CGTM NO. 137
SLAC

STEPHEN N. AU
AUGUST 1972

DATA

A FREE FIELD NUMERIC INPUT ROUTINE

ABSTRACT:

DATA is a Fortran callable subroutine which enables the user to read free field numeric data.

TABLE OF CONTENTS

1 INTRODUCTION

This routine and write-up were originally prepared by Bill Benson at the Lawrence Laboratory in Berkerly for a CDC 6600. It is coded entirely in Fortran and has been modified at SLAC to run under IBM/360 Fortran H. Users who keypunch their own data input cards may prefer the flexibility afforded by this subroutine to the rigid formats imposed by Fortran READ statements. In addition, DATA may be readily modified for special purpose applications, or to make keypunching more convenient, or to add new features, etc. See the references described below for several interesting ideas.

REFERENCES

The logic for subroutine DATA is in two layers. The inner routine (RDNUM) reads the input data cards, forms floating and fixed point numbers from the character strings, and isolates and identifies control information. This basic input routine follows very closely the routine described by A. Hassitt (DESIGN AND IMPLEMENTATION OF A GENERAL-PURPOSE INPUT ROUTINE , CACM, Volume 7, Number 6, June 1964, pp. 350-354). The outer routines interpret the control information and transmit the numbers found by RDNUM to the user's X arrays. The logic here follows that of the 7094 version of DATA written at Livermore in 1961.

RESERVED NAMES

Subroutine names are DATA, DATAI, RDNUM, and SHIFT. There are seven entry points in DATA (DATA1, DATA2, DATA3,... DATA7). REDMAN is a reserved name for a common block in DATA. These names must not be used for any other purpose. Since SHIFT is implemented only in the Fortran H compiler, the program must be compiled under the Fortran H compiler and the extended language option must be specified on the parameter list as 'XL'.

2 GENERAL DESCRIPTION

The subroutine DATA provides a flexible means to read numerical data from the input file. The input consists of the numbers to be stored and control information.

The type of number---floating or fixed---is determined by the way the number is typed on the data card. The locations where the data will be stored may be specified in the CALL DATA statement or on the input cards.

The Fortran statement

```
CALL DATAi(X1,N1,X2,N2,.....XI,NI)
```

does the following --- a list of N1 numbers are read, converted, and stored in X1(1), X1(2), ..., X1(N1). Then a list of N2 numbers are read into X2(1), X2(2),....., X2(N2), etc. When the last number has been stored, or the last argument pair has been terminated or skipped (see the use of S and C below), control returns to the next Fortran statement in the calling program. The first of each argument pair (the X) may be fixed or floating point variables or arrays. The second (the N) may be any fixed point expresssion. If some N is a fixed point variable appearing as one of the X earlier in the list, then the new value just read in will be used. If N is zero, the corresponding X will be skipped over. This is similar to the way items are entered into the list of a Fortran READ statement.

The DATA subroutine will accept a variable length argument list --- from one to seven argument pairs. The maximum length may easily be increased by modifying the subroutine DATA.

The i represents the number of input argument pairs. For example:

```
CALL DATA1(A,10)
CALL DATA3(A,10,B,2,C,4)
CALL DATA6(Q,5,R,7,S,2,T,10,U,10,V,5)
```

3 NUMBERS TO BE STORED

Decimal numbers to be stored may be punched anywhere on a card and are of the form

```
[<blanks><+ or -><digits>.<digits>E<blanks>
 <+ or -><digits>] ... <terminator>
```

where `<blanks>` means none, one or more blanks
`<digits>` means none, one or more decimal digits
`<+ or ->` means + or - or may be omitted
`.` means decimal point or may be omitted
`...` means the whole sequence may be repeated

The sequence <digits>.<digits> must contain at least one digit. E denotes exponent with base 10 and the sequence E<blanks><+ or -><digits> can be omitted. If E occurs it must follow the fractional part immediately with no intervening blanks. <terminator> denotes the terminating character which is usually a blank, but could be (1) a letter, (2) a special character (S C $ * R K), (3) a second appearance of the character . or the character E, or (4) a + or - sign following a digit. If a number has been started, then the end of a card (column 73) acts as a terminating character. Otherwise, if only blanks have appeared so far, the end of the card is ignored and a new card is read in. A number is converted to floating point format if it contains a . or an E or both. Otherwise it is stored as a Fortran fixed point integer.

Examples:

Valid Integers:

```
0
+62
-20
```

Valid Floating Point Numbers:

```
+5.9
6.8E-40
3.1412
-53E5
```

Example Of How The Same Set Of Number Can Be Typed On A Card In Various Ways:

```
6E20  1 -1  62  -20  6.8E-40   -53E5    0  +5.9
6E20,1,-1,62, -20 ,  6.8E-40,  -53E5  0   5.9,
6E20  1-1  62-20,6.8E-40-53E5,   0+5.9
6E20E1-1   62A-20,6.8E-40   -53E5+0,5.9
```

4 CONTROL INFORMATION

A series of numbers are normally stored in accordance with the CALL DATA statement, but you may override this specification by using special control symbols on the input data cards. A control symbol is none, one or more blank characters followed by one of the set S C $ * R K . In the following discussion N stands for a signed or unsigned decimal integer, and square brackets are used to set off the typographically awkward subscripts used to refer to a particular argument pair X[I], N[I]. For example X[2](1) stands for the Fortran variable X2(1). Quotes are used to set off the special control symbols S C $ * R K which stand for themselves.

S The character S can be used to terminate a list of input numbers or skip the entire list. In particular

1. If S is the first character, then the next number will be stored in X2(1).

2. If the last number stored was X[I](K), K being less than N[I], then the next number will be stored in X[I+1](1), that is, the remaining element in X[I] --- X[I](K+1), X[I](K+2)... X[I](N[I]) --- is skipped.

3. If the last number stored was the N[I]-th entry of the X[I]-th list, i.e., X[I](N[I]) or the last item in the list , then the next number stored will be X[I+2](1).

In cases 1 and 3, N successive 'S' control characters will skip N lists. In case 2, N successive 'S' control characters will skip N-1 lists.

C The control character 'C' terminates a list in exactly the same manner as 'S' does and enters the count of new entries in that list into the first location of the next list (numerical value of entry = number of elements stored in the previous list). Hence the next number to be stored is computed by subroutine DATA instead of coming from the data card.

$ Comments may be enclosed within '$' characters. Subroutine DATA ignores any intervening characters. Beware the chaos that will result if the '$' character terminating a comment is forgotten, since comments are allowed to overlap from one card to the next.

*M The following numbers will be stored in X[I](M), X[I](M+1), etc. The symbol specifically indicates the location of the number to be stored and it overrides any number stored in that location previously.

R M A number followed by 'R',M, is equivalent to repeating the number M times, this includes the first appearance of the number. In other words, an expression such as 'XXXX, R1' is equivalent to 'XXXX' and 'XXXX, R2' produces the following numbers, 'XXXX, XXXX' to be stored.

K M The increment between successive numbers stored in the array x[I] is set to M. For example, if the last number is stored in X[I](J), then succeeding ones will be stored in X[I](J+M), X[I](J+M+M) ..... The increment is reset to +1 when the next pair of arguments in the CALL statement is reached.

Only columns 1-72 are interpreted, and numbers may not overlap from one card to the next. Subroutine DATA reads one input data card at a time as needed, so that Fortran READ statements and CALL DATA statements may be interspersed as desired. Thus two CALL DATA statements cannot be used to read information from the same card, since the DATA subroutine immediately reads a new data card each time it is called (unless all N[I]=0).

5A ERROR MESSAGES

1. ERROR AT CARD XXXX, COL YY
   Generally indicates an error in control information format.

2. INPUT ERROR COLUMN XXX, CHARACTER YYY, CARD ZZZZ
   Indicates a character on the card. The offending character is YYY (decimal representation of the character).

3. INTEGER TOO LARGE
   Means that the number being formed is greater than or equal to 10**10. it is set to plus infinity (999999999) and the program proceeds.

4. DATA POSITIONED AT CARD XXXX, COLUMN YY
   Immediately before the RETURN statement in DATA is executed, the above comment is printed. This may be used to discover incorrect control information in data cards, incorrect CALL DATA statements, or programming errors in DATA itself.

5B ERROR HANDLING

This section discusses the incorrect usage of the control characters. For K, R, and *, it is necessary to associate an integer value with them. Any other information, e.g. floating point numbers, another control character, or other alphabetics will raise an error condition. In addition to the print out of an error message, the numerical representation of the control character (92 for *, 210 for K) will be stored in the list (immediately following the previous storage) except for R (the value of the previous storage is assigned instead), and the control character is ignored. For example:

```
CALL DATA1(X,10)
Data card =  7.10  R8.5, 108, 77 KR, 54.2 *7.8, 10,11.5 S

X(1)=7.10

ERROR AT CARD  1, COL 12

X(2)=7.10      (a floating point number--R8.5--instead of
               an integer)
X(3)=108
X(4)=77

ERROR AT CARD  1, COL 24

X(5)=210       (two control characters occurring
                together---KR)
X(6)=54.2

ERROR AT CARD  1, COL 35

X(7)=92       (floating point number ---*7.8)
X(8)=10
X(9)=11.5
```

In case of any illegal characters (e.g. ';', '<', '>',...) a message is printed out and any information read in after the last entry is stored and up to the encountering of the illegal character is ignored. For example,

```
Data card = 7.5, 4.1;  85

INPUT ERROR COLUMN 10 CHARACTER 96, CARD  1

In storage   X(1)=7.5       X(2)=85  (4.1 is ignored).
```

For other alphabetics or character string besides those control characters, an error message is printed out and the alphabetics or character string is ignored.

6 EXAMPLES

Example 1a.

```
CALL DATA3(X,4,Y,4,Z,4)
Card 1 = 1.  2+3.4,4E20$A+1$5.,6.R2C   *3-15S
```

```
In storage: X(1)=1.0   X(2)=2     X(3)=3.4    X(4)=4.0*10**20
            Y(1)=5.0   Y(2)=6.0   Y(3)=6.0
            Z(1)=3                Z(3)=-15

            DATA POSITIONED AT CARD 1, COLUMN 36
```

Example 1b.

```
CALL DATA3(A,4,B,4,C,4)
Card 2 =    *2K2,1.5, 2.0*1 3.C
Card 3 = S   S
```

```
In storage: A(1)=3.0   A(2)=1.5              A(4)=2.0
            B(1)=3

            DATA POSITIONED AT CARD 3, COLUMN 6
```

6 EXAMPLES

Example 2. In the following examples, deliberate errors are made and comments are enclosed by square brackets.

```
CALL DATA5(P,2,Q,5,R,7,D,1,E,9)
Card 4 =    *4 89,94.87,67E-59 R9S$IJRS$
Card 5 = XJ6.99 M E20,RK,112.98K2,4.7C

In storage:

P(4)=89
P(5)=94.87

Q(1)=67E-59
Q(2)=67E-59
Q(3)=67E-59
Q(4)=67E-59
Q(5)=67E-59

R(1)=67E-59
R(2)=67E-59
R(3)=67E-59
R(4)=67E-59

INPUT ERROR COLUMN  2 CHARACTER 231, CARD  5  [X]
INPUT ERROR COLUMN  3 CHARACTER 209, CARD  5  [J]

D(1)=6.99

INPUT ERROR COLUMN  9 CHARACTER 212, CARD  5  [M]
INPUT ERROR COLUMN 11 CHARACTER 197, CARD  5  [E]

E(1)=20

ERROR AT CARD 5 COl 16

E(2)=20          [Error in control character --RK]
E(3)=112.98
E(4)=4.7         [Control character K2 has no function
                  here because the list is skipped
                  before K2 can operate]

DATA POSITIONED AT CARD  5, COLUMN 30
```

6 EXAMPLES

Example 3.

```
CALL DATA7(AA,4,BB,5,CC,7,DD,2,EE,10,FF,6,GG,9)
Card 6 = 8.4  9.0  10  2E20
Card 7 = R2.5    3;  ,4,4.555 S
Card 8 =      *5.67 897, 1E50   >
Card 9 =           57.9  $448ABCR9$   K2 ,9,9.7   KC     C
Card 10= 1.7760954,,ABED45  CS   RK7.5 2000000000,9999.99,25E40
Card 11=   R2*5,6.9     7485   1145.9, SS

In storage:

AA(1)=8.4
AA(2)=9.0
AA(3)=10
AA(4)=2*10**20

ERROR AT CARD   7, COL  5

BB(1)=2*10**20    [Error in R2.5]

INPUT ERROR COLUMN 10 CHARACTER  94, CARD  7

BB(2)=4           [3 is ignored because of the associated
                   illegal character ';']
BB(3)=4.555

ERROR AT CARD  8 COL 11

CC(1)=92           [Error in *5.67]
CC(2)=897
CC(3)=1*10**50

INPUT ERROR COLUMN 24 CHARACTER 110, CARD  8
                  [Illegal character '>']

CC(4)=57.9
CC(5)=9
CC(7)=9.7
```

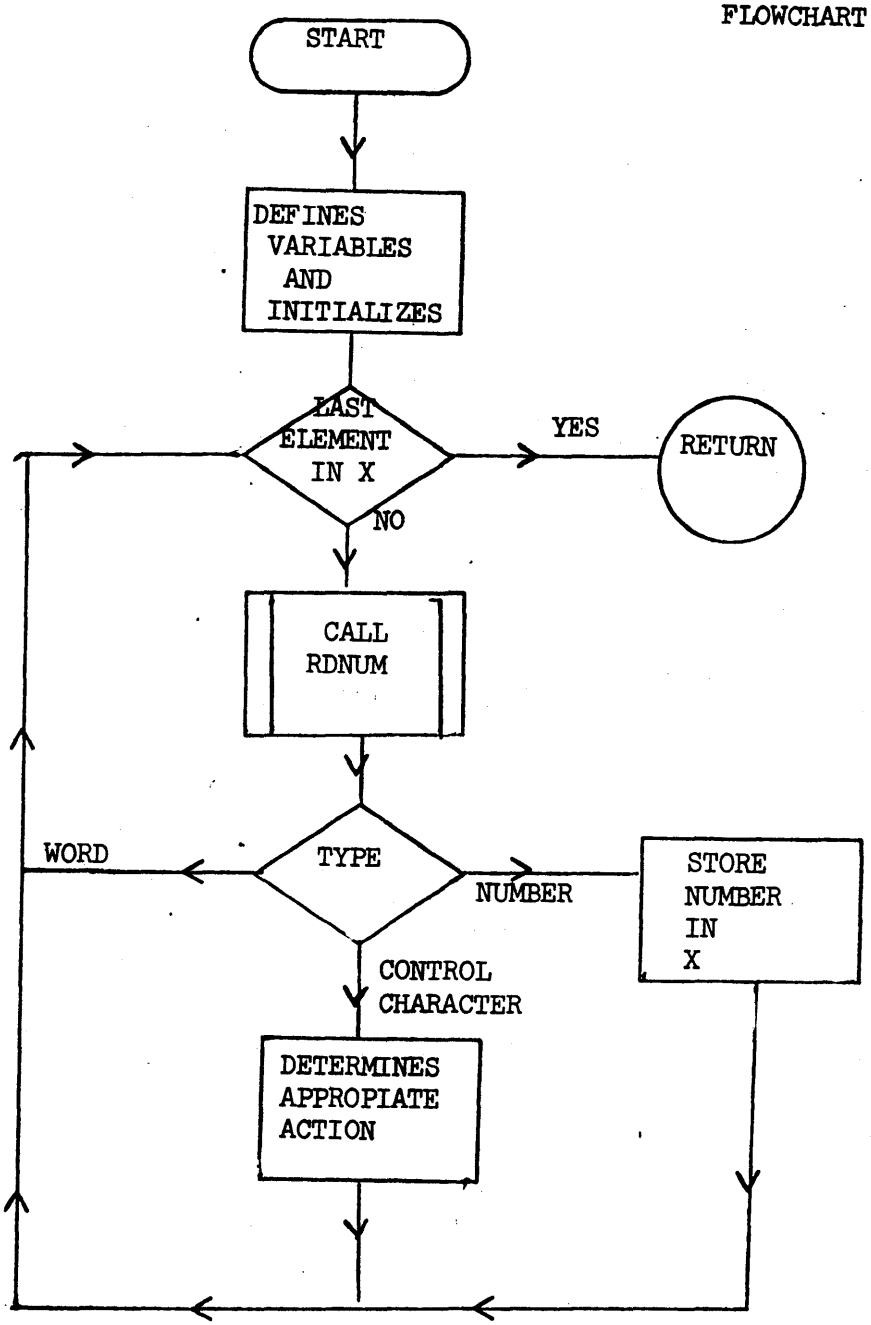
```
ERROR AT CARD  9, COL 44

CC(9)=210           [Error in control character -- KC]
                    [List DD is skipped due to control C]

EE(1)=0           [No number is stored in previous list]
EE(2)=1.7760954
EE(3)=45

FF(1)=3

ERROR AT CARD 10, COL 27

GG(1)=3             [Error in repetition factor -- RK7.5]
GG(2)=7.5

INTEGER TOO LARGE [Refers to 20000000000]

GG(3)=999999999     [Sets to infinity]
GG(4)=9999.99
GG(5)=25*10**40     [Override by 6.9 as shown below]
GG(6)=25*10**40     [Overrides by 7485 as shown below]
GG(5)=6.9           [Overrides previous storage-- *5,6.9]
GG(6)=7485          [*5,6.9 resets storage locations]
GG(7)=1145.9

DATA POSITIONED AT CARD 11, COLUMN 29
```

7 PROGRAM FLOWCHARTS

DATAI. DATAI organizes the input data into array X as directed by the control characters. Essentially, a value returned by RDNUM is analyzed; if the value is a number, it is stored in X. On the other hand, in case of a control character, the nature of the character is uncovered so as to determine further actions.

An error in the use of control information results in the printing of an error message. Execution continues, however, and diagnostics are generated by DATAI.

FLOWCHART FOR DATAI



7 PROGRAM FLOWCHARTS.

RDNUM. RDNUM reads in a single numerical data through the decision table each time it is called. The data treatment can be classified into four categories:

1. Numerical data. This includes floating point number as well as fixed point number. The number is stored in ISIT (dummy argument) and its type is set to be 1 (TYPE = 1).

2. Special character. Characters such as S, C, $, *, R and K are known as control characters throughout the entire subroutine DATA. Encountering such characters, RDNUM sets ISIT to have the numerical value of the character which has type 0.

3. Literal data (i.e. alphabetics besides control characters). An error message and the numerical representation of the constant is being printed out. No storage is provided for such data. The type is taken to be -1 and ISIT is assigned to be 0.

4. Illegal characters (e.g. '>',';','<','¬',... etc). An error message is printed out and the program reads in another piece of data.

In case 1, 2, and 3 control returns to the calling program after manipulation of data. In case 4, control attempts to read in another piece of data after error message is printed out.

FLOWCHART FOR RDNUM