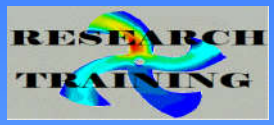


Introduction to Fortran95 Programming Part I

By Deniz Savas, CiCS, Shef. Univ., 2018

D.Savas@sheffield.ac.uk

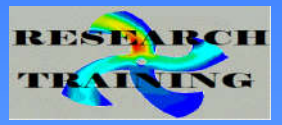
The University Of Sheffield.

RESEARCH TRAINING

Fortran Standards

- Fortran 2
- Fortran 4
- Fortran 66
- Fortran 77 : Character variables, File I/O
- Fortran 90 : Free Format Style, Object Orientation, Dynamic Storage
- Fortran 95 : Minor changes to 90 (Forall, Pure, Elemental , Structure & Pointer Default Initialisation
- Fortran 2003 : Interoperability with C/C++ , Exception Handling
- Fortran 2008 : Do-concurrent, command-line executions.

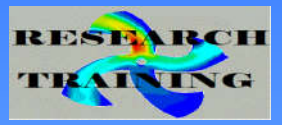
The University Of Sheffield.



Course Summary

- Writing and compiling your first program
- Anatomy of a Fortran Program
- Basic data types and variables
- Data input and output
- Program control statements

The University Of Sheffield.



Compilation

- A Fortran program is written in source form
  (human readable)
- A Fortran compiler converts source-code into a set of machine instructions and then links this with the support libraries provided by the compiler and the underlying operating system or third-party libraries.

- The executable-code can now be run and is entirely independent of the source files.

The University Of Sheffield.

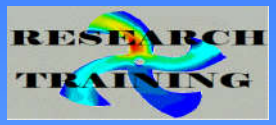
Development Steps

1) Use a text editor to write the source program
2) Compile & Link the source program
3) Run the generated executable.

- In step (2) compiling and linking are sometimes done separately
- Errors can occur during any one of these stages of development, requiring modifications to the program (step 1), followed by repeating steps 2 and 3.

The University Of Sheffield.

Selection of Compilers



- **On iceberg and sharc**

  To list what is available type: module avail compilers

  To use pgi compilers on iceberg : **module load compilers/pgi**
  To use pgi compilers on sharc : **module load dev/PGI-compilers**
  Compiler command : **pgf90**

  To use intel compilers on iceberg: **module load compilers/intel**
  To use intel compilers on sharc: **module load dev/intel-compilers**
  Compiler command : **ifort**

  To use GNU compilers on iceberg: **module load compilers/gcc**
  To use GNU compilers on sharc: **module load dev/gcc**
  Compiler command : **gfortran**

- **On Windows Platforms**

  Salford/silverfrost compiler

  Compiler command : **ftn95** (**slink** is the linker command)

The University Of Sheffield.

Exercise 1

Creating and running a program by using the PGI Fortran90 compiler

1. Copy the example programs into your directory by using the command;
   module load compilers/pgi
   cp –r /usr/local/courses/fortran .
2. Move into the directory named **intro** in the copied examples directory and edit the file named **hello.f90** by typing;
   gedit hello.f90
3. Compile and link the program in that file by typing;
   pgf90 hello.f90
4. Run the program by typing;
   ./a.out

The University Of Sheffield.

RESEARCH TRAINING

Creating and running your first program

- pgf90 command will perform the compile and link phases in one go. For example : pgf90 hello.f will compile and link and generate an executable with the default name of a.out .
- **-o** parameter can be used to give a meaningful name to the executable file. For example; pgf90 hello.f –o hello

  This will create an executable file named hello .
- The source file suffix determines the language compiler to be used (i.e. Fortran77,90,C,C++) and what action needs to be applied to the file during compilation (i.e. pre-process, compile, link)

The University Of Sheffield.

RESEARCH TRAINING

File-suffix associations

.f	Fortran77 or fixed format Fortran90 source
.f90	Free-format fortran90 source
.F , .F90	Fortran source with pre-processor instructions
.c	C source that can contain macro & preproc.
.C , .cc	C++ source that can contain macro & preproc.
.o	Object file
.a	Library of object files
.so	A library of shared-object files
.s	An assembly language file

The University Of Sheffield.

RESEARCH TRAINING

Creating and running a program

- It is possible to perform the compiling and linking stages separately as follow;
  - Compile: **pgf90 hello.f –c**
  - Link : **pgf90 hello.o**

  The compile stage generates a relocatable binary (object file) hello.o which can later be linked with other objects and libraries. Invoking pgf90 with **.o** files as its parameters initiate the link editor command ld which in turn uses the appropriate fortran libraries.
- This method is suitable for programs with the source code scattered over more than one file.
  - Example: **pgf90 –c sub1.f**

    pgf90 –c sub2.f

    pgf90 main.f sub1.o sub2.o –o myprog

The University Of Sheffield.

RESEARCH TRAINING

Pgi compiler debugger (pgdbg)

PGI provides a debugger utility for investigating misbehaving program.

To invoke the debugger, firstly compile the program with the –g debugger flag. E.g.;

pgf90 myprog.f90 –g –o myprog.exe

Next start up the debugger;

pgdbg myprog.exe

If you wish to run until the location of crashing simply type run or click the green RUN icon.

If you wish to take control from the beginning or from a certain routine click ; select Option > Locate Routine and enter the name of the routine or enter MAIN for the main program.

Once the routine is located you can click on the source window <event> column to set/remove break-points.

Select Control > Cont (or click resume icon) to continue executing the program.

You can terminate a debug session by File > Exit or by typing quit on command prompt.

The University Of Sheffield.

RESEARCH TRAINING

Source Program Formatting Style

This determines the way the program is laid out and can be one of;

FIXED FORMAT STYLE or FREE FORMAT STYLE

The compiler options –Mfixed or –Mfree can be used to force the pgf90 compiler to use one of these forms. Otherwise, default is determined from file prefix.

FIXED_FORMAT STYLE

(Fortran77 or earlier standards .f .f77 .for)

- Statements start at column seven
- Columns 1 to 5 are used for statement labels
- A character in column 6 means implies that the previous line is continuing on this line.
- The character C in column 1 implies that it is a comment line and should be ignored by the compiler.
- Statements should not exceed beyond column 72

The University Of Sheffield.

RESEARCH TRAINING

Source Program Formatting Style

FREE_FORMAT STYLE

(Fortran90 and later standards .f90 .f95)

- Each line of statement can be up to 132 characters long.
- More than one statement on a line is allowed if they are separated by ; 's Example: a=b ; c=b*d
- Ampersand (&) is used to indicate continuation. Example:

```
x = (-c + root)/ &
   &  (2.0*a )
```

- A line starting with a ! Is a comment line. Also comments can be added following a statement on the. Example:

```
sum = sum + a  ! Add to the total.
```

The University Of Sheffield.

RESEARCH TRAINING

General Layout Comments

- Blank lines are ignored by the compiler
- Spaces ‘unless in string constants’ are ignored by the compiler.
- Upper and lower case names are allowed but casing of the characters are ignored. For example: the names *Geo_Stationary1* and *GEO_STATIONARY1* both refer to the same variable.

The University Of Sheffield.

RESEARCH TRAINING

Structure of a Program

```
PROGRAM program_name
                :
  data and variable type declarations
                :
  executable statements
                :
  (optionally) internal subprograms
                :
END
```

The University Of Sheffield.

RESEARCH TRAINING

Basic data types

- Each variable or constant have a data type associated with it.
- Variables' values can be changed at any time but its data type remains the same. The following are the basic Fortran data types :

  INTEGER , REAL , LOGICAL,CHARACTER, COMPLEX
- **There are also** **user_defined_types**
- Integer variables can contain only whole numbers: `I = 42`
- Real variables can contain fractions: `PI = 3.14156`
- Character variables can only contain character strings.

  TITLE = 'Graph of Residuals'
- Logical variables can only be TRUE or FALSE

  CONVERGED = .TRUE.

The University Of Sheffield.

RESEARCH TRAINING

Variable Declarations

```
INTEGER :: index_number , levels,rank
!  three integer variables are declared
REAL :: tau    ! The strain constant
! a real variable declared, comments follows
REAL :: chi  = 1.56     ! Population index
!   a real variable declared and given an initial value
```

The University Of Sheffield.

RESEARCH TRAINING

Variable Declarations continued…

```
LOGICAL :: permit , passed , defect

COMPLEX :: voltage , current

CHARACTER*36 :: firstname , title

CHARACTER( LEN=80) :: title= ‘GRAPH-1’
```

The University Of Sheffield.

RESEARCH TRAINING

IMPLICIT statement

- In Fortran if a variable is used without being declared first then it is assumed to be an INTEGER variable if the first letter of it is I,J,K,L,M or N. Otherwise it is assumed to be a REAL variable.
- Starting with FORTRAN90 standards, users were strongly encouraged to declare all their variables.
- **IMPLICIT NONE** statement enforces this recommendation, flagging an error if a variable is used without being declared first.
- IMPLICIT NONE should come just before the declaration statements.

The University Of Sheffield.

RESEARCH TRAINING

Arrays … basics

```
REAL :: A( 20) , B(-1:5)  , C(40,40 )
INTEGER, PARAMETER :: N = 36
DOUBLE PRECISION ::  D(4,N,N)  ,  E(0:N,0:4)
INTEGER , DIMENSION(10,10) :: MX ,NX,OX
```

Note that up to 7 dimensional arrays are allowed.

Examples of referencing the above declared arrays individual elements:

X = A(12) ; i=4 ; Y=C(20, i) ; D(2, i ,i+1) = 5.678*A(i)

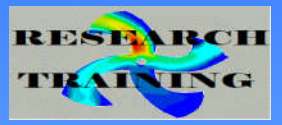
The University Of Sheffield.

RESEARCH TRAINING

Examples

```
PROGRAM VARS
 IMPLICIT NONE
 REAL :: A , B ,  ARK
 INTEGER :: I
 LOGICAL :: IS_OK
 CHARACTER*18  :: TITLE
       I = 3 ;    A = 1.0 ;   B = 1.344
      ARK = A + B
       I = I + 1
       IS_OK = ARK  .GT.  2.0
        TITLE = 'TRIALS'
 END
```

The University Of Sheffield.



Getting Information Out

- Numeric values & text strings can be output by using the **PRINT** and **WRITE** statements

```
PRINT *, N
PRINT *, 'N = ',N
```

WRITE(*,*) is the same as PRINT,*

Output can be formatted 'human-readable' or 'binary' computer representation.

The University Of Sheffield.

RESEARCH TRAINING

Reading information in

- A Fortran program can receive numbers using a **READ** statement

```
READ *,N,X
```

- Works best if a message is output first:

```
PRINT *,'Please enter N and X:'
READ * , N,X
```

The University Of Sheffield.

RESEARCH TRAINING

Exercises 1

See the file exercises.txt in the INTRO directory.

Perform exercises (1)(a) and (1)(b)

The University Of Sheffield.

RESEARCH TRAINING

Character input/output

- You must not enter non-numeric characters when the computer is expecting a real or integer number.
- On the other hand, when reading into variables defined as CHARACTER anything can be entered. However if a number is read into a character variable it is not treated as a number but as an ordinary character string.

The University Of Sheffield.

RESEARCH TRAINING

Exercises 2

Write a program that;

```
PROGRAM

REAL   :: A , B , C , S, AREA
```

- reads the length of the three sides of a triangle and

```
READ (*,* ) A , B , C
```

- calculates the area of that triangle

```
S= 0.5*(A+B+C)
AREA= SQRT(  S*(S-A)*(S-B)*(S-C)    )
```

- writes out the results

```
PRINT * , ‘  AREA OF THE TRIANGLE IS ’ ,  AREA


END
```

The University Of Sheffield.

RESEARCH TRAINING

Character input/output

```
CHARACTER*80 LINE
READ *, LINE      ! User must enter a string inside 'quote' characters
```

e.g. Enter 'Hello. It is me '

```
READ(*,100 ) LINE  ! Input does not need to be quoted
100 FORMAT( A )
```

e.g. Enter : Hello. It is me

The University Of Sheffield.

RESEARCH TRAINING

Numeric Expressions and assignment

- If you've studied algebra, you already know Fortran
  - Use * to imply multiplication.
  - Only one variable (NOT a constant) allowed on the LHS.

    Y = - A + B*X + C*X*X - 5.0
  - Use ** to mean *the power of*. X = Y ** Z
- Assignment Syntax is:

  variable = expression

  meaning the variable takes the numeric value of the expression after it is evaluated. The expression should normally evaluate to the type of the variable. If it does not, certain conversion rules may apply (e.g. integer expressions assign to a real variable will work) but don't rely on this.

The University Of Sheffield.

RESEARCH TRAINING

Built-in Functions

Many built in functions are available. For Example:

- Trigonometric and hyperbolic: **SIN , COS , TANH , COSH, ACOS**
- Exponent, logarithmic **EXP,LOG**
- Bit operations: **BTEST,IAND,IOR,IBCLR**
- Character handling:
  CHAR,IACHAR,TRIM
- Many others

The University Of Sheffield.

RESEARCH TRAINING

Example expressions and assignments

```
A = 1.0 + B
B = EXP( A) * 3.0
I = I + 1
TITLE = 'TINY'//'WEENY'  ! Concatenate strings.
GOD =  42/12
MONO = GOD.EQ.1  ! See logical expressions
```

The University Of Sheffield.

RESEARCH TRAINING

Exercises 3

- Perform Exercise 2 from the exercises sheet.

The University Of Sheffield.

RESEARCH TRAINING

Repeating ourselves

- **Do loops** are used to define a block of statements which will be executed repeatedly. The exact number of repeats are defined via the do loop variable of the DO statement. Statements to be repeated are bracketed by the DO statement and either the do-label (old syntax) or the END DO statement (new syntax).

The University Of Sheffield.

RESEARCH TRAINING

Repeating ourselves

- **Old Syntax:**

  DO *label* variable = exp1 , exp2 [, exp3]

  executable statements

  ***label* CONTINUE**

  (or ***label*** last-statement of the loop).

  For example: **DO 301 I= 1,10**

  executable statements

  301 CONTINUE

The University Of Sheffield.

RESEARCH TRAINING

Repeating ourselves

- **New Syntax:**

DO variable = exp1 , exp2 [, exp3]

executable statements

END DO

The University Of Sheffield.

RESEARCH TRAINING

Repeating ourselves

labelling the do loops

name: DO variable = exp1 , exp2 [, exp3]

executable statements

END DO name

The University Of Sheffield.

RESEARCH TRAINING

How Many Repeats ?

```
DO  I=1,10

DO  J=2,20,2

DO  K=1,20,2

DO  L=2,-7,-1
```

The University Of Sheffield.

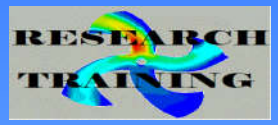
RESEARCH TRAINING

Variable number of repeats

```
READ *, n

 DO I=1,n
    etc
    etc
    etc
END DO
```

The University Of Sheffield.



Use EXIT to Leave a DO LOOP prematurely

```
DO I=1,n
  etc
  if ( … )  EXIT
  etc
  etc
END DO
```

The University Of Sheffield.

RESEARCH TRAINING

Nested DO LOOPS

```
DO  I = 1 , N
  B(I) = REAL(I/N)
  DO J = 1 , M
    A(I,J) = (I-J)/(I+J)
  END DO
END DO
```

In this example the inner loop index (J) varies fastest.

Nesting can be to any level. Above example is just for 2 levels. Do loop index variable for all levels must be different to each other to avoid ambiguity.

The University Of Sheffield.

RESEARCH TRAINING

DO WHILE loops

Repeat a loop while a logical condition holds.

Syntax: **DO WHILE (*logical expression*)**

Example:

```
SUM = 0.0
I = 0
DO WHILE ( SUM < 1.0)
    I  = I + 1
   SUM = SUM +A(I)/(A(I)+B(I) )
ENDDO
```

The University Of Sheffield.

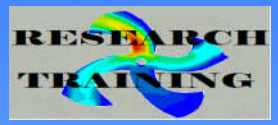
RESEARCH TRAINING

Example EXIT & CYCLE statements for DO loops

```
DO
   READ(*,*) X , Y
   IF( Y.LE.0.0 ) EXIT
   Z = X/Y
ENDDO
```

```
prim: DO  I = 1 , M
         IF( ICOUNT(I) .LE. 0 ) CYCLE prim
secon:    DO J = 1 , N
            IF ( IPOP(I,J) .LE. 0 ) CYCLE secon
                  :
          END DO  secon
       END DO prim
```

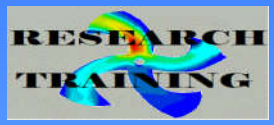
The University Of Sheffield.



Exercises 4

Perform exercise 4 from the exercises sheet.

The University Of Sheffield.



END OF PART 1