ED 195 425 SE 033 586

AUTHOR	Gales, Larry; Anderson, Lougenia
TITLE	User's Guide for Subroutine FFORM. Physical Processes in Terrestrial and Aquatic Ecosystems, Computer Programs and Graphics Capabilities.
INSTITUTION	Washington Univ., Seattle. Center for Quantitative Science in Forestry, Fisheries and Wildlife.
SPONS AGENCY	National Science Foundation, Washington, D.C.
PUB DATE	May 78
GRANT	NSF-GZ-2980; NSF-SED74-17696
NOTE	15p.; For related documents, see SE 033 581-597. Contains marginal legibility in computer printouts.
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	*Biology; College Science; *Computer Assisted Instruction; *Computer Programs; Ecology; Environmental Education; Higher Education; Instructional Materials; *Interdisciplinary Approach; *Physical Sciences; Science Education; Science Instruction

ABSTRACT

These materials were designed to be used by life science students for instruction in the application of physical theory to ecosystem operation. Most modules contain computer programs which are built around a particular application of a physical process. FFORM is a portable format-free input subroutine package which simplifies the input of values to computer programs. It is especially suitable for the input of directives which control the operation of interactive programs. FFORM input is controlled mainly by the user rather than the program and features a relaxed input syntax which assigns values to program variables by names without regard to position or order, and with blanks or comments freely interspersed. This module is based on the assumption that FFORM is already incorporated as a subroutine in an existing program. Three sample runs of typical FFORM input are included in the module. (Author/CS)

USER'S GUIDE FOR SUBROUTINE FFORM

by

Larry Gales and Lougenia Anderson

This instructional module is part of a series on Physical Processes in Terrestrial and Aquatic Ecosystems supported by National Science Foundation Training Grant No. GZ-2980

May 1978

USER'S GUIDE FOR SUBROUTINE FFORM: A FORMAT-FREE INPUT SYSTEM

Identification

FFORM	- A Subroutine Package Which Handles Free-Form Input
Authors	- Larry Gales and Lougenia Anderson
Date	- April 1978, Center for Quantitative Science in Forestry, Fisheries and Wildlife, University of Washington, Seattle, Washington 98195

Purpose

FFORM is a portable format-free input subroutine package which simplifies the input of values to computer programs. It is especially suitable for the input of directives which control the operation of interactive programs. Large data sets (e.g. more than 300 cards) are better handled by other means.

Conventional (formatted) input requires that users conform to the exact specifications dictated by the program which reads it. These specifications are often difficult to follow, are intolerant of errors, require that all input demanded by the program be present, and are peculiar to each individual program, thus requiring users to learn different input procedures for different programs.

FFORM input, however, is controlled mainly by the user, not by the program, and features a relaxed input syntax which assigns values to program variables by name without regard to position or order, and with blanks or comments freely interspersed. For example, the following input sets are all equivalent and assign the values 5 and 10 to variables named TIME and VELOC, respectively, and zero out the first 10 elements in an array named ARRAY. Each input set, which may span more than one card, is terminated by a dollar sign, and comments are enclosed within slashes.

```
VELOC = 10.0, TIME = 5.0, ARRAY = 10*0 $
ARRAY(1) = 10*0.0, TIME = 5, VELOC          = 10, $
TIME/OF DAY/ = 5/O'CLOCK/, ARRAY = 0,0, 8*0,
           VELOC/ITY/ = 1E1/MPH/ $
```

FFORM makes it easy for programs to handle default values so that users need input only those values which differ from their defaults. In addition, the input procedures for all programs which use FFORM are very similar and differ mainly in terms of the names, types, and dimensions of variables required by each program.

This write-up assumes that FFORM is already incorporated as a subroutine in an existing program. For a more thorough discussion of FFORM refer to its programmer's guide (Anderson and Gales, 1978).

Input Syntax

The input for a data set consists of a sequence of [input-expression]'s (abbreviated [in-exp]) separated by commas and terminated by a dollar sign, "$" (used to indicate the end of input), as follows:

```
[in-exp], [in-exp], ..., [in-exp], $
```

where the comma between the last [in-exp] and the dollar sign is optional.

Each [in-exp] is of the form

```
[name] = [value] or
[name] [subscript-list] = [value-list]
```

where

[name] is the name of an input variable. It must contain one to six letters or digits and must start with a letter.

Examples: ENERGY, H123, P2D

[subscript-list] consists of one, two, or three subscripts separated by commas and enclosed within parentheses.

Examples: (1), (3, 4, 10), (6, 1)

If a [subscript-list] consists of more than one subscript then the first varies most rapidly, the second varies less rapidly, and the third varies least rapidly.

[value-list]
consists of one or more [value-expression]'s separated by commas. Example: 4, 16, 9*26, 2*49

[value-expression]
is either of the form [value], representing a single value, or [integer]*[value] indicating that [value] is to be repeated [integer] number of times. Example: 9*26

[value]
is either an [integer], a [real-number], a [logical-constant], or a [text-string].

[integer]
is any number of the form d•••d or -d•••d where d is a digit (0-9).

Examples: 1, 473, -5, 6252

[real-number]
is any number of the form n or -n where

```
n = d•••d                      or
    d•••d.                     or
    .d•••d                     or
    d•••d.d•••d                or
    d•••d[exp]                 or
    d•••d.[exp]                or
    d•••d.d•••d [exp]
```

and where

```
[exp] = E ± d•••d              or
          Ed•••d
```

The E indicates that a number is to be multiplied by a power of ten.

	Examples: 0.5, -.5, 3E+2, 3E2, 4.5E-12, 3
[logical constant]	is either T, .T., or TRUE indicating the logical value "true" or F, .F., or FALSE indicating the logical value "false".
[text-string]	consists of any string of alphanumeric symbols enclosed within hash marks. The hash mark symbol, "#", may not appear anywhere in the string itself. The length of the string must not exceed 68 characters as the entire string must appear on one card image within columns 1 to 72.
	Examples: #THIS IS A TEXT STRING #, #AND SO IS THIS / * () ••• #, # 1 #

As many [in-exp]'s as desired may appear on any card, and a single [in-exp] or input set may span any number of cards provided that each card ends with a comma which separates [in-exp]'s or [value-expression]'s.

In addition, comments may appear anywhere on a card except within a [name] or a [value]. Comments must be enclosed within slashes, "/", and cannot contain internal slashes.

Restrictions

The maximum number of characters in a text string is 68.

The maximum number of different variable names within any one input set is 50.

All input handled by FFORM must lie within columns 1 and 72, inclusive.

All variable names are restricted to six characters or less.

Advanced Features

FFORM contains several advanced features which are of limited use to the computer novice and which are described in the programmer's guide for FFORM.

These include an output subsystem which is fully compatible with FFORM input, the input/output of complex and double precision values, and a means of specifying sublists which provide additional control over input/output.

Error Handling

FFORM checks for 25 types of input errors. When an error occurs, it outputs an appropriate error number, underscores that part of the input card in error, and skips down to the end of the current data set (indicated by a dollar sign). Any other error processing is left up to the program which calls FFORM. The error messages are listed in the Appendix.

Sample Runs

The listings on the next few pages contain three examples of typical FFORM input. The first two runs show only the input which a user might prepare for programs which accept FFORM directives--the programs themselves are not shown. Note that the hash mark symbol, '#', which is used to delimit text strings, is printed out by the CDC 6400 computer as '≡'. The third example is included mainly for the sake of completeness a should be skipped by users who are not familiar with Fortran programs. It shows the complete setup for a typical FFORM run: system control cards (used for the NOS/BE operating system on a CDC 6400 computer), a Fortran program which accepts FFORM input, the FFORM input itself, and the results of executing the program with the given data. The parameters in the FFORM routines QQINTL, QQREAD, and QQWRIT are explained in the programmer's guide for FFORM.

```
/ RUN 1:
/ SAMPLE INPUT TO A PROGRAM (NOT SHOWN)
/ WHICH CALCULATES THE VOLUME OF A
/ CYLINDER GIVEN ITS RADIUS AND HEIGHT.
/ ASSUME THAT THE INPUT VARIABLES ARE :
/      -RADIUS-   (WHICH MUST BE A [REAL-NUMBER])
/      -HEIGHT-   (WHICH MUST BE A [REAL-NUMBER])
/      -TITLE-    (WHICH MUST BE A [TEXT-STRING])
/      -FINIS-    (WHICH MUST BE A [LOGICAL-CONSTANT]
/ -TITLE- MAY HOLD UP TO 50 CHARACTERS, AND
/ -FINIS- IS USED TO TERMINATE THE PROGRAM
/
    RADIUS = 2.45/METERS/,  HEIGHT /ALSO IN METERS/ = 1.0,
    TITLE = =THIS IS THE FIRST INPUT SET=, $
/
    TITLE = =SECOND TIME AROUND=, RADIUS = 9.0,
    HEIGHT = .4E-2,
    $
/
    FINIS = .T., $
*EOR
```

```
/ RUN 2:
/ SAMPLE INPUT TO A PROGRAM (NOT SHOWN HERE - SEE RUN 3)
/ WHICH MULTIPLIES A 3 BY 3 ELEMENT ARRAY -A- BY A
/ 3 ELEMENT VECTOR -X- YIELDING B = A * X.  ASSUME
/ THE INPUT VARIABLES ARE:
/     -A-      (WHICH MUST CONTAIN [INTEGER] VALUES)
/     -X-      (WHICH MUST CONTAIN [INTEGER] VALUES)
/     -FINIS- (WHICH MUST BE A [LOGICAL CONSTANT])
/ -FINIS- IS USED TO TERMINATE THE PROGRAM
/
     A(1,1) = 3, A(2,1) = 4*9, A(3,2) = 5*4,
     X(3) = 46, X(1) = 21, X(2) = 13, FINIS = F, $
/
     X(1) = 2*4, A(1,1) = 9*6, X(3) = 1, $
/
     A(1,3) = 3*96, X(1) = 9, A(1,1) = 6*150,
     X(2) = 2*1080, $
/
     FINIS = T, $
*EOR
```

```
AMULT.                        RUN 3
ACCOUNT,XXXXXXXX,XXXXXXXX.
COMMENT. THE FOLLOWING ARE SYSTEM CONTROL CARDS:
MNF,L=0,B=LGO.
ATTACH,BFF,ID=BFF.
LOAD,LGO.
LOAD,BFF.
EXECUTE,MULMV.
COMMENT. THE FOLLOWING IS THE PROGRAM -MULMV-:
*EOR
      PROGRAM MULMV(INPUT,OUTPUT,TAPE1,TAPE5=INPUT,TAPE6=OUTPUT)
C
C     VARIABLES READ BY FREE-FORM
C
      COMMON//       A(3,3), X(3),    FINIS
      INTEGER        A,      X
      LOGICAL        FINIS
C
C     LOCAL VARIABLES
C
      INTEGER        B(3),   ERR
C     B =            RESULT OF MULTIPLYING A * X
C     ERR =          ERROR INDICATOR (WILL BE IGNORED IN THIS RUN)
C
C     INITIALIZE FREE-FORM SYSTEM
C
      WRITE(1,1)
      CALL QQINTL(0,1H ,1,6,  ERR)
C
C      READ AND ECHO USER-S INPUT
C
   10 WRITE(6,11)
      CALL QQREAD(5,0,6,  ERR)
      CALL QQWRIT(6   )
      IF (FINIS) STOP
C
C     CALCULATE B = A * X AND PRINT B:
C
      DO 30 I = 1,3
          B(I) = 0
          DO 20 J = 1,3
              B(I) = B(I) + A(I,J) * X(J)
   20     CONTINUE
   30 CONTINUE
      WRITE(6,21) (B(I),I=1,3)
      GOTO 10
C
C     FORMATS
C
    1 FORMAT(21H INTEGER A(3,3), X(3) /19H LOGICAL FINIS $ $   )
   11 FORMAT(39H0---CURRENT VALUES OF INPUT VARIABLES:    )
   21 FORMAT(23H0    B(1), B(2), B(3) = , 3I7)
      END
*EOR
/ HERE IS THE INPUT FOR RUN 3:
/ THE ABOVE PROGRAM MULTIPLIES A 3 BY 3
/ ELEMENT ARRAY -A- BY A 3 ELEMENT VECTOR
/ -X- YIELDING B = A * X. ASSUME THE INPUT VARIABLES ARE:
/     -A-     (WHICH MUST CONTAIN [INTEGER] VALUES)
/     -X-     (WHICH MUST CONTAIN [INTEGER] VALUES)
/     -FINIS- (WHICH MUST BE A [LOGICAL CONSTANT])
/ -FINIS- IS USED TO TERMINATE THE PROGRAM
/
    A(1,1) = 3, A(2,1) = 5*9, A(1,3) = 3*4,
    X(3) = 46, X(1) = 21, X(2) = 13, FINIS = F, $
/
    X(1) = 2*4, A(1,1) = 9*6, X(3) = 1, $
/
    A(1,3) = 3*96, X(1) = 9, A(1,1) = 6*150,
    X(2) = 2*1080, $
/
    FINIS = T, $
*EOR
*EOF
```

```
---CURRENT VALUES OF INPUT VARIABLES:
A       =          3,            9,             9,            9,
                   9,            9,             4,            4,
                   4,
X       =          21,           13,            46,
FINIS   =   F,
$

   B(1), B(2), B(3) =     364    490    490

---CURRENT VALUES OF INPUT VARIABLES:
A       =          6,            6,             6,            6,
                   6,            6,             6,            6,
                   6,
X       =          4,            4,             1,
FINIS   =   F,
$

   B(1), B(2), B(3) =      54     54     54

---CURRENT VALUES OF INPUT VARIABLES:
A       =         150,          150,           150,          150,
                  150,          150,            96,           96,
                   96,
X       =          9,          1080,          1080,
FINIS   =   F,
$

   B(1), B(2), B(3) =  267030 267030 267030

---CURRENT VALUES OF INPUT VARIABLES:
A       =         150,          150,           150,          150,
                  150,          150,            96,           96,
                   96,
X       =          9,          1080,          1080,
FINIS   =   T,
$
EOR
```

References

Anderson, L., and L. E. Gales. 1978. Programmer's guide for FFORM: a format free input system. Center for Quantitative Science in Forestry, Fisheries, and Wildlife, University of Washington, Seattle, Washington.

APPENDIX

ERROR MESSAGES

ERROR #	
1	ILLEGAL CHARACTER
2	UNRECOGNIZABLE SYNTACTIC SYMBOL
3	END-OF-CARD EXPECTED
4	RESERVED WORD EXPECTED
5	VARIABLE NAME EXPECTED
6	SYMBOL TABLE OVERFLOW
7	DUPLICATE DECLARATION
8	INTEGER SUBSCRIPT EXPECTED
9	LPAREN EXPECTED
10	RPAREN EXPECTED
11	IONAME ARGUMENT TO SUBROUTINE QQINTL DOES NOT MATCH ANY IO-LIST NAME
12	UNDECLARED VARIABLE NAME
13	EQUAL SIGN EXPECTED
14	VARIABLE DOES NOT APPEAR ON CURRENT IO-LIST
15	ONE OF SUBSCRIPTS EXCEEDS DECLARED SUBSCRIPT
16	COMMA EXPECTED
17	REPEAT VALUE MUST BE AN INTEGER
18	INTEGER VALUE EXPECTED
19	ARRAY OUT OF BOUNDS
20	REAL VALUE EXPECTED
21	TEXT VALUE EXPECTED
22	LOGICAL VALUE EXPECTED
23	DOUBLE PRECISION VALUE EXPECTED
24	COMPLEX VALUE EXPECTED
25	END-OF-FILE MARK EXPECTED