Counting loops

- These loops are used when you know how many times the body of the loop must execute prior to entering the loop
- Also used when you have an upper bound on the number of times a loop must execute (for example, search through 100 values looking for the first number > 15).
- Syntax

```
loop1:  DO  i = istart, iend, istep
            { Body of loop
        ENDDO loop1
```

comments

* i is called the loop index variable. The program controls its value for you.
* The value of i can be used in expressions in the loop, but the loop cannot assign a value to i (program won't compile if you try to do this)
* i must be an integer
* istart, iend, istep are the loop control parameters. These can be integer variables or static integer constants
* the loop name is optional

Example

loop to count to 10

```
DO  i = 1, 10, 1
    WRITE(**) 'i=', i
ENDDO
```

← value of i will start at 1, end at 10 and be incremented by 1 each pass through the loop

```
DO  i = 1, 10, 2
    WRITE(*,*) 'i=', i
ENDDO
```

← i starts at 1, ends at 10 and will be incremented by 2 each pass

- The most common value for the step (increment) is 1. If this is the case, you can leave off the step parameter

DO i = 1, 10 same as DO i = 1, 10, 1

- The control parameters can be variables

```
m = 1
n = 100
DO i = m, n  ⟶  DO i = 1, 100
```

Example

```
DO i = -4, 8, 9
   WRITE (*,*) 'i=', i
ENDDO
```

- How many times will a loop execute?

DO i = istart, iend, istep

$$ntimes = INT\left(\frac{iend - istart + istep}{istep}\right)$$

DO i = 1, 10, 1 (or 1, 10)

$$ntimes = INT\left(\frac{10 - 1 + 1}{1}\right) = 10$$

DO i = 1, 10, 2

$$ntimes = INT\left(\frac{10 - 1 + 2}{2}\right) = INT(5,5) = 5$$

DO i = -4, 8, 9

$$ntimes = INT\left(\frac{8 - (-4) + 9}{9}\right) = INT\left(\frac{21}{9}\right) = INT(2.\overline{3}) = 2$$

- Suppose you need to count backwards

```
DO i = 10, 1
   WRITE(*,*) 'i = ', i
ENDDO
```

$$ntimes = INT\left(\frac{1 - 10 + 1}{1}\right) = -8$$

If ntimes is <= 0 the loop is <u>empty</u>, meaning it will not execute.

```
DO i = 10, 1, -1
   WRITE(*,*) 'i = ', i
ENDDO
```

- loops can be nested (any type of loop can be nested within any other type of loop)

```
Do i = istart, iend
    {
    DO j = jstart, jend
        {
        DO K = Kstart, Kend
            {
            }
        ENDDO
    ENDDO
ENDDO
```

Can nest up to 7 levels deep

- CYCLE statement
  * This directs the loop to stop the current pass and go back to the top of the loop

```
Do i = 1, 10
    IF (MOD(i,3) == 0) THEN
        CYCLE
    ENDIF
    WRITE (*,*) 'i= ', i
ENDDO
```

→ can also give a loop name here to specify which loop to go to the top of.

output

```
i= 1
i= 2
i= 4
i= 5
i= 7
i= 8
i=10
```

- EXIT statement
  * Exits the current loop
  * can give a name to specify which loop to exit

```
loop1: DO i = 1, 10
loop2:    DO j = 1, 20
             IF (i+j > 15) THEN
                EXIT loop1
             ENDIF
          ENDDO loop2
       ENDDO loop1
```

- How do we use loops? We have seen a couple of examples, but will look at other applications of loops.
- Computing Sums

Example

compute $\sum_{i=1}^{100} i^2 + i$

counting loops make this type of sum easy to do

```
total = 0
DO i = 1, 100
   total = total + i**2 + i
ENDDO
```