Decision Making

Logical IF Statement

Used when we wish to execute a single statement or NOT execute the statement based on some condition being true or false.

Examples:

Form 1:	Form 2:
IF(hrs .GT. 40.0)otrate = rate *1.5	IF(hrs > 40.0)otrate = rate * 1.5
IF (index .EQ. 0) STOP	IF (index == 0) STOP
IF (age .LT. cutoff) WRITE(..)…	IF (age < cutoff) WRITE(..)…

If the condition is true then execute the statement and continue with the rest of the program. Otherwise, skip statement and continue with the rest of the program.

Logical Variable

Two Fortran logical constants: .TRUE. and .FALSE.

A logical variable can have a value either .TRUE. or .FALSE. and can be declared as:

```
LOGICAL :: variable_name1, variable_name2, variable_name3,……
```

Example:

```
INTEGER :: b = 8
      :
LOGICAL :: a
a = (b > 5)
IF(a) …..
```

A logical variable can be used as a condition in an IF statement In the above example the logical variable a will have the value .TRUE.

Logical Expressions & Conditions

Relation expressions are the most common type of Logical Expressions.

Logical Expressions are also called Boolean Expressions.

Relational Operators:

Operator	Form 1	Form 2
Equality	==	.EQ.
Greater than	>	.GT.
Less than	<	.LT.
Not equal to	/=	.NE.
Greater than or equal to	>=	.GE.
Less than or equal to	<=	.LE.

Relational Expressions Syntax:

<arithmetic expression> <relational operator> <arith. exp>

Examples:

(a > b)

((hours - 40.0) > 0.0)

(b**2 < 4*a*c)

(b**2 == 4*a*c)

(4*a + c /= b*c + 4)

(ABS(error) <= epsilon)

Logical IF Example

Problem:

Input: cols 1 - 4 account number, cols 6 - 8 test score

Determine: the average score, the number of A's
(i.e. score is $\geq$ 90) and the highest score.

Sample Program

```
PROGRAM test

IMPLICIT NONE
INTEGER ::  actnum, test_score, count=0, a_count=0, high_score=0, total=0, ios
REAL ::    average_score
OPEN(...)
OPEN(...)
READ (9, 100, IOSTAT=ios)actnum, test_score
DO
      IF (ios < 0) EXIT
      count = count + 1
      total = total + test_score
      IF(test_score >= 90) a_count = a_count + 1
      IF(test_score > high_score) high_score = test_score
      READ (9, 100, IOSTAT=ios) actnum, test_score
END DO
average_score = REAL(total) / count
WRITE(10,201) count
WRITE(10,202) average_score
WRITE(10,203) a_count
WRITE(10,204) high_score
STOP
100  FORMAT (...
201  FORMAT (...
202  FORMAT (...
203  FORMAT (...
204  FORMAT (...
END PROGRAM test
```

Note: comments and program headers have been omitted from the program examples in the class notes due to lack of space on the overheads.

What would be appropriate here ?

Desk Tracing

Hand Checking

Desk tracing (a.k.a. hand checking) involves making up sample data and executing/tracing the program by hand manually performing the actions that the machine would carry out.

Example Trace/Check of the previous program

Sample input data file contents:

1073	50
2975	90
2074	75
3398	95

Primary Memory Storage Variables

count	total	a_count	high_score	actnum	test_score
0	0	0	0	?	?
1	50	0	50	1073	50
2	140	1	90	2975	90
3	215	1	90	2074	75
4	310	2	95	3398	95

average_score
77.5

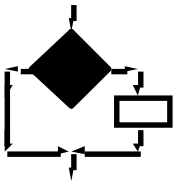
Block IF

Multiple Statement Selection

Usually it is desired to execute several statements based on some condition or boolean expression (evaluates to either true or false).

Syntax:

IF (condition) **THEN**
    statement
    statement
    •
    •
    •
END IF



Note: IF statements do NOT form a loop, they can be inside of a loop however.

Boolean Expression Considerations

Any variable used in the condition of an IF statement must have already been given a value! (initialized?)

Fortran (unlike English) does not allow implied subjects.

Example:

The mathematical expression: a < b < c

cannot be expressed as it is;

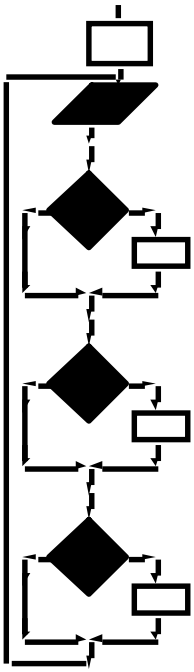
it must be stated as: ((a < b) .AND. (b < c))

Block IF Example

Problem

Consider the previous student/grades problem.Suppose that there are two classes of students (morning 0, and afternoon 1). Suppose that we want statistics for each class.

Sample Program



```
        •
        •
    DO
        .
        IF (class == 0) then
            morn_count = morn_count + 1
            morn_total = morn_total + test_score
            IF(test_score >= 90)morn_a = morn_a + 1
        END IF
        IF (class == 1) THEN
            aft_count = aft_count + 1
            aft_total = aft_total + test_score
            IF (testsc >= 90) aft_a = aft_a + 1
        END IF
        IF (test_score > high_score) THEN
            high_score = test_score
            high_actnum = actnum
        END IF
         •
    END DO
200 WRITE(..)...
        •
        •
```

NOTE indentation used and initialization of variable(s).

Block IF THEN ELSE

adds an ELSE Clause:

Syntax:

IF (condition) **THEN**
  statement
  - executed when (condition) is true
  -

ELSE
  statement
  - executed when (condition) is false
  -

END IF

Example

The first two if's of the previous example can be combined:

```
IF (class == 0) THEN
 morn_count = morn_count + 1
 morn_total = morn_total + test_score
 IF (test_score >= 90) morn_a = morn_a + 1
ELSE
 aft_count = aft_count + 1
 aft_total = aft_total + test_score
 IF (test_score >= 90) aft_a = aft_a + 1
END IF
```

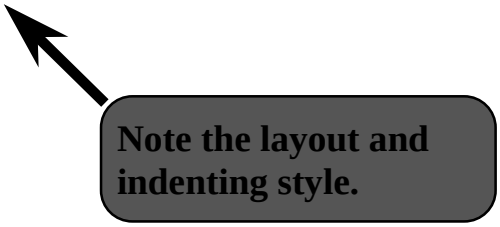
Conditions that are "mutually exclusive", (one condition being true excludes all others from being true), should be tested for with nested ifs, (as opposed to disjoint ifs), for efficiency.

Nested IF Statements

IF Statements Inside of IF Statements

Syntax:

```
IF (cond1) THEN
  statement A
  IF (cond2) THEN
    statement B
    statement C
  ELSE
    statement D
  END IF
ELSE
  statement E
  statement F
END IF
statement G
```



Under what (true) conditions, do each of the statements A..G get executed ?

Cond1	Cond2	Executed statements
true	true	A B C G
true	false	A D G
false	true	E F G
false	false	E F G

Nested IFs Example

Given three integer vars (a,b,c), having unique values, output the values in order.

```
IF (a > b) THEN
    IF (a > c) THEN                    { a is largest }
        IF (b > c) THEN
            WRITE(*,*) a,b,c
        ELSE
            WRITE(*,*) a,c,b
        END IF
    ELSE                               { a > b & c > a }
        WRITE(*,*) c,a,b
    END IF
ELSE
    IF (b > c) THEN                    { b is largest }
        IF (a > c) THEN
            WRITE(*,*) b,a,c
        ELSE
            WRITE(*,*) b,c,a
        END IF
    ELSE                               { b > a & c > b }
        WRITE(*,*) c,b,a
    END IF
END IF
```

Equivalent IFs

Given:

```
IF ( x  <= y) THEN
  M = 1
ELSE
  M = 2
END IF
```

Which of the following IFs are logically equivalent?

```
IF ( x > y) THEN
  M = 2
ELSE
  M = 1
END IF
```

```
IF ( y >= x) THEN
      M = 1
      ELSE
      M = 2
END IF
```

```
IF ( y <= x) THEN
  M = 1
ELSE
  M = 2
END IF
```

```
IF ( y <= x) THEN
      M = 2
      ELSE
      M = 1
END IF
```

Equivalent IFs (cont)

Trace all possible orderings of x & y.

Table of all possible ordering sample values.

		X ≤ Y	X > Y	Y ≥ X	Y < X	Y ≤ X
X	**Y**	**M**	**M**	**M**	**M**	**M**
1	**2**	**1**	**1**	**1**	**2**	**1**
2	**2**	**1**	**1**	**1**	**1**	**2**
3	**2**	**2**	**2**	**2**	**1**	**2**
			✔	✔		

Forming Equivalent IFs

Given:

```
IF (cond) THEN
    S1
    S2
ELSE
    S3
    S4
END IF
```

```
IF (.NOT. cond) THEN
    S3
    S4
ELSE
    S1
    S2
END IF
```

Compound Conditions

Logical (Boolean) Operators

.NOT. **.AND.** **.OR.** **.EQV.** **.NEQV.**

Logical operators allow relational expressions to be combined to form complex compound conditions.

Generally, logical expressions syntax:

(relational expression) .logical operator. (relational expression)

Examples:

((class == 0) .AND. (test_score >= 90))

((time == 930) .OR. (time == 1400))

((time == 930) .OR. (.NOT. (time ==1400)))

Complex Condition Evaluation

Truth tables for: (A .logical operator. B)
where A and B are relational expressions.

X	Y	.NOT. X	X .AND. Y	X .OR. Y
T	T	F	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	F

The logical operators .EQV. and .NEQV. will be discussed in class

Complex Condition Evaluation

Operator Hierarchy

0) (...) do what's in ( ) first

1) .NOT. **2**) .AND. **3**) .OR.

4) .EQV. and .NEQV.

Complex Cond Evaluation Example

Given:

X = 2.0 Y = 3.0 Z = 1.0

Evaluate

(Y == X + Z) .OR. (Y >= Z) .AND. .NOT. (X < Z)



All solutions, (for any values of X, Y or Z), can be determined by developing the truth table for the complex condition.

A	B	C	B.AND. .NOT.C	A.OR.B.AND..NOT.C
T	T	T	F	T
T	T	F	T	T
T	F	T	F	T
T	F	F	F	T
F	T	T	F	F
F	T	F	T	T
F	F	T	F	F
F	F	F	F	F

Week Day Program

Problem: Determine what day of the week any given date falls upon.

Given: the following formula which computes the day of the week for any date, (m, d, y; where y > 1752)

calc_day = d + 2m + INT(3(m+1)/5) + y + INT(y/4) - INT(y/100) + INT(y/400) + 1

weekday = MOD (calc_day, 7)

The INT function returns only the integer result.

where:

weekday = 0 -> Sunday
weekday = 1 -> Monday
• • •
• • •
• • •
weekday = 6 -> Saturday

The Fortran function MOD returns the remainder.

Note: the formula requires that Jan & Feb be treated as the 13th & 14th months, respectively, of the preceding year.

Day of Week Program

```
PROGRAM day_of_week
   IMPLICIT NONE
   INTEGER :: month, day, year, calc_day, weekday, ios
   OPEN ( 9, FILE = 'inday.dat')
   OPEN (10, FILE = 'outday.dat')
   WRITE(10,10)
10 FORMAT('This program determines the day of the', &
   ' week for any given date')
   READ (9,40, IOSTAT=ios) month, day, year
40 FORMAT( I2, I3, I5)
   DO
      IF (ios < 0) EXIT
      IF (month < 3) THEN
          month = month + 12
          year = year -1
      END IF
      calc_day = day + 2*month + (3*(month+1)/5) + year  &
      weekday = MOD(calc_day, 7 )

      IF (month > 12) THEN
          month = month - 12
          year = year +1
      END IF

      WRITE (10, *)
      IF (weekday == 0 ) THEN
          WRITE( 10, 100) month, day, year
      ELSE IF (weekday == 1 ) THEN
          WRITE(10, 101) month, day, year
      ELSE IF (weekday == 2 ) THEN
          WRITE(10, 102) month, day, year
      ELSE IF (weekday == 3 ) THEN
          WRITE(10, 103) month, day, year
      ELSE IF (weekday == 4 ) THEN
          WRITE(10, 104) month, day, year
      ELSE IF (weekday == 5 ) THEN
          WRITE(10, 105) month, day, year
      ELSE IF (weekday == 6 ) THEN
          WRITE(10, 106) month, day, year
      END IF
      READ (9,40, IOSTAT=ios) month, day, year
```

Day of Week Program (cont)

```
        END DO

        CLOSE ( 9)
        CLOSE (10)

        STOP

100     FORMAT ('The date ', 2(I2, '/'),I4,   &
                ' falls/fell upon a Sunday')
101     FORMAT ('The date ', 2(I2, '/'),I4,   &
                ' falls/fell upon a Monday')
102     FORMAT ('The date ', 2(I2, '/'),I4,   &
                ' falls/fell upon a Tuesday') &
103     FORMAT ('The date ', 2(I2, '/'),I4,
                ' falls/fell upon a Wednesday')
104     FORMAT ('The date ', 2(I2, '/'),I4,   &
                ' falls/fell upon a Thursday')
105     FORMAT ('The date ', 2(I2, '/'),I4,   &
                ' falls/fell upon a Friday')
106     FORMAT ('The date ', 2(I2, '/'),I4,   &
                ' falls/fell upon a Saturday')

        END PROGRAM day_of_week
```

ELSE IF statement

- **may appear only between a block IF & END IF**
- **never has a matching END IF statement**

Day of Week Program Output

Input:

```
 1  1 1900
11 11 1918
 3 12 1924
 9  1 1939
 7  4 1963
 2 29 1964
 5 11 1974
 3 14 1977
12 31 1996
 1  1 1997
```

Sample input and corresponding output for the day_of_week program are shown here. This is a good example of the use of many of the Fortran features covered to this point in the course. Be sure you understand it.

Output:

```
This program determines the day of the week for any given date

The date  1/ 1/1900 falls/fell upon a Monday

The date 11/11/1918 falls/fell upon a Monday

The date  3/12/1924 falls/fell upon a Wednesday

The date  9/ 1/1939 falls/fell upon a Friday

The date  7/ 4/1963 falls/fell upon a Thursday

The date  2/29/1964 falls/fell upon a Saturday

The date  5/11/1974 falls/fell upon a Saturday

The date  3/14/1977 falls/fell upon a Monday
```

CASE Construct

Like the block IF construct, the CASE construct allows to select a block of statements (or none) for execution from a set of blocks of statements depending on the value of a case selection expression. The syntax for the CASE construct is:

```
SELECT CASE (case _selection_expression)
CASE (case_selector)
        block_of_statements
CASE (case_selector)
        block_of_statements
        :
        :
END SELECT
```

The *case_selection_expression* must be of integer, character, or logical type but cannot be of real type. A CASE statement can take the form:

```
CASE (case_selector)
or
CASE DEFAULT
```

Note: CASE DEFAULT is optional

The *case_selector* determines which block of statements will be executed. The form of a *case_selector* can be :

case_value
low_value:
:high_value
low_value:high_value
or a list of any combination of the above

The block of code following the CASE DEFAULT statement is executed, if no values or value ranges match with the value of the *case_selection_expression*.

CASE Construct: Examples

Example 1:

```
SELECT CASE (class_code)
CASE (1)
    WRITE(*,*)  “Freshman”
CASE (2)
    WRITE(*,*) “Sophomore”
CASE (3)
    WRITE (*,*) “Junior”
CASE(4)
    WRITE (*,*) “Senior”
CASE DEFAULT
    WRITE (*,*) “Invalid Class code”
END SELECT
```

Example 2:

```
SELECT CASE (mid_term_score+final_score)
CASE(90:)
     grade_point = 4.0
     WRITE (*,*) “ Your course grade is A”
CASE (80:89)
     grade_point = 3.0
     WRITE (*,*) “ Your course grade is B”
CASE (70:79)
     grade_point = 2.0
     WRITE (*,*) “Your course grade is C”
CASE (60:69)
     grade_point = 1.0
     WRITE (*,*) “Your course grade is D”
CASE (:59)
     grade_point = 0.0
     WRITE (*,*) “Grade is not everything in life”
END SELECT
```