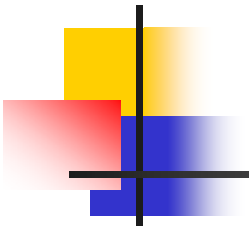
REPETITION



Why Repetition?

Read 8 real numbers and compute their average

```
REAL   X1, X2, X3, X4, X5, X6, X7, X8
REAL   SUM, AVG
READ *, X1, X2, X3, X4, X5, X6, X7, X8
SUM = X1+ X2 + X3 + X4 + X5 + X6 + X7 + X8
AVG = SUM / 8.0
PRINT*, ' THE  AVERAGE = ', AVG
END
```

Read 100 real numbers and compute their average

Read 1000 real numbers and compute their average

What about if we can do the following:

```
      REAL X, SUM
      SUM = 0
  repeat the following two statements 100 times

      READ*, X
      SUM = SUM + X
```

```
      REAL X , SUM
      SUM = 0
  repeat the following two statements 1000 times

      READ*, X
      SUM = SUM + X
```

```
      REAL X, SUM
      SUM = 0
  repeat the following two statements N times
      READ*, X
      SUM = SUM + X
```

```
      INTEGER N
      REAL X, SUM
      SUM = 0
      N = 100
      DO 5 K = 1, N, 1
         READ*, X
         SUM = SUM + X
5   CONTINUE
```

The DO LOOP

```
DO  N   index = initial, limit, increment

    block of FORTRAN statements

N   CONTINUE
```

How Many Times?

- The number of times the loop is executed is known before the loop execution begins.
- The number of times (iterations) the loop is executed is computed as follows:

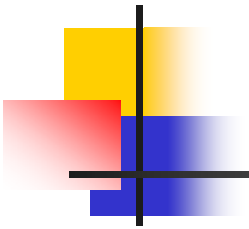
[(limit - initial) / increment] + 1

Examples on The DO Loop:

Write a FORTRAN program that reads the grades of 100 students and calculates and prints their average.

```
      REAL  GRADE, SUM, AVG
      INTEGER  K
      SUM = 0.0
      DO  10  K = 1, 100, 1
         READ*, GRADE
         SUM = SUM + GRADE
10    CONTINUE
      AVG = SUM / 100.0
      PRINT*, ' THE  AVERAGE = ' , AVG
      END
```

Increment



- Default increment is 1
- The increment can be negative
  It could be called then a decrement

What will be printed by the following do loop?

```
      DO 99  K = 15, 4 , -2
         PRINT*, K
99     CONTINUE
      END
```

15
13
11
9
7
5

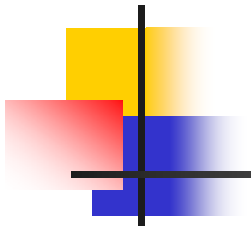
Examples on The DO Loop:

Write a FORTRAN program that evaluates the following series to the 7th term.

$$\sum_{i=1}^{N} 3^{i}$$

```
      INTEGER  SUM, K
      SUM =  0
      DO  11  K = 1, 7
         SUM = SUM + 3 **K
11    CONTINUE
      PRINT*, 'SUM = ' , SUM
      END
```

THE CONTINUE STATEMENT

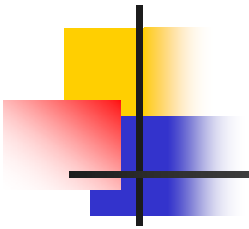


```
      REAL GRADE, SUM, AVG
      SUM = 0.0
      DO  3  I = 1 , 100 , 1
         READ *, GRADE
         SUM = SUM + GRADE
3     CONTINUE
      AVG = SUM /100.0
      PRINT*, ' THE AVERGE =  ', AVG
      END
```

```
      REAL GRADE, SUM, AVG
      SUM = 0.0
      DO  3  I = 1 , 100 , 1
         READ * GRADE
3        SUM = SUM + GRADE
      AVG = SUM /100.0
      PRINT*, ' THE AVERGE = ', AVG
      END
```

IF, GOTO, RETURN, STOP or another DO statement **can not** replace **CONTINUE** statements

Notes on the DO loop:



- In the **first iteration**, the **index** of the loop has the **value of initial**.
- Once the last statement “CONTINUE“ is executed, execution is transferred to the beginning of the loop.
- Before each iteration, the index is checked to see if it has passed the limit.
- If the index passed the limit, the loop iterations stop. Otherwise, the next iteration begins.

```
      DO 15  K = 1 , 5 , 2
         PRINT*, K
15    CONTINUE
```

- The loop above is executed 3 times. The value of K outside the loop is 7
- If the **increment is positive** the initial must be less than or equal to the limit. Otherwise the loop will not be executed.
- If the **increment is negative** the initial must be greater than or equal to the limit. Otherwise the loop will not be executed.
- If the values of the initial and the limit are equal, the loop executes only **once**.

DO loops rules

- **Index** of DO loop must be a variable of either INTEGER or REAL types.
- **Initial**, **limit**, and **increment** can be expressions of either INTEGER or REAL types.
- The value of the DO loop index cannot be modified inside the loop.
- The **increment** must **not** be **zero**, otherwise an error occurs.

```
      INTEGER  M
      DO 124  M = 1 , 10 , 0.5
         PRINT*, M
124   CONTINUE
      PRINT*, M
      END
```

- The index after the loop is the value that has been incremented and found to pass the limit.
- Branch into a DO loop is not allowed.
- Branch out of a DO loop before all the iterations are completed must not be used unless necessary.

DO loops rules (cont):

- The parameters (**initial** , **limit** , and **increment**) of the loop are evaluated **before** the loop execution begins.Once evaluated, changing their values will not affect the executing of the loop.

For an example, consider the following segment

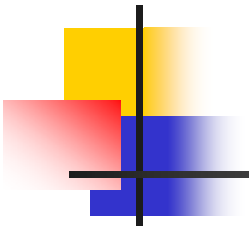
```
      REAL X , Y
      Y = 4.0
      DO 10  X = 0.0 , Y,  1.5
         PRINT*, X
         Y = Y + 1.0
         PRINT*, Y
10    CONTINUE
```

The output

0.0
5.0
1.5
6.0
3.0
7.0

This loop is executed [(4.0 - 0.0) /1.5)] + 1 = 3 times

Nested DO Loops



Example: Nested DO Loops

```
      INTEGER   M, J
      DO  111  M = 1, 2
         DO  122  J = 1, 6, 2
            PRINT*, M, J
122      CONTINUE
111   CONTINUE
      END
```

The output of the above program is:

1	1
1	3
1	5
2	1
2	3
2	5

Nested DO Loops

Example: Consider the following program.

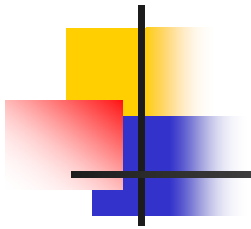
```
      INTEGER   M, J
      DO  111  M = 1, 2
          DO  122  J = 1, 6, 2
              PRINT*, M, J
122       CONTINUE
          PRINT*, M, J
111   CONTINUE
      PRINT*, M, J
      END
```

The output of the above program is:

```
 1   1
 1   3
 1   5
 1   7
 2   1
 2   3
 2   5
 2   7
 3   7
```

```
      INTEGER   M, J
      DO  111  M = 1, 2
          DO 122 J = 1, 6, 2
122           PRINT*, M, J
111       PRINT*, M, J
      PRINT*, M, J
      END
```

Exercises



```
      DO 1  K = 2, 3
         DO 2  M = 1, 4, 2
2           PRINT*, K, M
1        PRINT*, K, M
      PRINT*, K, M
      END
```

```
      DO 1  K = 2, 3
         DO 2  M = 1, 4, -2
2           PRINT*, K, M
1        PRINT*, K, M
      PRINT*, K, M
      END
```

```
      DO 1  K = 2, 3, -1
         DO 2  M = 1, 4, 2
2           PRINT*, K, M
1        PRINT*, K, M
      PRINT*, K, M
      END
```

The output	
2	1
2	3
2	5
3	1
3	3
3	5
4	5

2	1
3	1
4	1

2	???

What is the output of the following program?

```
      INTEGER  K, M, N
      N = 0
      DO 10 K = -5, 5
          N = N + 2
          DO 20 M = 3, 1
              N = N + 3
20        CONTINUE
          N = N + 1
10    CONTINUE
      PRINT*, N
      END
```

The output

33

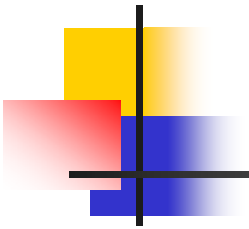
What is the output of the following program?

```
      INTEGER  K, M, N
      N = 0
      DO 10 K = 20, 2, -3
          N = N + 5
          DO 20 M = 13, 5, 4
20            N = N + 4
10        N = N + 3
      PRINT*, N, K, M
      END
```

The output

56 -1 13

The WHILE LOOP



```
DO WHILE ( condition )

    Block of statements

END DO
```

Examples on The WHILE LOOP:

Example 1: Write a FORTRAN program that reads the grades of 100 students in a course. The program then computes and prints the average of the grades.

Solution:

```
REAL  GRADE,  SUM,  AVG
INTEGER  NUM
NUM = 0
SUM = 0.0
DO WHILE (NUM .LT. 100)
    NUM = NUM + 1
    PRINT*, 'ENTER  GRADE  NUMBER' , NUM
    READ*, GRADE
   SUM = SUM + GRADE
END DO
AVG = SUM / NUM
PRINT*, ' THE AVERAGE OF THE GRADES = ', AVG
END
```

Example 2:

- A class has a certain number of students
- Read the grades of the students , the last input would include a negative grade
- Compute and print the average and the number of students in the class.

Solution:

```
REAL  GRADE,  SUM,  AVG
INTEGER  NUM
NUM = 0
SUM = 0.0
PRINT*, 'ENTER A GRADE'
READ*, GRADE
DO WHILE (GRADE .GE. 0)
    NUM = NUM + 1
    SUM = SUM + GRADE
    PRINT*, 'ENTER A GRADE'
    READ*, GRADE
END DO
AVG = SUM / NUM
PRINT*, ' THE AVERAGE OF THE GRADES = ', AVG
PRINT*, 'NUMBER OF STUDENTS IN THE CLASS = ', NUM
END
```

Example 3:Series Summation using a DO loop:
Question: Write a FORTRAN program which calculates the sum of the following series:

$$\frac{1}{2}+\frac{2}{3}+\frac{3}{4}+\frac{4}{5}+\ldots+\frac{99}{100}$$

Solution:

```
      REAL  N, SUM
      SUM = 0.0
      DO 100  N = 1, 99
         SUM = SUM + N / (N + 1)
100   CONTINUE
      PRINT*, 'SUM = ' , SUM
      END
```

Example 4: Series Summation using a WHILE loop:
Question: Write a FORTRAN program which calculates the sum of the following series:

$$\frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \ldots + \frac{99}{100}$$

Solution:

```
REAL  N, SUM
N = 1
SUM = 0.0
DO WHILE(N .LE. 99)
    SUM = SUM + N / (N + 1)
    N = N + 1
END DO
PRINT*, 'SUM = ' , SUM
END
```

Example 5: Alternating Sequences/ Series: Alternating sequences, or series, are those which have terms alternating their signs from positive to negative. In this example, we find the sum of an alternating series.
Question: Write a FORTRAN program that evaluates the following series to the 100th term.

1 - 3 + 5 - 7 + 9 - 11 + 13 - 15 + 17 - 19 + ...

Solution:

It is obvious that the terms differ by 2 and start at the value of 1.

```
      INTEGER  SUM , TERM , N
      SUM = 0
      TERM = 1
      DO  10  N = 1, 100
         SUM = SUM + (-1) ** (N + 1) * TERM
         TERM = TERM + 2
10    CONTINUE
      PRINT*, 'SUM = ' , SUM
      END
```

Example 6: Write a FORTRAN program that reads an integer number M. The program then computes and prints the factorial of M using a DO loop.

Solution:

```
      INTEGER  M ,TERM , FACT
      PRINT*, 'ENTER AN INTEGER NUMBER'
      READ*, M
      PRINT*, 'INPUT: ', M
      IF (M .GE. 0) THEN
         FACT = 1
         DO  100 TERM = M, 2 , -1
            FACT = FACT *TERM
100      CONTINUE
         PRINT*, 'FACTORIAL OF' , M , 'IS', FACT
      ELSE
         PRINT*, 'NO FACTORIAL FOR NEGATIVES'
      ENDIF
      END
```

Example 7: Write a FORTRAN program that reads an integer number M. The program then computes and prints the factorial of M using a WHILE loop.

Solution:

```
INTEGER   M, FACT
PRINT*, 'ENTER AN INTEGER NUMBER'
READ*, M
PRINT*, 'INPUT: ', M
IF (M .GE. 0) THEN
   FACT = 1
   DO WHILE (M .GE. 2)
      FACT = FACT *M
       M = M - 1
   END DO
    PRINT*, 'FACTORIAL  IS ', FACT
ELSE
   PRINT*, 'NO FACTORIAL FOR NEGATIVES'
ENDIF
END
```

Nested WHILE Loops

Example: Consider the following program.

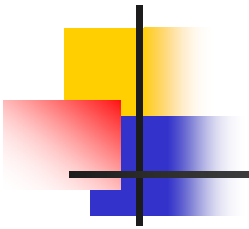
```
      INTEGER   M, J
      DO  111  M = 1, 2
          DO  122  J = 1, 6, 2
              PRINT*, M, J
122       CONTINUE
111   CONTINUE
      END
```

```
INTEGER  M, J
M = 1
DO WHILE ( M .LE. 2)
    J = 1
    DO WHILE (J .LE. 6)
        PRINT*, M, J
        J = J + 2
    END DO
    M = M + 1
END DO
END
```

The output of the above program is:

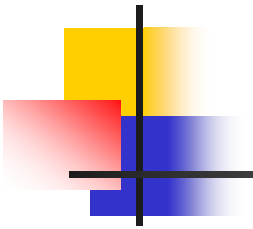
1	**1**
1	**3**
1	**5**
2	**1**
2	**3**
2	**5**

Infinite loop



```
INTEGER  X
X = 5
DO WHILE ( X .GT. 0)
     PRINT*, X
     X = X + 1
END DO
END
```

Implied Loops



Implied loops are only used in **READ** and **PRINT** statements.

READ*, (list of variables, index = initial, limit, increment)

PRINT*, (list of expressions, index = initial, limit, increment)

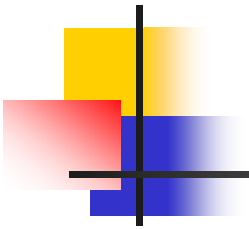
Example 1: Printing values from 100 to 87
The following segment prints the integer values from 100 down to 87 in a single line.

```
PRINT*, (K , K = 100, 87, -1)
```

Output:

```
100  99  98  97  96  95  94  93  92  91  90  89  88  87
```

Nested Implied Loops



Example: Consider the following program.

```
      INTEGER   M, J
      DO  111  M = 1, 2
          DO  122  J = 1, 6, 2
              PRINT*, M, J
122           CONTINUE
111     CONTINUE
        END
```

```
PRINT*, ((M, J, J = 1, 6, 2) , M = 1, 2)
```

```
 1    1    1    3    1    5    2    1    2    3    2    5
```

Nested Implied Loops

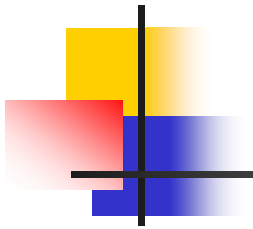
Example: Consider the following program.

```
      INTEGER   M, J
      DO  111  M = 1, 2
          DO  122  J = 1, 6, 2
              PRINT*, M, J
122           CONTINUE
          PRINT*, M, J
111   CONTINUE
      PRINT*, M, J
      END
```

PRINT*, ((M, J, J = 1, 6, 2) , M, J, M = 1, 2) , M, J

1 1 1 3 1 5 1 7 2 1 2 3 2 5 2 7 3 7

Exercises



```
PRINT*, ((K, M, M = 1, 4, 2) , K, M, K = 2, 3) , K, M
```

2 1 2 3 2 5 3 1 3 3 3 5 4 5

```
PRINT*, ((K, M, M = 1, 4, -2) , K, M, K = 2, 3) , K, M
```

2 1 3 1 4 1

```
PRINT*, ((K , M, M = 1, 4, 2) , K, M, K = 2, 3 , -1) , K, M
```

2 ???

Repetition Constructs in Subprograms

Exercise

What will be printed by the following program?

```
C     FUNCTION SUBPROGRAM
      LOGICAL FUNCTION PRIME(K)
      INTEGER N, K
      PRIME = . TRUE .
      DO 10 N = 2, K / 2
         IF (MOD(K , N) .EQ. 0) THEN
              PRIME = . FALSE .
             RETURN
          ENDIF
10    CONTINUE
      RETURN
      END

C     MAIN PROGRAM
      LOGICAL PRIME
      PRINT*, PRIME (5),  PRIME (8)
      END
```

The output

T F