Around and Around We Go
Loops and Iteration

Around and Around

- One of the most fundamental processes in a computer program is to repeat statements many, many (perhaps many, many, many) times.
- Computers never run out of patience
  - but loops are sometimes very slow compared to equivalent constructs, though this is mainly true in interpreted languages.

Loops

- Nearly every non-trivial program requires some form of looping or iteration. Examples:
  - Read all the lines in a file
  - Assign all values of an array
- In Fortran we have `do` and `do while` loops.

Fortran Do Loop

- Syntax

```
INTEGER  :: L, U
INTEGER  :: I, S
DO I=L,U,S
```

I: Loop variable

L: Lower bound

U: Upper bound

S: Stride. Equal to 1 if not present

S can be negative, in which case L must be greater than U.

Fortran Syntax

```
do i=1,maxtries
    statement
    statement
    more stuff
end do
```

- Can be written as one word enddo
- Loop variable cannot be changed, compiler will complain.

Quiz

- The standard requires that loop variables be integers. How would I implement loop variables that are real?
- How might real loop variables be a problem?

Leaving Early

- What on Earth is the purpose of the else?
  - Might be to set final value
  - Mostly it's used with early exits
- Fortran

  `exit`: leave loop

  `cycle`: skip rest of loop and go to next iteration

WHILE Loops

- While loops use a conditional to determine when to exit
- Make sure your expression evaluation changes!!!
- Fortran:

```
do while (<logical expression>)
    statement
    statement
    …
end do
```

- Equivalent to:

```
do
    if ( .not. <logical expression> ) exit
    statement
    statement
    ....
end do
```

NOTE: do while always tests at the *top* of the loop. The do … if/exit form can test anywhere, e.g. at the *bottom* to implement the repeat-until of some other languages.

Fortran

```
integer  :: x, y, z
x=-20
y=-10
 do while (x<0 .and. y<0)
        x=10-y
        y=y+1
        z=0
enddo
z=1
```

Fortran: Reading a File of Unknown Length

```
   nlines=0
   do
      read(unit=iunit, end=10) var
      nlines=nlines+1
   enddo
10 continue
```

Break/Continue

```
x=1.
do while (x>0.0)
    x=x+1.
   if (x>=10000.0) exit
   if (x<100.0) cycle
    x=x+20.0
enddo
```

Do Nothing (No-Op)

- Fortran

```
 continue
```

Infinite loops:

```
do while (.true.)
   continue
enddo
```

Repeat/Until

- Fortran

```
do
   statement
   statement
   statement
   update conditional
   if (true) exit
end do
```

Fortran Example

```
do
   z=y-3
   x=z-1
   if (x<0) exit
end do
```

Fortran Users: More complicated compile line

- On fir type
  module add intel
  man ifort

Marvel at the huge number of options.

Most common: -o <filename> -O

Debugging: use –g –CB (CB is Intel-specific, others use –C)

To name your output file something other than a.out:

  ifort –o hw2 hw2.f90