Fortran Handbook

- Fortran statements are written between columns 7 and 72. Numerical statement labels are placed between columns 1 and 5 for reference purposes. An arbitrary symbol, except a blank, in column 6 implies the present line is a continuation of the previous line.

- All specification statements, e.g., specifying real or integer variables, must be placed before the executable statements, e.g., open statements, read, etc.

- Fortran variables are allocated 4 bytes (32 bits) each for real and integer types. Other variables are double precision (real*8, 8 bytes), integer*2 (2-byte integers), complex (4-byte real, 4-byte imaginary) and logical (1 bit true or false). Character variables can be of any length.

- Fortran variable names are 6 characters long but most compilers allow more. The first character must be a letter. Upper and lower case letters are used interchangeably. Fortran, unlike C, is not case sensitive.

- By default, all variable names starting with the letters A-H and O-Z are specified “real” while all those starting with I, J, K, L, M and N are specified “integer.”

- Data can be stored in a memory location by
  - An arithmetic assignment:
    
    \[
    \begin{align*}
    b &= 4.25 \\
    c &= -9.82 \\
    a &= b \times c / 2.
    \end{align*}
    \]

  - A read statement:
    
    \[
    \begin{align*}
    \text{read} & (1,10) q \\
    10 & \text{ format(f10.4)}
    \end{align*}
    \]

  - Data initialization:
    
    \[
    \text{data} \quad d/4.25/,, e,f/6.22, 4.58/
    \]

- Data can be fetched from memory, b, by
  - having b on the right side of an arithmetic expression:
    
    \[
    c = b \times 2.0
    \]

  - having b on the argument list of a subroutine call or function call:
    
    \[
    \begin{align*}
    \text{call} & \quad \text{addnum}(b,c,\text{sum}) \\
    c &= \exp(b)
    \end{align*}
    \]

  - having b on a write statement:
    
    \[
    \begin{align*}
    \text{write} & (2,3) b \\
    3 & \text{ format(}’\text{The answer b=’}, f6.2)\text{)}
    \end{align*}
    \]

- The most often used formats are: “F” or “E” formats for real (floating point) variables; “I” formats for integer variables; “A” formats for character (alphanumeric) variables, “X” for spacing and “/” for new lines.

- A simple format can be placed inside the parentheses of a read or write statement without introducing a format statement, e.g.,
  
  \[
  \begin{align*}
  \text{write} & \quad (2,’(f5.2)’) b \\
  \text{read} & \quad (*,’(a9)’) \quad \text{ssn}
  \end{align*}
  \]

- A set of fortran statements can be repeated by bracketing them in a loop. The “DO” loop is started by the DO statement and ended by a statement with a label specified directly after the DO symbol, i.e.,

\[
\begin{align*}
&\text{DO } i=1,5 \\
&\quad \text{statement 1} \\
&\quad \text{statement 2} \\
&\text{END DO}
\end{align*}
\]
do 100 i=3,11,2  
    repeating statements
100 continue

In the DO statement, \( i \) is an integer counter which begins counting at 3, ends when \( i \) becomes greater than 11, and in each loop, the value of \( i \) is incremented by 2. For this particular case, \( i=3,5,7,9 \) and 11. Thus, the loop goes around 5 times. After the loop, the value of \( i \) is most likely equal to 13, depending on the compiler used.

- The “while a condition is true” loop is accomplished by placing a statement label on the first line of the loop and ends with a conditional go to statement, i.e.,

\[
100 \text{ first line of loop} \\
\text{other repeating statements} \\
\text{if(condition)go to 100} \\
200 \text{ continue}
\]

This type of loop is more dangerous than the DO loop because there is a chance that it would never end. Be sure the condition is changed each time the loop goes around. If not, the loop may never end. You can, however, stop the program externally in UNIX by striking control-C.

- The IF statement can be written as

\[
\text{if(a.lt.0.0)a=-a} \\
\text{if(i.eq.14)write (*,*) '14 lines completed'}
\]

for one-line decisions. For a larger set of statements to be included in a decision, use the if-then-endif set as

\[
\text{if(a.lt.0.0)then} \\
\quad a=-a \\
\quad c=a*4.5 \\
\text{endif}
\]

For decisions with more than one conditions, use the if-then-else...else-endif set as

\[
\text{if(j.eq.1)then} \\
\quad \ldots \ldots \\
\text{else if(j.eq.2)then} \\
\quad \ldots \ldots \\
\text{else} \\
\quad \text{if(k.eq.6)then} \\
\quad \quad \ldots \ldots \\
\text{else} \\
\quad \quad \ldots \ldots \\
\text{endif} \\
\text{endif}
\]

- Unlike C, scalar variables in fortran do not have to be specified if their name conform to the default naming convention. However, all vector variables, i.e., arrays, must be specified; otherwise, the compiler would think the variable name is a function name instead.

\[
\text{real a(3,4),b(200),c(15)} \\
\text{integer y(10),z(10)}
\]

If the array, \( b(200) \), is not specified as shown above, the reference of \( b(i) \) later in the program would make the compiler think that a function \( b \) is being called.

- You can alter the defaulted first character variable name scheme by introducing your own:

\[
\text{implicit real (i-n)}
\]

Then all variable names starting with I through N will have the floating point type.