CANADA

DEPARTMENT OF
ENERGY, MINES AND RESOURCES
MINES BRANCH

OTTAWA

A DATA BASE MANAGEMENT AND INFORMATION RETRIEVAL SYSTEM EMPLOYING COMPUTER PROGRAMS MNGFLE AND SRHFLE

F. J. KELLY

EXTRACTION METALLURGY DIVISION

OCTOBER 1973

Technical Bulletin TB 178

Price: 75 cents

Available by mail from Information Canada, Ottawa,
and at the following Information Canada bookshops:

HALIFAX
1687 Barrington Street

MONTREAL
640 St. Catherine Street West

OTTAWA
171 Slater Street

TORONTO
221 Yonge Street

WINNIPEG
393 Portage Avenue

VANCOUVER
800 Granville Street

or through your bookseller

Price 75 cents Catalogue No. M34-20/178

Price subject to change without notice

Information Canada
Ottawa, 1973

Mines Branch Technical Bulletin TB 178

A DATA BASE MANAGEMENT AND INFORMATION RETRIEVAL SYSTEM EMPLOYING COMPUTER PROGRAMS MNGFLE AND SRHFLE

by

F. J. Kelly*

ABSTRACT

Two computer programs have been developed to manage and retrieve information from document reference files or data bases. The programs have proven to be reliable for these tasks and economic in operation. This report describes the development of these computer programs and provides detailed information and instructions on their use.

*Research Scientist, Ore Treatment Section, Extraction Metallurgy Division, Mines Branch, Department of Energy, Mines & Resources, Ottawa, Canada.

Direction des mines

Bulletin technique TB 178

UN SYSTÈME DE RECHERCHE DOCUMENTAIRE ET DE GESTION POUR UN FICHIER CENTRAL UTILISANT LES PROGRAMMES MACHINES MNGFLE et SRHFLE

par

F. J. Kelly*

RÉSUMÉ

L'auteur a développé deux programmes machines dans le but de gérer et de rechercher de l'information provenant des fichiers de document de référence ou des fichiers centraux. L'auteur a montré que ces programmes pouvaient accomplir ces tâches avec fiabilité et étaient économiques du point de vue du fonctionnement. Ce rapport décrit le développement de ces deux programmes machines et fournit des enseignements et des instructions détaillés sur leur utilisation.

*Chercheur scientifique, Section du traitement des minerais, Division de la métallurgie extractive, Direction des mines, ministère de l'Énergie, des Mines et des Ressources, Ottawa, Canada.

INTRODUCTION

Every research organization is all too familiar with the problems created by the ever-increasing flood of technical literature. Not only is it impossible to read every article that might be pertinent to a given project, but it is a time consuming task to compile the list of documents from literature reference files.

Within the Extraction Metallurgy Division, there are several document reference files or data bases. The information stored in these systems is cross-indexed in accordance with a coding scheme. Manually scanning the index cards for selected groups of codewords or keywords is one method to retrieve desired information. Another is by passing a needle through selected holes punched in the perimeter of the index cards. The deck of cards is lifted and the sought-after references are found on those cards falling free of the needle. Both of these methods are time consuming and cumbersome to execute and become even more so as the volume of the data base increases.

A speed-up of the retrieval process was accomplished by placing the active portion of the data base on a time-sharing computer service. The utility program supplied by this service searched the data base for requested codeword combinations and returned the index numbers of those cards having the desired information. The program was inexpensive to operate and reliable. The installation of a Control Data Corp. 6400 computer system in the Department forced the cancellation of all outside computer contracts, and access to this program was terminated.

The information retrieval job was transferred to the departmental computer system. A system utility program, MARS, was available for this service. After several months of testing and use it was found that this program failed to retrieve many index numbers that bore the requested codeword combinations from the data base. Also, the operating charges for these jobs had increased to five times that of the previous service.

Because MARS was unreliable and costly to operate, programs MNGFLE and SRHFLE were developed to replace it. The functions performed by MNGFLE are as follows:

1) creation of the data base from source cards,
2) deletion or modification of existing records, and
3) insertion or addition of new records to the data base.

Program SRHFLE searches the data base for requested combinations of codewords and returns the card numbers of the records containing these combinations. More detailed explanations of these programs, nomenclature, FORTRAN IV listings, and job coding forms are given in Appendices 1 and 2.

DISCUSSION AND RECOMMENDATIONS

After development the programs were benchmarked on the CDC-6400 against the utility program MARS. The test jobs compared maximum computer memory requirements , central, peripheral and input/output processing times, file storage space on disc in terms of physical record units (PRUS), and the costs of data base creation, information retrieval, and disc storage.

The test data bases were created from the same source cards. Each of the 1,501 cards contained one complete record, i.e., index card number and codewords. There were between 2 and 16 words per record. Ten sets of request codeword combinations were selected from the source data base to be used in the retrieval test. The tasks, data base creation, and information retrieval were run through each program twice. The results given in Tables 1 and 2 are the calculated averages of each double run.

Inspection of the results (Tables 1 and 2) show that there are significant advantages to be gained from using programs MNGFLE and SRHFLE in place of MARS. On the data base creation task, MNGFLE reduces computer memory requirements by 35.7 %, file storage cost by 79.6 %, and the job operating cost by 63.1 %. On the information retrieval task, SRHFLE reduces computer memory requirement by 46.4 %, file storage cost by 79.6 %, and job operating cost by 95.2 %. These reductions mean that these programs will provide a turnaround service more quickly and more cheaply than does the utility program for the identical set of tasks.

The complete data base contains 6,671 records or 43,838 words. There are between 2 and 25 words per record. The cost of creating this data base with MNGFLE from source cards was $20.14. Running the data sets, used to search the test data bases, through the complete data base with SRHFLE cost $2.60. The projected costs of doing both jobs with MARS are $86.50 and $72.65 respectively. The cost of making a 1500-record update of the complete data base with MNGFLE was $18.34. As was expected, the cost of creating

TABLE: 1 COMPARING THE CREATION OF DATA BASE FILES WITH UTILITY PROGRAM MARS AND PROGRAM MNGFLE FROM 1501 SOURCE CARDS

COMPUTER: DEMR CDC-6400		PROGRAM	PROGRAM	REDUCTION
DATE RUN 03/04/73		MARS	MNGFLE	BY MNGFLE
				(%)
CORE STORAGE OCTAL WORDS		66500	43000	35.7
CORE STORAGE DECIMAL WORDS		28000	18000	35.7
CENTRAL PROCESSING TIME	(SEC)	79.4	14.4	81.8
PERIPHERAL PROCESSING TIME	(SEC)	250.0	69.8	72.1
INPUT/OUTPUT PROCESSING TIME	(SEC)	219.3	56.4	74.3
FILE STORAGE SPACE ON DISC	(PRUS)	752	153	79.6
COST OF CREATING FILE	($)	17.96	6.63	63.1
COST OF STORING FILE ON DISC	($/DAY)	1.50	0.305	79.6

TABLE: 2 COMPARING THE RETRIEVAL OF INFORMATION FROM DATA BASE FILES WITH UTILITY PROGRAM MARS AND PROGRAM SRHFLE

COMPUTER: DEMR CDC-6400		PROGRAM	PROGRAM	REDUCTION
DATE RUN 03/04/73		MARS	SRHFLE	BY SRHFLE
				(%)
CORE STORAGE OCTAL WORDS		66500	35000	46.4
CORE STORAGE DECIMAL WORDS		28000	15000	46.4
CENTRAL PROCESSING TIME	(SEC)	37.6	4.9	87.0
PERIPHERAL PROCESSING TIME	(SEC)	742.5	14.5	98.0
INPUT/OUTPUT PROCESSING TIME	(SEC)	578.1	5.3	99.1
FILE STORAGE SPACE ON DISC	(PRUS)	752	153	79.6
COST OF RETRIEVING INFORMATION	($)	15.07	0.72	95.2
COST OF STORING FILE ON DISC	($/DAY)	1.50	0.305	79.6

and retrieving information from the data base increases as the number of records increase.

Another factor checked during these tests was the ability of each program to produce the correct results. The test data bases created by MARS and MNGFLE were compared with the source data base for omissions. Both data bases proved to be void of errors. With each search of its data base MARS failed to retrieve the index numbers of two records having the requested codeword combinations. SRHFLE, on the other hand, correctly retrieved all possible index numbers each time. This test work has shown that SRHFLE is more relaible at the task of retrieving information from the data base than is the utility program MARS.

The reference files which these programs manage and from which information is retrieved, employ a numeric cross-reference system. However, punched card input to both programs is done under alphanumeric format specifications. Consequently the programs can be used to manage the research files that use the keyword cross-reference system.

The index numbers produced by the search program are used to manually locate desired documentation within the reference file. If a copy of the document file were available to the program, the system would be complete. With minor modification the program could then produce printed copies of the required information. This would speed-up the retrieval process and save man-hours.

The entire data base would be placed on two files. The first file contains the cross-reference records and the second file the document records. Both of these files are linked together by the record numbers and maintained by MNGFLE. A request for information first initiates a search of file-1 by SRHFLE for the pertinent document record numbers. After all requests have been processed, SRHFLE locates each document in file- 2 and prints a copy of the record. Having the cross-reference file separate from the document file will minimize central and peripheral processing time and computer memory requirement.

SUMMARY

Programs MNGFLE and SRHFLE were designed to manage and retrieve information from cross-indexed file systems. These FORTRAN IV programs operate on the Department's CDC-6400 computer system. Besides providing good turnaround service they have proven to be reliable and economic in operation. Unlike system programs, modification control of these programs remains in the hands of the user. For identical service, these programs have reduced the operating charges and computer memory requirement attributable to the utility program MARS; MNGFLE by 63 % and 35 %, and SRHFLE by 95 % and 46 %.

APPENDIX 1: PROGRAM MNGFLE

The appendix is divided into four parts: explanation, nomenclature, listing, and coding form. The program, written in FORTRAN IV, consists of a main driving routine and six subroutines. The function of these routines are as follows; MNGFLE either creates or updates a data base or file, XFN inserts, modifies and deletes records from the file, FOSFN transfers a record from the old file to the new file, FOBFN transfers one logical record from the old file to the new file (a logical record is a consecutive series of individual records), RTAPE reads logical records from a file located on a peripheral storage device, WTAPE writes logical records to a file located on a peripheral storage device and RCARD reads new records from source cards.

In addition, the program uses four system routines: FLOAT converts fixed-point values to floating-point values; IFIX converts floating-point values to fixed-point values; EOF and ENDFIL are end of file test and marker, respectively. The function of all variables used in the program is given in the nomenclature. There are only four floating-point variables in the program A, BPI, BPW, and TLR. All other variables are integer type,either fixed-point or alphanumeric. Card input to the program is done under alphanumeric A-type format. All data written on, or read from files located on peripheral devices is in unformatted binary form.

Records read from cards can be of variable length. The minimum number of words per record is 2 and the maximum is 99. Under the present format (RCARD, 1 FORMAT),1 to 5 character alphanumeric words are punched in five-column fields. An eightcolumn card will hold sixteen words. The first word of the record must contain the source file identification number of the record. The record number can be alphanumeric and is always punched in columns 1 to 5 of the first card. If the number of cross-reference words exceeds 15, the remainder are punched on succeeding cards, but columns 1 to 5 on those cards are left blank. Examples of records (cards 2.1 to 9.1) coded for punching are shown on the coding forms.

On the first entry into RCARD, a card is read and the data is stored in array Y. A check is made to determine if an end of file mark has been encountered. If it has, control returns to the main program. If not, each non-blank word in Y is counted and packed into the array X starting at the second location. A second card is then read into Y. If an end of file is not encountered,the first word in Y is tested to determine if it is a blank word. If it is blank,it indicates a record continuation and the non-blank words are counted and added to X. If the first word in Y is not blank, it indicates the start of a new record and the data is held in Y. The number of non-blank words in the previous record is stored in the first location of X and control returns to the main program so that processing of the record can continue. Note that RCARD adds one word to each record.
On succeeding entries into RCARD, the information stored in Y is processed first and the above cycle repeats itself until an end of file mark is encountered.

The processing of records in MNGFLE is done on the record number. The program requires that the records read from cards be sequenced in ascending order. The numbering system employed is alphanumeric and has five units per number. This system contains the twenty-six letters of the alphabet and the digits zero to nine for a total of thirty-six units. The system can reference 11,881,375 records. In the CDC-6400 system, unit, A<B< ----Z<0<1< ----9. Leading and/or embedded blanks are not permitted in a record number. However, all characters including blanks can be used anywhere in any other record word (see coding form for example).

Records processed by MNGFLE are stored consecutively in the array FN. Every time a record is entered into FN, the record word count plus one is added to the total number of words contained in FN. Each time subroutines XFN, FOSFN and FOBFN are called,they check the FN word count plus that of the next record,against the value 2045. If the combined total is less than or equal to 2045 the record is transferred to FN,and if greater,WTAPE is called. WTAPE writes one logical record, i.e., the number of words in FN, the number of words in the last record entered,and the contents of FN, to a file located on a peripheral storage device. After incrementing the logical record counter and initializing the FN word counter, control returns to the calling routine. RTAPE reads logical records created by MNGFLE from peripheral storage files in the same manner. This cycle repeats itself until all records have been processed.

The manipulation of records occurs between statements 15 and 35 in the MNGFLE routine. The manner in which the records are processed depends on a series of decision tests. If the test at statement 31 of the listing is satisfied, control goes to 16 and new records are added to the end of the file. If the next test after 31 is satisfied, control goes to 21 and a part or an entire logical record is transferred from the old to the new file. If the test at statement 33 is not satisfied, one record is transferred from the old to the new file. If the test at statement 34 is not satisfied, a record is either deleted or updated. If the test at statement 35 is not satisfied, a record is inserted into the file. Processing continues until the test at statement 20 is satisfied and control passes to statement 50. The output section produces the final version of the file. The number of logical records contained in the file is written on the new file and the contents of the working file are copied to this file. A printed record of the files content, along with relevant file statistics are printed and the program stops. The new version of the file is copied on to magnetic tape or a private disc pack for storage purposes.

An example of a data deck coded for punching is given on the coding form. Card (1.1) is the first card in all data decks. The value of the variable LRO (card 1.1, col 10) is used as a control switch at the start of the program. If the value is 0, a new file is created and all record input will be from source cards. If the value is 1, records are processed from an accessible storage file and updated in accordance with the source cards. This would be the case for the data shown on the coding form. The record numbers (cards 2.1 to 9.1, columns 1 to 5) as noted previously appear in ascending order. Also columns 1 to 5 on the continuation cards (4.2, 6.2, 6.3) of records 4 and 6 are left blank. Cards containing only a record number (e.g. 3.1 and 7.1) will cause those records to be deleted from the file. The other records will update a record, be inserted between records, or be added to the end of the file. The last card in the data deck is the end-of-file card which consists of a 7-8-9 multi-punched in Column 1,and 1 and 5 punched in columns 2 and 3 respectively.

Before operating the program,make a visual check of the data deck. If the deck contains more than a hundred cards, request an off-line listing from the terminal operator. Make sure that the record numbers are in ascending order and correct all punching errors. The program produces a complete printed record of the file each time it is run. The cost of printing this record is approximately 62 % of the total operating charge. The printing section can be controlled by the user by making the following changes in the MNGFLE routine:

1. Change statement READ (CR, 2) LRO
   to READ (CR, 2) LRO,NP

2. After statement ENDFIL TC
   insert statement IF (NP .LE. ZR) TO TO 70

3. After statement 60 CONTINUE
   change statement REWIND TC to 70 REWIND TC

The value of NP can be punched in column 15 of card 1.1 on the coding form without a change to format statement 2. If NP = 0, printing does not occur. If NP = 1, a printed record of the file is produced. The produced listing gives the record count, the octal display number of the reference file card index number, the number of computer words per record, the reference file card index number, and the codewords or keywords associated with each record in the file.

NOMENCLATURE

Variable — Function

A — Stores the estimate of the number of lines printed per record. Used in the regulation of the line printer paging control.

.AND. — Logical operator meaning conjunction.

BPI — Constant = 800, the number of bits stored per inch of magnetic tape.

BPW — Constant = 60, the number of bits in a CDC-6400 computer word.

CR — Constant = 1, the logical unit number that references the card reader.

.EQ. — Relational operator meaning equal to (=).

FN — Array used to store logical records processed on the new file.

FO — Array used to store logical records read from the old file.

.GE. — Relational operator meaning greater than or equal to (≥).

.GT. — Relational operator meaning greater than (>).

I — Index counter used during a DO loop execution.

IC — Counter used to count the number of lines printed. Every time the value of IC = 59 it is reset to 1 and printing starts at the top of the next page.

IN — Index counter for the array FN. Within routine WTAPE, IN equals the number of words in a logical record. Before returning from WTAPE, the value of IN is set to zero.

Variable	Function
IO	Index counter for the array FO. The value of this counter is set to zero each time routine RTAPE is called.
ISW	A control switch that is used to signal the end of data input via the card reader. The value of this variable is set to one in routines XFN and RCARD. If ISW = 0, input from cards continues and, if ISW = 1, it is completed.
ITS	A control switch that controls the reading and processing of records in routine RCARD. If ITS = 0, data is read from a card. If ITS > 0, data stored in array Y is processed before reading more data from cards. After each read at statement 5, the value of ITS is set to 1. If an end of file is encountered, the value of ITS is set to 2. This delays the setting of ISW until the record is processed by routine MNGFLE with a call to routine XFN. The testing of the data in Y to signal the end of a record has been outlined in a previous section.
IX	Index counter for the array X in routine XFN. Used as a counter in routines FOSFN and FOBFN. The value of IX is set to 1 on entry to each of these routines.
J	Index counter used during a DO loop execution.
K	Counts the number of cards read by routine RCARD during program execution.
KK	Counts the number of records contained in the new file.

Variable — Function

L — Counts the total number of words contained in the new file. Included in the word count are the number of words per record plus those added in routines RCARD (1 per record), WTAPE (2 per logical record) and MNGFLE (1 per file).

.LE. — Relational operator meaning less than or equal to (≤).

LP — Constant = 3, the logical unit number that references the line printer.

LRN — Counts the number of logical records written on the new file. The value of LRN is incremented by one each time routine WTAPE is called. The final value of LRN is the first word written on the new file TC.

LRO — Initial value is equal to the number of logical records contained on the old file TA. The value of LRO is decreased by one each time routine RTAPE is called.

.LT. — Relational operator menaing less than (<).

.NE. — Relational operator meaning not equal to (≠).

NF — Counter used in routine RCARD to total the number of non-blank words in a record.

NFC — Constant = 16, the number of five-column fields in an eighty-column card. Used to control the read DO loops in routine RCARD. Also used in routine MNGFLE to calculate the number of printed lines per record.

NFT — Equals the number of words per logical record in routine FOBFN and per record in routines FOSFN and XFN.

Variable Function

NLR The number of words contained in the last record of a logical record. The value of NLR is set in routines FOBFN, FOSFN, and XFN. Also the value of the second word of a logical record transferred to the new file each time routine WTAPE is called.

NR Same meaning as NLR. Used in routine RTAPE to calculate the location of the record number of the last record in array FO.

NT The value calculated in routine RTAPE to give the location of the last record number in array FO. Used in routine MNGFLE to determine if a record being processed is to be included within the records stored in FO.

NW The number of words in a logical record that will be stored in the array FO. It is used in the calculation of NT in routine RTAPE. Also used in routines MNGFLE and FOBFN to determine if all records stored in FO have been processed.

NWN Constant = 2045, the maximum number of words that can be stored in the array FN. If it is desired to alter the length of a logical record, the value of this constant along with the dimensions of arrays FN and FO must be changed.

ON Constant = 1.

Variable	Function
TA	Constant = 4, the logical unit number that references the peripheral storage file from which logical records are transfered into the array FO.
TB	Constant = 5, the logical unit number that references a scratch peripheral storage file.
TC	Constant = 6, the logical unit number that references the peripheral storage file on which the new file is stored.
TES	Alphanumeric constant = 5 blanks, used in routine RCARD to test for blank words.
TLR	The approximate footage of magnetic tape used to store the data base.
TO	Constant = 2.
X	Array used to store recrods processed in routine RCARD.
Y	Array used to store data read via the card reader in routine RCARD.
ZR	Constant = 0.

```
      PROGRAM MNGFLE (INPUT,OUTPUT,TAPE1=INPUT,TAPE3=OUTPUT,TAPE4,
     1                TAPE5,TAPE6)
C
C                                 PROGRAM MNGFLE
C
C     PROGRAMED BY F.J.KELLY  EXT MET DIV  MINES BRANCH  E M AND R
C
      INTEGER CR,FN,FO,ON,TA,TB,TC,TES,TO,X,Y,ZR
C
      COMMON/ALL/CR,IN,IO,ISW,ITS,K,LRO,LRN,NFC,NFT,NT,NW,NLR,NWN,
     1           ON,TA,TB,TES,TO,ZR
      COMMON FN(2048),FO(2048),X(100),Y(20)
C
      DATA CR,LP,TA,TB,TC,TES/1,3,4,5,6,5H     /
      DATA ZR,ON,TO,NFC,NWN,BPI,BPW/0,1,2,16,2045,800.,60./
C
      REWIND TA
      REWIND TB
      REWIND TC
      BPI = BPW / BPI
      IC = ZR
      K = L = LRN = ISW = JC = JT = KK = IN = ITS = NFT = ZR
      IO = NW = NWN
      WRITE (LP,1)
      READ (CR,2) LRO
      IF (LRO .GT. ZR) GO TO 19
15    CALL RCARD
      IF (ISW .GT. ZR) GO TO 50
16    CALL XFN
      GO TO 15
19    READ (TA) LRO
20    IF (IO .GE. NW .AND. LRO .LE. ZR .AND. ISW .GT. ZR) GO TO 50
      IF (ISW .LE. ZR) GO TO 30
      IF (IO .GE. NW .AND. LRO .GT. ZR) CALL RTAPE
21    IF (IO .NE. ZR) GO TO 25
      CALL FOBFN
      GO TO 20
25    CALL FOSFN
      IF (IO .GE. NW) GO TO 20
      GO TO 25
30    IF (IO .GE. NW .AND. LRO .GT. ZR) CALL RTAPE
      IF (JC .GT. ZR) GO TO 31
      CALL RCARD
      JC = ON
      IF (ISW .GT. ZR) GO TO 20
31    IF (IO .GE. NW .AND. LRO .LE. ZR) GO TO 16
      IF (LRO .GT. ZR .AND. X(TO) .GT. FO(NT)) GO TO 21
      IF (JT .GT. ZR) GO TO 35
33    IF (FO(IO+TO) .GE. X(TO)) GO TO 34
      CALL FOSFN
      JT = ZR
      GO TO 30
34    IF (FO(IO+TO) .NE. X(TO)) GO TO 35
      CALL XFN
      IO = IO + FO(IO+ON) + ON
      JC = ZR
      JT = ON
      GO TO 30
35    IF (X(TO) .GE. FO(IO+TO)) GO TO 33
      CALL XFN
      JC = ZR
      GO TO 30
C
C     OUTPUT SECTION
C
50    IF (IN .EQ. ZR) GO TO 51
      CALL WTAPE
51    REWIND TA
      REWIND TB
      WRITE (TC) LRN
      L = L + ON
      DO 52 I = ON,LRN
      READ (TB) NW,NLR,(FN(J),J = ON,NW)
52    WRITE (TC) NW,NLR,(FN(J),J = ON,NW)
      ENDFIL TC
      REWIND TC
      READ (TC) LRN
      DO 60 I = ON,LRN
      READ (TC) NW,NLR,(FN(J),J = ON,NW)
      IN = TO
      IO = FN(ON) + ON
54    KK = KK + ON
      IC = IC + ON
      IF (IC .LT. 59) GO TO 56
      IC = ON
      WRITE (LP,1)
56    WRITE (LP,3) KK,FN(IN),FN(IN-ON),(FN(J),J = IN,ON)
      A = FLOAT(FN(IN-ON)) / NFC
      IC = IC + IFIX(A + 0.99) - ON
      IF (A.EQ.1..OR.A.EQ.2..OR.A.EQ.3..OR.A.EQ.4..OR.A.EQ.5..OR.
     1    A.EQ.6..OR.A.EQ.7.) IC = IC + ON
      IF (IO .GE. NW) GO TO 60
      IN = IN + FN(IN-ON) + ON
      IO = IO + FN(IO+ON) + ON
      GO TO 54
60    CONTINUE
      REWIND TC
      TLR = (L * BPI + 3.0 * LRN + 0.75) / 12.0 + 2.0
      WRITE (LP,4) K,L,LRN,TLR
      WRITE (LP,5)
      STOP
C
C     FORMAT SECTION
C
1     FORMAT (1H1/)
2     FORMAT (5X,5I5)
3     FORMAT (5X,I5,1X,O10,I5,16(1X,A5)/26X,16(1X,A5)/26X,16(1X,A5)/
     1 26X,16(1X,A5)/26X,16(1X,A5)/26X,16(1X,A5)/26X,16(1X,A5))
4     FORMAT (1H1///5X,'NO OF CARDS READ=',20X,I7//5X,
     1 'NO OF WORDS WRITTEN ON TAPE= 'I15//5X,
     2  'NO OF LOGICAL RECORDS WRITTEN= '6X,I7//5X,
     3  'APPROXIMATE LENGTH OF TAPE RECORD=',F10.1,'(FT)')
5     FORMAT (1H1/////' PROCESSING COMPLETED')
      END


      SUBROUTINE XFN
C
      INTEGER CR,FN,FO,ON,TA,TB,TES,TO,X,Y,ZR
C
      COMMON/ALL/CR,IN,IO,ISW,ITS,K,L,LRO,LRN,NFC,NFT,NT,NW,NLR,NWN,
     1           ON,TA,TB,TES,TO,ZR
      COMMON FN(2048),FO(2048),X(100),Y(20)
C
      IF (ITS .EQ. TO) ISW = ON
      IF (X(ON) .EQ. ON) RETURN
      NFT = X(ON) + ON
      IF ((IN + NFT) .GE. NWN) CALL WTAPE
      IX = ON
5     IN = IN + ON
      FN(IN) = X(IX)
      L = L + ON
      IF (IX .GE. NFT) GO TO 9
      IX = IX + ON
      GO TO 5
9     IX = IN - NFT + ON
      NLR = FN(IX)
      RETURN
      END


      SUBROUTINE FOSFN
C
      INTEGER CR,FN,FO,ON,TA,TB,TES,TO,X,Y,ZR
C
      COMMON/ALL/CR,IN,IO,ISW,ITS,K,L,LRO,LRN,NFC,NFT,NT,NW,NLR,NWN,
     1           ON,TA,TB,TES,TO,ZR
      COMMON FN(2048),FO(2048),X(100),Y(20)
C
      NFT = FO(IO+ON) + ON
      IF ((IN + NFT) .GE. NWN) CALL WTAPE
      IX = ON
5     IN = IN + ON
      IO = IO + ON
      FN(IN) = FO(IO)
      L = L + ON
      IF (IX .GE. NFT) GO TO 9
      IX = IX + ON
      GO TO 5
9     IX = IN - NFT + ON
      NLR = FN(IX)
      RETURN
      END
      SUBROUTINE FOBFN
C
      INTEGER CR,FN,FO,ON,TA,TB,TES,TO,X,Y,ZR
C
      COMMON/ALL/CR,IN,IO,ISW,ITS,K,L,LRO,LRN,NFC,NFT,NT,NW,NLR,NWN,
     1           ON,TA,TB,TES,TO,ZR
      COMMON FN(2048),FO(2048),X(100),Y(20)
C
5     IX = ON
      NFT = FO(IO+ON) + ON
      IF ((IN + NFT) .GE. NWN) CALL WTAPE
6     IN = IN + ON
      IO = IO + ON
      FN(IN) = FO(IO)
      L = L + ON
      IF (IO .GE. NW) RETURN
      IF (IX .EQ. NFT) GO TO 9
      IX = IX + ON
      GO TO 6
9     IX = IN - NFT + ON
      NLR = FN(IX)
      GO TO 5
      END


      SUBROUTINE RTAPE
C
      INTEGER CR,FN,FO,ON,TA,TB,TES,TO,X,Y,ZR
C
      COMMON/ALL/CR,IN,IO,ISW,ITS,K,L,LRO,LRN,NFC,NFT,NT,NW,NLR,NWN,
     1           ON,TA,TB,TES,TO,ZR
      COMMON FN(2048),FO(2048),X(100),Y(20)
C
      READ (TA) NW,NR,(FO(I),I = ON,NW)
      NT = NW - NR + ON
      LRO = LRO - ON
      IO = ZR
      RETURN
      END


      SUBROUTINE WTAPE
C
      INTEGER CR,FN,FO,ON,TA,TB,TES,TO,X,Y,ZR
C
      COMMON/ALL/CR,IN,IO,ISW,ITS,K,L,LRO,LRN,NFC,NFT,NT,NW,NLR,NWN,
     1           ON,TA,TB,TES,TO,ZR
      COMMON FN(2048),FO(2048),X(100),Y(20)
C
      WRITE (TB) IN,NLR,(FN(J),J = ON,IN)
      L = L + TO
      LRN = LRN + ON
      IN = ZR
      RETURN
      END
      SUBROUTINE RCARD
C
      INTEGER CR,FN,FO,ON,TA,TB,TES,TO,X,Y,ZR
C
      COMMON/ALL/OR,IN,IO,ISW,ITS,K,L,LRO,LRN,NFC,NFT,NT,NW,NLR,NWN,
     1           ON,TA,TB,TES,TO,ZR
      COMMON FN(2048),FO(2048),X(100),Y(20)
C
      IF (ISW .GT. ZR) GO TO 20
      IF (ITS .GT. ZR) GO TO 6
      READ (CR,1) (Y(J),J = ON,NFC)
      ITS = ZR
      IF (EOF(1) .NE. ZR) ISW = ON
      IF (ISW .GT. ZR) GO TO 20
      K = K + ON
      GO TO 7
5     READ (CR,1) (Y(J),J = ON,NFC)
      ITS = ON
      IF (EOF(1) .NE. ZR) GO TO 15
      K = K + ON
      IF (Y(ON) .NE. TES) GO TO 10
      GO TO 8
6     ITS = ZR
7     NF = ZR
8     DO 9 I = ON,NFC
      IF (Y(I) .EQ. TES) GO TO 9
      NF = NF + ON
      X(NF+ON) = Y(I)
9     CONTINUE
      GO TO 5
10    X(ON) = NF
      RETURN
15    ITS = TO
      X(ON) = NF
20    RETURN
C
C     FORMAT SECTION
C
1     FORMAT (16A5)
      END
```

MNGFLE: CODED DATA DECK

Date ____________

Name ____________ Page ______ of ______

```
1.1           1
2.1  AAAAA    7  200   35 4750  7531259     3  670  CUØ  5AY  651H2SØ4   10
3.1  A0BYX
4.1  B1L4R  245    1 5250   10  640   63   90  400   29 6357   18   30   11 752    48
4.2           3  120   85  200  FEØ   39
5.1  ZØ12A   63 5250  HCL
6.1  000AZ    1    2    3    4    5    6    7    8    9   10    A    B    C   DD    E
6.2           F    G    H    I    J    K    L    M    N    Ø    P    Q    R    S    T
6.3           U    V    W    X    Y    Z 4500
7.1  00001
8.1  00150  100  Ø25   CU  MGØ  98016543   10
9.1  99999  100  200 1157   16   10
10.1 7515
```

O = Ø Z = Ƶ ZERO = 0

APPENDIX 2: PROGRAM SRHFLE

This appendix is divided into four parts: explanation, nomenclature, listing, and coding form. The program, written in FORTRAN IV, is a stand-alone package that uses two system routines. FLOAT converts fixed-point values to floating-point values, and IFIX converts floating-point values to fixed-point values. All variables are integer type either fixed-point or alphanumeric. Card input into the program is done under alphanumeric (A) and integer (I) format. All input from peripheral storage files is in unformatted binary form.

The function of SRHFLE is to search the data base produced by MNGFLE for selected combinations of codewords or keywords and return the card numbers of the records on which those combinations appear. The program consists of three parts: the requested codeword combination input section (statements 10-15), file search and information retrieval section (statements 15-30) and a section that provides a printed record of the search results (statements 30-95).

File search request sets are read in from cards. A data set consists of a control header card and from 1 to 10 sets of requested codeword combinations. Cards 1.00 to 1.30 and 2.00 to 2.50 of the coding form are examples of data sets coded for keypunching. Cards 1.00and 2.00 of each set are control cards. The functions of the variables on the control card are given in the nomenclature. For this data base file, codewords are punched in five-column fields, right justified, 16 words per card. Cards 1.10 to 1.30 of set 1 and 2.10 to

2.50 of set 2 are codeword combination sets. The number of data sets that can be included in one data deck is limited only by the time policy of the computer facility and/or the user's budget. The last data set in the deck is followed by a processing termination card (Card 3.00 on the coding form) that has the word END punched in column 1 to 3.

A data set in which the number of codewords per request is less than or equal to sixteen will be read in at statement 11. If any request in a data set contains more than sixteen codewords, then the number of words in each combination set is punched, right justified, in columns 1 to 5 of the first card in each set. The first card of each request is then read in at statement 12, and the continuation cards at statement 11. Cards 1.00 to 1.30 on the coding form show an example of a data set in which each request contains fewer than sixteen words. Cards 2.00 to 2.50 constitute a data set in which a request (Cards 2.20 - 2.21) contains more than sixteen words. The maximum number of codewords per request set is forty-five.

The program begins a search by reading a control card at statement 10 of the listing. If the logical test at statement 10+1 is satisfied, control is transfered to statement 99 and processing is terminated. If not, a search of the data base is made. Each request combination set is read at statements 11 or 12 and stored one request per row in the multi-dimensioned array RS. After all the requests in the data set have been entered, the non-blank words contained in each request are counted and stored in the array NCF. Control then passes to the search section at statement 15.

The search section first reads the number of logical records contained in the file TA and stores this value in the counter LRB. If the logical test at statement 18 is satisfied, all records in the file have been searched and control is transferred to the output section at statement 90. If not, a logical record and the number of words contained in it are read from the file TA at statement 18+1. The logical record is stored in the array XX. Each logical record contains a series of individual records. The logical record counter is decreased by one each time a record is read from the file.

The program next takes the number of words contained in the first record stored in XX, subtracts one, and compares the value with the number of words contained in each request set at statement 20+2. If the number of words in the record is greater than or equal to the number of words in any request set, control transfers to statement 24. If not, the word counter is incremented and compared with the number of words (NW) contained in the logical record. If it is greater than or equal to NW, control transfers to statement 18. If not, control transfers to statement 20 and the next record in the series is processed in the same manner.

At statement 24, the record is transferred from the array XX to the working array X. The same testing procedure outlined above is done for each request set at statement 25+4. If the number of words in the record minus one is less than the number of words in a request, the record is not searched and control transfers to statement 30. Otherwise each word in the record is compared with each word in the request set at statement 25+9. Every time a word in the record is found equal to a word in the request set, a counter RJ is incremented at statement 25+10 and control transfers to statement 27.

After all the words in the request set have been compared, the value of RJ is compared to the number of words in the request set at statement 27+1. If they are not equal, control transfers to statement 30. If they are equal, the counter,which totals the numbers of records located containing the requested codeword combination, is incremented by one and stored in the array CHJ at statement 27+2. The value of this counter is tested at statement 27+3. If it is less than or equal to NER, the catalog number of the record is stored in the multi-dimensioned array CN at statement 27+5 and control transfers to statement 30. If it is greater than NER, it indicates that the row vector in array CN is filled, so control transfers to statement 28. At statement 28, the records located counter for that request set is set to one. The contents of arrays RS and CN for that request set are printed and the record number that has been located is then stored in CN. At statement 30, the program transfers control to either statement 18 or 20. The cycle repeats itself until all records in the file have been processed at which time control is transferred to statement 90.

Between statements 90 and 95 the contents of arrays RS and CN are printed. The number of a request set is given along with a record of the requested codewords at statement 91+1. Next a printed record of all the catalog numbers associated with this request set is printed at statement 94+1. After all the information has been printed, control returns to statement 10. If this test indicates that all data sets have been processed, control is transferred to statement 99 and processing stops.

SRHFLE was designed to compare all of the requests in a data set with each record before searching the next record. This minimizes peripheral processing time because the program reads the data base file from peripheral storage into computer memory once per data set of 1 to 10 requests. Central processing time is also minimized because the program searches only those records whose codeword counts are equal to or greater than the word count of a request set.

NOMENCLATURE

Variable	Function
CN	Array used to store the catalog numbers of records containing the information for which the file is being searched.
CNJ	Array used to store the number of located records associated with each request.
CR	Constant = 1, the logical unit number that references the card reader.
.EQ.	Relational operator meaning equal to (=).
FIN	Alphanumeric constant = ENDbb (b = Blank).
.GE.	Relational operator meaning greater than or equal to (≥).
.GT.	Relational operator meaning greater than (>).
I	Index counter used during DO loop executions.
IC	Counter used to count the number of non-blank words in each request set.
II	Row storage location indicator for the array RS in the data input section.
IO	Index counter for the array XX. The value of this counter is set to zero each time a logical record is read from the file TA.
J	Index counter used during a DO loop execution. Also used to store other values during the execution of the program.

Variable	Function
JOB	Aphnumeric control card variable punched in columns 1 - 5. If JOB = FIN processing is terminated (See card 3.00 on coding form).
K	Array index location indicator. Also used to store other values during program execution.
KK	Index counter used during a DO loop execution.
KR	Equals the number of codewords contained in a request and is used during a record search.
L	Control card variable punched in column 15. If L = 0 all requests in a data set contain 16 or less codewords. If L > 0 one or more requests contain more than 16 codewords .
.LE.	Relational operator meaning less than or equal to (≤).
LP	Constant = 3, the logical unit number that references the line printer.
LRB	The number of logical records contained in file TA.
.LT.	Relational operator meaning less than (<).
M	Counts the number of request sets processed by the program.
MC	Counts the number of lines printed. Every time the value of MC = 59 it is reset to 0 and printing starts at the top of the next page.
N	Row index counter for the array RS.
NC	The number of words contained in a data file record.
NCF	Array used to store the number of non-blank words contained in each request.

Variable	Function
.NE.	Relational operator meaning not equal to (≠).
NER	Constant = 150, equals the number of values that can be stored in each row of the array CN.
NFR	Constant = 16, the number of five-column fields in an eighty-column card.
NJ	Intermediate storage for the number of requests contained in one data set.
NN	Index counter used during a DO loop execution.
NONE	Alphanumeric constant = bNONE; the value of this constant is printed whenever a search of the data base file does not yield a catalog number containing the requested codeword combination.
NOR	Control card variable columns 9-10, right justified, its value is set equal to the number of codeword combination requests included in a data set. Can have any value ranging from 1 to 10.
NRF	If L = 0, NRF is set equal to NFC. If L is greater than 0, NRF is set equal to the number of codewords contained in a request. This value is punched in columns 1-5, right justified, on the first card of each request included in the data set (See data set 2.00 to 2.50 on coding form)
NW	The number of words in a logical record. Used to determine if all records in a logical record have been processed.
ON	Constant = 1.

Variable	Function
RJ	The value of this counter increases by one each time a record word is found equal to a request codeword. After all record and request words have been compared, the value of RJ is compared with the word count of the request for equality. If the values are equal a record has been located and its identification number is stored in the array CN. If not equal, RJ is set to zero and the next request is processed.
RS	Array used to store request information data. Each request set is stored in a row vector of RS. The maximum number of words per request is 45.
TA	Constant = 4, the logical unit number that references the peripheral storage file from which logical records are transferred into the array XX.
TES	Alphanumeric constant = 5 blanks, used to test for non-blank words in each request set.
TO	Constant = 2.
X	Array in which individual records are stored for processing.
XX	Array used to store logical records transferred from the file TA.
ZR	Constant = 0.

```
      PROGRAM SRHFLE (INPUT,OUTPUT,TAPE1=INPUT,TAPE3=OUTPUT,TAPE4)
C
C                          PROGRAM SRHFLE
C
C     PROGRAMED BY F.J.KELLY  EXT MET DIV  MINES BRANCH  E M AND R
C
      INTEGER CN,CNJ,CR,FIN,ON,RJ,RS,TA,TO,TES,X,XX,ZR
C
      COMMON XX(2048),CN(10,150),RS(10,45),X(100),CNJ(10),NCF(10)
C
      DATA CR,LP,TA,FIN,NONE/1,3,4,5HEND  ,5H NONE/
      DATA ZR,ON,TO,NFR,NER,TES/0,1,2,16,150,5H     /
C
      M = ZR
      II = ON
10    READ (CR,2) JOB,NOR,L
      IF (JOB .EQ. FIN) GO TO 99
      REWIND TA
      WRITE (LP,4)
      MC = ZR
      NJ = NOR
      NRF = NFR
      DO 15 I = ON,NJ
      IC = CNJ(I) = ZR
      IF (L .GT. ZR) GO TO 12
11    READ (CR,3) (RS(I,J),J = II,NRF)
      GO TO 13
12    READ (CR,1) NRF,(RS(I,J),J = II,15)
      IF (NRF .LT. NFR) GO TO 13
      II = NFR
      GO TO 11
13    II = ON
      DO 14 K = ON,NRF
      IF (RS(I,K) .EQ. TES) GO TO 14
      IC = IC + ON
14    CONTINUE
      NCF(I) = IC
15    CONTINUE
      READ (TA) LRB
18    IF (LRB .LE. ZR) GO TO 90
      READ (TA) NW,IO,(XX(J),J = ON,NW)
      IO = ZR
      LRB = LRB - ON
20    IO = IO + ON
      DO 21 I = ON,NJ
      IF (XX(IO) - ON .GE. NCF(I)) GO TO 24
21    CONTINUE
      IO = IO + XX(IO)
      IF (IO .GE. NW) GO TO 18
      GO TO 20
24    NC = XX(IO)
      DO 25 I = ON,NC
      IO = IO + ON
25    X(I) = XX(IO)
```

```
      DO 30 KK = ON,NJ
      K = KK
      RJ = ZR
      IF (NC - ON .LT. NCF(K)) GO TO 30
      KR = NCF(K)
      DO 27 NN = ON,KR
      N = NN
      DO 26 L = TO,NC
      IF (RS(K,N) .NE. X(L)) GO TO 26
      RJ = RJ + ON
      GO TO 27
26    CONTINUE
27    CONTINUE
      IF (RJ .NE. NCF(K)) GO TO 30
      CNJ(K) = CNJ(K) + ON
      IF (CNJ(K) .GT. NER) GO TO 28
      J = CNJ(K)
      CN(K,J) = X(ON)
      GO TO 30
28    CNJ(K) = ON
      I = K + M
      L = NCF(K)
      WRITE (LP,5) I,(RS(K,J),J = ON,L)
      WRITE (LP,6)
      WRITE (LP,7) (CN(K,J),J = ON,NER)
      WRITE (LP,8) I
      CN(K,ON) = X(ON)
      J = IFIX((FLOAT(NCF(K)) / 15.) + 0.99) + 15
      IF (MC + J .LT. 59) GO TO 29
      MC = ZR
      WRITE (LP,4)
29    MC = MC + J
30    CONTINUE
      IF (IO .GE. NW) GO TO 18
      GO TO 20
90    WRITE (LP,4)
      MC = ZR
      DO 95 I = ON,NJ
      M = M + ON
      K = NCF(I)
      L = CNJ(I)
      J = IFIX((FLOAT(K)/15.) + 0.99) +
     1    IFIX((FLOAT(L)/18.) + 0.99) + 4
      IF (MC + J .LT. 59) GO TO 91
      WRITE (LP,4)
      MC = ZR
91    MC = MC + J
      WRITE (LP,5) M,(RS(I,J),J = ON,K)
      K = CNJ(I)
      IF (K .GT. ZR) GO TO 94
      CN(I,ON) = NONE
      K = ON
94    WRITE (LP,6)
      WRITE (LP.7) (CN(I,J),J = ON,K)
95    CONTINUE
      GO TO 10
```

```
99    REWIND TA
      WRITE (LP,9)
      STOP
C
C     FORMAT SECTION
C
1     FORMAT (I5,15A5)
2     FORMAT (A5,5I5)
3     FORMAT (16A5)
4     FORMAT (1H1)
5     FORMAT (//' JOB',I3,' REQUESTED FIELDS;',15(2X,A5)/26X,
     1 15(2X,A5)/26X,15(2X,A5)/26X,15(2X,A5)/26X,15(2X,A5)/26X,
     2 15(2X,A5)/26X,15(2X,A5))
6     FORMAT (/' REFERENCE FILE CARD NUMBERS FOLLOW;')
7     FORMAT (5X,18(2X,A5)/5X,18(2X,A5)/5X,18(2X,A5)/5X,18(2X,A5)/
     1 5X,18(2X,A5)/5X,18(2X,A5)/5X,18(2X,A5)/5X,18(2X,A5)/
     2 5X,18(2X,A5))
8     FORMAT (/' OTHER CARDS HAVING THE ABOVE FIELD(S) FOR JOB',
     1 I3,' ARE LISTED BELOW')
9     FORMAT (1H1/////5X,'PROCESSING COMPLETED')
      END
```

SRHFLE: CODED JOB REQUEST DECK

Date ______________

Name ______________________ Page _____ of _____

```
1.00 JØB      3    0
1.10   200   10
1.20     H    A    V    E 4500
1.30 16543   CU  100


2.00 JØB      5    1
2.10     2  200   10
2.20    17    1    2    3    4    5    6    7    8    9    B    C    Ø    H    A    V
2.21     E 4500
2.30     316543   CU  100
2.40     8 6357    1  200 5250  752   63    3  FEØ
2.50     1  HCL
3.00 END
```

O = Ø   Z = Ƶ   ZERO = 0