I. Basics of F90/F77 programming

PROGRAM  program-name

IMPLICIT  NONE

[specification part]

[execution part]

stop
end

[start program part]

[subprogram part]

Define variables, constants, arrays

The program code!

Subroutines (e.g. matrix diagonalization

START AT 7th COLUMN!!!
Specification part-- variables

* INTEGER: the variables in list can hold integers
* REAL: the variables in list can hold real numbers
* COMPLEX: the variables in list can hold complex numbers
* LOGICAL: the variables in list can hold logical values (i.e., true or false)
* CHARACTER: the variables in list can hold character strings

Examples:

integer I, j, n, ntotal, naverage
double precision Hamil, Q, Energy, Z
character(2) Element
Specification part-- variable arrays

Examples:

double precision a(10), b(10), c(10)
double precision H(10,10)

In this example, everything is declared as real variables. Each of a, b, and c are one-dimensional arrays of length 10. In contrast, H is a two-dimensional array which is 10x10.

In other words, a, b, and c are like vectors, and H is a matrix!
Parameter statements

• Not really parameters, but instead constants
• Can be integer, real, etc.
• Should declare variables as real, integer, etc. before parameter statement

PARAMETER  pi=3.141592
PARAMETER  MAX=100

Once set, parameters cannot be changed within your code!
Comments

- Comments start with `c`
- No function other than notes for programmer
- Still important!

`c` Program to compute eigenfunctions and eigenvalues
`c` for a particle in a one-dimensional potential

Comments are especially important for anyone else who inherits your code!
The execution part is control statements, subroutine calls, functions, and intrinsic functions.
**If/then/else**

- Control statement
- Can nest as many as one needs

Just a single if statement... no then or else:

```plaintext
if (x<0) x=0
```

...alternately, if(x.lt.0) x=0

Or, when several conditions are desired:

```plaintext
if (x<0) then
  x=0
else
  if (x>100) x=100
endif
```

Notice indentation!

Terminate with “endif”
Do loops...

- Another control statement
- Useful for repeated calculations

```fortran
INTEGER n
REAL x,y,pi
pi=4.0d0*atan(1.0)
do n=1,100
   x=pi*n
   y=cos(x)
   write(6,*) n,y
endo do
```

Integer n takes on Values 1,2,3,...,100

Again indentation

Terminate do loop
Goto statements

- Often best to avoid as much as possible
- Very often used in conjunction with “if” statements

```plaintext
REAL x

if (x<0) goto 100

... x=0

100
```

Statements that are only done if x>0

Line number 100 (just a tag or label). Start in first column!
**Intrinsic functions**

- Mathematical functions
- Intrinsic to f90 compiler

```fortran
real x,y

y = cos(x)  \quad \text{Cosine of argument } x
y = exp(x)  \quad y = e^x
y = atan(x) \quad y = \tan^{-1}(x)
```

Links to intrinsic function list given in my webpage
**Input/output**

- Input data usually from “read” statement... Input file
- Output results from “write” statement... screen or output file
- Everything needs a unit number
- Format statements... useful, sometimes needed, can be a pain

```plaintext
INTEGER  n,i,j,k
REAL    x,y,z

open(unit=10,file='input.dat')
open(unit=11,file='output.dat')

read(10,*) i,j,k,x,y
n=i+j+k
z=x+y
write(11,*) n,z
write(6,*) n,z
```

Unformatted input and output

unit=6 reserved for output to screen
For the first project...

c Simulation of radioactive decay

program decay
C declare variables/arrays (specification part)
  implicit none
  double precision n_uranium(100), t(100), tau, dt
  integer n

call initialize(n_uranium, t, tau, dt, n)
call calculate(n_uranium, t, tau, dt, n)
call store(n_uranium, t, n)

stop
end
Initialize subroutine

subroutine initialize(nuclei,t,tc,dt,n)
double precision nuclei(100),t(100),tc,dt,time
integer n
print *,’initial number of nuclei ‘
read(5,*) nuclei(1)
print *,’time constant’
read(5,*) tc
print *,’time step’
read(5,*) dt
print *,’total time’
read(5,*) time
t(1)=0.0d0
n=min(int(time/dt),100)
return
end
subroutine calculate(n_uranium, t, tau, dt, n)
  implicit none
  double precision n_uranium(n), t(n), tau, dt
  integer n, i
  do i = 1, n-1
    n_uranium(i+1) = n_uranium(i) * (1.0d0 - dt/tau)
    t(i+1) = t(i) + dt
  end do
  return
end
Store subroutine

subroutine store(n_uranium, t, n)
  double precision n_uranium(n), t(n)
  integer i, n
  open(unit=1, file='decay.dat')
    do i=1, n
      write(1, 20) t(i), n_uranium(i)
    enddo
  format(1x, 1p, 2(f12.5, 2x))
return
end