Fortran: what is it and what's the difference between F90 and F77?

Overview

- Who decides what Fortran is?
- F90 vs. F77

Who decides what Fortran is?

- Standards are defined by a committee (http://www.j3-fortran.org/)
- Compilers are written
  - Commercial: e.g. Intel (ifort), many others
  - Non-commercial: g95, gfortran
- May also have coding guidelines/standards

Standards

- Fortran IV (1966)
- Fortran 77 (old but popular version)
- Fortran 90 (major revision to F77)
- Fortran 95 (minor revision to F90)
- Fortran 2003 (current standard)
- Fortran 2008 (will be next standard)

Why write standard compliant code?

- Should work with any compiler. Not tied to a specific compiler
- Easier to port to new computer
- Good practice: deleted features are removed for good reasons
- New standards have useful new features

Compilers

- Some behaviour is compiler dependent (not specified in standard) e.g.
  - Flags
  - Values of un-initialised variables
  - Warnings

Coding guidelines

- Local working practices
- Particularly useful for large projects with many developers.
- Easier to work with a uniform style
- Standards still permit undesirable coding practices

Example coding guidelines

- http://xmm.esa.int/sas/7.1.0/doc/devel/coding.html
- http://www.cgd.ucar.edu/cms/ccm4/codingstandard.shtml
- http://vlm089.citg.tudelft.nl/swan/online_doc/swanpgr/node2.html

Fortran 90 vs. Fortran 77

- Fortran 90 was a major revision to Fortran 77
- Fortran 77 is a complete sub-set of Fortran 90
- F90 introduced major new features
- Also introduced many useful minor features which can be gradually introduced

You may already use ...

- Longer names (only 6 character variable names in F77)
- `IMPLICIT NONE`
- Comparison operators: `.LT.` and `.GT.` or `<` and `>`

Fixed format source code

- F77 used fixed format source code
- Makes sense if your code is on fixed width punched cards
- A pointless inconvenience on a modern computer

Free format source code

- Start in any column, go up to column 132 (may be more readable <90 columns)
- Comments star t with a `!`
- Put a `&` at end of line to continue

Loops

- End a `DO` loop with `END DO`
- Go on to next iteration with `CYCLE`
- Break out of loop with `EXIT`
- Label your loops for safety
- `DO WHILE`, `DO` with out an index.

```
program loop_test

  implicit none

  integer, parameter :: missing_data = -99
  integer            :: data, status

  open (status="old", unit=60, file="data.dat")

  data_loop: do
    read(60,*, iostat=status) data
    if ( status /= 0 ) exit data_loop
    if ( data == missing_data ) cycle data_loop
    call process_data (data)
  end do data_loop

  close(60)

end program loop_test
```

```
      PROGRAM BADPROG

      DO 10 I=1. 10
         WRITE(*,*) I
10    CONTINUE

      END
```

New intrinsics

- Many new intrinsic functions and subroutines in F90
- Use of intrinsics saves writing extra code and should be quick
- Handling strings is much easier in F90
- Several useful array intrinsics

Some examples

- `DOT_PRODUCT, MATMUL`
- `SUM, ALL, ANY, MAXVAL, MINVAL`
- `DATE_AND_TIME`
- `RANDOM_NUMBER`
- `TRIM, ADJUSTL, ADJUSTR`
- and many more

Arrays

- F90 has dynamic memory allocation
- Avoids problems with hard-wired array sizes
- Needs care to avoid memory leaks
- Also assumed size and automatic arrays
- Can work with array sections
  `a(:)=b(1,:)`

```
program array_example

  implicit none

  real, allocatable :: spectral_cube(:,:,:)
  real, allocatable :: image(:,:)
  real              :: exp_time
  integer           :: nx, ny, nfreq

  call get_size (nx, ny, nfreq, "image.fits")

  allocate( spectral_cube (nx, ny, nfreq) )
  allocate( image (nx, ny) )

  call read_cube( spectral_cube, exp_time, nx, ny, nfreq )

  image(:,:) = SUM(spectral_cube, dim=3)
  image(:,:) = image(:,:) / exp_time

  call write_image(image, nx, ny)

  deallocate ( image )
  deallocate ( spectral_cube )

end program array_example
```

There's more ...

- User defined data types and modules
- Subroutines: recursion, optional arguments, intent
- Pointers
- WHERE
- FORALL
- SELECT CASE construct
- etc

Useful links

- http://www.star.le.ac.uk/~cgp/f90course/f90.html
- http://www.nsc.liu.se/~boein/f77to90/f77to90.html
- http://www.pcc.qub.ac.uk/tec/courses/f77tof90/stu-notes/f90studentMIF_1.html