Blog

BLOG

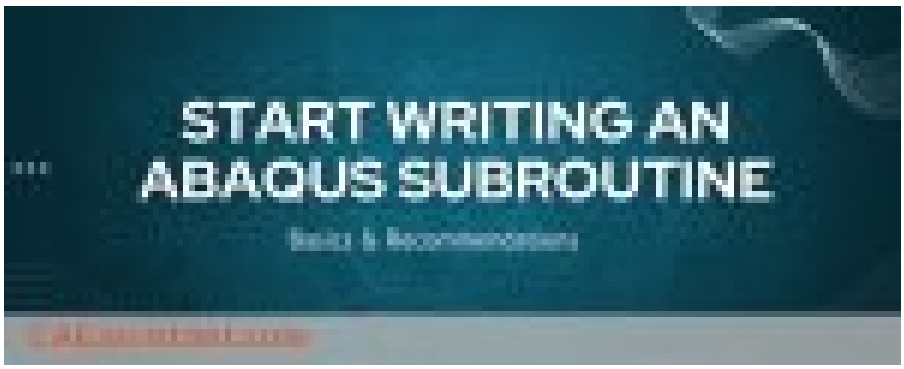
Fortran ‘Must Knows’ for Writing Subroutines in Abaqus (PART I)

Posted by Writers Of CAE Assistant Group

19 SEP

Abaqus Fortran “must knows”

In Abaqus, user subroutines are primarily written in Fortran, although developing your code in C or C++ is also possible. If you are newer to Abaqus subroutine, you should start from here:



Start Writing an Abaqus Subroutine : Basics & Recommendations

The Abaqus user subroutine allows the program to be customized for particular applications which are not available through the main ...
Continue reading

CAE Assistant 3

Fortran is one of the easiest programming languages to learn because it can do virtually nothing other than basic low-level operations common to all languages. Its syntax is also very similar to MATLAB. Furthermore, you need to know only an even smaller subset of FORTRAN to write a subroutine code in Abaqus. Here is a handy package of almost all the things you will need... Enjoy this article to know Abaqus Fortran basics.

1. Basics

A Fortran program is just a sequence of lines of text. It has to follow a specific *structure*. Take a look at this simple piece of code:


```
      PROGRAM CIRCLE
      REAL R, AREA

C THIS PROGRAM READS A REAL NUMBER R AND PRINTS
C THE AREA OF A CIRCLE WITH RADIUS R.

      WRITE (*,*) 'GIVE RADIUS R:'
      READ  (*,*) R   !THIS IS RADIUS OF THE CIRCLE
      AREA = 3.14159*R*R
      WRITE (*,*) 'AREA = ', AREA

      STOP
      END
```


Originally, all Fortran programs had to be written in all uppercase letters. However, Fortran is not case-sensitive, so “X” and “x” are the same variable.

Column Position Rules

Fortran has very few rules for how to format the source code. The most important rules are the column position rules:

» Col. 1 Blank, or a "**C**" for comments

» Col. 1-5 Statement label (optional)

» Col. 6 Continuation of the previous line (optional)

» Col. 7-72 Statements

Comments

The lines that begin with a "**C**" here are *comments* and have no purpose other than to make the program more readable. You may also use the exclamation mark (**!**) for comments. This type of comment can be any place among your piece of code (look at '**!**THIS IS RADIUS OF THE CIRCLE').

Statement label

Statement labels are used to mark positions in the program. Typically, many loops and other statements in a single program require a statement label. The programmer is responsible for assigning a unique number to each label in each program (or subprogram). Look at **4.1. Loops Section (Do Loop)** in the second part of this article (**Under Construction**) to see a practical use of statement labels.

Continuation

Sometimes, a statement does not fit into the 66 available columns (7-72) of a single line. One can then break the statement into two or more lines and use the continuation mark in position 6 (I mean, you put 5 spaces and then type **+**). Example:


```
C THE NEXT STATEMENT GOES OVER TWO PHYSICAL LINES
      AREA = 3.14159265358979
     +       * R * R
```

Any character can be used instead of the plus sign (+) as a continuation character. Also, we may use digits (using 2 for the second line, 3 for the third, and so on).

Keep going reading this post to know main points of Abaqus Fortran subroutine writing.

2. Data Types in Fortran

Variable names

Variable names in Fortran consist of the letters **A-Z**, the digits **0-9** and the underscore (_). The first character must be a letter.

Like many other languages, the words which make up the Fortran language are called *reserved words* and cannot be used as names of variables. Some examples are **PROGRAM**, **REAL**, **STOP**, **END**, etc.

Variable Types

Fortran supports several different data types to make data manipulation easier:

Data Type	Description	Example

Numerical	1		Integer	a whole number (not a fraction). can be positive, negative, or zero.	23, -32768, 0
	2		Real	can be set to a real number.	2.344, 2.01E6
	3		Complex	can be set to a complex number.	2.+i*2.5, (0,2)
Non-Numerical	4		Logical	can only be set to **.TRUE.** or **.FALSE.**	.TRUE.
	5		Character	a character or sequence of characters (a piece of text) known as a **string.** Eighter **'** or **"** can be used on both sides.	'A', "AM@67"

The most frequently used data types are **integer** and **real** (floating point).

Integers

An integer is a number that can be written without a fractional component. For example, 21, 4, 0, and −2048 are integers. Fortran has only one type for integer variables. They are usually stored as 4 bytes variables.

Floating-point variables

When a number is not an integer, it is considered a decimal. Fortran has two different types of decimal (floating point) variables, called **REAL** and **DOUBLE PRECISION**. Numerical calculations in subroutines usually need very high precision, and **DOUBLE PRECISION** should be used. Usually, a real is a 4-byte variable, and the double-precision is 8 bytes. You may use the syntax **REAL*8** to denote 8-byte floating-point variables in Abaqus subroutines.

Declarations

Every variable *should* be defined in a *declaration*. This establishes the *type* of the variable. The most common declarations are:


```
INTEGER           Variable1, Variable2,…
REAL              Variable1, Variable2,…
DOUBLE PRECISION  Variable1, Variable2,…
```


Each variable should be declared exactly once. If a variable is undeclared, Fortran uses a set of *implicit rules* to establish the type. This means all variables starting with the letters **I**, **J**, **K**, **L**, **M** and **N** are **integers** and **all others** are **real**. ***You have to be very careful with variable names.***

However, you can make a habit of putting a simple piece of code at the top of each of your subroutines:

```
      IMPLICIT NONE
```

Then, you must explicitly declare every variable that your subroutine uses or needs.

Parameters

Some constants appear many times in a program. Therefore, it is often desirable to define them only once, at the beginning of the program. This is called **PARAMETER** in Fortran:

- To reduce the number of typos (i.e., typing errors).
- To facilitate changing a constant that appears many times in a program.
- To increase the readability of the code.

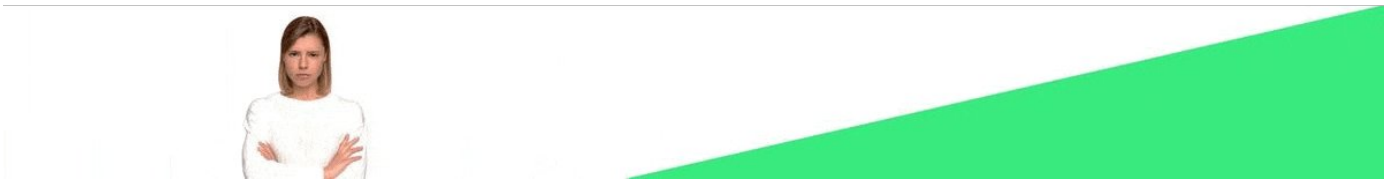
For example, we can rewrite the example code above (circle area) as:

```
      PROGRAM CIRCLE
      REAL R, AREA, PI
      PARAMETER (PI = 3.14159)
C THIS PROGRAM READS A REAL NUMBER R AND...
      .
      .
      .
      END
```


The syntax of the parameter statement is


```
      PARAMETER (name1 = value1, name2 = value2, …)
```


The PARAMETER statement(s) must come before the first executable statement.



Arrays

Many computations in Abaqus Fortran subroutines use vectors and matrices. The data type Fortran uses for representing such objects is the array. A one-dimensional array corresponds to a vector, while a two-dimensional array corresponds to a matrix.

1D arrays

It is just a linear sequence of elements stored consecutively in memory. For example,

```
      REAL A(6)
```

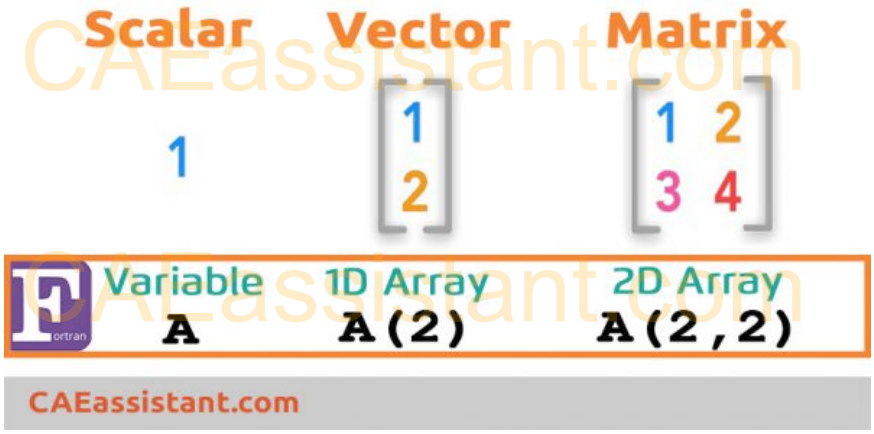
declares A as a real array of length 6. By convention, Fortran arrays are indexed from 1 and up. Each element of an array can be thought of as a separate variable. You reference the **i**th element of array **A** by **A**(**i**).

2D arrays

Matrices are very important in linear algebra. They are usually represented by two-dimensional arrays. For example, the declaration

```
      REAL A(3,5)
```

defines a two-dimensional array of 3 × 5 = 15 real numbers. It is useful to think of the first index as the row index and the second as the column index.



This article has a second part **(Fortran 'Must Knows' for Writing Subroutines in Abaqus (PART II))**

References:

1. ***FORTRAN 77 for Engineers and Scientists with an Introduction to Fortran 90***, *L. Nyhoff*

and S. Leestma, Prentice Hall

2. ***Fortran 77 Tutorial***, *Stanford University*

ABAQUS | Fortran | Subroutine | UMAT | VUMAT

NEWER

OLDER

LEAVE A REPLY

You must be logged in to post a comment.

ADDRESSES

Carrer de Jaume II ,46015,Valencia ,Spain

REON INTERNATIONAL GROUP LTD, 21 Hill Street, Unit 5, Haverfordwest, Dyfed, United Kingdom, SA61 1QQ (Sales Representative)

Enviroflex GmbH, Sterngasse 3/2/6 1010, Vienna, Austria (Sales Representative)

CAE Assistant
Your Assistant in CAE Challenges

With our assistance, consider your simulation project is done; we brought together a set of services and tutorial material to meet all your needs in CAE.

LINKS

- Contact Us
- Privacy Policy
- Terms & Conditions
- Cookie Policy
- Join Us
- FAQs
- Site Map