DO Loops

Counting Loop

When a programmer knows, (or can determine through computation) how many times a loop is to be executed, a DO-loop should be used for efficiency.

Syntax:

```
DO count = start, endval, increment
    statement
    statement
END DO
```

Example 1:

```
isum = 0
DO i = 1, 10, 1
    isum = isum + i
END DO
```

Example 2:

```
isum = 0
i = 1
DO
    IF (i > 10) EXIT
    isum = isum + i
    i = i + 1
END DO
```

\[ \sum_{i=1}^{N} i = \frac{N(N + 1)}{2} \]
Flow Chart

DO Loop Representation

Almost identical to a while loop representation.
DO Loop Components

DO Loop Elements

**count** is a simple integer variable (not an expression) used to control
the number of times the loop gets executed.

**start** is the value to which count is initialized the first time through the loop

**endval** stops the execution of the loop, i.e. when **count > endval**
(or < if the loop is counting down) the loop is halted, at which point
execution of the program continues with the statement following the
END DO statement.

**increment** is the value that is added to count each time the loop is executed.

Count, start, endval, and increment are called loop control variables (**LCV**).

Note:

Start, endval, and increment must be either:

1) constant     (2, -16)
2) variable     j, count, k
3) expression   2 * j + 1, i / 3

and for our purposes they will always be INTEGER.
Examples

Simple DO Loops

```
    do  i = 1, 10, 2
    do  j = 10, 200, 5
    do  count = init, 500, 1
    do  k = i, j, n
    do  l = 1, jmp
      Note: increment omitted ---> 1
    do  k = 20, 2, 2
```

Non-trivial and invalid DO Loops

```
    do  j = 20, 2, -2
    do  i = j, n - 1
    do  l = 0, 20
    do  L = 1, L, 2             <---------- invalid, why?
    do  x + 1 = 1, y            <---------- invalid, why?
    do  j = 20, -1, 1
```

How many times does the body of the loop get executed?

\[
\frac{\text{endval} - \text{start}}{\text{increment}} + 1
\]

• Provided the loop gets executed at all!
Control Flow

- PRE-tested (make the first condition test before first loop)
- NOT every do loop gets executed at least once.
- The count variable may be used inside the loop.

- Do NOT change any of the loop control variables inside the loop.
Problem

Write a program for a weight-control center.

Given a data file that contains a client’s id number, age, height (feet & inches) and their weight. Determine if the client is under-weight, over-weight or ok and how many pounds they should lose or gain.

Note: the first line of the data file contains an integer count of the number of clients in the file.

The following formula for height|age <= weight ratio is to used:

A person’s optimal weight = 2.5 pounds for every inch of height plus 5 pds for every year of age over 28 and -5 pds for every year of age below 28.

A person is to be considered ok if they are within 5 pds of their optimal weight.
PROGRAM weightcenter  
   IMPLICIT NONE  
!  Variable declarations  
   INTEGER :: client_id, age, feet, inches, clients, client  
   REAL ::   wtact, wtopt, wtdiff  
   LOGICAL :: wtok, overwt  

!  Constant declarations  
   INTEGER, PARAMETER :: keyage=28  
   REAL, PARAMETER :: wtrange=5.0, pdsyear=5.0, pdsinch=2.5  

!  data files  
   OPEN ( 9, FILE = 'CLIENT.DAT')  
   OPEN (10, FILE = 'CLIENT.RST')  

!  Write table header to output file  
   WRITE (10, 90)  
   WRITE (10, 91)  

!  Input number of clients  
   READ (9,100) clients  

! weightcenter Program Continued  
! process client info  
   DO    client = 1, clients  
      READ (9,200) client_id, age, feet, inches, wtact  

! Determine optimal weight & difference  
   inches = feet * 12 + inches  
   wtopt  = pdsinch * inches + (pdsyear * (age-keyage))  
   wtdiff = ABS( wtact - wtopt)  
   inches = inches - feet * 12  

! Check weight status  
   wtok   = ( wtdiff <= wtrange)  
   overwt = ( wtact > wtopt )
DO Example (cont)

! Output client’s weight status
IF ( wtok ) THEN
    WRITE (10,300) client_id, age, feet, inches, wtact
ELSE IF ( overwt ) THEN
    WRITE (10,400) client_id, age, feet, inches, wtact, &
                   (wtact- (wtopt+5.0))
! If not wtok and not overwt then must be underweight
ELSE
    WRITE (10,500) client_id, age, feet, inches, wtact, &
                   (wtopt - (wtact+5.0))
END IF

END DO

CLOSE(9)
CLOSE(10)

STOP

90 FORMAT (’Client Age Weight Status Amount’)  
91 FORMAT (’---------------------------------------------’)
100 FORMAT (I3)
200 FORMAT (I4, TR1, I3, TR1, I1, TR1, I2, TR1, F5.1)
300 FORMAT (I4, TR6, I3, TR2, I1, TR1, I2, TR3, F7.1, ’ OK’)
400 FORMAT (I4, TR6, I3, TR2, I1, TR1, I2, TR3, F7.1, ’ Lose’, F9.1)
500 FORMAT (I4, TR6, I3, TR2, I1, TR1, I2, TR3, F7.1, ’ Gain’, F9.1)
END PROGRAM weightcenter
DO Example (cont)

Input:

<table>
<thead>
<tr>
<th>Client</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>Status</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>22</td>
<td>5</td>
<td>10</td>
<td>155.2</td>
<td>5.2</td>
</tr>
<tr>
<td>1001</td>
<td>20</td>
<td>5</td>
<td>8</td>
<td>135.0</td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>37</td>
<td>5</td>
<td>6</td>
<td>160.6</td>
<td>44.4</td>
</tr>
<tr>
<td>1003</td>
<td>43</td>
<td>6</td>
<td>4</td>
<td>225.0</td>
<td>35.0</td>
</tr>
<tr>
<td>1004</td>
<td>47</td>
<td>6</td>
<td>0</td>
<td>265.8</td>
<td>4.2</td>
</tr>
</tbody>
</table>

Output:

<table>
<thead>
<tr>
<th>Client</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>Status</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>22</td>
<td>5</td>
<td>10</td>
<td>Lose</td>
<td>5.2</td>
</tr>
<tr>
<td>1001</td>
<td>20</td>
<td>5</td>
<td>8</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>37</td>
<td>5</td>
<td>6</td>
<td>Gain</td>
<td>44.4</td>
</tr>
<tr>
<td>1003</td>
<td>43</td>
<td>6</td>
<td>4</td>
<td>Gain</td>
<td>35.0</td>
</tr>
<tr>
<td>1004</td>
<td>47</td>
<td>6</td>
<td>0</td>
<td>Gain</td>
<td>4.2</td>
</tr>
</tbody>
</table>

Hmm. Well, looks like the formula used here isn’t exactly ideal. Still, the example makes the basic points.