Exercises

Day 2

Exercise #2a

- Add an INTERFACE block to the program OR place the relevant subroutine(s) in a module to create an implicit INTERFACE.

```fortran
program main
real, dimension(:), pointer :: my_data
integer :: n,status

n = 12
call sub(my_data,n,status)
print*,'status = ',status,associated(my_data)

call init(my_data,n,status)
do i=1,n
   print*,'my_data(i) = ',my_data(i)
enddo

end program main

subroutine sub(data,n,info)
real, dimension(:), pointer :: data
integer :: n,info

allocate(data(n),stat=info)
end subroutine sub

subroutine init(data,n,info)
real, dimension(*) :: data
integer :: n,info

do i=1,n
   data(i) = -i
enddo
end subroutine init
```

Exercise #2b

- Extend the program written in exercise #1b to:
  - Place the (global) data in a module.
  - Place the initialization part of the program and the file output in two separate subroutines.
  - Add an optional argument to the output routine to accept the time step counter. Call the output routine with the optional argument inside the time step loop (to produce files for post processing). If the optional argument is present construct the file name as: diff<STEP>.dat.

    (hint: use: WRITE(string,'(A,I6.6,A)') 'diff.',step,'.dat')
  - Place the copying of the new-to-old data field in a subroutine.

Exercise #2b

- Hint:
  - To compile do:
    - `f90 -free -c modulefile.f`
      - Creates: modulefile.o and modulename.mod (the extension of module file is compiler specific)
    - `f90 -free -c mainprg.f`
      - Creates: mainprg.o
  - To link do:
    - `f90 mainprg.o modulefile.o -o runme`

Exercise #2b

- Hint:
  - Collect commands in a file (fx runme.sh):

```
f90 -free -c modulefile.f
f90 -free -c mainprg.f
f90 mainprg.o modulefile.o -o runme
```

  - `Run this file using:`
    - `chmod u+x runme.sh (once)`
    - `./runme.sh`