

Goals for This Lecture:

- Learn about subroutines
- Understand the behavior of the argument list
- Understand the notion of variable scope

Subprograms a.k.a. Procedures

- Two types of subprograms in FORTRAN:
 - Subroutines (invoked by a call statement)
 - Function subprograms, a.k.a. functions (invoked like an intrinsic function)
- Advantages of using subprograms:
 - Allow independent testing of parts of a code (unit testing)
 - Promotes the development of reusable code (design a subprogram to do a single task)
 - Helps isolate code from unintended side-effects (encapsulation)

Subroutines: An Toy Example

```
program sub_ex1
implicit none
real :: x, y, z
x = 1.5
y = 2.5
z = 3.5
call print_sub(x)
call print_sub(y)
call print_sub(z)
stop
end program sub_ex1
```

```
subroutine print_sub(a)
implicit none
real :: a
write(*,*) ' variable = ',a
return
end subroutine print_sub
```

Subroutines

Basic Form:

```
subroutine subroutine_name(parameter_list)  
  declaration section  
  ...  
  execution section  
  ...  
return  
end subroutine subroutine_name
```

- The parameter list is a list of variables that will be inputs to the subroutine
 - These variables are sometimes called dummy arguments
- Any changes to these variables will be seen outside the subroutine

The **CALL** statement

Basic Form:

```
call subroutine_name(argument_list)
```

- The argument list is a list of variables, constants, or expressions that will be inputs to the subroutine
- If the argument is a variable, any changes to the corresponding dummy variable made in the subroutine will be seen outside the subroutine
- Subroutines can change values of variables in the code calling the subroutine!

Argument and Parameter Lists

Very, Very Important Point:

- The argument and parameter lists must correspond in number of items and in type of items
 - Must have same number of arguments as parameters
 - Each argument must be of the same type as the corresponding parameter
- Failure to follow these two rules will result in code that compiles but executes incorrectly!

Scope and Local Variables

- Variables declared in the subroutine are unknown outside of the subroutine
- These variables are often referred to as local variables
- They exist only within the subroutine
 - We say the scope of the variable lies within the subroutine
- Changes to variables within the subroutine have no effect unless they are dummy variables
- Variables outside the subroutine can have the same name as a variable inside the subroutine but they are unrelated unless they appear as an argument/parameter pair

The INTENT attribute

- Dummy variables can be declared with the INTENT attribute which clarifies to the compiler the intention of the programmer as to whether the dummy variables should be permitted to change or not inside the subroutine

- Form:

type, INTENT(IN) :: *name*

type, INTENT(OUT) :: *name*

or

type, INTENT(INOUT) :: *name*

- Example:

```
subroutine print_sub(x)
  implicit none
  real, intent(in) :: x
  write(*,*) ' variable = ',x
  return
end subroutine print_sub
```

Reading Assignment

- Read Sections 7.1, 9.1