# Chapter 3
# Selective execution

Example: How to use a program to solve quadratic equations

$$Ax^2 + Bx + C = 0$$

Formula of real roots:

$$\begin{cases} \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} & \text{if } B^2 - 4AC \geq 0 \\ \text{No real roots} & \text{otherwise} \end{cases}$$

Logical constants: `.TRUE.` and `.FALSE.`

Relational-operator:

| | |
|---|---|
| $<$ | `.LT.` |
| $>$ | `.GT.` |
| $==$ | `.EQ.` |
| $<=$ | `.LE.` |
| $>=$ | `.GE.` |
| $/=$ | `.NE.` |

Simple logical expressions: logical constants, logical variables or relational expressions

Relational expressions:

*expression relational-operator expression*

Examples:

```
X < 5.2
Number == -99
B ** 2 >= 4.0 * A * C
1.0 >= 24.0
"cat" < "dog"
"cat" < "cattle"
```

The value of relational expressions is either `.true.` or `.false.`

The character strings are compared according to their coding (most common standard coding schemes is ASCII).

See `http://www.ascii-code.com/` for ASCII codes.

Comparison is from left to right.

Shorter strings are appended blanks.

Logical operations:

`.NOT.`

`.AND.`

`.OR.`

`.EQV.`

`.NEQV.`

The logical operations are defined at tables in pp.46.

If a logical expression contains arithmetic operations, relational operations, and logical operations, the operations are performed in the following order:

1. Arithmetic operations and functions
2. Relational operations
3. Logical operations in the order `.NOT.`, `.AND.`, `OR.`, `.EQV.` `(or .NEQV.)`

Examples:

```
N**2 + 1 > 10 .AND. .NOT. N < 3
N == 3 .OR. N == 4
.NOT. M > N .AND. X < Z .EQV. M <= N .AND. X >= Z
```

Wrong examples:

```
N == 3 .OR. SIN(X)
N > 2 .AND. N < 1
X .NOT. Y + 3
X .NEQV. Y ** 2
```

Simple IF construct:

```
IF (logical-expression) THEN
   statement-sequence
END IF
```

If the value of `logical-expression` is true, then the statement-sequence will be executed.

If the value of `logical-expression` is false, then the statement-sequence will be ignored.

```
IF (B*B - 4.0*A*C >= 0) THEN
  PRINT *, "This quadratic equation has real solutions!"
END IF
```

In this example, if $B^2 - 4AB \geq 0$, then the value `B*B - 4.0*A*C` will be true and the computer will print out `This quadratic equation has real solutions!`

Otherwise, the computer will not print it.

The Fortran has a simpler if statement called logical IF statement.

```
IF (logical-expression) statement
```

Example

```
IF (1.5 <= X .AND. 2.5 >= x) PRINT *, X
```

If $1.5 \leq X \leq 2.5$, the computer will print out the value of $X$.
Otherwise it will not print the value of $X$.

General form of the IF construct

```
IF (logical-expression) THEN
  statement-sequence1
ELSE
  statement-sequence2
END IF
```

If the `logical-expression` is TRUE, then do `statement-sequence1` otherwise do `statement-sequence2`

```
Discriminant = B ** 2 - 4.0*A*C
IF (Discriminant >= 0) THEN
   Discriminant = SQRT(Discriminant)
   Root_1 = (-B + Discriminant) / (2.0 * A)
   Root_2 = (-B - Discriminant) / (2.0 * A)
   PRINT *, "The roots are:", Root_1, Root_2
ELSE
  PRINT *, "There are no real roots"
END IF
```

IF-ELSE IF Constructs

```
IF (logical-expression1) THEN
   statement-sequence1
ELSE IF (logical-expression2) THEN
  statement-sequence2
ELSE IF (logical-expression3) THEN
  statement-sequence3
......

ELSE
  statement-sequence
END IF
```

If (logical-expression1) is true, Then do statement-sequence1 and go to END IF

If (logical-expression1) is false, then check (logical-expression2). If (logical-expression2) is true, do statement-sequence2 and go to END IF.

... ...

If all the above logical-expressions are false, then do statement-sequence

Named constructs

IF and IF-ELSE IF constructs may be named by attaching a label at the beginning and the end of the construct. Example

```
Update: IF (X > Largest ) THEN
   Largest = X
   Position = N
END IF Update
```

```
EmpType: IF (EmployeeType == "s") THEN
  PRINT *, "Enter employee's annual salaty:"
  READ *, Salary
  Pay = Salary / 52
ELSE
  PRINT *, " Enter hours worked and hourly rate:"
  READ *, HoursWorked, HourlyRate
  Overtime: IF (HoursWorked > 40.0) THEN
    Pay = 40.0 * HourlyRate &
     + Multiplier * HourlyRate * (HoursWorked - 40.0)
    ELSE
     Pay = HoursWorked * HourlyRate
    END IF Overtime
END IF EmpType
```

The CASE construct

```
SELECT CASE (selector)
  CASE (label-list1)
    statement-sequence1
  CASE (label-list2)
    statement-sequence2
 ......
END SELECT
```

where selector is an expression of integer, or character, of logical. Each of the label-list is a list of one or more possible values of selector, or the word `DEFALULT`.

It is also can use named case constructs

```
Class: SELECT CASE (ClassCode)
  CASE (1)
    PRINT *, "Freshman"
  CASE (2)
    PRINT *, "Sophomore"
  CASE (3)
    PRINT *, "Junior"
  CASE (4)
    PRINT *, "Senior"
  CASE (5)
    PRINT *, "Graduate"
  CASE DEFAULT
    PRINT *, "Illegal class code", ClassCode
END SELECT class
```

Example:

The level of air pollution in a city is measured by a pollution index. Readings are made at 12:00 P.M. at three locations. The integer average of these three readings is the pollution index. A value of 50 parts per million or greater for this index indicates a hazardous condition. A index below 25 parts million indicates a good condition. Otherwise, the condition is fair. The city environmental statistician would like a program that calculates the pollution index and then determines the air condition.

It is not a good idea to start writing computer program right after reading the problem.

General steps:

- Problem analysis and specification
- Algorithm design
- Coding
- Testing
- Maintenance

Analysis and specification:

Need to calculate the index, which required the 3 readings. The formula of the index is the average.

Need to decide the condition, which is determined by compare the index with some constants.

Design:

Decide variables, constants, input and output, the algorithm (pseudo codes or diagram)

Coding: pp. 56

Testing:

```
PROGRAM Pollution
IMPLICIT NONE
INTEGER :: Level_1, Level_2, Level_3, Index
INTEGER, PARAMETER :: LowCutoff = 25, HighCutoff = 50
PRINT *, "Enter 3 pollution readings:"
READ *, Level_1, Level_2, Level_3
Index = ( Level_1 + Level_2 + Level_3) / 3
SELECT CASE (Index)
  CASE (:LowCutoff - 1)
    PRINT *, "Good condition"
  CASE (LowCutoff : HighCutoff -1))
    PRINT *, "Fair condition"
  CASE (HighCutoff :)
    PRINT *, "Poor condition"
END SELECT
END PROGRAM pollution
```

The logical data type has two possible values: .true. or .false.

The output of the logical constant is: `T` or `F`. The input is decided by the first letter of the input word is T or F.

```
PROGRAM HALF_ADDER
IMPLICIT NONE
LOGICAL :: A, B, Sum, Carry

PRINT *, "Enter logical input A and B:"
READ *, A,B
Sum = (A .OR. B) .AND. .NOT. (A .AND. B)
Carry = A .AND. B
PRINT *, "Carry, Sum = ", Carry, Sum
END PROGRAM HALF_ADDER
```

Sample runs:

```
Enter logical input A and B:
T T
Carry, Sum = T F

Enter logical input A and B:
T F
Carry, Sum = F T

Enter logical input A and B:
F T
Carry, Sum = F T

Enter logical input A and B:
F F
Carry, Sum = F F
```