Numerical Modelling in FORTRAN day 1

Paul Tackley, 2017

Today’s Goals

- Review class structure (see http://www.gfd.geophys.ethz.ch/%7Epjt/FORTRAN/FortranClass.html)
- Review background/history of Fortran
- Example programs to introduce the basics
- Edit, compile and test simple programs

Project
(optional, 1 KP)

1. Chosen topic, agreed upon with me (suggestions given, also ask the advisor of your MSc or PhD project).
   - Due end of Semesterprüfung (16 Feb 2018)
   - Start planning soon!

Project: general guidelines

- Choose something either
  - related to your research project and/or
  - that you are interested in
- Effort: 1 KP => 30 hours. About 4 days' work.
- I can supply information about needed equations and numerical methods that we have not covered

Some ideas for a project

- Involving solving partial differential equations on a grid (like the convection program)
  - Wave propagation
  - Porous flow (groundwater or partial melt)
  - Variable-viscosity Stokes flow
  - Shallow-water equations
  - 3-D version of convection code
- Involving other techniques
  - Spectral analysis and/or filtering
  - Principle component analysis (multivariate data)
  - Inversion of data for model parameters
  - N-body gravitational interaction (orbits, formation of solar system, ...)
  - Interpolation of irregularly-sampled data onto a regular grid

History of Fortran

FORmulaTRANslation

(see http://en.wikipedia.org/wiki/Fortran)

- Invented by John Backus at IBM in 1953
  - *"Much of my work has come from being lazy. I didn't like writing programs, and so, when I was working on the IBM 701, writing programs for computing missile trajectories, I started work on a programming system to make it easier to write programs."*
- First compiler 1957
  - First widely-used high-level language
- Standardised (by ANSI) in 1966: FORTRAN 66
- New ANSI standard in 1978: FORTRAN 77.
  - Became out of date: many features lacking compared to Pascal, C, Ada etc.
- New standard in 1992: FORTRAN 90.

History (2)

- FORTRAN 90: A big leap forwards!
  - free source format, modern control structures, precision specification, whole array processing (like MATLAB), dynamic arrays, user defined types, modules. But backward-compatible with F77
- FORTRAN 95 (1996): A few small fixes & improvements.
- FORTRAN 2003: Major additions mainly related to object-oriented programming
- FORTRAN 2008: Minor improvements
- FORTRAN 2015: planned; minor upgrades

Recommendation

- Use Fortran95 not 77 for new codes
  - f2003 is not yet fully implemented by all compilers
- f95 even has some advantages to C++
  - easier to understand, learn and write (typically)
  - easier to debug (e.g., no worry about pointers)
  - codes run faster (usually)
  - built-in complex numbers, array operations, multidimensional arrays, etc.
  - built-in parallel computing constructs & versions
  - doesn’t have such advanced object oriented programming but this is addressed with f2003&2008

Example program 1

```
program Temp_conversion

  implicit none
  real :: Deg_F, Deg_C, K

  print*,"Please type in the temperature in F"
  read*, Deg_F

  Deg_C = 5.*(Deg_F-32.)/9.
  print*,"This is equal to",Deg_C,"C"

  K = Deg_C + 273.15
  print*, "and", K , "K"

end program Temp_conversion
```

Analysis

- program....end program delineates the code
- Specification of variables comes first
  - **implicit none** means that all variables must be explicitly declared. Optional, but helps to avoid bugs (problems)
  - **real** is one of 5 variable types (also integer, logical, character, complex)
- Execution part comes next
  - print* and read* are the simplest I/O operations to stdout and stdin (often the screen and keyboard)

Notes

- Case doesn’t matter: e.g., PROGRAM, program, PrOgRaM all mean the same, deg_c and Deg_C refer to the same variable
- Doesn’t matter what column statements start (indent for legibility)
- Extra lines, and spaces within lines, can be added to improve legibility

f77 version

```
      program Temp_conversion

      implicit none
      real  Deg_F, Deg_C, K

      print*,"Please type in the temperature in F"
      read*, Deg_F

      Deg_C = 5.*(Deg_F-32.)/9.
      print*,"This is equal to",Deg_C,"C"

      K = Deg_C + 273.15
      print*, "and", K , "K"

      end
```

- Not much difference because simple
- statements have to begin in >=7th column
- No "::" in variable declarations

EXERCISE 1

- write, compile and run a simple program that writes a message to the screen (e.g., “Hello World”)
  - edit a file ending in “.f90” (or “.f95”), using a text editor like emacs, vi, etc.
  - on linux or macosx, compile using “ifort program.f90” or “gfortran program.f90”. This will make an executable “a.out”
    - To specify a different name use -o, e.g.,
    - “gfortran –o myname program.f90”
  - Type “a.out” to execute it
    - If the computer doesn’t find it type “./a.out”

Beware of integer constants!

- If you write numbers without a "." you may get unexpected results:
- 1/3 = 0
- 1./3.=0.33333
- 1.0/3.0=0.33333

EXERCISE 2

- Write a program that asks the user to input three real numbers, then calculates the arithmetic mean, harmonic mean and geometric mean and prints them to the screen.

Example Program 2

```
program loopdemo

  implicit none
  integer :: i
  integer,parameter :: low=3, high=5

  ! This program does nothing useful

  do i = 1,10        ! repeats loop with i=1,2,3...10

     if (i>high) then
        print*,i," is greater than 5"
     else if (i<=low) then
        print*,i," is less than or equal to 3"
     else
        print*,i," is nothing special"
     end if

  end do

end program loopdemo
```

Notes

- the "**parameter**" label indicates that these things have a fixed value that cannot be altered later
- "!" indicates a comment: everything after it is ignored
- The **do...end do** loop construct
- The **if...else if...else...end if** construct
- Comparison operators are <, >, ==, /=, >=, <=
  - in f77 these are .lt. .gt. .eq. .ne. .ge. .le.
- Indentation helps see the structure
- So do editors that auto-colour (xemacs in this case)

f77 version

```
      program loopdemo

      implicit none
      integer i, low, high
      parameter (low=3, high=5)

c     This program does nothing useful

      do i = 1,10                   ! repeats loop with i=1,2,3...10

         if (i.gt.high) then
            print*,i," is greater than 5"
         else if (i.le.low) then
            print*,i," is less than 3"
         else
            print*,i," is nothing special"
         end if

      end do

      end
```

- ‘c’ in column 1 indicates a comment line
- ‘!’ comments and do...end do are not strict f77 but are supported by all modern compilers

Notes

- So far, 2 types of variable:
  - **Real**: floating-point number
  - **Integer**: whole number
- Soon we will come across 2 more types:
  - **Logical**: either .true. or .false.
  - **Character**: one or more characters
- And there is also
  - **Complex**: has real and imaginary parts

Exercise 3

- Write and test a program that
  - asks the user for a positive integer number
  - checks whether it is positive and prints an error message if not
  - calculates the factorial of the number
  - prints the result to the screen

Homework (due next class)

- Finish the 3 exercises
- Study the section “Introduction and Basic Fortran” at http://www.cs.mtu.edu/%7eshene/COURSES/cs201/NOTES/intro.html
- write a 4th program as specified on the next page, and hand in all programs by email. Deadline: next class
- Send f90 files to ETHfortran@gmail.com

Exercise 4

- Write a program that calculates the mean and standard deviation of an series of (real) numbers that the user inputs
  - Ask the the user to input how many numbers (this will be an integer)
  - check the user has input a positive number
  - input each number in turn (these will be real, and could be +ve or –ve) and add to the sum and (sum of squared) immediately so you don't have to store them
  - after all numbers are input, calculate the mean and standard deviation from the sum, sum_of_squared and number

Appendix: Do while…construct

- N=0
- Do while (n<1)
-   read*,n
-   if (n<1) print*,"number not positive"
- End do

Appendix: Mathematical formulae

Arithmetic, geometric and harmonic means:

$$\overline{a}_{arithmetic} = \frac{1}{N}\sum_{i=1}^{N} a_i \qquad \overline{a}_{geometric} = \left(a_1 \times a_2 \times \ldots \times a_N\right)^{\frac{1}{N}} \qquad \overline{a}_{harmonic} = \frac{N}{\left(\frac{1}{a_1} + \frac{1}{a_2} + \ldots + \frac{1}{a_N}\right)}$$

In fortran, 'to the power of' is written **, e.g., a**b

Factorial: $N! = 1 \times 2 \times 3 \times \ldots \times N$

Standard deviation: $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N} a_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N} a_i\right)^2}$

In fortran, the square root function is sqrt()