Lecture 3

- Control Constructs
- Program Units and Procedures

william.lee@ul.ie

Outline

william.lee@ul.ie

Control Constructs

Control constructs are used for

- Branching: calling different code under different circumstances
- Looping: calling the same code again and again (and . . .)

Fortran Control Constructs

- The (controversial) `go to` statement
- The `if` statement and construct
- The `case` construct
- The `do` construct

Fortran: go to statement

Move directly to numbered statement.

```
        x=y+3.0
        goto 4
3       x=x+2
4       z=x+y
```

- To another statement in the same block
- To the end of the construct
- To a statement outside the construct

GO TO considered harmful

- GO TO statements make code hard to understand because it interrupts linear narrative
- Avoid if at all possible!

Despite its many detractors GO TO is useful for dealing with errors and essential for breaking out of nested loops.

Fortran: IF statement

For when a single conditional statement suffices

```
if(flag) go to 6
if(x-y > 0.0) x=0.0
if(cond .or. p<q) s(i,j)=t(j,i)
```

Fortran: IF construct

```
if(i<0) then
  x=x+y
else if(j<0) then
  x=x-y
else
  x=0
end if
```

- Only the first and last statements are essential.
- Each block can contain multiple statements.
- Indenting is not essential but highly recommended. Most IDEs/editors indent automatically if you hit TAB.

Fortran: Case construct

I prefer IF construct.

```
select case(number) ! number is integer
case(:-1) ! number is -1 or lower
  n_sign=-1
case(0)   ! number is 0
  n_sign=0
case(1:)  ! number is 1 or higher
  n_sign=-1
end select
```

Fortran: DO loops

In nearly every other language these are FOR loops.

```
sum=0.0
DO i=1,10
   sum=sum+a(i)
   IF(sum<0e0) EXIT
END DO
```

Matlab control constructs

- IF construct
- FOR loop
- WHILE loop

Matlab: IF construct

```
if a ~=0
   sq=sqrt(b*b-2*a*c);
   x(1)=0.5*(-b+sq)/a;
   x(2)=0.5*(-b-sq)/a;
elseif b~= 0
   x(1)=-c/b;
elseif c~=0
   disp('Impossible equation')
else
   disp('The equation is a (boring) identity.')
end
```

Matlab: FOR loop

```
f(1)=1; f(2)=1
for i in [3 4 5 6]
  f(i)=f(i-1)+f(i-2);
end
```

Matlab: WHILE loop

```
f(1)=1; f(2)=1; k=3;
while k<=6
   f(k)=f(k-1)+f(k-2)
   k=k+1
end
```

Outline

Breaking a program up

- All but the simplest programs should be broken up into subtasks
- Single programs: functions, subroutines, m-files
- Reuseability
- Do one thing and do it well

Fortran

- Main program
- Functions
- Subroutines
- Modules

Main program

A Fortran program must contain one and only one main program

```
PROGRAM test
   print *, 'Hello world'
END PROGRAM test
```

The only thing that isn't optional is the END statement.

Functions

Functions return a single object but may also modify their arguments. (Pass by reference to C, C++ programmers.)

```
FUNCTION poly(n)
  REAL poly, x
  INTEGER n
  x=n**2 ! ** is exponentiation n^2
  RETURN x
END
```

Functions may appear in the same file as the main program or in different files.

Subroutines

Subroutines modify the values of their parameters and do not return a value

```
SUBROUTINE total(x,y,z)
   REAL x,y,z
   x=x+y+z
   y=y+z
END
```

Modules

Used for

- Global variables
- Interfaces
- Black boxes

Matlab, Functions

Can be contained in m files

```
function [out1,...,outn]=name(in1,...,inm)
% NAME this is the message given by  help name
...
return
```

Worksheet 3.

william.lee@ul.ie