IISEE lecture for group training (Seismological course)

Fortran programming for beginner seismologists Lesson 6

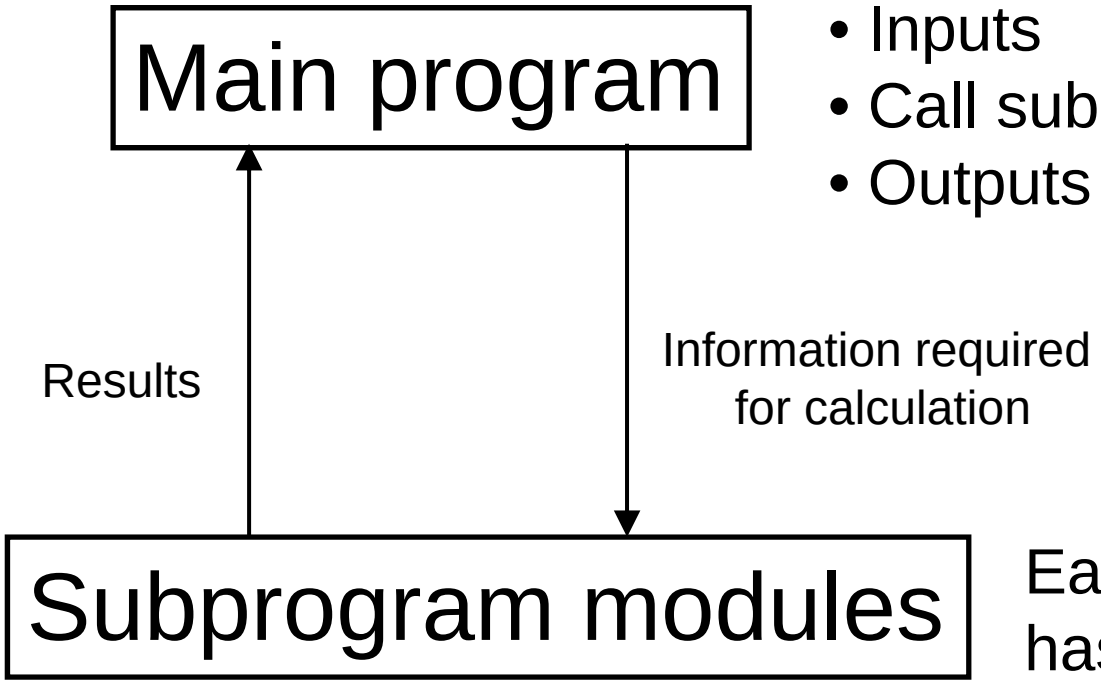
Lecturer

Tatsuhiko Hara

Subprogram modules

- It is convenient to make a subprogram module which has a specific functionality that is frequently used in calculations and analyses.
- In FORTRAN, there are two types of subprogram modules:
    - Subroutine
    - Function

Main program and subprogram modules



Main program does:

- Inputs
- Call subprogram modules
- Outputs

Each subprogram module has a certain functionality

Subroutine

The syntax of *Subroutine* statement is:

```
subroutine subr
```

or

```
subroutine subr(arg1 [, arg2, …])
```

where *subr* is a name of a subroutine, *arg1*, *arg2*, … are arguments.

Arguments transfer information from a main program to a subroutine. Also, they transfer information from a subroutine to a main program when computation returns.

Example (1)

- The following is a simple example of subroutine without arguments:

```
call hello
end

subroutine hello
write(*,*) ‘hello’
end
```

Example (2)

- The following is another simple example of subroutine without arguments:

```
call errmsg
end

subroutine errmsg
write(*,*) ‘ERROR!’
end
```

Example (3)

- The following is a simple example of subroutine with one argument. The type of the argument in the main program should be the same of that in the subroutine.

```
implicit none
real a
data a/1.0/
call samplesub3(a)
end

subroutine samplesub3(a)
real a
write(*,*) ‘a in samplesub1: ’, a
end
```

Example (4)

- This is an example of a wrong program. Note that the type of the argument in the main program is different from that in the subroutine.

```
implicit none
real a
data a/1.0/
call samplesub4(a)
end

subroutine samplesub4(a)
integer a
write(*,*) ‘a in samplesub1: ’, a
end
```

Example (5)

- In this example, the value of the argument is changed in the subroutine.

```
implicit none
real a
data a/1.0/
call samplesub5(a)
write(*,*) a
end

subroutine samplesub5(a)
real a
write(*,*) ‘a in samplesub1: ’, a
a = a*2.
end
```

Example (6)

- In this example, `a` is the input, and `b` is the output.

```
implicit none
real a, b
data a/1.0/
call samplesub6(a,b)
write(*,*) a, b
end

subroutine samplesub6(a,b)
real a, b
b = a*2.
end
```

Function

The syntax of *Function* statement is:

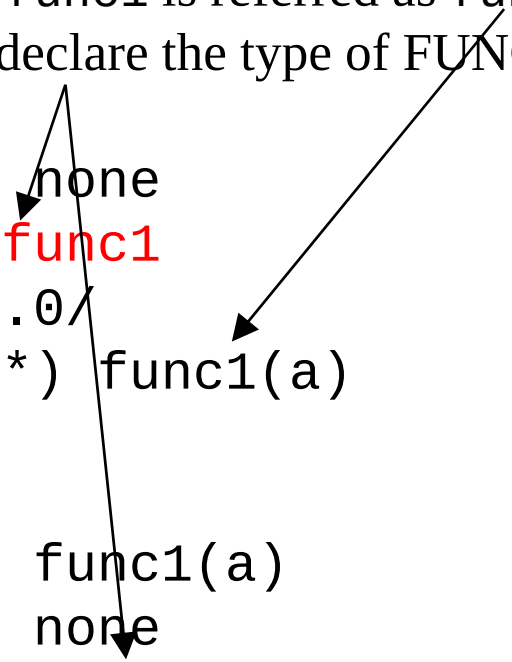
```
function func(arg1 [, arg2, …])
```

where *`func`* is a name of a function, *`arg1`*, *`arg2`*, … are arguments.

Function subprogram is referred as a variable in a main program like intrinsic functions such as `sin(x)`, `log(x)`.

Example (7)

- The FUNCTION `func1` is referred as `func1(a)`.
- It is necessary to declare the type of FUNCTION to use FUNCTION.



```
implicit none
real a, func1
data a/1.0/
write(*,*) func1(a)
end

function func1(a)
implicit none
real a, func1
func1 = a*2.
end
```

Example (8)

- Compilation of the following program fails, since the type of `func1` is not declared.

```
implicit none
real a, func1
data a/1.0/
write(*,*) func1(a)
end

function func1(a)
implicit none
real a
func1 = a*2.
end
```

Example (9)

- Another way to declare the type.

```
implicit none
real a, func1
data a/1.0/
write(*,*) func1(a)
end

real function func1(a)
implicit none
real a
func1 = a*2.
end
```

Example (10)

- This program can be compiled successfully, since *implicit typing* is active.

```
implicit none
real a, func1
data a/1.0/
write(*,*) func1(a)
end

function func1(a)
real a
func1 = a*2.
end
```

Exercise 6-1

- Let’s make a subroutine and functions to calculate epicentral distance between an earthquake and a station and azimuth of an earthquake measured from a station.
- The formula and programs are shown in the following slides. Try to run the program.

Formula

Epicentral angular distance (Δ) and azimuth (θ) of an earthquake measured form a station is given by:

$$\cos\Delta = A_E A_s + B_E B_S + C_E C_S$$

$$\sin\theta = \frac{A_S B_E - B_S\ A_E}{\sin\Delta\cos\phi_S},\quad \cos\theta = \frac{C_E - C_S\cos\Delta}{\sin\Delta\cos\phi_S}$$

where

$$A_E = \cos\phi_E\cos\lambda_E,\qquad B_E = \cos\phi_E\sin\lambda_E,\qquad C_E = \sin\phi_E$$

$$A_S = \cos\phi_S\cos\lambda_S,\qquad B_S = \cos\phi_S\sin\lambda_S,\qquad Cs = \sin\phi_S$$

and (ϕ_E, λ_E) and (ϕ_S, λ_S) are (latitude, longitude) of the earthquake and stationm, respectively.

Original program

```
      implicit none
      real phi_s, lambda_s, phi_e, lambda_e, pi, deg2rad, rad2deg
      real ae, be, ce, as, bs, cs
      real cos_delta, delta, sin_theta, cos_theta, theta
c     Constants
      pi = acos(-1.0)
      deg2rad = pi/180.
      rad2deg = 180./pi
c     Inputs
      write(*,*) 'Station latitude and longitude:'
      read(*,*) phi_s, lambda_s
      write(*,*) 'Earthquake latitude and longitude:'
      read(*,*) phi_e, lambda_e
c     Conversion from deggree to radian
      phi_s = phi_s * deg2rad
      lambda_s = lambda_s * deg2rad
      phi_e = phi_e * deg2rad
      lambda_e = lambda_e * deg2rad
c     --- Calculation starts ---
      ae = cos(phi_e) * cos(lambda_e)
      be = cos(phi_e) * sin(lambda_e)
      ce = sin(phi_e)
      as = cos(phi_s) * cos(lambda_s)
      bs = cos(phi_s) * sin(lambda_s)
      cs = sin(phi_s)
      cos_delta = ae*as + be*bs + ce*cs
      delta = acos(cos_delta)
      sin_theta = (as*be-bs*ae) / sin(delta) / cos(phi_s)
      cos_theta = (ce - cs * cos_delta) / sin(delta) / cos(phi_s)
      theta = atan2(sin_theta, cos_theta) !--- Calculation ends ---
c     Output
      delta = delta * rad2deg
      theta = theta * rad2deg
      write(*,*) 'delta theta: ', delta, theta
      end
```

Subroutine

```
      Subroutine delaz(phi_s,lambda_s,phi_e,lambda_e,delta,theta)
      implicit none
      real phi_s, lambda_s, phi_e, lambda_e, pi, deg2rad, rad2deg
      real ae, be, ce, as, bs, cs
      real cos_delta, delta, sin_theta, cos_theta, theta
c     Constants
      pi = acos(-1.0)
      deg2rad = pi/180.
      rad2deg = 180./pi
c     Inputs
      write(*,*) 'Station latitude and longitude:'
      read(*,*) phi_s, lambda_s
      write(*,*) 'Earthquake latitude and longitude:'
      read(*,*) phi_e, lambda_e
c     Conversion from deggree to radian
      phi_s = phi_s * deg2rad
      lambda_s = lambda_s * deg2rad
      phi_e = phi_e * deg2rad
      lambda_e = lambda_e * deg2rad
c     --- Calculation starts ---
      ae = cos(phi_e) * cos(lambda_e)
      be = cos(phi_e) * sin(lambda_e)
      ce = sin(phi_e)
      as = cos(phi_s) * cos(lambda_s)
      bs = cos(phi_s) * sin(lambda_s)
      cs = sin(phi_s)
      cos_delta = ae*as + be*bs + ce*cs
      delta = acos(cos_delta)
      sin_theta = (as*be-bs*ae) / sin(delta) / cos(phi_s)
      cos_theta = (ce - cs * cos_delta) / sin(delta) / cos(phi_s)
      theta = atan2(sin_theta, cos_theta) !--- Calculation ends ---
c     Output
      delta = delta * rad2deg
      theta = theta * rad2deg
      write(*,*) 'delta theta: ', delta, theta
      end
```

INPUTS

OUTPUTS

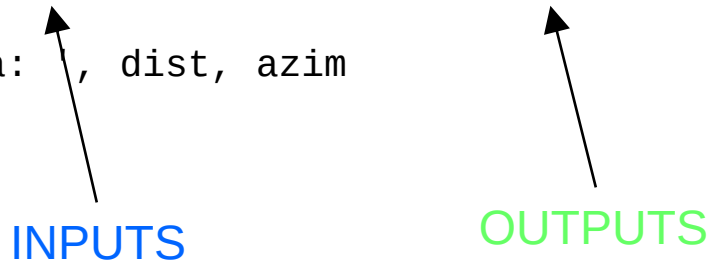
Main program

```
implicit none
real phi_s, lambda_s, phi_e, lambda_e, dist, azim

write(*,*) 'Station latitude and longitude:'
read(*,*) phi_s, lambda_s
write(*,*) 'Earthquake latitude and longitude:'
read(*,*) phi_e, lambda_e

call delaz(phi_s,lambda_s,phi_e,lambda_e,dist,azim)

write(*,*) 'delta theta: ', dist, azim
end
```



Notice that the names of variables in the main program (`dist` and `azim`) are different from those in the subroutine (`delta` and `theta`).

Function 1

OUTPUT

INPUTS

```
      Real function delta(phi_s,lambda_s,phi_e,lambda_e)
      implicit none
      real phi_s, lambda_s, phi_e, lambda_e, pi, deg2rad, rad2deg
      real ae, be, ce, as, bs, cs
      real cos_delta, delta, sin_theta, cos_theta, theta
c     Constants
      pi = acos(-1.0)
      deg2rad = pi/180.
      rad2deg = 180./pi
c     Inputs
      write(*,*) 'Station latitude and longitude:'
      read(*,*) phi_s, lambda_s
      write(*,*) 'Earthquake latitude and longitude:'
      read(*,*) phi_e, lambda_e
c     Conversion from deggree to radian
      phi_s = phi_s * deg2rad
      lambda_s = lambda_s * deg2rad
      phi_e = phi_e * deg2rad
      lambda_e = lambda_e * deg2rad
c     --- Calculation starts ---
      ae = cos(phi_e) * cos(lambda_e)
      be = cos(phi_e) * sin(lambda_e)
      ce = sin(phi_e)
      as = cos(phi_s) * cos(lambda_s)
      bs = cos(phi_s) * sin(lambda_s)
      cs = sin(phi_s)
      cos_delta = ae*as + be*bs + ce*cs
      delta = acos(cos_delta)
      !sin_theta = (as*be-bs*ae) / sin(delta) / cos(phi_s)
      !cos_theta = (ce - cs * cos_delta) / sin(delta) / cos(phi_s)
      !theta = atan2(sin_theta, cos_theta) !--- Calculation ends ---
c     Output
      delta = delta * rad2deg
      !theta = theta * rad2deg
      write(*,*) 'delta theta: ', delta, theta
      end
```

Function 2

OUTPUT

INPUTS

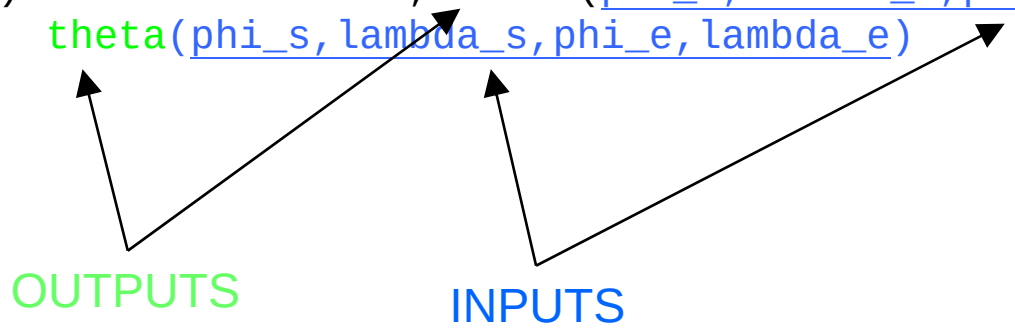
```
      Real function theta(phi_s,lambda_s,phi_e,lambda_e)
      implicit none
      real phi_s, lambda_s, phi_e, lambda_e, pi, deg2rad, rad2deg
      real ae, be, ce, as, bs, cs
      real cos_delta, delta, sin_theta, cos_theta, theta
c     Constants
      pi = acos(-1.0)
      deg2rad = pi/180.
      rad2deg = 180./pi
c     Inputs
      write(*,*) 'Station latitude and longitude:'
      read(*,*) phi_s, lambda_s
      write(*,*) 'Earthquake latitude and longitude:'
      read(*,*) phi_e, lambda_e
c     Conversion from deggree to radian
      phi_s = phi_s * deg2rad
      lambda_s = lambda_s * deg2rad
      phi_e = phi_e * deg2rad
      lambda_e = lambda_e * deg2rad
c     --- Calculation starts ---
      ae = cos(phi_e) * cos(lambda_e)
      be = cos(phi_e) * sin(lambda_e)
      ce = sin(phi_e)
      as = cos(phi_s) * cos(lambda_s)
      bs = cos(phi_s) * sin(lambda_s)
      cs = sin(phi_s)
      cos_delta = ae*as + be*bs + ce*cs
      delta = acos(cos_delta)
      sin_theta = (as*be-bs*ae) / sin(delta) / cos(phi_s)
      cos_theta = (ce - cs * cos_delta) / sin(delta) / cos(phi_s)
      theta = atan2(sin_theta, cos_theta) !--- Calculation ends ---
c     Output
      delta = delta * rad2deg
      theta = theta * rad2deg
      write(*,*) 'delta theta: ', delta, theta
      end
```

Main program

```
 implicit none
 real phi_s, lambda_s, phi_e, lambda_e, delta, theta

 write(*,*) 'Station latitude and longitude:'
 read(*,*) phi_s, lambda_s
 write(*,*) 'Earthquake latitude and longitude:'
 read(*,*) phi_e, lambda_e

 write(*,*) 'delta theta: ', delta(phi_s,lambda_s,phi_e,lambda_e),
&            theta(phi_s,lambda_s,phi_e,lambda_e)
 end
```



Function can return only a single value such as sin(x).

Passing array

- When you pass an array to a subprogram module, it is necessary to pass the name and size of the array (*adjustable dimensioning*).

Passing one dimensional array

```
implicit none
integer i
real x(3), ave
read(*,*) (x(i),i=1,3)
call cal_ave(x,3,ave)
write(*,*) ‘ave.: ‘, ave
end

subroutine cal_ave(y,n,ave)
implicit none
integer i, n
real y(n), ave
ave = 0.
do i=1, n
    ave = ave + y(i)
end do
ave = ave / float(n)
end
```

Passing multi dimensional array

- The first dimension decides orders of mapping the array elements into memory of computer.
- Therefore, if the first dimension of an array in the main program is different from that in the subprogram, the program does not work properly.

Passing Matrix (1)

```
implicit none
integer msize
parameter(msize=5)
real a(msize,msize)
integer i,j,m,n

do i=1, msize
do j=1, msize
   a(j,i) = 0.0
end do
end do

write(*,*) 'size of matrix?'
read(*,*) m,n

call readmatrix(a,msize,m,n)
```

```
do i=1, msize
   write(*,*) (a(i,j),j=1,msize)
end do

end

subroutine readmatrix(a,msize,m,n)
implicit none
integer msize
real a(msize,msize)
integer i,j,m,n

write(*,*) 'Input a matrix', m, ' by ', n
do i=1, m
   read(*,*) (a(i,j),j=1,n)
end do
end
```

Passing Matrix (2)

```
implicit none
integer msize
parameter(msize=5)
real a(msize,msize)
integer i,j,m,n

do i=1, msize
do j=1, msize
   a(j,i) = 0.0
end do
end do

write(*,*) 'size of matrix?'
read(*,*) m,n

call readmatrix(a,m,n)
```

The first dimension in the main program is different from that in the subprogram

```
do i=1, msize
   write(*,*) (a(i,j),j=1,msize)
end do


end


subroutine readmatrix(a,m,n)
implicit none
integer i,j,m,n
real a(m,n)

write(*,*) 'Input a matrix', m, ' by ', n
do i=1, m
   read(*,*) (a(i,j),j=1,n)
end do
end
```

Exercise 6-2

- Try the examples of passing arrays shown in the previous slides.