UR1986_44

1986/44. CORLIBRARY - a core data index.

A. L. Telfer

Abstract

The CORLIBRARY program suite is designed to provide a record of the location of all diamond core, cuttings, auger samples and any other geological materials retained by the Tasmania Department of Mines. Facilities are provided for adding to and searching the data-base. A sample input form is included.

USING THE PROGRAMS

The programs are run on the Geological Survey Perkin-Elmer minicomputer.

Adding data

Data are recorded on a standard input form (Appendix 1) before entry. Data input is started by typing CORADD, then the data are entered from the standard input forms in response to the prompts of the program. Provision is made for up to ten different core locations per hole. Input of core locations per hole is terminated by entering a location of 0 (zero). The input session is terminated by entering 0 (zero) as the analysis number.

A proof sheet is printed in the computer room, and the data are saved in file CORTEMP.TMP. This file should be corrected using the editing facilities (see Appendix 2) before continuing.

The corrected temporary file CORTEMP.TMP is added to the end of the existing data base using the CORMERGE command. The data base should be sorted periodically using the CORSORT command to maintain the entries in numerical order and update the number index file (CORINDEX.IND).

Corrections to the main data base (CORIND.DAT) would normally be made by deleting entries using the system editor in hexadecimal mode and then putting the corrected entries in using CORADD.

Data base searching

The data base is searched by typing CORSRCH. An entry will be retrieved if each of the specified search options is found in the entry.

The following search options may be specified or left blank;

(1) REFERENCE NUMBER - one or more reference numbers on a single line, each terminated by a /. e.g. 3253/3254/3255/

(2) STORE LOCATION - one or more store locations, each terminated by a /.

(3) RACK NUMBER - one or more rack numbers, each terminated by a /.

(4) CORE TYPE - one or more core types, each terminated by a /.

PROGRAM FUNCTIONS

CORADD (Appendices 3, 4)

This program prompts the user for data and writes it to a temporary file CORTEMP.TMP. This file is corrected using the computers editing facilities (see Appendix 2 for the most useful commands).

CORMERGE (Appendices 5, 6)

The corrected data in CORTEMP.TMP is converted to hexadecimal format. The data base is copied to a temporary work file and the corrected data in CORTEMP.TMP is converted to hexadecimal format and added to the end of the work file. The combined file is then copied back to the original data base file.

CORSORT (Appendices 7, 8)

The data are sorted into ascending DOM sample number order and written to a temporary file in the new order. The temporary file is then copied back to the original data base file. A reference number index (CORINDEX.IND) is created so that the CORLIBRARY data base may be accessed from LOGSRCH also.

CORSRCH (Appendices 9, 10)

The program prompts the user with the available search options (see *Data-base searching* above), takes the values of the options selected, and searches each record for the occurrence of every selected option.

[18 July 1986]

APPENDIX 1

TASMANIA DEPARTMENT OF MINES CORE LIBRARY INDEX

CORLIB

REFERENCE NUMBER: _ _ _ _ _ (5)

Store Location* S	P	D	Q	Rack No.	Depths From	To	Units* m	ft	?	Type* d	c	a	o
1	2	3	4	_ _ _ _ _ (5)	_ _ _ _ (4)	_ _ _ _ (4)	1	2	3	1	2	3	4
				_ _ _ _ _	_ _ _ _	_ _ _ _							
				_ _ _ _ _	_ _ _ _	_ _ _ _							
				_ _ _ _ _	_ _ _ _	_ _ _ _							
				_ _ _ _ _	_ _ _ _	_ _ _ _							
				_ _ _ _ _	_ _ _ _	_ _ _ _							
				_ _ _ _ _	_ _ _ _	_ _ _ _							
				_ _ _ _ _	_ _ _ _	_ _ _ _							
				_ _ _ _ _	_ _ _ _	_ _ _ _							
				_ _ _ _ _	_ _ _ _	_ _ _ _							

* Tick one only S = Store; P = Plant, D = Domain; Q = Queenstown

APPENDIX 2

Editing Facilities

To examine and change a file, type 'EDIT file name' (in this case, 'filename' = CORTEMP.TMP).

Attached is a short list of the most useful editing commands. A full summary of commands is found in the 'EDIT User Guide' in the computer room.

ALTER COMMAND

The ALTER command modifies entire lines of text.

Format:

```
                    ( ␣          )          NOTE:
                    ( \          )          The characters used to
      (   cr     ) ( @          )          alter the line can be used
ALTER ( line no ) ( %          )          more than once on the
                    ( %/string/ )          same line.
                    ( x          )
```

cr — entering a carriage return immediately following the the ALTER command readies the current line to be altered.

line no — specifies the line number to be altered.

␣ — represents a space. Spaces have no effect on the current line. The cursor is moved by depressing the space bar.

\ — is the backslash character that replaces the current character with a space.

@ — is the at sign character that deletes the current character and compresses the rest of the line.

% — is the percent sign character that inserts a single space before the percent sign.

%/string/ — is the percent sign followed by a slash, a character string, and a slash, that inserts the character string before the percent sign.

x — indicates any character other than a blank, \, @, and %. The character replaces the current character in that position.

Note: This decription of the ALTER command applies only when in Edit's NOBIOC mode. In the BIOC mode the ALTER commands functions in a different way, using the CONTROL key to position the cursor and to operate the delete and insert functions. See the 'EDIT User Guide' for details.

Functional Details:

After the ALTER command is entered, Edit responds by printing the specified line as it currently exists. The line number is then printed again. Changes to the original line now can be entered in the corresponding character positions. The alter line can be terminated at any character position. All remaining characters in the line are retained.

A carriage return in the first character position will terminate ALTER.

As soon as 80 characters are typed on a terminal, the line automatically wraps around to the next line. To alter the second part of a line that exceeds 80 characters, space through the first part of the line in order to access the second part.

Examples:

```
>T1-3
    1   This text is part of a new document
    2   The first thing we must do is revise it
    3   This is an example of a long line that wraps
        around when it is displayed.

>AL 1
    1   This text is part of a new document
    1   >                          old
    1   This text is part of a old document
    1   >                         n
    1   This text is part of anold document
    1   >                          %
    1   This text is part of an old document
        >              %/one /
    1   This text is one part of an old document
    1   >

>AL 2
    2   The first thing we must do is revise it
    2   >                 too@@@@@
    2   The first thing toodo is revise it
    2   >                    \
    2   The first thing to do is revise it
    2   >

>AL 3
    3   This is an example of a long line that wraps
        around when it is displayed.
    3   >
        >                      print@@
    3   This is an example of a long line that wraps
        around when it is printed.
    3   >
```

DELETE COMMAND

The DELETE command deletes data from a file.

Format:

DELETE [(line no 1)][-(line no 2)]

Parameters:

line no 1 is the line number, or first line number of a range of lines to be deleted.

line no 2 is the ending line number of a range of data to be deleted.

Examples:

>DELETE 10-20

Delete lines 10 through 20.

>DELETE

Delete the current line.

DONE COMMAND

The DONE command saves the current file and ends the editing session.

Format:

DONE

SCREEN COMMAND

The SCREEN command displays a full screen of data starting, ending, or centred at the current line.

Format:

```
          ( E )
SCREEN    ( C )
          ( S )
```

Parameters:

E displays a full screen of data ending at the current line. If this parameter is omitted, the default is S.

C displays a full screen of data centered at the current line. If this parameter is omitted, the default is S.

S displays a full screen of data starting at the current line.

TOP COMMAND

The TOP command moves the current line pointer to the first line of the file.

Format:

```
TOP
```

Examples:

```
>TY 1-
    1     The TOP command
    2     moves the current line pointer
    3     to the first line of the file
>TOP
    1     The TOP command
```

TYPE COMMAND

The TYPE command displays lines of data to the list device.

Format:

```
TYPE  [(line no 1)][-(line no 2)]
```

Parameters:

line no 1 — is the line or first line number of a range of lines to be displayed to the list device.

line no 2 — is the ending line number of the range to be displayed to the list device.

Examples:

```
>TYPE 10-20
```

Display lines 10 through 20 of the current file.

```
>TYPE 8
```

Display line 8 of the current file.

```
>TYPE 20-
```

Display lines 20 to the last line of the current file.
file.

APPENDIX 3

```
*CORADD.CSS
*FOR RUNNING CORADD AND PRINTING A PROOF SHEET
$IFX CORADD.TMP; $WRITE  **CHECK LAST CORMERGE**; $EXIT; $ENDC
PRE ETM; AL CORTEMP.TMP,IN,162
*SET UP THE OUTPUT FILE
L CORADD,3; AS 6,CORTEMP.TMP; *LOAD PROGRAMME AND ASSIGN LU 6
ST
PRINT CORTEMP.TMP
$WRITE  FILE IS CORTEMP.TMP; ENA ETM; $EXIT
```

APPENDIX 4

```
$TITL CORADD.FTN - ADD CORE LOCATIONS TO THE CORE LOCATION INDEX
C           TAKES KEYBOARD CHARACTER INPUT AND PREPARES A PROOF SHEET
C
      CHARACTER*5 RACNO(10)
      CHARACTER*4 DEPFR(10),DEPTO(10)
      CHARACTER*1 STOLOC(10),UNIT(10),TYPE(10),SHEET
      INTEGER*2 REFNO
C
C           INPUT IS REFERENCE NUMBER, RACK NUMBER, CORE DEPTHS (FROM
C           AND TO), STORE LOCATION, UNITS OF LENGTH USED AND SAMPLE
C           TYPE. A MAXIMUM OF 10 RACKS OF CORE PER HOLE IS SET
C           ********ONLY 200 SHEETS CAN BE ENTERED IN ONE SESSION*****
C
      OPEN(UNIT=5,FILE='CON:')
C
C           OPEN THE CONSOLE FOR DATA INPUT. THE OUTPUT FILE HAS BEEN
C           SET ON LU 6 BY THE CSS FILE.
C
   20 WRITE(5,100)
  100 FORMAT(' REFERENCE NUMBER (TYPE *0* IF SESSION FINISHED)')
      READ(5,*) REFNO
      IF (REFNO .EQ. 0) GOTO 50
C
C           END OF BATCH SO END
C
      DO 99 J=1,10
       WRITE(5,101)
  101  FORMAT(' STORE LOCATION :TYPE *0* TO END SHEET')
       READ(5,201) STOLOC(J)
  201  FORMAT(A1)
       IF (STOLOC(J) .EQ. '0') GOTO 999
C
C           END OF SHEET SO WRITE TO OUTPUT FILE
C
       WRITE(5,102)
  102  FORMAT(' RACK NUMBER')
       READ(5,203) RACNO(J)
  203  FORMAT(A5)
       WRITE(5,103)
  103  FORMAT(' DEPTH: FROM...')
       READ(5,202) DEPFR(J)
  202  FORMAT(A4)
       WRITE(5,104)
  104  FORMAT(' DEPTH:...TO')
       READ(5,202) DEPTO(J)
       WRITE(5,105)
  105  FORMAT(' UNITS')
       READ(5,201) UNIT(J)
       WRITE(5,106)
  106  FORMAT(' DRILL TYPE')
       READ(5,201) TYPE(J)
   99 CONTINUE
C
C           END OF LOOP WHICH READS THE INPUT DATA
C           HAVING READ ONE DATA SHEET, WRITE TO THE DATA FILE
C
```

```
  999 WRITE(6,300) REFNO,(STOLOC(I),RACNO(I),
     .DEPFR(I),DEPTO(I),UNIT(I),TYPE(I),
     .I=1,J-1)
  300 FORMAT (I5,10(1X,A1,1X,A5,1X,A4,1X,A4,1X,A1,1X,A1,/5X)/)
C
C          WRITE DATA TO FILE WITH A LINE SPACE BETWEEN EACH ENTRY
C
       GOTO 20
C
C           BACK TO THE START FOR ANOTHER SHEET OF DATA
C
   50 CLOSE(UNIT=6, STATUS='KEEP')
C
C           OR END THE SESSION
C
       STOP
       END
```

APPENDIX 5

```
*CORMERGE.CSS
* FOR MERGING CORRECTED FILE WITH MAIN FILE
PRE ETM; XDE MRGTMP.TMP; *DELETE SCRATCH FILE
AL MRGTMP.TMP,IN,162; L CORMERGE,10; * AL SCRATCH FILE AND LOAD PROG
REP CORTEMP.TMP,FF00; AS 6,CORTEMP.TMP,ERO; AS 4,MRGTMP.TMP; ST
$IFNE 0; $WRITE TRANSLATE ERROR; ENA ETM;  $EXIT; $ENDC
REP MRGTMP.TMP,FF00
$BUILD COPY.CMD
IN CORIND.DAT
AL SYSF:TEMP.DAT,IN,162/6/5
OUT SYSF:TEMP.DAT
COPY *,*
IN MRGTMP.TMP
COPY *,*
END
$ENDB
L COPY32,50; ST ,COM=COPY.CMD,LI=NULL:,LO=NULL:
$IFNE 0; $WRITE COPY-MERGE ERROR; ENA ETM; $EXIT; $ENDC
REP CORIND.DAT,0; REP SYSF:TEMP.DAT,FF00
DE COPY.CMD,CORIND.DAT
$BUILD COPY.CMD
IN SYSF:TEMP.DAT
AL CORIND.DAT,IN,162/10/3
OUT CORIND.DAT
COPY *,*
REW I
REW O
VERIFY *,*
END
$ENDB
L COPY32,50; ST ,COM=COPY.CMD,LI=NULL:,LO=NULL:
$IFNE 0; $WRITE COPY BACK FAILED; ENA ETM; $EXIT; $ENDC
REP CORIND.DAT,FF00; REP SYSF:TEMP.DAT,0; REP CORTEMP.TMP,0
REP MRGTMP.TMP,0
DE COPY.CMD,SYSF:TEMP.DAT,MRGTMP.TMP,CORTEMP.TMP
ENA ETM
$EXIT
```

APPENDIX 6

```
$TITL CORMERGE.FTN - CONVERT PROOF FILE TO MASTER FILE FORMAT
C
      CHARACTER*4 DEPFR(10),DEPTO(10)
      CHARACTER*5 RACNO(10)
      CHARACTER*1 STOLOC(10),UNIT(10),TYPE(10)
      INTEGER*2 REFNO
C
C          DECLARATION OF INPUT AND OUTPUT VARIABLES
C
      CHARACTER*1 BUFF(162)
      EQUIVALENCE (BUFF(1),REFNO),
     .      (BUFF(3),STOLOC),(BUFF(13),RACNO),(BUFF(63),DEPFR)
     .      ,(BUFF(103),DEPTO),(BUFF(143),UNIT),(BUFF(153),TYPE)
C
C          A BINARY BUFFERED ARRAY FOR FASTER DATA-BASE ACCESS
C
      OPEN(UNIT=4,FORM='BINARY',RECL=162)
C
C          OPEN THE OUTPUT FILE FOR BINARY OPERATIONS
C
   10 I=1
C
C          READ CORTEMP.TMP (THE CORADD DATA FILE) AND REWRITE AS A
C          BINARY FILE.
C
      READ(6,300,END=29) REFNO,STOLOC(I),RACNO(I),DEPFR(I),DEPTO(I),
     .      UNIT(I),TYPE(I)
  300 FORMAT(I5,1X,A1,1X,A5,1X,A4,1X,A4,1X,A1,1X,A1)
   20 I=I+1
      READ(6,301,END=30) STOLOC(I),RACNO(I),DEPFR(I),DEPTO(I),
     .      UNIT(I),TYPE(I)
  301 FORMAT(6X,A1,1X,A5,1X,A4,1X,A4,1X,A1,1X,A1)
      IF (STOLOC(I) .EQ. ' ') GOTO 40
C
C          WHEN AN EMPTY LINE IS ENCOUNTERED THE REST OF THE ARRAY
C          IS PADDED WITH ZEROS, TO FACILITATE A QUICK SEARCH.
C
      GOTO 20
   40 DO 45 J=I,10
         STOLOC(J)='0'
         RACNO(J)='0'
         DEPFR(J)='0'
         DEPTO(J)='0'
         UNIT(J)='0'
         TYPE(J)='0'
   45 CONTINUE
C
C          READ NULL VALUES INTO THE REST OF THE ARRAY
C
      WRITE(4) BUFF
C
C          BINARY WRITE TO OUTPUT FILE
      GOTO 10
C
C          BACK AROUND
```

```
C
   29 I=I+1
C
C          PAD OUT THE LAST ARRAY WITH ZEROS.
C
   30 DO 55 J=I,10
         STOLOC(J)='0'
         RACNO(J)='0'
         DEPFR(J)='0'
         DEPTO(J)='0'
         UNIT(J)='0'
         TYPE(J)='0'
   55 CONTINUE
      WRITE(4) BUFF
C
C          WRITE TO THE BINARY FILE, CLOSE ALL FILES AND END.
C
      CLOSE(UNIT=4,STATUS='KEEP')
      CLOSE(UNIT=6,STATUS='KEEP')
      END
```

APPENDIX 7

```
*CORSORT.CSS
* FOR SORTING CORE LIBRARY FILES INTO REFERENCE NO. ORDER
PRE ETM
L CORSORT,10; AS 4,CORIND.DAT,ERO; XAL SYSF:TEMP.DAT,IN,162/5/2
AS 6,SYSF:TEMP.DAT; XAL CORINDEX.IND,IN,4000; AS 8,CORINDEX.IND
ST
$IFNE 0; $WRITE SORT ERROR; ENA ETM; $EXIT; $ENDC
$BUILD COPY.CMD
IN SYSF:TEMP.DAT
AL CORIND.DAT,IN,162/10/5
OUT CORIND.DAT
COPY *,*
REW I
REW O
VERIFY *,*
END
$ENDB
REP SYSF:TEMP.DAT,FF00; REP CORIND.DAT,0; DE CORIND.DAT
L COPY32,50; ST ,COM=COPY.CMD,LI=NULL:,LO=NULL:
$IFNE 0; $WRITE SORT-COPY ERROR; ENA ETM; $EXIT; $ENDC
REP CORIND.DAT,FF00; REP SYSF:TEMP.DAT,0; DE SYSF:TEMP.DAT,COPY.CMD
ENA ETM; $EXIT
```

APPENDIX 8

```
$TITL CORSORT.FTN - SORT THE CORE LOG FILES INTO ASCENDING REF NO.
C
      CHARACTER*4 DEPFR(10),DEPTO(10)
      CHARACTER*5 RACNO(10)
      CHARACTER*1 STOLOC(10),UNIT(10),TYPE(10)
      INTEGER*2 REFNO,REFNUM(8000),REFN1(2000),REFN2(2000)
      INTEGER*2 REFN3(2000),REFN4(2000)
C
C         DECLARATION OF INPUT AND OUTPUT VARIABLES
C
      CHARACTER*1 BUFF(162)
      EQUIVALENCE (REFNUM(1),REFN1(1)),(REFNUM(2001),REFN2(1)),
     .            (REFNUM(4001),REFN3(1)),(REFNUM(6001),REFN4(1))
      EQUIVALENCE (BUFF(1),REFNO),
     .     (BUFF(3),STOLOC),(BUFF(13),RACNO),(BUFF(63),DEPFR)
     .     ,(BUFF(103),DEPTO),(BUFF(143),UNIT),(BUFF(153),TYPE)
C
C         A BINARY BUFFERED ARRAY FOR FASTER DATA-BASE ACCESS
C
       INTEGER*2 INDEX(8000),IA(8000)
C
C         THESE INDEXES ARE FOR THE SORT: UP TO 8000 ENTRIES ALLOWED
C
      OPEN(UNIT=4,FORM='BINARY',ACCESS='DIRECT',RECL=162)
C
C         OPEN THE INPUT FILE FOR RANDOM AND SEQUENTIAL ACCESS
C
      OPEN(UNIT=8,FORM='BINARY',RECL=4000)
C
C         OPEN THE INDEX FILE FOR LINKING TO LOGSRCH
C
       REWIND 4
       IREC=0
C
C         COUNTS NUMBER OF RECORDS
C
      DO 5 I=1,8000
         REFNUM(I)=0
    5 CONTINUE
C
C       FILL REFNUM ARRAY WITH ZEROS TO BE ABLE TO FIND THE LAST
C       REFNO LATER IN LOGSRCH
C
C
   10   READ(4,END=20) BUFF
        IREC=IREC+1
        IF(IREC .GT. 8000) STOP 'MORE THAN 8000 ENTRIES'
        INDEX(IREC)=IREC
C
C         FILL SORT INDEX
C
        IA(IREC)=REFNO
        GO TO 10
C
C         READ THE NEXT FILE
C
```

```
   20   CONTINUE
C
C        AND NOW TO BUSINESS
C
        IF(IREC .EQ. 0) STOP 'NO RECORDS IN DATA FILE'
        CALL BUBSTI(INDEX,IA,IREC)
C
C        SORT INTO ASCENDING NUMBER - I.E. CHRON ORDER
C
       OPEN(UNIT=6,FORM='BINARY',RECL=162)
C
C        OPEN OUTPUT FILE
C
       DO 30 I=1,IREC
       READ(4,REC=INDEX(I)) BUFF
       WRITE(6) BUFF
C
C        COPY TO NEW ORDER
C
       REFNUM(I)=REFNO
C
C        STORE THE REFERENCE NUMBERS
C
    30 CONTINUE
       WRITE(8) REFN1
       WRITE(8) REFN2
       WRITE(8) REFN3
       WRITE(8) REFN4
C
C        BINARY WRITE TO FILE OF REF NOS
C
C        CLOSE FILES AND END
C
       CLOSE(UNIT=6,STATUS='KEEP')
       CLOSE(UNIT=4,STATUS='KEEP')
       CLOSE(UNIT=8,STATUS='KEEP')
       END
C
C*************************************************
C
       SUBROUTINE BUBSTI(IR,IA,N)
       INTEGER*2 IA(N)
       INTEGER*2 IR(N)
       LOGICAL NSWAP
       IF (N .LE. 1) RETURN
C
C        NOTHING TO SORT
C
       NM1=N-1
       DO 30 J=1,NM1
       NSWAP=.TRUE.
       IRI=IR(1)
       DO 40 I=1,NM1
       IP1=I+1
       IRIP1=IR(IP1)
       IF (IA(IRI) .LE. IA(IRIP1)) GOTO 40
       NSWAP=.FALSE.
       IR(I)=IRIP1
```

```
      IR(IP1)=IRI
      IRIP1=IRI
   40 IRI=IRIP1
      IF (NSWAP) RETURN
   30 CONTINUE
      RETURN
      END
```

APPENDIX 9

```
*CORSRCH.CSS - SEARCH THE CORE INDEX FILE
L CORSRCH,10; AS 4,CORIND.DAT,ERO; XAL SYSF:CORIND.TMP,IN,162/3/2
AS 6,SYSF:CORIND.TMP; REW 6; AS 5,CON:; TEMPFILE 3,IN,80; ST
$IFX SYSF:CORIND.TMP; PRI SYSF:CORIND.TMP,DEL; $ENDC; $EXIT
```

APPENDIX 10

```
$TITL  CORSRCH.FTN - SEARCH CORE INDEX FILE
C
C
      CHARACTER*4 DEPFR(10),DEPTO(10)
      CHARACTER*5 RACNO(10),CREFNO
      CHARACTER*1 STOLOC(10),UNIT(10),TYPE(10),SPRINT
      INTEGER*2 REFNO,IREFNO(30)
C
C         DECLARATION OF INPUT AND OUTPUT VARIABLES
C
      CHARACTER*1 BUFF(162)
      EQUIVALENCE (BUFF(1),REFNO),
     .     (BUFF(3),STOLOC),(BUFF(13),RACNO),(BUFF(63),DEPFR)
     .     ,(BUFF(103),DEPTO),(BUFF(143),UNIT),(BUFF(153),TYPE)
C
C         A BINARY BUFFERED ARRAY FOR FASTER DATA-BASE ACCESS
C
      CHARACTER*80 SREFNO,SSTOLOC,SRACNO,STYPE,SQUAD
C
C         STRINGS FOR INPUT DATA
C
      INTEGER*2 FFEED
      DATA FFEED/3072/
C
C         THE FORM FEED CHARACTER IN A1
C
      CHARACTER*8 STOCON(4),UNITCON(3),TYPECON(4)
      CHARACTER*8 ASTOLOC(10),AUNIT(10),ATYPE(10)
      DATA STOCON/'MORNSTOR','MORNPLNT',' DOMAIN ',' Q.TOWN '/
      DATA UNITCON/' METRES ','  FEET  ',' OTHER  '/
      DATA TYPECON/'DIAM.COR','CUTTINGS',' AUGER  ',' OTHER  '/
C
C         THE STORE LOC'N, UNITS OF LENGTH AND SAMPLE TYPE ABBREVS.
C
      CHARACTER*1 ALINE(63)
C
C         SCREEN OUTPUT VARIABLE
C
      LOGICAL NOTIN
      INTEGER*2 FCOUNT,LCOUNT
C
C         END OF DECLARATIONS. NOW WE CAN START THE DATA SEARCH.
C
      OPEN(UNIT=4,FORM='BINARY',RECL=162)
C
C         OPEN THE INPUT FILE
C
      WRITE(5,100)
  100 FORMAT(' SEARCHING CORE LIBRARY INDEX')
      WRITE(6,400)
  400 FORMAT(' SEARCH OPTIONS ARE:')
      WRITE(5,101)
  101 FORMAT(' FOR EACH OPTION ENTER EITHER A BLANK LINE OR'/
     . ' THE DESIRED VALUES FOR THE OPTION, EACH TERMINATED BY'
     . ,' A /'/' EG. 2234/4536/7658/')
```

```
  102 FORMAT(' REFERENCE NUMBER?')
      READ(5,201) SREFNO
      IF (SREFNO(1:5) .EQ. '     ') GOTO 20
C
C         TEST TO SEE IF REFERENCE NUMBER SEARCH REQUIRED.
C         IF SEARCH REQUIRED, CONVERT THE INPUT REFERENCE NUMBERS FROM
C         CHARACTER TO INTEGER FORMAT BY LOCATING THE /'S AND
C         THEN CONVERTING EACH NUMBER IN TURN.
C
      NREFNO=0
      WRITE(6,401) SREFNO
  401 FORMAT(' REFERENCE NUMBER',10X,A80)
      ILAST=1
   21 ICHAR=INDEX(SREFNO(ILAST:80),'/')
      IF(ICHAR .EQ. 0) GOTO 20
      NREFNO=NREFNO+1
      IREFNO(NREFNO)=CTOI(SREFNO(ILAST:ILAST+ICHAR-1),ITEMP)
C
C         CONVERTED TO AN INTEGER
C
      ILAST=ILAST+ICHAR
      IF (ILAST .NE. 79) GOTO 21
C
C         GO BACK AND CONVERT THE NEXT REF'NCE NO. TO INTEGER FORMAT
C
C         NOW THE REST OF THE SEARCH OPTIONS
C
   20 CONTINUE
      WRITE(5,103)
  103 FORMAT(' STORE LOCATION : MORNINGTON CORE STORE = 1',
     .                /16X,': MORNINGTON PLANT STORE = 2',
     .                /16X,': DOMAIN CORE STORE = 3')
      READ(5,201) SSTOLOC
  201 FORMAT(A80)
      IF (SSTOLOC(1:1) .NE. ' ') WRITE(6,402) SSTOLOC
  402 FORMAT(' STORE LOCATION',10X,A80)
      WRITE(5,104)
  104 FORMAT(' RACK NUMBER ')
      READ(5,201) SRACNO
      IF (SRACNC(1:1) .NE. ' ') WRITE(6,403) SRACNO
  403 FORMAT(' RACK NUMBER',10X,A80)
      WRITE(5,105)
  105 FORMAT(' SAMPLE TYPE ( CORE=1,CUTTINGS=2,AUGER SAMPLES=3 )')
      READ(5,201) STYPE
      IF (STYPE(1:1) .NE. ' ') WRITE(6,404) STYPE
  404 FORMAT(' SAMPLE TYPE',10X,A80)
C
C         END OF OPTION INPUT
C
C-------------------------------------------------------------------------
C
C         NOW FOR THE SEARCH
C
      FCOUNT=0
      LCOUNT=6
C
C         COUNTER FOR NUMBER OF FOUND ENTRIES, AND NUMBER OF LINES
C         WRITTEN TO THE OUTPUT FILE.
```

```
C
      WRITE(6,107)
  107 FORMAT(/'  REFRNCE¦  STORE   ¦ RACK  ¦    DEPTH    ',
     .       '¦          ¦'/2X'NUMBER ¦ LOCATION ¦NUMBER ¦',
     .       ' FROM ¦  TO  ¦  UNITS   ¦  TYPE'/1X,63('-'))
C
C          HEAD UP PAGE THEN TEST FOR A MATCHING STRING FOR EACH
C          OF THE SEARCH OPTIONS (USING THE SUBROUTINE 'NOTIN')
C
   27 READ(4,END=40) BUFF
      IF (NREFNO .NE. 0) THEN
      DO 28 I=1,NREFNO
        IF(IREFNO(I) .EQ. REFNO) GOTO 29
   28 CONTINUE
      GOTO 27
   29 CONTINUE
      ENDIF
C
C          IF MATCHING REFERENCE NUMBER NOT FOUND, READ NEXT LINE.
C          OTHERWISE TEST OTHER SEARCH OPTIONS.
C
      I=0
   30 I=I+1
      IF (STOLOC(I) .EQ. '0') GOTO 27
      IF (NOTIN(SSTOLOC,STOLOC(I))) GOTO 30
      IF(NOTIN(SRACNO,RACNO(I))) GOTO 30
      IF (NOTIN(STYPE,TYPE(I))) GOTO 30
C
C          THE REMAINING SEARCH OPTIONS ARE TESTED. IF A MISMATCH IS
C          FOUND, A NEW DATA LINE (BUFF) IS READ. IF A MATCH IS FOUND,
C          THE PROGRAMME CONTINUES.
C
      IF (LCOUNT .GE. 58) THEN
        WRITE(6,106) FFEED
  106   FORMAT(1A1/'  REFRNCE¦  STORE   ¦ RACK  ¦    DEPTH    ',
     .         '¦          ¦'/2X'NUMBER ¦ LOCATION ¦NUMBER ¦',
     .         ' FROM ¦  TO  ¦  UNITS   ¦  TYPE'/1X,63('-'))
        LCOUNT=3
      ENDIF
      FCOUNT=FCOUNT+1
C
C          HEAD UP PAGE (IF NECESSARY) . INCREMENT FOUND ENTRY AND
C          WRITTEN LINE COUNTER.
C
      DO 33 I=1,10
        IF(STOLOC(I) .EQ. '0') GOTO 34
        CALL NMONIC(STOLOC(I),ASTOLOC(I),STOCON,4)
        CALL NMONIC(UNIT(I),AUNIT(I),UNITCON,3)
        CALL NMONIC(TYPE(I),ATYPE(I),TYPECON,4)
   33 CONTINUE
C
C          SUBSTITUTE MNEMONICS FOR THE CODE NUMBERS BEFORE PRINTING
C
   34 CREFNO=ITOC(REFNO,ITEMP)
      WRITE(6,302) CREFNO,ASTOLOC(1),RACNO(1),DEPFR(1),DEPTO(1)
     .             ,AUNIT(1),ATYPE(1)
  302 FORMAT(3X,A5,3X,A8,3X,A5,3X,A4,3X,A4,3X,A8,3X,A8)
      DO 47 J=2,I-1
```

```
        WRITE(6,303) ASTOLOC(J),RACNO(J),DEPFR(J),DEPTO(J)
     .              ,AUNIT(J),ATYPE(J)
  303   FORMAT(11X,A8,3X,A5,3X,A4,3X,A4,3X,A8,3X,A8)
   47 CONTINUE
      LCOUNT=LCOUNT+I-1
      GOTO 27
C
C         WRITE THE DATA TO THE OUTPUT FILE, THEN BACK TO THE START
C         FOR THE NEXT DATA LINE.
C
   40 CONTINUE
C
C         AT THE END OF THE RUN, CHECK IF A SCREEN LISTING IS WANTED.
C
      WRITE(5,115) FCOUNT
  115 FORMAT(I7,' RECORDS WERE FOUND'/' DO YOU WANT A',
     .       ' SCREEN LISTING (Y OR N)?')
      READ(5,200) SPRINT
      IF (SPRINT .EQ. 'N') GOTO 503
C
C         IF NO SCREEN LIST WANTED, CHECK FOR PRINTOUT.
C
      REWIND (6)
C
C         REWIND THE FILE SO IT CAN BE SENT TO THE SCREEN
C
  500 ICOUNT=0
C
C         COUNTER FOR NUMBER OF LINES ON SCREEN
C
  501 READ(6,304,END=503) ALINE
      WRITE(5,304) ALINE
  304 FORMAT(63A1)
      ICOUNT=ICOUNT+1
      IF (ICOUNT .LT. 20) GOTO 501
C
C         ALLOW 20 ENTRIES PER SCREENFULL
C
      WRITE(5,117)
  117 FORMAT(' LIST MORE ENTRIES ON SCREEN (Y OR N)?')
      READ(5,200) SPRINT
      IF (SPRINT .EQ. 'Y') GOTO 500
C
C          ROUND AGAIN FOR ANOTHER SCREEN FULL, OR ASK ABOUT PRINTOUT.
C
  503 WRITE(5,121) FCOUNT
  121 FORMAT(I7,' RECORDS, DO YOU WANT A PRINTOUT?')
      READ(5,200) SPRINT
  200 FORMAT(A1)
      IF (SPRINT .NE. 'Y' .OR. FCOUNT .EQ. 0) THEN
       CLOSE(UNIT=6,STATUS='DELETE')
       STOP
C
C         PRINTOUT NOT WANTED, SO CLOSE AND DELETE FILE THEN STOP
C
      ENDIF
      CLOSE(UNIT=4,STATUS='KEEP')
C
```

```
C           CLOSE THE INPUT FILE.
C
      STOP
      END
C
C*****************************************************************
C
      LOGICAL FUNCTION NOTIN(SSTRNG,RSTRNG)
C
C           SSTRNG CONTAINS A SERIES OF STRINGS TERMINATED BY /'S
C           E.G.    M99/M100/M101/
C           AND RSTRNG IS SEARCHED FOR THE OCCURENCE OF ONE OF THESE
C           STRINGS. IF ONE IS FOUND TO MATCH NOTIN IS .FALSE.
C           IF STRNG IS EMPTY NOTIN IS .FALSE.
C
      CHARACTER*80 SSTRNG
      CHARACTER* (*) RSTRNG
      NOTIN=.FALSE.
      IF (SSTRNG(1:2) .EQ. '  ') RETURN
C
C           NOTHING IN STRING, SO NO NEED TO LOOK
C
      ILAST=1
C           COUNTER FOR POSITION IN SSTRNG
   20 ICHAR=INDEX(SSTRNG(ILAST:80),'/')
C           LOOK FOR / AS TERMINATOR OF STRING
      IF (ICHAR .EQ. 0) GOTO 25
C           HAVENT FOUND A MATCH BY THE END OF THE LINE
      IF (INDEX(RSTRNG,SSTRNG(ILAST:ILAST+ICHAR-2)) .NE. 0) RETURN
C           LOOK BETWEEN /S IN SSTRNG AND IF FIND A MATCH RETURN
      ILAST=ILAST+ICHAR
C           POSITION AFTER THE /
      IF (ILAST .LE. 79) GOTO 20
C           BACK ROUND IF NOT AT END OF LINE
C
C           ELSE END
   25 CONTINUE
      NOTIN=.TRUE.
C           NO MATCH
      RETURN
      END
C
C*****************************************************************
C
      SUBROUTINE NMONIC(CVAR,STRVAR,CONSTS,NOPT)
      CHARACTER*1 CVAR
      CHARACTER*8 CONSTS(1),STRVAR
C
C           USED TO PUT A STRING CONSTANT FROM CONSTS INTO STRVAR
C           DEPENDING ON THE CHARACTER IN CVAR
C           IF CVAR IS BLANK STRVAR IS SET BLANK
C           NOPT IS THE NUMBER OF VALID OPTIONS FOR CVAR
C
      JVAL=CTOI(CVAR,K)
C           CHARACTER TO INTEGER CONVERSION TO GET THE VALUE
      IF (JVAL .LE. NOPT) STRVAR=CONSTS(JVAL)
      RETURN
      END
```