS(II),L)-ISORT(INADS(I2),L)) 270,280,290
```

```
C          PLACE THE I1 CHARACTER STRING BEFORE THE I2 CHARACTER STRING
  270 CONTINUE
      INAD(K)=INADS(I1)
      I1=I1+1
      L=2
      IF (I1.EQ.I2S) GO TO 300
      GO TO 340
C          THE L-TH WORDS OF THE CHARACTER STRINGS ARE IDENTICAL, COMPARE
C          THE (L+1)ST WORDS
  280 CONTINUE
      IF (L.EQ.LENLIM-1) GO TO 290
      L=L+1
      GO TO 240
C          PLACE THE I2 CHARACTER STRING BEFORE THE I1 CHARACTER STRING
  290 CONTINUE
      INAD(K)=INADS(I2)
      I2=I2+1
      L=2
      IF (I2.GT.LAST) GO TO 320
      GO TO 340
C          THE I1 CHARACTER STRING LIST IS EXHAUSTED
  300 CONTINUE
      KP=K
      DO 310 M=I2,LAST
      KP=KP+1
      INAD(KP)=INADS(M)
  310 CONTINUE
      GO TO 350
C          THE I2 CHARACTER STRING LIST IS EXHAUSTED
  320 CONTINUE
      I1P=I2S-1
      KP=K
      DO 330 M=I1,I1P
      KP=KP+1
      INAD(KP)=INADS(M)
  330 CONTINUE
      GO TO 350
  340 CONTINUE
C          SAVE THE MOST RECENT ORDER FOR THE NEXT CYCLE OR FOR THE
C          STORAGE OPERATION TO FOLLOW
  350 CONTINUE
      DO 360 IL=1,LIMUP
      INADS(IL)=INAD(IL)
  360 CONTINUE
  370 CONTINUE
C
C     . . . . . .
C
C     WRITE THE CHARACTER STRINGS ON SCRATCH MASS STORAGE IN THE ORDER
C     SPECIFIED BY INADS. IF THE TOTAL NUMBER OF SUBSETS OF CHARACTER
```

```
C     STRINGS ORDERED FOR THIS CATALOG LISTING IS ODD, WRITE ON TAPE1;
C     IF EVEN, WRITE ON TAPE2.
C
C     . . . . . .
C
      NTF=NTF+1
      ITEST=XTEST=NTF/2.0
      NFWRT=1
      IF (ITEST.EQ.XTEST) NFWRT=2
      NF(NFWRT)=NF(NFWRT)+1
      LF(NFWRT,NF(NFWRT))=NLIST
      DO 380 I=1,NLIST
      N=ISORT(INADS(I),1)
      WRITE (NFWRT) N,(ISORT(INADS(I),K),K=2,N)
  380 CONTINUE
      IF (ENDATA.EQ.1) GO TO 390
      GO TO 40
C
C     . . . . . .
C
C     MERGE THE DATA STORED ON TAPE1 AND TAPE2 TO PRODUCE A SINGLE
C     ORDERED SET OF ALL CHARACTER STRINGS. SUBSETS FROM TAPE1 ARE
C     MERGED WITH THOSE FROM TAPE2. THE RESULTING MERGED SUBSETS ARE
C     WRITTEN ALTERNATELY ON TAPE3 AND TAPE4. WHEN ALL SUBSETS FROM
C     TAPE1 AND TAPE2 HAVE BEEN MERGED, THE PROCESS REVERSES AND SUBSETS
C     FROM TAPE3 AND TAPE4 ARE MERGED WITH THE RESULTS BEING WRITTEN
C     ON TAPE1 AND TAPE2. THIS PROCESS CONTINUES UNTIL A FINAL ORDERED
C     LIST IS ACHEIVED.
C
C     . . . . . .
C
  390 CONTINUE
C         BEGIN FOR NEW SUBSETS
      NFTEST=1
      REWIND 1
      REWIND 2
      REWIND 3
      REWIND 4
      NFP(1)=NFP(2)=NFP(3)=NFP(4)=0
  400 CONTINUE
C         BEGIN FOR NEW LISTS
      NOREAD1=NOREAD2=0
      NR1=NR2=0
      IF (IDIR.NE.-1) GO TO 410
C         WILL READ FROM UNITS 1 AND 2, WRITE ON UNITS 3 OR 4
      NFRD1=1
      NFRD2=2
      NFP(1)=NFP(1)+1
      NFP(2)=NFP(2)+1
      NFTEST=-1*NFTEST
```

```
      NFWRT=3
      IF (NFTEST.EQ.1) NFWRT=4
  410 CONTINUE
      IF (IDIR.NE.1) GO TO 420
C         WILL READ FROM UNITS 3 AND 4, WRITE ON UNITS 1 OR 2
      NFRD1=3
      NFRD2=4
      NFP(3)=NFP(3)+1
      NFP(4)=NFP(4)+1
      NFTEST=-1*NFTEST
      NFWRT=1
      IF (NFTEST.EQ.1) NFWRT=2
  420 CONTINUE
      IF (NF(NFRD2).EQ.0) NOREAD2=1
      IF (NF(NFRD1).EQ.0) GO TO 520
  430 CONTINUE
      IF (NOREAD1.EQ.1.AND.NOREAD2.EQ.1) GO TO 510
      IF (NOREAD1.EQ.1.AND.NWTEST1.EQ.0) GO TO 500
C         READ FROM NFRD1
      READ (NFRD1) N,(ICA(K),K=2,N)
      NWTEST1=1
      ICA(N+1)=10H;;
      NR1=NR1+1
      IF (NR1.EQ.LF(NFRD1,NFP(NFRD1))) NOREAD1=1
      IF (NOREAD1.EQ.1) NF(NFRD1)=NF(NFRD1)-1
      IF (NOREAD2.EQ.1.AND.NWTEST2.EQ.0) GO TO 480
      IF (NR2.EQ.0) GO TO 440
      GO TO 450
  440 CONTINUE
      IF (NOREAD1.EQ.1.AND.NOREAD2.EQ.1) GO TO 510
      IF (NOREAD2.EQ.1) GO TO 480
C         READ FROM NFRD2
      READ (NFRD2) M,(ICB(K),K=2,M)
      NWTEST2=1
      ICB(M+1)=10H;;
      NR2=NR2+1
      IF (NR2.EQ.LF(NFRD2,NFP(NFRD2))) NOREAD2=1
      IF (NOREAD2.EQ.1) NF(NFRD2)=NF(NFRD2)-1
      IF (NOREAD1.EQ.1) GO TO 500
  450 CONTINUE
C         COMPARE ICA AND ICB
      L=2
  460 CONTINUE
      IF (ICA(L).GE.0.AND.ICB(L).GE.0) GO TO 470
      IF (ICA(L).LT.0.AND.ICB(L).LT.0) GO TO 470
      IF (ICA(L).GE.0) GO TO 480
      GO TO 500
  470 CONTINUE
      IF (ICA(L)-ICB(L)) 480,490,500
  480 CONTINUE
```

```
      WRITE (NFWRT) N,(ICA(K),K=2,N)
      NWTEST1=0
      L=2
      GO TO 430
  490 CONTINUE
      IF (ICA(L).EQ.10H;          .AND.L.EQ.LENLIM-1) GO TO 500
      L=L+1
      GO TO 460
  500 CONTINUE
      WRITE (NFWRT) M,(ICB(K),K=2,M)
      NWTEST2=0
      L=2
      GO TO 440
  510 CONTINUE
C          MERGING OF TWO SUBSETS IS COMPLETE
      IF (NWTEST1.EQ.1) WRITE (NFWRT) N,(ICA(K),K=2,N)
      IF (NWTEST2.EQ.1) WRITE (NFWRT) M,(ICB(K),K=2,M)
      NF(NFWRT)=NF(NFWRT)+1
      LF(NFWRT,NF(NFWRT))=NR1+NR2
      IF (NR2.NE.0) NTF=NTF-1
      GO TO 400
  520 CONTINUE
      IF (NTF.EQ.1) GO TO 530
      IDIR=-1*IDIR
      GO TO 390
  530 CONTINUE
C          A SINGLE ORDERED LIST HAS BEEN ACHEIVED
      DO 540 I=1,3,2
      IF (NF(I).EQ.1) NFWRT=I
  540 CONTINUE
      REWIND NFWRT
C
C     . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C     PREPARE THE FINAL ORDERED LIST OF CHARACTER STRINGS FOR PRINTING
C     AND PRINT THEM AS THEY ARE PROCESSED. THE PREPARATION REQUIRES A
C     CHARACTER BY CHARACTER EXAMINATION OF EACH CHARACTER STRING TO
C     CONVERT COLONS TO BLANKS, TO ELIMINATE EXCESS BLANKS, AND TO
C     INSERT - AS A SEPARATOR BETWEEN DATA ITEMS.
C
C     . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
      IF (IND(1).NE.11) PRINT 710
      IPRNTSV=10H
      NCOUNT=0
      LST=LF(NFWRT,1)
C          PROCESS EACH CHARACTER STRING
      DO 690 L=1,LST
      READ (NFWRT) N,(IODATA(K),K=2,N)
C          INITIALIZE IPRINT
```

```
      LAST=N
      DO 550 I=1,LAST
      IPRINT(I)=0
  550 CONTINUE
C         SPACE DATA FOR OUTPUT CLARITY
      IPOS=IPOSP=ISPACE=0
C         EXAMINE EACH CHARACTER IN IODATA(I)
      DO 640 I=2,LAST
      IF (IODATA(I).EQ.10H;;          ) IODATA(I)=0
      DO 630 J=1,10
      ICT=IODATA(I).AND.IMASK(J)
      IF (ICT.NE.0B) GO TO 580
C         CHARACTER IS A COLON (SPACE)
      ISPACE=ISPACE+1
      IF (ISPACE.GT.2) GO TO 630
      IF (ISPACE.EQ.1) GO TO 570
C         END OF DATA ITEM, INSERT -
      IPOS=IPOS+1
      IPOSP=IPOSP+1
      IF (IPOSP.EQ.11) IPOSP=1
      IWORD=(IPOS-1)/10+1
      IPRINT(IWORD)=IPRINT(IWORD).OR.SHIFT(1L-,IBIT(IPOSP))
C         IF LISTING BY STUDY BEGIN EACH STUDY ON NEW PAGE
      IF (IWORD.NE.1) GO TO 570
      IF (IND(1).NE.11.OR.IPRINT(1).EQ.IPRNTSV) GO TO 560
      PRINT 710
      NCOUNT=0
  560 CONTINUE
      IPRNTSV=IPRINT(1)
  570 CONTINUE
C         CONVERT COLON TO A BLANK
      ICT=5555555555555555558.AND.IMASK(J)
      GO TO 590
  580 CONTINUE
C         CHARACTER NOT A COLON (SPACE)
      ISPACE=0
  590 CONTINUE
C         MAKE CHARACTER PART OF CHARACTER STRING TO BE PRINTED (630)
      IPOS=IPOS+1
      IPOSP=IPOSP+1
      IF (IPOSP.EQ.11) IPOSP=1
      IWORD=(IPOS-1)/10+1
C         DETERMINE NUMBER OF BITS CHARACTER IN ICT MUST BE SHIFTED FOR
C         STORAGE IN IPRINT(IWORD) (620)
      IF (IPOSP-J) 610,600,600
  600 CONTINUE
C         SHIFT TO LOWER ORDER BITS OR DO NOT SHIFT AT ALL
      ISHIFT=IPOSP-J+1
      GO TO 620
  610 CONTINUE
```

```
C          SHIFT TO HIGHER ORDER BITS
       ISHIFT=10-J+IPOSP+1
  620 CONTINUE
       IPRINT(IWORD)=IPRINT(IWORD).OR.SHIFT(ICT,IBIT(ISHIFT))
  630 CONTINUE
  640 CONTINUE
       IF (IPOSP.EQ.10) GO TO 660
C          BLANK FILL REMAINDER OF IPRINT(IWORD)
       DO 650 I=IPOSP,10
       IPRINT(IWORD)=IPRINT(IWORD).OR.(555555555555555555558.AND.IMASK(I
      1)
  650 CONTINUE
  660 CONTINUE
       NLINE=1+(IWORD-2)/12
       IF (NCOUNT+NLINE.LE.58) GO TO 670
C          NOT ENOUGH ROOM ON PAGE TO PRINT ENTIRE CHARACTER STRING,
C          EJECT PAGE
       PRINT 710
       NCOUNT=0
  670 CONTINUE
C          PRINT CHARACTER STRING WITHOUT EJECTING PAGE
       NCOUNT=NCOUNT+NLINE+1
       IF (NLINE.EQ.1) GO TO 680
C          CHARACTER STRING REQUIRES MORE THAN ONE LINE FOR PRINTING
       PRINT 700, (IPRINT(J),J=1,13)
       PRINT 720, (IPRINT(J),J=14,IWORD)
       GO TO 690
  680 CONTINUE
C          CHARACTER STRING REQUIRES ONLY ONE LINE FOR PRINTING
       PRINT 700, (IPRINT(J),J=1,IWORD)
  690 CONTINUE
       PRINT 730, LISTTOT
C          CATALOG LISTING COMPLETED, CHECK FOR ADDITIONAL LISTINGS
       GO TO 20
C
C
  700 FORMAT (/1X,13A10)
  710 FORMAT (1H1//)
  720 FORMAT (11X,12A10)
  730 FORMAT (///20X,I5,28H ENTRIES APPEAR ON THIS LIST)
       END
```

```
      SUBROUTINE INPUT
      COMMON /A/ IND(20),NITEM
      DIMENSION NAME(10)
      DATA IRETURN/1/
      DATA NAME/4HITEM,7HENDCASE,6HENDJOB/
C
      PRINT 80
      NITEM=0
   10 CONTINUE
      READ 50, INAME,ILOC,X
      PRINT 60, INAME,ILOC,X
      DO 20 I=1,10
      IF (INAME.EQ.NAME(I)) GO TO 30
   20 CONTINUE
      IRETURN=0
      PRINT 70, INAME
   30 CONTINUE
      IF (INAME.NE.NAME(1)) GO TO 40
      IND(ILOC)=X
      NITEM=NITEM+1
   40 CONTINUE
      IF (INAME.EQ.NAME(2).AND.IRETURN.EQ.1) RETURN
      IF (INAME.EQ.NAME(3)) STOP 1
      GO TO 10
C
C
   50 FORMAT (A10,I5,F10.0)
   60 FORMAT (1X,A10,I5,F10.1)
   70 FORMAT (5H ***,1X,A10,5H*****,21H IS NOT IN DICTIONARY)
   80 FORMAT (1H1)
      END
```