Decision Making using the IF Statement

Logical Control Structures
Methods of executing instructions are : Sequence Selection (IF-THEN-ELSE) Iteration (PERFORM) Case (EVALUATE)

The IF-THEN-ELSE structure permits us to execute one or more instruction depending on the content of fields.

Basic Conditional Statement.

The Instruction format for an IF Statement.

A **Conditional statement** is one that performs operations depending on an existence of some condition. In COBOL, such statement generally begins with the word IF and are called IF-THEN-ELSE or selection structures.

The basic instruction format for IF statement is as follows:

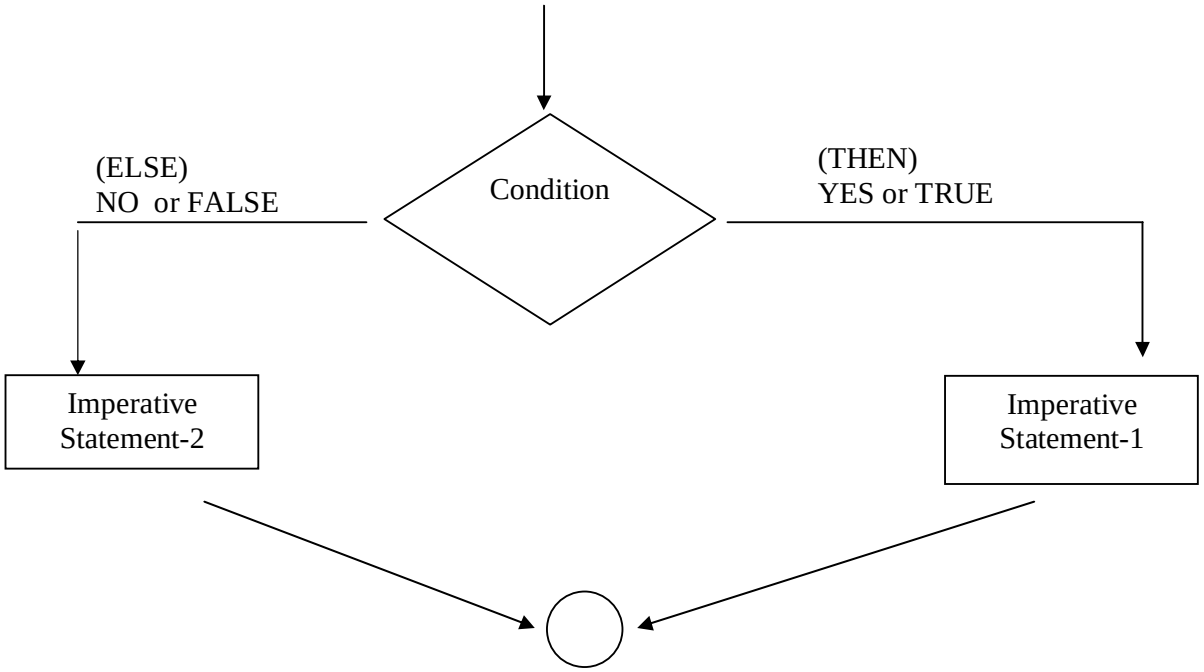
```
IF    Condition-1
[ THEN]
          imperative statement-1  ..
[ELSE
          imperative statement-2  .. ]
[END-IF]
```

An imperative statement, as opposed to a conditional statement, is one that performs an operation regardless of any existing conditions. ADD AMT-IN to AMT-OUT and MOVE NAME-IN to NAME-OUT are examples of imperative statements. They **do not** test for a condition **but** simply perform operations.

So, COBOL statements are divided into 2 broad categories :-

1) Imperative Statements: Statements that perform operations.
2) Conditional Statements: Statements which test for existing of one or more conditions.

Flowchart for the IF-THEN-ELSE Statement



A condition may test for a specific relation. A **Simple Condition** may be a single relational test of the following form :

SIMPLE RELATIONAL CONDITIONS

```
IF  identifier-1  IS  EQUAL     TO        identifier-2
IF  identifier-1  IS  LESS THAN           identifier-2
IF  identifier-1  IS  GREATER THAN        identifier-2
```

Example :

```
IF   AMT1   IS  EQUAL TO    AMT2
        DIVIDE   QTY  INTO    TOTAL
ELSE
        ADD UNIT-PRICE TO  FINAL-TOTAl.
```

```
IF   AMT1   IS  LESS  THAN    AMT2
        MOVE NAME          -IN    TO  NAME-OUT
        MOVE DESCRIPTION-IN   TO  DESCRIPTION-OUT
ELSE
        ADD   AMT1  TO   TOTAL1
        ADD   AMT2  TO   TOTAL2.
```

Using REALTIONAL Operators in Place of Words.

The following symbols for simple relational conditions are valid within a COBOL statement:

Relational Operators

Symbol	Meaning
<	IS LES THAN
>	IS GREATER THAN
=	IS EQUAL TO
<=	IS LESS THAN OR EQUAL TO
>=	IS GREATER THAN OR EQUAL TO
Not =	NOT EQUAL

Examples:

```
IF  AMT1    <= ZERO
        MOVE    ' NOT  POSITIVE  '  TO  CODE-OUT.
```

```
IF  AMT1 < ZERO    OR     AMT1  =     ZERO
        MOVE  'NOT POSITIVE  '  TO  CODE-OUT.
```

```
IF  AMT1   =   AMT2  +   500
        PERFORM  100-A-OK
```

Notes:

1) Do not mix field types in a comparison.

```
IF  CODE-IN = '123'  Then
        Move Name-In  to Name-Out.
```

CODE-IN must be of NONNUMERIC Type field.

```
IF   CTR1  =  CTR2
        ADD  AMT1  TO   TOTAL
```

BOTH CRT1 AND CRT2 fields must be of the same data type. (Either numeric or non numeric)

2) Numeric fields should not contain blanks.

```
IF  AMT1-IN  IS  EQUAL TO  10  ADD  1   TO   COUNTER.
```

IF AMT1-IN were a field defined as numeric but actually contained blanks, the instruction will result in **DATA EXCEPTION ERROR,** which causes program interrupt. The reason because ***Blanks are not valid numeric characters.***

How Comparisons Are Performed.

When Comparing numeric fields, the following are all considered equal:

012 12.00 12 +12

When comparing nonnumeric fields, the following are considered equal :

ABC ABCb (b denotes blank position) ABCbb … etc.

How about ABCb and bABC ????

ASCII and EBCDIC Collating Sequence.

When performing an alphanumeric comparisons, the hierarchy of the comparison called the **collating sequence**, depends on the computer being used. The 2 types of internal codes that are most commonly used are :

	EBCDIC	**ASCII**
Low	Special Characters	Special Characters
↓	a-z	0 – 9
↓	A – Z	A – Z
High	0 – 9	a – z

Examples :-

1) On both types of computers, numeric and alphanumeric comparisons will be performed properly. 012 < 022 < 042 .. etc.
2) Both types of computers will be able to determine if data is arranged alphabetically using UPPERCASE Letters, because A is considered LESS than B. Thus ABC < BBCD < XBCD.
3) Note that on ASCII computers UPPERCASE letters are LESS THAN LOWERCASE letters where as the RERVERSE is TRUE with EBCDIC computers. Similarly, if alphanumeric fields are being compared where they may be ***mixed of letter and digits,*** the result of comparison will differ depending on whether you are running the program on an EBCDIC or ASCII computer.

EBCDIC	ASCII
Smith < SMITH	SMITH < Smith
Stu < Sam	SAM < Stu

```
IF  ADDRESS-IN <  '100 MAIN ST'
        ADD  1  to   TOTAL.
```

If ADDRESS-IN has the value ' ROUTE 109 ' then on EBCDIC 'ROUTE 109' < '100 MAIN ST' Because R is < 1. But the reverse is true on ASCII machines

Ending Conditional Sentences with a Period.

To definitely specify the boundaries of an IF, You would use the PERIOD or END IF.

```
IF PRICE1  < PRICE2
THEN
      ADD  PRICE1  TO  TOTAL
      MOVE  2  TO  ITEM1
ELSE
       ADD PRICE2  TO   TOTAL
END - IF
MOVE  0   TO  ITEM2.
```

What happens if we omit the END IF statement ??????

The NEXT SENTENCE or CONTINUE Clause

The COBOL expression NEXT SENTENCE will enable you to :-

1) Avoid performing any operation if a condition exists.
2) To execute instructions only if the ELSE condition is met.

Example:-

```
IF  AMT1  =  AMT2
        Next Sentence
Else
        ADD  1  to  TOTAL.
```

Note:-

1) IF the NEXT SENTENCE is coded, then it must be the only imperative statement following the condition.

This is illegal coding

```
IF  A  =  B
        ADD  A  TO  TOTAL
        NEXT SENETNCE
ELSE
        ADD  1  TO  COUNTER.
```

2) The Next Sentence may not be used with END IF (Check the Compiler).

3) Try not to use the NEXT SENTENCE and rewrite the logic

```
IF  AMT1  NOT  =  AMT2
        ADD  1  TO  TOTAL
END  - IF
```

<u>Selection Using Other Options of the IF Statement</u>

1) <u>Nested Conditional.</u>

A nested conditional is a conditional in which an IF statement itself can contain additional IF clauses.

```
IF Condition-1
        Statement-1
ELSE
   IF Condition-2
        Statement-2
   ELSE
        Statement-3
   END  - IF
END – IF
```

```
IF Condition-1
        IF Condition-2
                Statement-1
        ELSE
                Statement-2
        END  - IF
ELSE
        Statement-3
END – IF
```

Example :-

```
IF  AMT  =  6
   IF  TAX  =  10
        PERFORM  500-RTN-5
    ELSE
        PERFORM   400-RTN-4
   END – IF
ELSE
    PERFORM  100-RTN-1.
```

```
IF  AMT NOT  =  6
        PERFORM 100-RTN-1
 END – IF.
 IF AMT  = 6 AND TAX  =  10
        PERFORM    500-RTN-5
 END – IF.
 IF AMT  = 6 AND TAX NOT = 10
        PERFORM  400-RTN-4
 END – IF.
```

*2) **Compound Conditional.***

This provides greater flexibility.

OR in a Compound Statement

To perform an operation or a series of operations *if any of several conditions exists*, use a compound condition with conditions separated by ***OR***.
Examples:

```
IF AMT1 = AMT2  OR AMT2 > AMT3
      PERFORM 500-TOTAL-RTN
END - IF


IF AMT1 < AMT2 OR  AMT1 = AMT4
      ADD AMT1 TO TOTAL
ELSE
      PERFORM 600-ERR-RTN
END – IF
```

Invalid Syntax

```
IF A IS EQUAL TO B OR IF B IS EQUAL TO C
      PERFORM 500-RTN-5
END - IF


IF TOTAL-1 OR TOTAL-2 = 7 ….
```

AND in a Compound Statement

If a statement or statements are to be executed only when all of several conditions are met, use the word AND in a compound conditional. *All* conditions must be met when AND is used in a compound condition. The ELSE clause is executed if *any one* of the stated conditions is not met.

Examples:

```
IF AMT1 = AMT2  AND AMT2 > AMT3
      PERFORM 500-TOTAL-RTN
END - IF


IF AMT1 < AMT2 AND  AMT1 = AMT4 AND AMT2 =  AMT4
      ADD AMT1 TO TOTAL
ELSE
      PERFORM 600-ERR-RTN
END – IF
```

Using AND and OR in the Same Statement.

There are times when BOTH the AND and the OR are required within the same compound condition

Example:

```
IF A = B OR C = D AND E = F
        PERFORM 600-PARA-1
END – IF
```

Be careful when it comes to the order of evaluation.

1) Conditions that are surrounding the AND are evaluated first
2) Conditions that are surrounding the OR are evaluated last
3) When there are several AND or OR connectors, The AND conditions are evaluated first, as they appear in the statement, from left to right. Then the OR conditions are evaluated, also from left to right.
4) To override Rules 1-3, use parentheses around conditions you want to be evaluated first

SIGN TEST

We can test whether a filed is POSITIVE, NEGATIVE, or ZERO with a sign test.

-382 is negative 383 is positive + 385 is positive

0 is neither negative nor positive in this context unless it is indicated as -0 or +0.

Example

```
IF AMT1 IS POSITIVE
        ADD 1 TO TOTAL
END – IF

IF AMT5 IS NEGATIVE
        MULTIPLY -1 BY AMT5 GIVING ABS-AMT5
END – IF
```

CLASS TEST

We can test for the type of data in a filed by coding IF identifier-1 IS NUMERIC or IF identifier-1 IS APHABETIC. The class test is useful tool for minimizing program errors. It is used for validating data.
Example:

```
IF   AMT1   IS  NUMERIC
     PERFORM   300-CALC-RTN
ELSE
     PERFORM  1000-ERR-RTN
```

IF AMT1 CONTAINS 123B THEN THE ELSE PART WILL BE EXECUTED.

Alphabetic Class Test in COBOL

Reserve Word	Meaning
ALPHABETIC	A-Z, a-z, Blanks
ALPHABETIC–UPPER	A-Z, Blanks
ALPHABETIC- LOWER	a-z, Blanks

Example:

```
IF   AMT1   IS  ALPHABETIC-UPPER
THEN
     PERFORM   300-UPPER-CASE-RTN
ENS – IF
```

3) Negating Conditionals.

Negating Simple Conditions

All simple relational , class, or sign tests may be coded using ***negated conditional.***

```
IF  identifier-1  IS NOT  { { GREATER  THAN
                           { LESS  THAN     } identifier-2
                           { EQUAL  TO
                           { NUMERIC
                           { POSITIVE
                           { NEGATIVE
                           { ZERO
                           { ALPHABETIC
                           { ALPHABETIC-UPPER
                           { ALPHABETIC-LOWER
THEN
        { Statement-1
        { NEXT SENTENCE
ELSE
        { Statement-2
        { NEXT SENTENCE
END-IF
```

Example:
The following 2 statements are the same

```
IF  AMT1   IS  EQUAL TO AMT2
        PERFORM 100-EQUAL-RTN
ELSE
        PERFORM   200-NOT-EQUAL-RTN
END-IF
```

```
IF AMT1 IS NOT  EQUAL TO AMT2
    PERFORM 200-NOT-EQUAL-RTN
ELSE
    PERFORM  100-EQUAL-RTN
END-IF
```

Practice:

1) Write a single IF statement to PERFORM 50-PARA-5 if A is between 3 and 13, ***inclusive of the end points.***
2) Write a single IF statement to PERFORM 50-PARA-5 if A is between 3 and 13, ***exclusive of the end points.***
3) Write a single if statement tp perform 300-PARA-3 if the following conditions are all met; otherwise perform 200-PARA-2

a) A = B;
b) C = D;
c) E = F.

Example Using Nested IF

Consider the following Decision Table :-

Condition	Condition	Action
A = B	C = D	Perform 100-RTN-A
A = B	C NOT = D	Perform 200-RTN-B
A NOT = B	Anything	Perform 300-RTN-C

We could code the decision table as nested condition.

```
IF   A  =   B
     IF  C  = D
              PERFORM    100-RTN-A
     ELSE
              PERFORM    200-RTN-B
ELSE
     PERFORM   300-RTN-C.
```

```
IF   A  =   B
          IF  C  = D
                   PERFORM    100-RTN-A
          ELSE
                   PERFORM    200-RTN-B
          END-IF
ELSE
          PERFORM   300-RTN-C
END-IF.
```