12.010 Computational Methods of Scientific Programming
Fall 2008

12.010 Computational Methods of Scientific Programming

Lecturers

Thomas A Herring

Chris Hill

Review and today's lecture

- So far we have covered most of the features of Fortran 77 although there are still a number of structures and methods that we have not explored.
- It is important to remember that there is nearly always multiple ways, all valid, of doing the same thing.
- Today we look at the features of Fortran90 that make it different from Fortran77. In some cases, these features are one that have commonly been available in many versions of Fortran77 (so called language extensions).

Syntax improvements

- Lines are no longer of fixed format (i.e., no need to start in column 7 and limited to column 72).
- Variable names can be longer 32 characters and _ is officially allowed in variable names.
- Any characters after a ! are treated as comments (many fortran77 compilers allow this already)
- If lines are to be continued then a & character is used at the end.
- The logical expressions >, <, ==, <=, => and /= replace the .gt. .lt. etc forms. (Note: the negating character is /)
- Multiple commands can appear on same line separated by ;
- The “do … end do” structure is official in f90

Variable attributes

- The methods of declaring variables has been extended and enhanced. The basic form is:
  <type>, <attribute> :: name=initialization
- Examples are
  integer, parameter :: n = 100
  real (kind=8), dimension(n,n) :: a,b
  (The real*8 form also works)
- For variables being passed into subroutines there is the INTENT attribute
  real, intent(in) :: real_in; real, intent(out) :: real_out
  Allows specification of how variables will be used in subroutines.
  (inout) used for both in and out variables.
- The new feature (making f90 more like c) the ALLOCTABLE attribute:
  real (kind=8), allocatable, dimension(:,:) :: matrix

Array features

- Array manipulation: *overloading* of the math symbols for array manipulation. If a, b, c are arrays, then
  c = a*b ; ! Multiply the elements of a and b and save result in c.
  All the arrays must be same size and works for multi-dimensional arrays
- The allocate(matrix(n,m), status=istat) can be used to set the size of an allocatable array; deallocate (matrix) frees the memory; and allocated(matrix) is true if matrix has already been allocated in program
- Array sections can addressed: a(n:m:inc) where inc is assumed 1 if not specified.
- Array constructors: a = (/ (i,i=1,20) /) ! The arrays must be conformable (i.e., elements on LHS and RHS must be the same. a(1:19:2) = (/ (i,i=1,20,2) /) will assign every second element

INTERFACE BLOCKS

- These are designed to ensure that subroutine and functions are consistently called

```
INTERFACE
    subroutine sub1(array,n)
    real :: array(n)
    end subroutine
end interface
```

- With the interface statements included, the compiler can check that subroutines are called with the correct arguments (incorrect arguments is a common cause of run-time segmentation violations.)
- Similar to the proto-type function in C

MODULE and USE statement

- This construct is used declare arrays that can be shared between routines and to define procedures
- Two forms: data sharing method
  MODULE test
  ! Declare data to share
  <declarations of arrays etc>
  end module test
- In subroutines and programs, immediately after program or subroutine declaration
  USE test
  allows access to the contents of the declared arrays. The SAVE statement should be used in the module declaration to ensure that memory contents is not lost.

MODULE and USE Statement

- The other form is for procedure declarations
  MODULE mysubs
  CONSTAINS
  subroutine sub1(arguments)
  <declarations and code for sub1>
  end subroutine
  end module
- In program and subroutines add
  use mysubs
  immediately after the program/subroutine statement to make mysubs routines available. In this form the interface statements are automatically generated.

Array inquiry functions

- F90 allows the sizes of arrays to be determined inside subroutines (similar to character string features in f77)
- SIZE returns the size of an array
  SIZE(array, *dim*) dim is optional and returns an array of sizes for higher dimension arrays
  SHAPE(array) returns an array with the shape (number of dimensions, and sizes) of any "array"
  LBOUND(array, *dim*) returns lower bounds on array indices
  UBOUND(array, *dim*) returns upper bounds on array indices
- SIZE can be used in a dimension statement in a subroutine so that the correct size is allocated if a copy of an array is needed

```
subroutine sub(a)
real, dimension(:) :: a
real, dimension(size(a)) :: b
b = a
```

Array transformation function

- F90 supports a number of intrinsic function that allow manipulations of arrays.
  Array construction functions
  - o SPREAD, PACK, RESHAPE, ...

  Vector and matrix multiplication
  - o DOT_PRODUCT, MATMUL

  Reduction functions
  - o SUM, PRODUCT, COUNT, MAXVAL, ANY, ALL...

  Geometric location functions
  - o MAXLOC, MINLOC

  Array manipulation functions
  - o CSHIFT, EOSHIFT, TRANSPOSE
- A fortran 90 manual will explain the uses of these functions. One feature is that they all allow a logical MASK to be specified that sets the elements to be operated on. (Similar to some MATLAB features).

KEYWORD and OPTIONAL arguments

- When interface or module structure is used, subroutines can be called with argument names:
  real function calc (first, second, third) ! In module
  then usage can be:
  a = calc (second = 2., first=1., third = 3.)
- This can be powerful when combined with the OPTIONAL attribute in the function/subroutine declarations of variables i.e., in function
  real, intent(in), optional :: third
  PRESENT(third) will be true of third argument was passed.

Summary of f90 changes

- Many of the changes in f90 reflect the growing need to keep modules consistent and to allow better compiler detection of problems in the code.
- The array manipulation features allow more compact code to be written
- In f90 arrays are best thought of as “objects” which carry not only data in them but also information about what the array is.
- The concepts of objects will appear again in c++ and matlab.