Introduction to FORTRAN

by

Dr. Ibrahim A. Assakkaf

ENCE 202
Spring 2000
Department of Civil and Environmental Engineering
University of Maryland

Structured Programming

Control Structures

- In a structured program, the logical flow can be of the following types:
  - Sequential ➔ straight-line programs
  - Selection ➔ If statements
  - Repetition ➔ Do loops
Structured Programming

- Logical Expressions
  - Relational Operators
    
    Expression-1 relational-operator expression-2
    
    - Relational Operators are:
      .LT.  Is less than
      .GT.  Is greater than
      .EQ.  Is equal to
      .LE.  Is less than or equal to
      .GE.  Is greater than or equal to
      .NE.  Is not equal to

- Logical Operators
  - They can be used to combine the previous operators or negate them
  - Logical operators are:
    .NOT.  ➔ negation
    .AND.  ➔ both true
    .OR.   ➔ one is true
    .EQV.  ➔ both true or false
    .NEQV. ➔ negation of .EQV.
**Structured Programming**

### Logical IF Statement

IF(logical-expression) statement

- Example

```fortran
IF (1.5 .LE. X .AND. X .LE. 2.5) PRINT *, X
```

### Block IF Statement

- Type I

```
IF (logical-expression) THEN
    statement 1
    statement 2
    ...
END IF
```

```fortran
Block 1
```

Structured Programming

- Block IF Statement
  - Example: Type 1

  ```fortran
  IF (X .GT. 0) THEN
    Y = X * X
    Z = SQRT (X)
    Slope = TAN (Z)
  END IF
  ```

Structured Programming

- Block IF Statement
  - Type II

  ```fortran
  IF (logical-expression) THEN
    Block-1
  ELSE
    Block-2
  END IF
  ```
Structured Programming

Example: Type II

PROGRAM POLLUT
C Program that reads 3 pollution LEVELS, calculates a pollution INDEX as their average, and then displays a "safe condition" message if this index is less than some CUTOFF value, otherwise displays a "hazardous condition" message.

INTEGER CUTOFF, LEVEL1, LEVEL2, LEVEL3, INDEX
PARAMETER (CUTOFF = 50)

PRINT *, 'ENTER 3 POLLUTION READINGS:
READ *, LEVEL1, LEVEL2, LEVEL3
INDEX = (LEVEL1 + LEVEL2 + LEVEL3) / 3.0
IF (INDEX .LT. CUTOFF) THEN
   PRINT *, 'SAFE CONDITION'
ELSE
   PRINT *, 'HAZARDOUS CONDITION'
END IF
END

Sample runs:
ENTER 3 POLLUTION READINGS: 55, 39, 48
SAFE CONDITION
ENTER 3 POLLUTION READINGS: 68, 49, 57
HAZARDOUS CONDITION

Nested IF Statements

– They are used for multi-alternative selection structure as follows:

   IF (logical-expression-1) THEN
      Block-1
   ELSE
      IF (logical-expression-2) THEN
         Block-2
      ELSE
         Block-3
      END IF
   END IF
Structured Programming

- Nested IF Statements
  - An alternative format for the previous nested IF structures is as follows:
    
    \[
    \begin{align*}
    \text{IF (logical-expression-1) THEN} & \quad \text{Block-1} \\
    \text{ELSE IF (logical-expression-2) THEN} & \quad \text{Block-2} \\
    \text{ELSE IF (logical-expression-3) THEN} & \quad \text{Block-3} \\
    \vdots & \quad \vdots \\
    \text{ELSE} & \quad \text{Block-n} \\
    \text{END IF}
    \end{align*}
    \]

- Repetition Structure (The DO and CONTINUE statements)
  
  \[
  \text{DO \ n, control-variable = initial-value, limit, step-size} \\
  \text{statement} \\
  \vdots \\
  \vdots \\
  \text{n continue}
  \]

  NOTE that the default for step size = 1
  Also, note that statement number must be in columns 1 to 5
### Structured Programming

#### Nested DO Loops

<table>
<thead>
<tr>
<th>Legal Structure</th>
<th>Illegal Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO 5, I = 1, 10</td>
<td></td>
</tr>
<tr>
<td>Statement-Set-1</td>
<td></td>
</tr>
<tr>
<td>DO 6, J = 1, 5</td>
<td></td>
</tr>
<tr>
<td>Statement-Set-2</td>
<td></td>
</tr>
<tr>
<td>CONTINUE</td>
<td></td>
</tr>
<tr>
<td>Statement-Set-3</td>
<td></td>
</tr>
<tr>
<td>CONTINUE</td>
<td></td>
</tr>
</tbody>
</table>

| DO 5, I = 1, 10  |
| Statement-Set-1 |
| IF (expression) THEN |
| Statement-Set-2 |
| END IF           |
| Statement-Set-3 |
| CONTINUE         |

#### The WHILE (repetition) Statement

- The WHILE loop is similar to the DO loop with unknown number of repetitions. The number of repetitions is determined by a logical expression.

Two forms for WHILE statement can be used:

<table>
<thead>
<tr>
<th>Form I of WHILE Loop</th>
<th>Form II of WHILE Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHILE (logical expression) DO</td>
<td></td>
</tr>
<tr>
<td>statements</td>
<td></td>
</tr>
<tr>
<td>END WHILE</td>
<td></td>
</tr>
<tr>
<td>DO WHILE (logical expression)</td>
<td></td>
</tr>
<tr>
<td>Statements</td>
<td></td>
</tr>
<tr>
<td>END DO</td>
<td></td>
</tr>
</tbody>
</table>
Structured Programming

GO TO Statement

– Form I

\[ n \quad \text{IF (logical-expression)} \quad \text{THEN} \]

\[ \begin{aligned}
\text{Statements} \\
: \\
: \\
\text{GO TO } n
\end{aligned} \]

END IF

GO TO Statement

– Form II

\[ n \quad \text{statements} \]

\[ : \\
: \\
\text{IF (logical-expression)} \quad \text{GO TO } n \]

NOTE: Standard FORTRAN does not include a WHILE statement. Nevertheless, this important control structure can be implemented in standard FORTRAN by using a GO TO statement within an IF construct.
Structured Programming

Examples: DO WHILE & GO TO

<table>
<thead>
<tr>
<th>DO WHILE</th>
<th>GO TO</th>
</tr>
</thead>
<tbody>
<tr>
<td>(standard Fortran)</td>
<td></td>
</tr>
<tr>
<td>DO WHILE (SUM LE LIMIT)</td>
<td>IF (SUM LE LIMIT) THEN</td>
</tr>
<tr>
<td>NUMBER = NUMBER + 1</td>
<td>NUMBER = NUMBER + 1</td>
</tr>
<tr>
<td>SUM = SUM + NUMBER</td>
<td>SUM = SUM + NUMBER</td>
</tr>
<tr>
<td>END DO</td>
<td>GO TO 10</td>
</tr>
</tbody>
</table>

Input and Output

- Non-formatted input and output was discussed earlier
- Formatted Output
  - There are two output statement in FORTRAN, the PRINT statement and the WRITE statement. The PRINT statement is the simpler of the two and has the form

  • PRINT format-identifier, output-list
**Input and Output**

- **EXAMPLE: Formatted Output**
  
  ```fortran
  PRINT 10, N, Y, Z  
  10 FORMAT (list of format descriptors)  
  OR  
  10 FORMAT (1X, I5, 2F8.2)  
  ```

  *NOTE: the 2 is to repeat the instruction*

---

**Input and Output**

- **Control Characters**
  
  - `I` = integer
  - `F` = real number in decimal format
  - `E` = real number in scientific format
  - `D` = `F` or `E` input/output, depending on value
  - `nX` = horizontal spacing
  - `/` = vertical spacing
  - `A` = character data

**Example:**

```fortran
  FORMAT (1X, 2(A, F6.2))  
```
Input and Output

- Formatted Input
  - The READ statement is for reading from the default input device (such as the keyboard)

```
READ format-identifier, input-list
```

- It is common to use non-formatted (or called free format) read as follows:

```
READ *, input-list
```

The WRITE Statement

- This statement is used to write to output files

```
WRITE (control-list) output-list
```

- Control-list consists of the following:
  - unit-specifier for output device such as printers (*prints on screen)
  - Format-identifier statement (*means free format)
### Input an Output

**EXAMPLES: WRITE Statement**

- `WRITE (6, *) X, Y`
- `WRITE (UNIT = 6, FMT = *) X, Y`
- `WRITE (*, *) X, Y`
  - same as `PRINT *, X, Y`

### Input and Output

**The General READ Statement**

- `READ (control-list) input-list`

**Examples:**

- `READ (UNIT = 5, FMT = *) X, Y`
- `READ (5, *) X, Y`
- `READ (*, *) X, Y`  ==> same as `READ, * X, Y`
- `READ (12, *, END = 50) X, Y`

The last statement means go to statement 50 when the end of data is encountered.
Input and Output

File Processing

- It is common to have a need for large input/output. Files can be on magnetic tapes, disks, or hard drive. Using files requires the following steps:

  • To open an existing file or create a new file:
    ```fortran
    OPEN (open-list)
    ``

  • To close a file:
    ```fortran
    CLOSE (close-list)
    ```

Example: File Processing

- OPEN (UNIT = 13, FILE = `File-name`, STATUS = `NEW`)
  (You may use also STATUS = `OLD` to open an existing file)

- CLOSE (13)
  (where 13 is the unit number)
Input and Output

- File Processing
  - Other features
    - REWIND unit
      (go to initial point for unit such as 13 (use a number))
    - BACKSPACE unit
      (go to the beginning of preceding record in unit such as 13 (use a number))