Reflection Seismology (SCPY 482)

An Introduction to Fortran 90 Programming

Chaiwoot Boonyasiriwat

September 18, 2014

Outline

- Why Fortran 90?
- Hello World
- Compilation and Execution
- Recommended Program Structure
- Basic and User-Defined Data Types
- Flow Controls (if, else, do, goto, continue)
- Input and Output
- Subroutine, Function, and Module
- More about Fortran: 95/2003/2008

Why Fortran 90?

- Easy to program compared to C
- Similar to MATLAB
- Fast compared to MATLAB
- Many codes and libraries available in Fortran
- Parallel programming with MPI/OpenMP

Drawbacks of Fortran

- Incomplete object-oriented programming
- Graphics through non-standard library, e.g. F03GL

Hello World

hello.f90

```
! Hello World Program
program hello
implicit none
write(*,*) 'Hello World'
end program hello
```

Compilation and Execution

- Fortran compilers: GNU, G95, Intel, PGI, PathScale, NAG
- Compiler command: gfortran
- Code Compilation

```
>>gfortran hello.f90 -o hello.x
```

- Program Execution

```
>>./hello.x
```

Recommend Program Structure

```
! Program description
program program_name
implicit none ! No implicit data
! Comment

...

! Comment

...
end program program_name
```

Basic Data Types

- Integer: `integer`
- Floating-point number
  - Single-precision: `real`
  - Double-precision: `double precision`
- Character: `character`
- True/False: `logical`
- Complex: `complex`

Basic Data Example

```
program simpledata
implicit none
integer                 :: n
real                    :: x
complex                 :: y
logical                 :: iszero, isempty
n = 1; x = 1.0; y = (0.0,1.0)
iszero = .false.; isempty = .true.
write(*,*) 'n = ', n, `, x = `, x
write(*,*) 'y = ', y
write(*,*) 'iszero = ', iszero
end program simpledata
```

Static and Allocatable Arrays

- A static array has a fixed size that must be declared prior to its use.
- An allocatable array can change its size anywhere in the program.

Array Data Example

```
program arraydata
implicit none
integer                    :: n(10)
integer, allocatable :: m(:)
allocate(m(5))
n = 1; n(1) = 2; n(8:10) = 5
m = 2; m(3) = 1
write(*,*) 'm = ', m
deallocate(m); allocate(m(10))
m = 1; m(8:10) = 2
write(*,*) 'm = ', m
end program arraydata
```

User-Defined Data Types

Declaration:

```
type student
    character(100) :: name
    integer        :: ID
end type student
```

Usage:

```
type(student) :: s1
s1%name = 'Jack'
s1%ID = 1234
```

Flow Control: if, else, endif

Example

```
if (a < 0) then
     b = 1
elseif (a == 0) then
     b = 2
else
     b = 3
endif
```

Flow Control: do, enddo

Example

```
do i=1,10          ! Increment by 1
    write(*,*) i
enddo
do j=1,10,2        ! Increment by 2
    write(*,*) j
enddo
do k=10,1,-1       ! Decrement by 1
    write(*,*) k
enddo
```

Flow Control: cycle, exit

Example

```
do i=1,10
     if (i == 1) then
       cycle
     elseif (i == 5) then
       exit
     endif
enddo
```

Flow Control: goto, continue

Example

```
do i=1,10
     if (i == 5) goto 100
     ...
enddo
100 continue
...
```

Command Line Data Input

Example

```
program input1
implicit none
integer :: n
write(*,*) 'Please enter a number'
read(*,*) n
write(*,*) 'n = ', n
end program input1
```

Output a Text (ASCII) File

Example

```
program output1
implicit none
open(10,file='num.txt',form='formatted')
write(10,*) 123
close(10)
end program output1
```

Output a Binary File

Example

```
program output2
implicit none
real :: a(5)
a = 1.0
open(10,file='data.bin',access='direct',recl=4*5)
write(10,rec=1) a
close(10)
end program output2
```

Input a Text (ASCII) File

Example

```
program input2
implicit none
integer :: n
open(10,file='num.txt',form='forma
tted')
read(10,*) n
close(10)
write(*,*) 'n = ', n
end program input2
```

Input a Binary File

Example

```
program input3
implicit none
real :: a(5)
open(10,file='data.bin',access='direct',recl=4*5)
read(10,rec=1) a
close(10)
write(*,*) 'a = ', a
end program input3
```

Subroutine and Function

```
program test
implicit none
integer :: a, b, c, d
a = 1; b = 2; c = add(a,b)
call subtract(a,b,d)
write(*,*) 'c = ', c
write(*,*) 'd = ', d
contains
function add(x,y)
integer :: x, y, add
add = x+y
end function add
```

Subroutine and Function

(continued)

```
subroutine subtract(x,y,z)
integer :: x, y, z
z = x-y
end subroutine subtract
end program test
```

Module: Declaration

(math.f90)

```
module math
implicit none
contains
function add(x,y)
integer :: x, y, add
add = x+y
end function add
subroutine subtract(x,y,z)
integer :: x, y, z
z = x-y
end subroutine subtract
end module math
```

Module: Usage

(main.f90)

```
program test
use math
implicit none
integer :: a, b, c, d
a = 1; b = 2; c = add(a,b)
call subtract(a,b,d)
write(*,*) 'c = ', c
write(*,*) 'd = ', d
end program test
```

Compiling Multiple Source Files

Compile source codes to object files:

```
>>gfortran -c math.f90 -o math.o
>>gfortran -c main.f90 -o main.o
```

Linking object files into an executable:

```
>>gfortran math.o main.o -o main.x
```

make and Makefile

Makefile:

```
main: object link
object: math.f90 main.f90
     gfortran -c math.f90 -o math.o
     gfortran -c main.f90 -o main.o
link: math.o main.o
     gfortran math.o main.o -o main.x
```

Run: `make`

More About Fortran

- More features have been added in the standards of Fortran 95/2003/2008