"The utility of the system stems from the possibility of labeling the data unit and treating it as some common type of mathematical entity."

An Integrated Data Input/Output Manager for Microcomputers

by

Salvador F. Reyes, Ph.D.*

INTRODUCTION

When microcomputers are used in data processing of the "number crunching" category (such as in many scientific and engineering applications) user intervention or interaction may well be limited to the input and output phases of the processing. This suggests that an all-purpose input/output manager along with utilities (subroutines) that can be linked to the applications program module could permit specific coding for the applications program to be focused on the computational aspects and still provide, in the running of the application, a sufficiently friendly environment for users.

In such a scheme the input/output manager creates a file from keyed in data which then is read (with the help of the utilities) by the applications module. These same utilities may subsequently be used to transform the results generated by the applications module to an output file which the input/output manager can later convert to the desired displayed or printed form.

FEATURES OF I.D.I.O.M.

This note describes a simple implementation which includes a stand-alone integrated input/output manager (I.D.I.O.M.) and a set of utilities (subroutines in CBASIC source code) which can be linked to any applications module. The data file organization selected for the implementation is sequential and ASCII format in order to facilitate data interchange among independent program modules written possibly in a variety of high level source codes.

For simplicity, I.D.I.O.M. is designed to process only numbers which, in the file, are in the form of ASCII fields delimited by commas and carriage return marks. The basic data unit is the data member consisting of one or more such fields (i.e., an array). The utility of the system, however, stems from the possibility of labeling the data unit and treating it as some common type of mathematical entity. In the current implementation the possible mathematical data types are scalar, vector, matrix, hypermatrix (partitioned matrix in which the submatrices are of equal size), and table. Although a table of numerical elements is structurally similar to a matrix the distinction permits assigning labels to the columns of the table. Moreover, for tabular data it is meaningful to modify the number of rows when editing the member or in the course of processing.

The logical structure of a data member is conveyed by a header which precedes the numeric array itself. Normally the details concerning the various forms of the header (described in Appendix "A") need not concern the user. I.D.I.O.M. interprets the information contained therein and presents these in the proper forms in the displays or printouts. Moreover, during the input phase, the user will be guided by prompters in specifying the desired data structure. It will, therefore, suf-

*Professor of Civil Engineering, U.P. College of Engineering, Diliman, Quezon City, Philippines.

fice to note that the information in the headers is contained in one or more character data fields (enclosed in quotation marks) with each field being identified as such by a preceding "P" (for parameter) field.

It is possible to include arbitrary character data in the headers. Identified as "C" records, these are displayed during data input or editing. Thus a data header consists of "P" records and optional "C" records. A sequence of one or more data members (each consisting of a header and numeric array elements) constitute a data set. If a file is to contain more than one data set each set should be terminated by an "E" mark.

I.D.I.O.M. can be used to generate input data files under control of a predefined set of headers. The file containing such sets of headers may be regarded as a dummy file (or template file) tailored to the requirements of the applications program at hand. As noted in Appendix "A" certain required data parameters can either be fixed in the header or made specific only when data are about to be keyed in.

The basic facilities provided in I.D.I.O.M. are therefore the following:

1. Header generation
2. Header display or printing
3. Header editing
4. (Numeric) data entry
5. Data display or printing
6. Data editing
7. Data merging

The first three are for the purpose of developing dummy data files or header files; the remaining four are concerned with true data files. On startup I.D.I.O.M. guides the user in setting up a file directory. This directory can be updated after every successful (or aborted) execution of any of the above tasks. Full guidance is given by display when some response is expected from the user. Suffice it to note that in the management of the files (up to 20 files can be simultaneously active in the present implementation) the magnetic tape metaphor is adopted. In general all facilities provided by the operating system for handling sequential files can be availed of.

For the data input phase, the facilities for entering or editing the files are aimed at reasonably approximating the ease with which these tasks can be carried out on a deck of punched cards. For display and printing purposes numeric data formats can be specified. Again a degree of flexibility is afforded in that the formats of the number fields are made specific only when the data is about to be printed or displayed.

EXAMPLE

Appendix "B" lists the source code of a typical applications program which includes the necessary utilities for reading and writing I.D.I.O.M. files. This simple program computes the vertical normal stress at a specified set of points in a semi-infinite elastic medium due to a set of vertical loads. Accordingly the required input consists of a table of LOADS giving the x,y (horizontal plane) coordinates and magnitude of each of the given loads, and a table of POINTS containing the x, y, z coordinates of the points where the vertical stresses are needed. As soon as the two tables are at hand the program calculates the vertical stresses and outputs these along with the point coordinates in a table of RESULTS.

Plate 1 lists a possible dummy data file or header file for the program. In I.D.I.O.M. it is possible to repetitively use or skip a prepared header in a dummy file during data entry. Hence, tables of POINTS and RESULTS can be interleaved arbitrarily or stacked in the input file being created. In a given run the output will thus consist of one or more tables of stress values. The headers for the tables of RESULTS are written with the help of the utilities as are the needed file initialization statements. Thus, only simple input/output statements are needed in the applications program module itself. Plate 2 lists sample data and Plate 3 the tabulation of results. Both are printouts produced by I.D.I.O.M.

CONCLUSION

An integrated input/output manager has been described and illustrated. Apart from the benefit of a uniform environment for data inputs and outputs that such a system affords, security of specialized applications software can be easily maintained since the input/output phases of data processing can be done separately from the purely computational phase.

APPENDIX "A"
I.D.I.O.M. Data Headers

A typical header is a sequence of ASCII data records (delimited character data fields each of which is enclosed by quotation marks (") and preceded by a one-character data record identifier, (either "C" or "P"). If preceded by "C", the record is interpreted as a commentary and is displayed during data entry or editing of the member in question. Hence, "C" records effectively function as prompters.

"P" records convey information which defines the type, appropriate labels, and dimensions of the data. The character subfields within the record are delimited by blanks. The first subfield is the data type symbol. Valid data type symbols are S (scalar), V (vector), M (matrix), H (hyper matrix), and T (table). The type symbol will be followed by a member (or header) name or label (an alphanumeric string of reasonable length). For scalars these two subfields complete the "P" record. For vectors, matrices and hypermatrices these will be followed by the proper number of positive integers representing the array dimensions (number of elements of vectors, number of rows and number of columns of matrices, number of rows/columns and number of submatrix rows/columns of hyper-matrices).

For tabular data the prescribed dimensions are the number of columns and the number of rows (in that order) followed by the column labels. For header files it is possible to indicate that the required dimensions will be entered from the keyboard when the data elements themselves are entered. An asterisk ("*") is included in the record in lieu of the dimension to indicate that the latter will be specified during data entry. The header written to the data file will necessarily contain the actual dimensions of the data member. Following are typical headers for the different types of data members.

```
"C", "Header for a Scalar Data Member"
"P", "S ASCALAR"
"C", "A vector with number of elements to be keyed in"
"C", "just before element values are entered"
"P", "V V *"
"C", "Matrix – number of rows fixed; number of columns"
"C", "to be keyed in with the data"
"P", "M TYPMATRIX 10 *"
"C", "Hypermatrix"
"P", "H HYPER 2 2 * *"
"C",, "Tabular data with three columns labeled 'FIELDA', "
"C", "'FIELDB', 'FIELDC', and 'LONG. FIELD. C'. Note that"
"C", "two or more 'P' records may be used if necessary."
"P", "T TABLE 3 * FIELDA FIELDB FIELDC"
"E"
```

As many 'P' records as required may be used for specifying field (column) labels of tabular data.

PLATE 1
SAMPLE HEADER DATA SET

```
HEADER No.  1  (Type - TABLE)
VERTICAL LOADS ON A
SEMI-INFINITE ELASTIC MEDIUM
NAME        LOADS
Field (Column) NAMES:
  X-COORD  Y-COORD  LOAD
Number of Rows from the Keyboard
HEADER No.  2  (Type - TABLE)
COORDINATES OF POINTS WHERE
STRESSES ARE TO BE DETERMINED
NAME        POINTS
Field (Column) NAMES:
  X-COORD  Y-COORD  Z-COORD
Number of Rows from the Keyboard
```

PLATE 2
SAMPLE DATA SET

LOADING SET 1
LOADS

	X-COORD	Y-COORD	LOAD
1	0.00	0.00	1000
2	5.00	0.00	1000
3	5.00	5.00	1000
4	0.00	5.00	1000

COORDINATES OF POINTS DIRECTLY BENEATH ONE FOOTING
POINTS

	X-COORD	Y-COORD	Z-COORD
1	0.00	0.00	1.00
2	0.00	0.00	2.00
3	0.00	0.00	3.00
4	0.00	0.00	4.00
5	0.00	0.00	5.00

COORDINATES OF POINTS BENEATH CENTER OF TOWER
POINTS

	X-COORD	Y-COORD	Z-COORD
1	2.50	2.50	1.00
2	2.50	2.50	2.00
3	2.50	2.50	3.00
4	2.50	2.50	4.00
5	2.50	2.50	5.00

PLATE 3
COMPUTED VERTICAL NORMAL STRESSES

STRESSES DIRECTLY BENEATH A FOOTING
RESULTS

	X-COORD	Y-COORD	Z-COORD	Pz
1	0.00	0.00	1.00	477.8
2	0.00	.0.00	2.00	121.2
3	0.00	0.00	3.00	57.4
4	0.00	0.00	4.00	36.4
5	0.00	0.00	5.00	27.1

STRESSES BENEATH CENTER OF TOWER
RESULTS

	X-COORD	Y-COORD	Z-COORD	Pz
1	2.50	2.50	1.00	2.9
2	2.50	2.50	2.00	13.8
3	2.50	2.50	3.00	24.1
4	2.50	2.50	4.00	28.2
5	2.50	2.50	5.00	27.7

APPENDIX "B"
SAMPLE PROGRAM LISTING

```
REM ====================================================================
REM FILENAME:   IDIOMEX1.BAS
REM A simple applications program utilizing IDIOMatic data
REM Programmed by S. F. Reyes                              15 JUNE 1985
REM ====================================================================
REM
REM Arrays worked on by the utilities:
REM Size of FD$(.) must be at least equal to the number of columns
REM in tabular data members (4 in the present case)
REM SZ%(.), for member dimensions, has a maximum of 4 elements
REM (for hypermatrices)
DIM LB$(1),SZ%(3),FD$(3),CD$(24)
LB$(0)="LOADS":LB$(1)="POINTS":NL%=2
REM
REM ================== I.D.I.O.M. UTILITIES ===========================
REM
REM Pause Function
DEF FN.WAIT%
        PRINT "Press <ENTER>"
        WHILE NOT CONSTAT%:WEND
        I%=CONCHAR%:PRINT CHR$(8);" ";CHR$(8):FN.WAIT%=0:RETURN
        FEND
REM
REM Prompt for a valid keyboard response
DEF FN.PROMPTER%(A$)
```

```
        WHILE -1
                WHILE NOT CONSTAT%:WEND
                T%=MATCH(UCASE$(CHR$(CONCHAR%)),A$,1)
                PRINT CHR$(8);" ";CHR$(8);
                IF T%>0 THEN FN.PROMPTER%=T%:PRINT:RETURN ELSE
        WEND
        FEND
REM
REM YES/NO Response
DEF FN.QUERY%
        PRINT "  ( ";US$;"Y";UT$;"ES / ";US$;"N";UT$;"O ) ?";
        FN.QUERY%=FN.PROMPTER%("YN")-2:RETURN
        FEND
REM
REM Comments to OUTFILE%
DEF FN.COMMENTS%
        T%=0
        WHILE T%<NC%
                PRINT CHR$(12)
                IF T%>0 THEN FOR TT%=1 TO T%:PRINT \
                        CHR$(14);CD$(TT%-1);CHR$(15):NEXT TT% ELSE
                PRINT "TYPE Next Comment Line"
                INPUT LINE T$:CD$(T%)=T$
                PRINT # OUTFILE%; "C",T$:T%=T%+1
        WEND:RETURN
        FEND
REM
REM Write "E" into file M%
DEF FN.WRITEE%
        PRINT "Write 'E' MARK to OUTPUT FILE";
        IF FN.QUERY% THEN PRINT # OUTFILE%; "E" ELSE
        FN.WRITEE%=-1:RETURN
        FEND
REM
REM Locate the next record field. Return pointer in I%, length in J%
REM Read a new record, DD$, from INFILE% if necessary
REM Return value of -1 and record in DD$ if successful, 0 otherwise
REM If not successful, return error message is MS$
REM If invalid record identifier, return HD$="?"
REM Return 'C' records in CD$(.), number of 'C' records in NC%
REM  Work variables: K%
DEF FN.NXTFIELD%
        MS$=""
        WHILE -1
                IF END # INFILE% THEN 13.004
                WHILE DD$=""
                        HD$="":READ # INFILE%; HD$:HD$=UCASE$(HD$)
                        IF LEN(HD$)>1 THEN GO TO 13.002
                        K%=MATCH(HD$,"ECPN",1)
                        IF K%=0 OR K%=4 THEN GO TO 13.006
                        IF K%=1 THEN GO TO 13.005
                        IF END # INFILE% THEN 13.001
                        READ # INFILE%; DD$
                        IF K%=2 THEN DD$="" ELSE I%=1
                WEND
                T$=" ":K%=LEN(DD$)
                WHILE I%<=K%:T$=MID$(DD$,I%,1)
                        IF T$=" " THEN I%=I%+1 ELSE K%=0
                WEND
                WHILE T$<>" ":J%=MATCH(" ",DD$,I%)
                        IF J%=0 THEN J%=LEN(DD$)+1-I% ELSE J%=J%-I%
                        FN.NXTFIELD%=-1:RETURN
                WEND:DD$=""
        WEND
13.001  MS$="PREMATURE END of FILE"
13.002  HD$="?"
13.003  FN.NXTFIELD%=0:RETURN
13.004  MS$="END of FILE":GO TO 13.003
13.005  MS$="END of DATA SET":GO TO 13.003
13.006  MS$="INVALID RECORD IDENTIFIER":GO TO 13.003
        FEND
```

```
REM
REM Read and Interpret a Header
REM Value of VN% is index of matching name in LB$(.)
REM Variable label in VN$; type in VT%;
REM Size parameters in SZ%(.)
REM SZ%(I%-1)=0 means size parameter I% to be input from keyboard
REM DP$(I%-1)="*" is set; otherwise DP$(I%-1)=""
REM For tabular data, return field names in FD$(.)
REM Return value of -1 if successful; 0 if not
REM Work variables:     K%,T%,KK%,SY%,T$,DD$
DEF FN.INTERPRT%
REM Get member type
        DD$="":T%=FN.NXTFIELD%
        IF NOT T% THEN GO TO 13.011 ELSE T$="SVMHT"
        KK%=MATCH(UCASE$(MID$(DD$,I%,1)),T$,1)
        IF KK%=0 THEN PRINT "INVALID Variable TYPE" \
                :GO TO 13.011 ELSE I%=I%+J%
REM Get member name
        IF NOT FN.NXTFIELD% THEN GO TO 13.013
        VN$=MID$(DD$,I%,J%)
        T%=0:VN%=0
        WHILE T%<NL%
                IF VN$=LB$(T%) THEN VN%=T%+1:T%=NL% ELSE T%=T%+1
        WEND
        IF VN%=0 AND NL%>0 THEN GO TO 13.014
REM If NL%=0 any name is valid
REM Get dimension(s) if not a scalar
        VT%=KK%
        WHILE KK%<>1
                IF VT%=2 THEN KK%=1
                IF VT%=3 OR VT%=5 THEN KK%=2
                IF VT%=4 THEN KK%=4
                FOR N%=1 TO KK%:I%=I%+J%
                        IF NOT FN.NXTFIELD% THEN GO TO 13.013
                        T%=VAL(MID$(DD$,I%,J%))
                        IF T%<=0 THEN GO TO 13.012 ELSE \
                            SZ%(N%-1)=T%
                        NEXT N%:KK%=VT%
REM If tabular, get field labels
                WHILE KK%=5
                        FOR KK%=0 TO SZ%(0)-1:I%=I%+J%
                            IF NOT FN.NXTFIELD% THEN GO TO 13.013
                            FD$(KK%)=MID$(DD$,I%,J%)
                            NEXT KK%:KK%=0
                WEND:KK%=1
        WEND:FN.INTERPRT%=-1:RETURN
13.011  FN.INTERPRT%=0:RETURN
13.012  MS$="INVALID DIMENSION":GO TO 13.011
13.013  IF HD$="?" THEN GO TO 13.011 ELSE \
                MS$="INCOMPLETE HEADER":GO TO 13.011
13.014  MS$="INVALID HEADER NAME":GO TO 13.011
        FEND
REM
REM Check name for invalid characters in VT$
DEF FN.VALIDNAM%
        IF MATCH(" ",VT$,1)>0 OR MATCH(",",VT$,1)>0 OR \
                MATCH(CHR$(34),VT$,1)>0 THEN FN.VALIDNAM%=0 \
                ELSE FN.VALIDNAM%=-1
        RETURN
        FEND
REM
REM Extend (or write to OUTFILE% and reinitialize) template data
REM Return value of 0 if nothing written; -1 otherwise
DEF FN.WRITEDD%
        FN.WRITEDD%=0
        T%= LEN(HD$)+LEN(T$)
        IF T%>72 THEN FN.WRITEDD%=-1:PRINT # OUTFILE%; "P" \
                :PRINT # OUTFILE%; HD$ \
                :HD$="":T%=0 ELSE
        HD$=HD$+T$
        RETURN
        FEND
```

```
REM
REM Form a data template and write it to OUTFILE.
DEF FN.FORMDD%
        HD$=MID$("SVMHT",VT%,1)+" "+VN$
        T$="":T%=FN.WRITEDD%
        T%=VT%
        WHILE T%>1        REM Array dimensions
                IF VT%=2 THEN TT%=0
                IF VT%=3 OR VT%=5 THEN TT%=1
                IF VT%=4 THEN TT%=3
                FOR I%=0 TO TT%:T%=SZ%(I%)
                        IF T%<=0 THEN T$=" *" ELSE T$=" "+STR$(T%)
                        T%=FN.WRITEDD%:NEXT I%:T%=0
        WEND
        IF VT%=5 THEN FOR I%=0 TO SZ%(0)-1:T$=" "+FD$(I%) \ Fld. Labels
                :T%=FN.WRITEDD%:NEXT I% ELSE
        PRINT # OUTFILE%; "P",HD$:FN.FORMDD%=-1:RETURN
        FEND
REM
REM Set up INPUT and OUTPUT files
REM Names used: INFILE$,INFILE%,OUTFILE$,OUTFILE%
DEF FN.SETUPIO%
        INPUT "INPUT  FILE Name? ";INFILE$
        INPUT "OUTPUT FILE NAME? ";OUTFILE$
        INFILE%=1:OUTFILE%=2
        IF END # INFILE% THEN 13.022
        OPEN INFILE$ AS INFILE%
        IF END # OUTFILE% THEN 13.021
        OPEN OUTFILE$ AS OUTFILE%
        CLOSE OUTFILE%
        PRINT CHR$(14);OUTFILE$;CHR$(15);" has DATA. OVERWRITE";
        IF NOT FN.QUERY% THEN CLOSE INFILE%:FN.SETUPIO%=0:RETURN ELSE
13.021  CREATE OUTFILE$ AS OUTFILE%
        FN.SETUPIO%=-1:RETURN
13.022  PRINT CHR$(14);VN$;CHR$(15);" Non-existent."
        FN.SETUPIO%=0:RETURN
        FEND
REM
REM ================== End of UTILITIES ==============================
REM
100 PRINT CHR$(12)
PRINT "==========================================="
PRINT "* Calculation of     VERTICAL NORMAL STRESSES *"
PRINT "* at a SET of POINTS in an Elastic Half Plane *"
PRINT "* Due to a SET of VERTICAL CONCENTRATED LOADS *"
PRINT "*           (The  BOUSSINESQ Problem)          *"
PRINT "* S. F. REYES                        14 JUNE 85 *"
PRINT "==========================================="
PRINT
PRINT "Program reads a table of  ";CHR$(14);"LOADS";CHR$(15);
PRINT "  and a table "
PRINT "of  ";CHR$(14);"POINTS";CHR$(15);"  from the ";
PRINT "designated Input File."
PRINT "After computing for the desired stresses, "
PRINT "the table of  ";CHR$(14);"RESULTS";CHR$(15);"  is written";
PRINT " to the"
PRINT "designated Output File."
PRINT
REM
DEF FN.RSQ(ILOAD%,IPOINT%)=(ABS(POINTS(IPOINT%,0)-LOADS(ILOAD%,0)))^2 \
        +(ABS(POINTS(IPOINT%,1)-LOADS(ILOAD%,1)))^2
REM Set up input and output files
WHILE NOT FN.SETUPIO%
        PRINT "Try AGAIN";
        IF NOT FN.QUERY% THEN STOP
WEND
REM
100.2 REM Data input phase
GO%=0:NPOINTS%=0:NLOADS%=0
WHILE -1          REM Process until 'E' or error in input file
```

```
REM Interpret the header
        WHILE NOT FN.INTERPRT%
                PRINT MS$
                IF HD$="E" THEN PRINT "Data Set END Sensed. CONTINUE"; \
                        :T%=FN.QUERY% ELSE T%=0
                IF NOT T% THEN STOP ELSE GO TO 100.2
        WEND
REM If interpretation successful read data to the proper array
        IF END # OUTFILE% THEN 13.102
        T%=VN%
        WHILE T%=1      REM Read load table
                IF VT%<>5 OR SZ%(0)<>3 THEN GO TO 13.103
                NLOADS%=SZ%(1)
                DIM LOADS(NLOADS%-1,2)
                FOR I%=0 TO NLOADS%-1
                        FOR J%=0 TO 2
                                READ # INFILE%; LOADS(I%,J%)
                                NEXT J%
                        NEXT I%:T%=0
        WEND
        T%=VN%
        WHILE T%=2      REM Read points table
                IF VT%<>5 OR SZ%(0)<>3 THEN GO TO 13.103
                NPOINTS%=SZ%(1)
                DIM POINTS(NPOINTS%-1,2),VSTRESS(NPOINTS%-1)
                FOR I%=0 TO NPOINTS%-1
                        FOR J%=0 TO 2
                                READ # INFILE%; POINTS(I%,J%)
                                NEXT J%
                        NEXT I%:T%=0
        WEND
        T%=NPOINTS%*NLOADS%
        WHILE T%>0
REM Write the Output Header
                VN$="RESULTS":SZ%(0)=3
                INPUT "Number of COMMENT LINES for Output?";NC%
                IF NC%>0 THEN T%=FN.COMMENTS%
                SZ%(0)=4:FD$(3)="Pz"
                T%=FN.FORMDD%
REM Calculate the stresses at the prescribed points.
                PRINT CHR$(12);"Computed VERTICAL STRESSES:"
                FOR JJ%=0 TO NPOINTS%-1:Z=POINTS(JJ%,2):ZZ=Z*Z:T=0
                        FOR II%=0 TO NLOADS%-1
                                RR=FN.RSQ(II%,JJ%)
                                T=T+0.477464829*LOADS(II%,2) \
                                        * Z*ZZ / (RR+ZZ)^2.5
                                NEXT II%:VSTRESS(JJ%)=T:PRINT JJ%+1,T
                        NEXT JJ%
REM Write results
        FOR II%=0 TO NPOINTS%-1
                PRINT # OUTFILE%;POINTS(II%,0),POINTS(II%,1), \
                        POINTS(II%,2),VSTRESS(II%)
                NEXT II%:T%=0
        WEND
WEND
13.102  PRINT "Premature END of FILE":STOP
13.103  PRINT "INVALID DATA":STOP
END
```