Fortran 90

presented by

Chandan Kumar Gupta

Variables, Declarations, Types

1) **integer** (in the range from about - 2 billion to + 2 billion)

2) **real** (single precision real variable)

3) **double precision** (double precision real variable)

4) **character** (string variable)

5) **complex** (complex variable)

6) **logical** (logical variable)

The layout of Fortran 90 statements

- A line can be up to 132 characters long (including spaces).
- An **&** at the end of a line indicates that the statement continues on the next line. If the item to be continued is a character constant or a format statement the next line should start with a second **&.**
- If a line contains an exclamation mark (**!**), everything from the exclamation mark to the end of the line is a comment
- Several statements can appear on a line, separated by semi-colons (**;**).

Formatting input and output

f4.2→ real number, floating point defined the decimal point between the 3rd and 4th digit. Ex 12.34

i5 → five digit integer . Ex:- 12365

a5 → five text string (character)

2x → two digits to be skipped

/ → vertical space (line feed)

d → double precision real number, exponential notation

e → single precision real number, exponential notation

t → tab indicator

E format → 1.234567 x 10^ -12 will be written as *0.1234567E-11*

write(*,10)n,x,y 10 forma t (i4,4x,f10.4,2x,f10.4)	Integer n printed using 4 places, then 4 spaces, then real numbers x and y printed with 2 spaces between, each using 10 places and 4 decimal places
write (*,20) area 20 format ("The area is ",f8.5)	string in quotes is printed, then the real number area is printed, using 8 places with 5 decimal places
write (*,30) "The area is ", area 30 format (a,f8.5)	same output as immediately above
write (*,40) x, y, z 40 format (3d20.14)	3 double precision numbers x, y, z printed, each reserving 20 spaces, with 14 decimal places
write (*,50) student, score 50 format (a20,4x,i3)	student, a text string up to 20 characters, is printed, then 4 spaces, then score, an integer using a maximum of 3 places
write (*,60) r, A 60 format (t10,f4.2,/,t10,f6.2)	tabs to column 10, prints real number r, goes to next line, tabs to column 10, prints real number A

Intrinsic functions

Function	Description
abs(x)	absolute value of x
cos(x)	cosine of x
acos(x)	arccosine of x
sin(x)	sine of x
asin(x)	arcsine of x
tan(x)	tangent of x
atan(x)	arctangent of x
cosh(x)	hyperbolic cosine of x

tanh(x)	hyperbolic tangent of x
sinh(x)	hyperbolic sine of x
dble(x)	converts x to double precision type
exp(x)	exponential function of x (base e)
log(x)	natural logarithm of x (base e)
mod(n,m)	remainder when n is divided by m
real(x)	converts x to real (single precision) type
sign(x,y)	changes the sign of x to that of y
sqrt(x)	square root of x

A simple program for format

```
program format1
implicit none
integer:: i, j
real:: x
character(len=10):: mychars
!
! Simple program to demonstrate the use of the format statement
!
write(*,*)' Enter digits 0 to 9 twice in succession '
read(*,100)mychars, i, x, j
100 format(1x, a5, i5, f6.2, i3)
write(*,*)mychars, i, j, x
stop
end program format1
```

Output of above program

```
^Cchandan@chandan:~/chandan$ gfortran format1.f90
chandan@chandan:~/chandan$ ./a.out
  Enter digits 0 to 9 twice in succession
01234567890123456789
 12345               67890               789   1234.5601
```

Arrays

- Fortran 90 allows arrays to have up to 7 dimensions.
- 3-element vector may be declared as
  real,dimension(3)::r or *real::r(3)*
- Similarly, a 2 by 3 integer matrix might be declared as:
  integer,dimension(2,3)::mymat or *integer::mymat(2,3)*

Example of arrays introduced by types of declarations:

real a(10), b(5)	one-dimensional arrays a and b of real variables, indexed from 1 to 10 and from 1 to 5, respectively
integer n(3:8), m	one-dimensional array n of integers, indexed from 3 to 8, and an integer variable m
double precision c(4,5)	two-dimensional array c of double precision real numbers, the first index running from 1 to 4, and the second from 1 to 5
character student(30)*20	one-dimensional array *student* of strings, indexed from 1 to 30, each string up to 20 symbols long
real num(0:5,1:10,-3:3)	three-dimensional array *num* of single precision real numbers, the first index running from 0 to 5, the second from 1 to 10, and the third from -3 to 3
do i = 1, 5 write (*,10) (A(i,j), j = 1, 6) end do 10 format (6f7.3)	5 rows and 6 columns of real numbers, with each row of the matrix printed on its own line :

A program for single dimension array.

```
program vector1
implicit none

real::rlength
real,dimension(3)::vect1,vect2,vectsum
write(*,*)'Enter the three components of vector 1'
read(*,*)vect1
write(*,*)'Enter the three components of vector 2'
read(*,*)vect2
vectsum=vect1+vect2
rlength=sqrt(sum(vectsum**2))
write(*,10)vectsum,rlength
10 format('the sum of vectors is :  ',3f6.2/'the length of sum of
vectors = :  ',f6.2 )
stop
end program vector1
```

Whole array operations

product	product(a)	Product of all the elements in array a
sum	sum(a)	Sum of all the elements in array a
dot_product	dot_product(v1, v2)	Dot product of vectors Vv1 and v2
matmul	matmul(a,b)	Multiply two conformal matrices a and b
maxval	maxval(a)	Maximum value in array a
minval	minval(a)	Minimum value in array a
maxloc	maxloc(a)	Array of location(s) of the maximum value in array a
minloc	minloc(a)	Array of location(s) of the minimum value in array a

Working with subsections of arrays

Selecting individual array elements

mymat(1:2,2:3) would select these elements from a 4 by 4 matrix:

```
-  *  *  -
-  *  *  -
-  -  -  -
-  -  -  -
```

mymat(3,3:4) Row 3, columns 3 - 4

mymat(3,2:) Row 3, all columns from column 2 to the end column

mymat(3,:) Row 3, all columns

mymat(3:3,3:3) Single element: row 3, column 3.

Function	Examples	Description of function
all	**all(a>1.0)** **all(a>0.0, dim=1)**	Returns logical value **.true.** if the mask condition is true for all values (in given dimension, if specified).
any	**any(a>=1.0)** **any(a>0.0,dim=1)**	Returns logical value **.true.** if the mask condition is true for any values.
count	**Count(a>0.0)**	Number of values which match the mask condition.
maxval	**maxval(a,dim=1)** **maxval(a,a<=0.0)**	Maximum value in the given dimension or among elements that match the specified mask.
minval	**minval(a,a>b)**	As maxval, but gives the minimum value.

Question on Array

Imagine that your department has asked you to write a program that records students' exam marks. Every student who scores more than 50% has passed the exam. You will need to use arrays to record the students' names, their marks and whether they have passed. The department has also asked that you calculate the total number who have passed, the average mark, and identify the person with the top mark. Write a prototype program called **results1.f90** which processes results for 5 students and produces output similar to the following:

Student: Fred Susie Tom Anita Peter

Mark: 64 57 49 71 37

Pass? P P P

No. of passes = 3

Average mark = 55.6

```
write(*,100)student
100 format('student: 'a3,'    ',a3,'    ',a3,'    ',a3,'    ',a3)
write(*,1000)marks
1000 format('marks:   ',i2,'     ',i2,'     ',i2,'     ',i2,'     ',i2)
write(*,1001)pass1
1001 format('pass?    ',a3,'     ',a3,'    ',a3,'     ',a3,'     ',a3)

write(*,10)npass,avmarks
10 format(/'no. of passes= ',i1//'Average marks= ',f6.2)
do i=1, 5, 1
if(marks(i)==maxv)then
write(*,12)student(i)
 12 format(/'Prize awarded to  ',a3)
end if
end do
stop
end program results1
```

Program for the previous question.

```
Program result1
implicit none

character(len=10),dimension(5)::student,pass1
integer,dimension(5)::marks
integer::npass,i,maxv
real::avmarks
write(*,*)'Enter the name of students'
read(*,13)student
13 format (a3)
write(*,*)'Enter the marks'
read(*,*)marks
write(*,*)'write P if marks > 50'
read(*,*)pass1
npass=count(marks>=50)
avmarks=sum(marks)/5
maxv=maxval(marks,marks>=50)
```

DO loops

```
do index=istart,iend,incr
statement 1
statement 2
statement 3
end do
```

```
program roots1
implicit none
integer::i,n
real::rooti
write(*,*)'Enter an integer'
read(*,*)n
do i=2, 2*n, 2
rooti=sqrt(real(i))
write(*,*)i,rooti
end do
stop
end program roots1
```

IF statements

if (logical expression) then
statement 1
statement 2
end if

Operator	Alternative (older) form	Example	Explanation
==	**.eq.**	if(i == j) then…	equals
>	**.gt.**	if(i.gt.j) then…	greater than
>=	**.ge.**	if(i.ge.j) then…	greater than or equal to
<	**.lt.**	if(i < j) then…	less than
<=	**.le.**	if(i<=j) then…	less than or equal to
/=	**.ne.**	if(i.ne.j) then…	not equal to
	.not.	if(.not. k) then…	**.true.** if **k** is **.false.** and **.false.** if **k** is **.true.**
	.or.	if(i>j.or.j<k) then…	logical **or**
	.and.	if(i>j.and.j<k) then…	logical **and**

DO WHILE loops

```
do while(logical expression)
statement 1
statement 2
statement 3
etc
end do
```

Implied DO loops

read(5,*)(r(i), i=1,3)

This statement is a shorthand way of telling the program **r(i)** to read in **i** taking the values 1 to 3. The structure **i=1,3** here has the same meaning as it does in any normal **do** loop.

write(6,20)((mymatrix(i, j), j=1,3), i=1,2)
20 format(3i6)

Question

Exercise

In part 3 of this course you wrote a program **results1.f90**, which recorded the exam marks of 5 students. The head of department is so pleased with your program that he would like a version that can handle large classes. Unfortunately, your original output layout is not suitable for larger numbers of students. Modify your program so that it uses a **do** loop to display the arrays in a vertical layout rather than a horizontal one, e.g:

Student:	Mark:	Pass?
Fred	64	P
Susie	57	P
Tom	49	
Anita	71	P
Peter	37	

No. of passes =	3
Average mark =	55.6
Prize awarded to	Anita

Test the program by entering marks for a larger number of students.

Program for solution

```
program results2
implicit none
character(LEN=3),dimension(5)::student,pass1
integer,dimension(5)::marks
integer::npass,i,maxv
real::avmarks

write(*,*)'Enter the name of students'
read(*,*)student
write(*,*)'Enter the marks'
read(*,*)marks

npass=count(marks>=50)
avmarks=sum(marks)/5
write(*,*)'student:          marks:          Pass?'
```

```
do i=1, 5, 1
if(marks(i)>=50)then
write(*,100)student(i),marks(i)
100 format(/a3,'                ',i2,'            P')
else
write(*,11)student(i),marks(i)
11 format(/a3,'                ',i2)
end if
end do
write(*,10)npass,avmarks
10 format(/'no. of passes= ',i1//'Average marks= ',f6.2)
do i=1, 5, 1
if(marks(i)==maxval(marks,marks>50))then
write(*,12)student(i)
12 format(/'Prize awarded to  ',a3)
end if
end do
stop
end program results2
```

Subprograms

Function

It is used by referring to their name (exactly as you would refer to the intrinsic function and they produce one answer.

Example:--

```
program fn1
implicit none
real::a,b,c,bigroot,root1,root2,test
write(*,*) 'Enter the coefficient of a,b and c '
read(*,*)a,b,c
test=b**2-4.0*a*c
root1= (-b+sqrt(test))/(2.0*a)
root2= (-b-sqrt(test))/(2.0*a)
write(*,20)root1,root2
20 format('The roots are :',2f12.6)
write(*,10)bigroot(a,b,c)
10 format('The largest root is :',f12.6)
stop
```

```
end program fn1

function bigroot(a,b,c)
implicit none
real::bigroot,a,b,c,test,root1,root2
test=b**2-4.0*a*c
if(test>=0.0) then
root1= (-b+sqrt(test))/(2.0*a)
root2= (-b-sqrt(test))/(2.0*a)
if (root2>root1) then
bigroot=root2
else
bigroot=root1
end if
else
bigroot=-9.0e35
end if
return
end function bigroot
```

Program to find factorial of a number.

```
    program factorial
 implicit none
 integer :: n, factor
 write(*,*) 'Enter the value of n '
 read(*,*)n
 write(*,10)factor(n)
10 format('The factorial of n is :',i12)
stop
end program factorial
function factor(n)
implicit none
integer::n,factor,i
factor=1
do i=1,n,1
factor=i*factor
end do
return
end function factor
```

Subroutine

It is a more general form of subprogram. They can perform complicated tasks that return one or more answers.
Alternatively, a subroutine that generates a file or some graphics might not return any answer to the main program.
The subroutine is invoked or 'called' by the call statement in the main program:

Example:--

```
program subrout1
implicit none
real::a,b,c,root1,root2
logical:: roots
write(*,*) 'Enter the coefficient of a,b and c '
read(*,*)a,b,c
call solvit(a,b,c,root1,root2,roots)
if (roots)then
write(*,20)root1,root2
20 format('The roots are :',2f12.6)
```

```
    else
    write(*,*)'sorry! There are no real roots'
    end if
    stop
    end

  subroutine solvit(a,b,c,root1,root2,roots)
  implicit none
  real::a,b,c,test,root1,root2
  logical::roots
  test=b**2-4*a*c
  if(test>=0.0) then
  root1= (-b+sqrt(test))/(2.0*a)
  root2= (-b-sqrt(test))/(2.0*a)
  roots=.true.
  else
  roots=.false.
  end if
  return
 end
```


Thank
You!