12.010 Computational Methods of Scientific Programming

Lecture 15: Fortran

Summary

- Fortran: Programming concepts and syntax
  - Codes for demonstration
    - ifs.f90: Branching structures
    - loops.f90: Types of iterator code
    - subs.f90: Functions and subroutines
    - vars.f90: Things not to do when passing variables (may need -fallow-argument-mismatch option passed to gfortran compiler)

Fortran: Formula Translation

- Start examining the FORTRAN language
- Development of the language
- “Philosophy” of language: Why is FORTRAN still used (other than you can’t teach an old dog new tricks)
- Basic structure of its commands
- Communication inside a program and with users
- A summary of Fortran 77 is available from (also posted): http://www.math.utah.edu/~beebe/software/fortran-documentation/ftnsum.pdf
- Modern flavor is Fortran 95 to Fortran 2008.

FORTRAN (Formula Translation).

- History
  - Developed between 1954-1957 at IBM
  - FORTRAN II released in 1958
  - FORTRAN IV was released in 1962 (standard for the next 15 years)
  - FORTRAN 66 ANSI standard (basically FORTRAN IV).
  - FORTRAN 77 standard in 1977
  - Fortran 90 Added new features in 1997 Fortran 95 was released (started phasing out some FORTRAN 77 features).
  - Fortran 90 introduced new concepts of modules and array tracking
  - Fortran 2000 and 2008 have been released. These updates tend to be incremental and tightening of standards.

FORTRAN Philosophy

- FORTRAN developed at a time when computers needed to do numerical calculations.
- Its design is based on the idea of having modules (subroutines and functions) that do calculations with variables (that contain numeric values) and return the results of those calculations.
- FORTRAN programs are a series of modules that do calculations, with typically the results of one module passed to the next.
- Usually, programs need some type of IO to get values to computations with and to return the results.

FORTRAN

- Commands are divided into executable and non-executable ones. All non-executable commands must appear before the executable ones.
- Between commands, lines that start with ! are considered "comments" and are ignored. (C and * work as comment lines in Fortran 77)
- Basic Structure:
  - Module types:
    - Program (only one per program, optional)
    - Subroutine — Multi-argument return
    - Function —single return (although not forced)
    - Module (Fortran 90 and above)

Variables

- All results of calculations must be saved and saved in quantities called “variables”. In most Fortran programs, there is no distinction between the memory location of a variable (pointer) and its contents (the values stored). The address is the start of bytes associated with the variable. Number of bytes and interpretation depends on the declared type.
- Implicit types: i-n integer, all others real. Use IMPLICIT NONE
- Case insensitive unless specified.
- Variable types (Fortran 77 style)
  - Integer*n where n is number of bytes (2/4)
  - Real*n where n is 4 or 8
  - Complex*n where n is 4 or 8
  - Character*(m) where m is number of characters in string variable
  - Logical*n is n is 2 or 4
  - Byte (equivalent to integer*1)

Declaring types

- Fortran 90+ has more verbose and specific declarations.
  TYPE, options, attributes :: name = <initial value>
- TYPE: Integer, real, complex, character
- Options:
  - (KIND=<#bytes) ! Sets precision, range
  - DIMENSION(n,m,k) ! Set array sizes
  - LEN=nn ! Length of character strings (padded with spaces, ASCII 32).
  - PARAMETER ! Fixed value that can't changed
  - INTENT (IN,OUT,INOUT) ! Used in functions and subroutines

Fortran: I/O and operators.

- Constants: Numerical, strings or defined in parameter statement
- I/O
  - Read (various forms of this command)
  - Write (again various forms)
  - Print (useful for debug output)
  - Command line arguments (getarg intrinsic)
  - Format defines how results are read and written.
- **Math symbols**: * / + - ** (power) = (assignment). Operations in parentheses are executed first, then **. * and / have equal precedence (left to right), + - equal precedence left to right.
- Array calculations are element by element (F90 and above) with intrinsic functions for matrix multiplies etc.

Fortran control and communications*

- Control
  - If statement (various forms): ifs.90
  - Do statement (looping control, various forms): Loops.f90
  - Forall (F90 and above for setting array elements).
- Termination
  - End (appears at the end of every module)
  - Return (usually at the end of modules, except program; does not take an argument)
- Communication between modules
  - Variables passed in module calls. Two forms:
    - Pass by address (memory address of the variable is passed)
    - Pass by value (actual value passed, rarer and usually only applies when constants or expressions to be evaluated are passed)

Communication

- Communications between modules
  - Return from functions
  - Common blocks (specially assigned variables that are available to all modules)
  - F90 : MODULE with save and USE module.
  - Save (ensures modules remember values)
  - Data statement presets values before execution (during compilation)
  - Parameter (method for setting constants).

Other types of commands

- Other types of commands
  - Open (opens a device for IO)
  - Close (closes a device for IO)
  - Inquire (checks the status of a device)
  - Backspace and rewind change the position in a device, usually a file).
  - External (maybe later)
  - Include (includes a file in source code)
  - Implicit

Syntax

- Fortran 77: Relatively rigid (based on punched cards)
  - Commands are in columns 7-72 (option exists for 132 columns but not universal).
  - Labels (numerical only) columns 1-5
  - Column 6 is used for “continuation” symbol for lines longer than 72 characters.
  - Case is ignored in program compilation (but strings are case sensitive, i.e., a does not equal A)
  - Spaces are ignored during compilation (can cause strange messages)
- Fortran 90: Free format with & to continue lines

CODE Layout

- Modules are:
  - Program
  - Subroutine – returns values through calling arguments or modules.
  - Function – returns a value as the function name
  - The last line is always END (may have name added)
- Layout:
  - Name
  - Use statements (F90)
  - Declarations (variable definitions)
  - Data/save statements
  - Code
- Unlike other languages, declarations and code must be separated.

Program Layout

- Actual layout of program in memory depends on machine (reason why some “buggy” programs will run on one machine but not others).
- Typically executable layout in memory.

MEMORY
Program and subroutines
Variables/Constant/ Common and data
Stack (changes size during execution)
Other programs etc.

- Not all machines use a Stack which is a place in memory that temporarily stores module variables.
- It is good practice to assume stack will be used, and that memory is “dirty.” i.e., random values are in the memory.

Compiling and linking

- Source code is created in a text editor.
- Fortran compilers come with different names (e.g., f77 fort, gfortran). The most common open-source version is gfortran from the GNU foundation
- To compile and link:
  - gfortran <options> prog.f90 subr.f90 libraries.a -o prog
- Where prog.f90 is the main program plus maybe functions and subroutines
  - subr.f90 are more subroutines and functions
  - libraries.a are indexed libraries of subroutines and functions (see ranlib)
  - prog is name of executable program to run.
- <options> depend on specific machine (see man gfortran or gfortran --help, some compilers use just -help)

Functions

- Real*8 function func(list of variables)
- Invoked with
  - Result = func(same list of variable types)
- Example: Value returned as function name
  - Real*8 function eval(i,value)
  - Integer*4 I
  - Real*8 value
  - eval = I*value
- In main program or subroutine or function
  - Real*8 result, eval
  - Integer*4 j
  - Real*8 sum
  - Result = eval(j,sum)

subroutines

- Subroutines normally return more than one value or array.
- In general, Fortran “passes” variables to a subroutine or function by address. The subroutine is then able to change the values stored at that address.
- Subroutines are used to “break” the program into logical pieces e.g., subroutines for initialization, input, computational elements, and output. Each piece can be coded and tested separately, with the overall flow coordinated.
- When codes are compiled from separate source files, there is no check that variable types and/or dimensions match between the program and subroutines.

Functions (subs.f90/vars.f90*)

- Functions can be of any of the variable types
- The last action of the function is to set its name to the final result
- The function type must be declared in the main program (looks like any other variable)
- There are other forms of the declaration. Often a function is used and the type is declared in the function.
- The function must always appear with the same name and type.
- Fortran has special functions called intrinsic, which do not need to be declared.
- Passing of arguments: vars.f90 (not the way to do code!).

Summary

- There are many more features for Fortran:
- It advantages are numeric speed; handling of matrices and arrays (compared to C), simple memory layout with defined structures
- Still common due to heritage and parallel efficiency.

MIT OpenCourseWare

https://ocw.mit.edu

12.010 Computational Methods of Scientific Programming, Fall 2024