Topic: FORTRAN Programming

Objectives

- Basics of FORTRAN programming
- Subroutines
- Format statements

Subroutines

- Invoked by *call* statement
- Can be written in the same file (both as a part of the same program or after the end program line) as the main program or in a separate file
- If the subroutine is written a separate file, care need to be taken during compilation. For eg: main.f90 and sub.f90

```
gfortran main.f90 sub.f90  OR
    gfortran -c main.f90
    gfortran -c sub.f90
    gfortran main.o  sub.o -o main.x
```

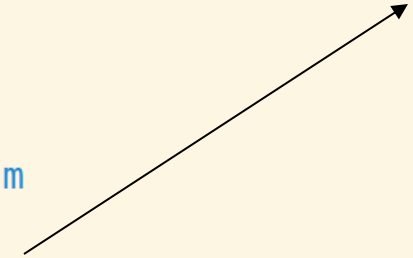
Subroutines – Hello World

```
program test
 implicit none

 call hello

end program test


subroutine hello
 implicit none

 write(*,*) "Hello World"

end subroutine hello
```

```
program test
 implicit none

 call hello

 contains
  subroutine hello
   implicit none

   write(*,*) "Hello World"

  end subroutine hello
end program test
```

Topic: FORTRAN Programming

```
program test
  implicit none

  integer :: i
  real    :: s(100), total

! initialize the varaible s

  do i = 1, 100

      s(i) = i

  enddo

! main part of the program

   call testsub(s,total)

   write(*,*) ' Sum ',  total

end program test
```



```
subroutine testsub(a,total)
  implicit none

  real :: a(*)
  real :: total
  integer :: i

  total=0
  do i = 1, 100

     total = total + a(i)

  enddo

end subroutine testsub
```

Subprogram(s)

Topic: FORTRAN Programming

```
program test
  implicit none

  integer :: i
  real    :: s(100), total

! initialize the varaible s

  do i = 1, 100

      s(i) = i

  enddo

! main part of the program

  total=sum(s)

   write(*,*) ' Sum ',  total

end program test
```

Output

Sum 5050.00000

sum is an internal function

Introducing internal functions

Internal functions

- Internal functions

`ABS(A)`	Returns the absolute value of A. If complex returns $\sqrt{real^2 + imag^2}$
`ACOS(X)`	Returns the arcosine of X.
`AIMAG(Z)`	Returns the imaginary part of the complex argument Z.
`ASIN(X)`	Returns the arcsine of X.
`ATAN(X)`	Returns the arctan of X.
`ATAN2(Y,X)`	Returns the arctan of Y/X in the range of $-\pi$ to π
`COS(X)`	Returns the cosine of X.
`COSH(X)`	Returns the hyperbolic cosine of X.
`DIM(X,Y)`	Returns X-Y if > 0, otherwise returns 0. Both X and Y must be of the same type and kind.
`DOT_PRODUCT(Vector_1,Vector_2)`	Performs the mathematical dot product of the two rank 1 arrays.
`DPROD(X,Y)`	Returns the double precision product of X and Y.
`EXP(X)`	Returns e^x.
`FLOOR(A,kind)`	Returns the largest integer $\leq$ A.
`LOG(X)`	Returns the natural logarithm of X

Dynamic memory allocation

- Use allocate statement

```
real, allocatable ::  coor(:,:)


n=10

m=10


allocate(coor(n,m))



deallocate(coor)
```

Topic: FORTRAN Programming

Dynamic memory allocation

```
program test
  implicit none

  integer :: i
  real    :: s(100), total

! initialize the varaible s

  do i = 1, 100
      s(i) = i
  enddo

! main part of the program

  total=0.0
  do i = 1, 100
   total = total + s(i)
  enddo

   write(*,*) ' Sum ',  total

end program test
```

```
program test
  implicit none

  integer :: i,N,total

  real,allocatable    :: s(:)

! initialize the varaible s

  write(*,*) "Enter the size of the array"
  read(*,*) N

  allocate(s(N))

  do i = 1, N
      s(i) = i
  enddo

! main part of the program

  total=0
  do i = 1, N
   total = total + s(i)
  enddo

   write(*,*) ' Sum ',  total

  deallocate(s)

end program test
```

Topic: FORTRAN Programming

```
do atom = 1, natoms
   write(101,"(a5,2x,3F15.7)") atm_name(atom), atm_coor(atom,:)
enddo
```

OR

```
do atom = 1, natoms
   write(101,100) atm_name(atom), atm_coor(atom,:)
enddo
100 format(a5,2x,3F15.7)
```

format statement

Descriptors: separated by commas and in parentheses.

Syntax

- Integer: iw
- Real: fw.d, ew.d
- Character: aw
- Space: x

Examples:

- 10 FORMAT(1X, A5, 2X, I3, 4X, A6, 2X, F6.2)
- 1001 format (i5, f5.2, e12.3)
- 2001 format (3(i5,e15.3))

Integer format descriptor

```
program test
 implicit none

 integer :: ii,ij

 ii=12345; ij=12345
 write(*,"(i4,i4)") ii,ij
 write(*,"(i6,i6)") ii,ij
 write(*,"(i6,2x,i6)") ii,ij
 write(*,"(i6,2x,i6)") ii,ij
 write(*,"(i10,2x,i10)") ii,ij
 write(*,"(2(i10,2x))") ii,ij

end program test
```

Output

```
********
 12345 12345
 12345   12345
 12345   12345
     12345       12345
     12345       12345
```

Real format descriptor

```
program test
 implicit none

 real(kind=8) :: ri, rj

 ri=12345; rj=12345.12345d0

 write(*,"(f10.3,f10.3)") ri,rj
 write(*,"(2(f10.3,2x))") ri,rj
 write(*,"(f10.7,2x,f14.7)") ri,rj
 write(*,*)
 write(*,"(e10.3,e10.3)") ri,rj
 write(*,"(e12.6,2x,e12.6)") ri,rj

end program test
```

Output

```
 12345.000 12345.123
 12345.000   12345.123
**********   12345.1234500

 0.123E+05 0.123E+05
0.123450E+05  0.123451E+05
```

character format descriptor

```
program test
 implicit none

 character(len=8) :: ca

 ca='hpc-cour'

 write(*,"(a8,a8)") ca,ca
 write(*,"(2x,a8,2x,a8)") ca,ca
 write(*,"(2(a5))") ca,ca
 write(*,"(2(a,2x))") ca,ca

end program test
```

Output

```
hpc-courhpc-cour
  hpc-cour  hpc-cour
hpc-chpc-c
hpc-cour  hpc-cour
```

hands-on

1. Write a program to read pdb file (input file provided) and write the output using the format statements (the output should look the same as input file)
2. Write a program to output the path of a projectile launched from the ground. The input values should be read from the standard output (velocity, angle, time interval)
3. Write a subroutine which calculates the distance between any two points. Coordinates are (x1,y1,z1) and (x2,y2,z2)
4. Write a subroutine to evaluate the following expression ($0 < x < 1$)

$$\sum_{i=1}^{5} \frac{X^{2i}}{2i}$$

Tips

- Don’t worry about declaring variables initially. Identify the main part of the program and start writing
- All real numbers should be in double precision (add d0 in the end), eg. 10.0d0
- Always use indentation, leave black spaces to improve readability
- Always use ‘parameter’ in case when assigning the values to integer datatype
- Use internal functions to convert datatypes, eg. real(x)
- Read compiler error messages more carefully
- For debugging, use ‘write’ statement at several places in the program and check for the output
-

Topic: FORTRAN Programming

FORTRAN – Reading material

- Please go through this FORTRAN program for a quick overview,
  - https://learnxinyminutes.com/docs/fortran95/
- Please go through this document for quick overview of FORTRAN
  - https://www.ldeo.columbia.edu/~mspieg/mmm/Fortran.pdf
- Book: Computer Programming in Fortran 90 and 95, V. Rajaraman