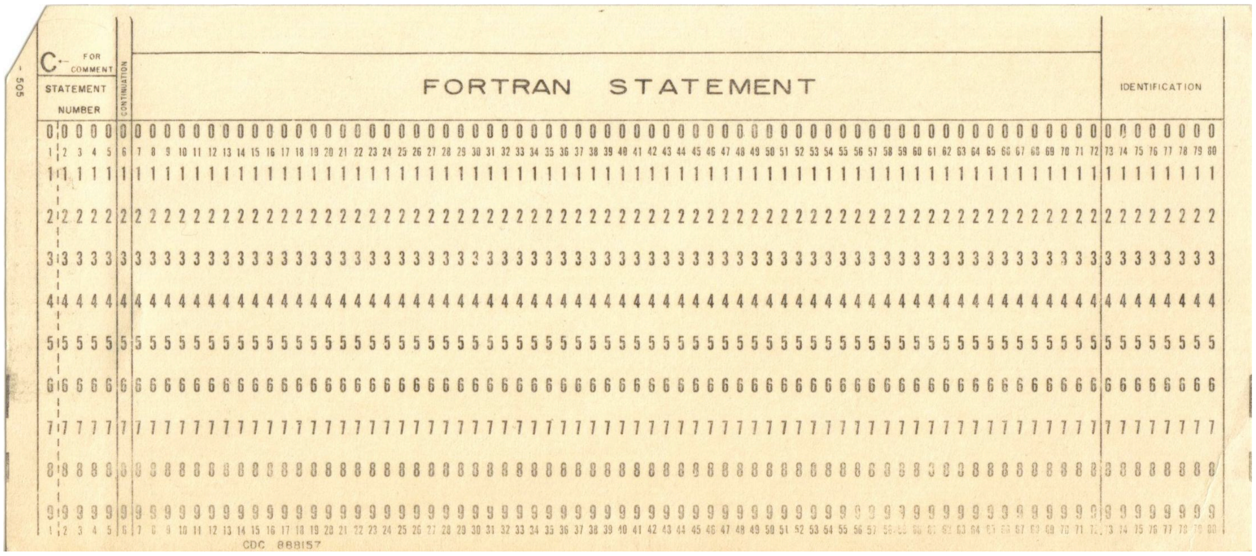
M.E. Plesha

Some Guidelines for Fortran Code

A nice online textbook for Unix and Fortran (author: Prof. Hung Leung Wong, Dept. of Civil Engineering, Univ. of Southern California) can be found at:

http://www-classes.usc.edu/engr/ce/108/

Statements: A Fortran "statement" (some old-timers call this a "card") consists of 80 characters, as follows:



Column 1: Used if the statement is a comment. Enter C or * if the statement is a comment, leave blank otherwise.

Columns 1-5: Used to assign an optional statement number. Leave blank if a statement number is not needed.

Column 6: Used if the statement is a continuation of the previous statement*. If this is the case, then place a period in this column, or 2, 3 or something like that.

Columns 7-72: Used for the instructions to be executed by the statement.

Columns 73-80: Used for an identification for the statement (optional). Most people simply leave these columns blank.

* A statement provides 66 columns for the instructions to be executed. If this is not enough, then one or more continuation statements may be used. To help with alignment of columns in a program and in data input files, it's helpful to use a font, such as `Courier`, where each character has the same length.

Comments on Fortran

1) Fortran has integer variables and arrays (INTEGER), and real variables and arrays (REAL). Unless you specify otherwise, the default naming convention is that a variable or array whose name begins with any of the characters I-N is an INTEGER, and a variable or array whose name begins with any of the characters A-H or O-Z is a REAL. Variables may contain upper case and lower case characters.

You may over-ride the default for any variables and arrays you would like. For example, imagine you have a subroutine where you want `Le` to be the length of a finite element, and `IA` to be the area moment of inertia (both REALS). This can be accomplished as follows:

```
      SUBROUTINE BEAM(X,KFIX,NOD,IADRES,MATNUM,S,FEXT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z),INTEGER(I-N)
      DOUBLE PRECISION Le,IA
C23456789 123456789 123456789 123456789 123456789 123456789
```

Notice in the first three statements above that they all begin in column 7, and the last statement is a comment that you might find handy for alignment.

2) The old Fortran standard required variable names to be six characters or less, and you will see this throughout FEMCOD. The newer standards allow variables with longer names.

3) Standard precision for REALS usually provides numbers with eight digits. Double precision usually provides numbers with 16 digits. For some compilers, these numbers are higher. You should use double precision for all REALs.

4) FEMCOD uses formatted input. This means that input information must be placed in exactly the correct columns. FEMCOD also uses formatted output, which provides the programmer great control on how printed numbers will appear, and where they will be placed. Details on the formatting that FEMCOD expects for input files are given in the FEMCOD user guide. Formatted input can be a pain sometimes, and you can use unformatted input and output if you like. For example, the following statements illustrate unformatted input and output:

```
      READ(IIN,*)R
      PI=4.*ATAN(1.)
      Area=PI*R**2
      WRITE(IOUT,*)R,Area
C23456789 123456789 123456789 123456789 123456789 123456789
```

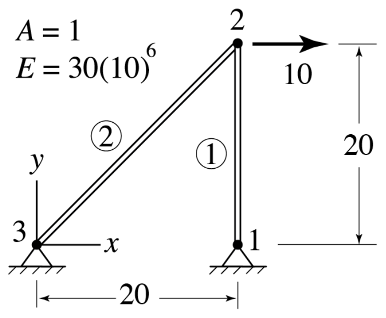
5) For debugging in FEMCOD, you should consider making liberal use of the subroutine WI (meaning *write integer*), and subroutine WR (meaning *write real*). These subroutines provide a handy way to print both the name of the variable and its value. See FEMCOD for examples of how these are used.

Illustration of Compiling and Executing FEMCOD

First, we examine the input file (twobardata) to be used for the FEA analysis shown below, using `Courier` font. This is the same model we analyzed in lecture (3 nodes, 2 dof per node, 2 elements, 2 nodes per element, 2 space dimensions, and 1 material set)

```
BAR TEST PROGRAM
    3    2    2    2    2    1
    1    11            20.          0.
                        0.          0.
    2    00            20.         20.
                      +10.          0.
    3    11             0.          0.
                        0.          0.
    1    1    2    1
    2    1    3    2
    1        1.    30.E06

123456789 123456789 123456789 123456789 123456789 123456789 123456789
```



Next, we illustrate compiling and executing FEMCOD on Plesha's Macintosh, using the gfortran compiler (available for free for MacOS, Windows, and Linux from http://gcc.gnu.org/wiki/GFortranBinaries). On the computer you use, the Fortran compiler may be called something different than gfortran.

Open the Terminal program (this is the UNIX shell underlying MacOS). Then enter the following UNIX commands:

> ls	List the contents of the current directory.
> cd "FEMCOD example folder"	Moves from the current directory to the subdirectory entitled *FEMCOD example folder*. If the directory name has spaces, then it should be enclosed in quotes, as shown here. If there are no spaces, then the quotes may be omitted. Also use quotes for a file name with spaces.
> ls	List the contents of the current directory.
> gfortran femcod.f -o femcod.o	Use gfortran to compile femcod.f and place the object code that's produced in a new file called femcod.o Most of the time, you may disregard *warnings* produced by the compiler. *Errors* need to be fixed.
> ls	List the contents of the current directory. Verify that the object code file is now present.

> ./femcod.o	Execute femcod.o
ENTER INPUT FILE NAME twobardata	FEMCOD instructs you to provide the name for the file containing input data.
ENTER OUTPUT FILE NAME twobaroutput	FEMCOD instructs you to provide the name for the file that will contain output data.
> ls	List the contents of the current directory. verify that the output file is now present.

Examine the output file, which should show the following (viewed using `Courier` font):

```
 BAR TEST PROGRAM


 PROGRAM CONTROL DATA
 --------------------
 NUMNP   =         3
 NDOF    =         2
 NUMEL   =         2
 NNPE    =         2
 NSD     =         2
 NMAT    =         1
 NEQ     =         6



 REAL AND INTEGER ARRAY MEMORY POINTERS
 --------------------------------------
 MPCORD  =         1
 MPFEXT  =         7
 MPDISP  =        13
 MPTEMP  =        19
 MPWORK  =        22
 MPSTIF  =        28
 IPKFIX  =         1
 IPNOD   =         7
 IPIADR  =        11
 IPMAT   =        17
 IPEND   =        19
 MAXREL  =     40000
 MAXINT  =      4000
```

```
NODAL DATA  (#,X,Y,Z,K1 THRU K6,F1,F2,...)
----------
   1 0.200E+02 0.000E+00 0.000E+00    110000 0.000E+00 0.000E+00
   2 0.200E+02 0.200E+02 0.000E+00    000000 0.100E+02 0.000E+00
   3 0.000E+00 0.000E+00 0.000E+00    110000 0.000E+00 0.000E+00



ELEMENT DATA  (#,MAT#,NODE1,NODE2,...)
------------
   1    1    2    1
   2    1    3    2



MATERIAL DATA  (#,RMAT1,RMAT2,...)
-------------
   1  0.1000E+01  0.3000E+08  0.0000E+00  0.0000E+00  0.0000E+00



STIFFNESS MATRIX STORAGE PARAMETERS
-----------------------------------
MBAND   =         1
LENGTH  =         7
MPEND   =        35



NODAL POINT SOLUTION  (NODE #, NODAL DOF #, VALUE OF THE DOF)
--------------------
   1    1  0.000000E+00      1    2  0.000000E+00      2    1  0.255228E-04
   2    2 -0.666667E-05      3    1  0.000000E+00      3    2  0.000000E+00



ELEMENT NO., ELEMENT STRESS, ELEMENT LOAD,
-----------------------------------------
   1   -0.1000E+02   -0.1000E+02
   2    0.1414E+02    0.1414E+02
```

After the output file is produced, look at the end of the stiffness matrix storage parameters to verify that no errors occurred during the factorization of the stiffness matrix. In particular, check to see if IPOSDF=−1 is reported ... if so, the stiffness matrix is not positive definite and there is a problem. If there is no report on the value of IPOSDF, then the stiffness matrix is positive definite; while this is certainly good, it is not a guarantee of accuracy. Also check to see if there is any warning associated with diagonal decay, which is an indication of an ill-conditioned stiffness matrix ... also not good.