FORTRAN 77
Chapter 3

Satish Chandra

www.satish0402.weebly.com

LOOP STATEMENTS

- These statements are used to execute a set of statements a fixed number of times or until a particular condition is satisfied.
- For repeated execution of similar things, loops are used. Fortran 77 has only one loop construct, called the do-loop. The do-loop corresponds to what is known as a for-loop in other languages.

Do-Loop

- The general form of a do-loop statement is;

```
do st. no variable = Initial, Final, Increment
statement 1
.....
statement n
st. no continue
```


Do-Loop

- In particular,

```
do 10 x= m,n, p
statements
10 continue
```

- Here the block of statements are executed after assigning x = m and checking if it is less than n. Then x is increased by p, if the new value of x is less than n again the block is executed. It continues till x exceeds n.

Do-Loop

- Example 1

```
sum=0
do 10 i = 1 , n
sum=sum + i
10 continue
```

- With these statements we get the sum of first n natural numbers.
- Note that here p=1, if so it need not be particularly mentioned.

Do-Loop

- The Fortran code to calculate N factorial for positive value of N would be

```
n_factorial = 1
do 10 i = 1, n
n_factorial = n_factorial * i
10 continue
```

- If n is 5, the do loop parameters will be initial = 1, final = 5, and increment = 1. The factorial will be: 1*2*3*4*5= 125

Do-Loop

```
      integer i, n, sum
c     cumulative sums of integers from 1 through n
      sum = 0
      do 10 i = 1, n
      sum = sum + i
      write(*,*) 'i =', i
      write(*,*) 'sum =', sum
 10   continue
      stop
      end
```


Nested Loops

- With in a do loop another loop may come like

```
      do 10 i= 1, m
      do 20 j= 1, n
      read (*,*) a (i, j)
   20 continue
   10 continue
```

- If two loops are to be nested, one of them must lie completely within the other one.

Nested Loops

```
c     program nested loops
      integer i, j, product
      do 20 i = 1, 3
            do 10 j = 1, 3
            product = i * j
            write (*,*) i, ' * ', j, ' = ', product
 10         continue
 20   continue
      stop
      end
```


Nested Loops

- The results are

```
1 * 1 =   1
1 * 2 =   2
1 * 3 =   3
2 * 1 =   2
2 * 2 =   4
2 * 3 =   6
3 * 1 =   3
3 * 2 =   6
3 * 3 =   9
```


Problem 1:

- Find the sum of squares of first n natural numbers.
- Write the Fibonacci sequence.
- Evaluate the series $\Sigma (1/n^2)$ for various n.

CONTROL STATEMENTS

- Normally the Fortran statements are executed sequentially as written in the program. That is after executing one statement it goes to the next line. This is called normal flow of control.
- But in certain case we may have to restrict the normal flow especially when the actions are based on the result of testing condition. For this we use control statements.
- The condition is normally a logical expression obtained by connecting expressions using relational operators.

GO TO Statement

- The Syntax is,

  go to n
- where n is the label of an executable statement to which the control is transferred. For example:

  statements1
  go to 10
  statements2
  10 statement 3
- Here without executing statements2 the flow of execution is directed to the statement labeled 10

Computed GO TO Statement

- The syntax is

  go to (s1,s2,s3,s4…..) , integer expression
- Here when the control is transferred to the statements labeled s1,s2,s3,…. According as the integer expression takes the value 1,2,3 …… respectively.
- Example


```
      go to ( 10,20,30), icode
10    hra=300
      cca=50
      go to 40
20    hra=500
      cca=100
      go to 40
30    hra=1000
      cca=125
40    salary=pay+hra+cca
      write (*,50) salary
50    format(1x, 'your salary is Rs. ', f8.2, '/-')
```


Logical IF statement

- The syntax is

  if (logical expression) statement1
- Example:

  if (x < 0) go to 30
  statements2
  30 stop
- Here the statement1 (go to 30) is executed if logical expression is TRUE and if not statemsnt2 are executed.


```
c     Example program 1 for logical if
      real a, b, c
c     choosing the largest of three numbers
      read (*,*) a, b, c
      if (a.GT.b) go to 20
            if (b.GT.c) go to 10
10          print (*,*) b
            stop
20          If (a.GT.c) go to 30
            print (*,*) c
            stop
30    print (*,*) a
      stop
      end
```



```
c     Example program 2 for logical if
      real a, b, c
c     choosing the largest of three numbers
      read (*,*) a, b, c
      big = a
      if (b.GT.big) big = b
            if (c.GT.big) big = c
            print (*,*) big
            stop
      end
```

Block IF statement

- The block IF statement allows a block of statements to be executed if a logical expression is TRUE. The syntax is

```
if (logical expression) then
        statements
end if
```


Block IF statement

- Example:

```
    if (x <0) then
    write (*, 10)
10  format(1x, 'the number is negative')
    end if
```


Block IF statement

- It allows another set of statements to be executed if the logical expression is either TRUE or FALSE. The syntax is

```
if (logical expression) then
        block1
else
        block2
end if
```


Block IF statement

```
    if (x <0) then
    write (*, 10)
10  format(1x, 'x is negative & root can't be found')
    else
    write(*,15)
15  format(1x, 'x is nonnegative')
    y=sqrt(x)
    write(*,20) y
20  format(1x, 'the square root is ', f10.3)
    end if
```


Block IF statement

- If there are various conditions to be tested, we use IF- ELSE IF Statements

Block IF statement

```
if (logical expression 1) then
    block1
else if (logical expression 2) then
    block2
else if (logical expression 3) then
……..
else
    block n
end if
```

```
      if (mark > 90) then
      write (*, 10)
10    format(1x, 'excellent')
      else if (mark > 80) then
      write (*, 15)
15    format(1x, 'distinction')
      else if (mark > 60) then
      write( *,20)
20    format(1x, ' first class')
      else if (mark > 40) then
      write (*, 30)
30    format(1x, 'good')
      else
      write (*, 40)
40    format(1x, 'work hard')
      end if
```


Nested Block IF structure

- In the block If-end if or if-else-end if block there can be another blocks of if-end if or if-else-end if block.
- The syntax is


```
if (logical expression 1) then
    statement 1
    if (logical expression 2) then
        block1
    else
        block2
    end if
else
    statement2
    if (logical expression 3) then
        block3
    end if
end if
```



```
if (x .GT. 0) then
  if (x .GE. y) then
  write(*,*) 'x is positive and x >= y'
  else
  write(*,*) 'x is positive but x < y'
  end if
else
  if (x .LT. 0) then
  write(*,*) 'x is negative'
  else
  write(*,*) 'x is zero'
  end if
end if
```



```
c     Example program 3 for nested block if
      real a, b, c
c     choosing the largest of three numbers
      read (*,*) a, b, c
      if (a.GT.b) then
          if (a.GT.c) than
              print (*,*) a
              stop
          else
              print (*,*) c
              stop
          end if
```



```
      else
          if (b.GT.c) then
              print (*,*) b
              stop
          else
              print (*,*) c
              stop
          end if
      end if
      end
```

Arithmetic IF Statement

- It causes transfer of control depending on the value of an arithmetic expression. The syntax is

```
    if(arithmetic expression) s1,s2,s3
s1  continue
    block1
s2  continue
    block2
s3  continue
    block3
```


Arithmetic IF Statement

- Where s1, s2 and s3 are statement numbers of executable statements, not necessarily distinct.
- If the arithmetic expression is < 0, Statement labeled s1 is executed, if zero control is transferred to s2 and if > 0 to s3.
- Here continue is only to continue the execution. Even if it is not used it will work
- Example: Let $d= b^2 - 4ac$. The d can be positive, zero or negative.


```
    if (d) 10,20,30
10  continue
    write(*,15)
15  format(1x, 'the roots are imaginary')
    go to 40
20  continue
    write(*, 25)
25  format(1x, ' the roots are equal')
    go to 40
30  continue
    write (*,35)
35  format(1x, ' the roots are real and distinct')
40  stop
    end
```


Problem 2:

1. Solve the quadratic equation $ax^2+bx+c=0$
2. Read a set of marks and write the marks above average.

STOP statement

- This is to terminate the execution of a program and to display the result on the out put device.
- The syntax is STOP or STOP 'message'. In the later case the 'message' will also be displayed

CHAPTER 3

ENDS