COBOL - CONDITION STATEMENTS

Conditional statements are used to change the execution flow depending on certain conditions specified by the programmer. Conditional statements will always evaluate to true or false. Conditions are used in IF, Evaluate and Perform statements. The different types of conditions are as follows:

- IF Condition Statement
- Relation Condition
- Sign Condition
- Class Condition
- Condition-Name Condition
- Negated Condition
- Combined Condition

IF Condition Statement

IF statement checks for conditions. If a condition is true the IF block is executed; and if the condition is false, the ELSE block is executed.

END-IF is used to end the IF block. To end the IF block, a period can be used instead of END-IF. But it is always preferable to use END-IF for multiple IF blocks.

Nested-IF : IF blocks appearing inside another IF block. There is no limit to the depth of nested IF statements.

Syntax

Following is the syntax of IF condition statements:

```
IF [condition] THEN
   [COBOL statements]
ELSE
   [COBOL statements]
END-IF.
```

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM1 PIC 9(9).
   01 WS-NUM2 PIC 9(9).
   01 WS-NUM3 PIC 9(5).
   01 WS-NUM4 PIC 9(6).

PROCEDURE DIVISION.
   A000-FIRST-PARA.
   MOVE 25 TO WS-NUM1 WS-NUM3.
   MOVE 15 TO WS-NUM2 WS-NUM4.

   IF WS-NUM1 > WS-NUM2 THEN
      DISPLAY 'IN LOOP 1 - IF BLOCK'

      IF WS-NUM3 = WS-NUM4 THEN
         DISPLAY 'IN LOOP 2 - IF BLOCK'
      ELSE
```

```
         DISPLAY 'IN LOOP 2 - ELSE BLOCK'
      END-IF

   ELSE
      DISPLAY 'IN LOOP 1 - ELSE BLOCK'
   END-IF.

STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
IN LOOP 1 - IF BLOCK
IN LOOP 2 - ELSE BLOCK
```

Relation Condition

Relation condition compares two operands, either of which can be an identifier, literal, or arithmetic expression. Algebraic comparison of numeric fields is done regardless of size and usage clause.

For non-numeric operands

If two non-numeric operands of equal size are compared then the characters are compared from left with the corresponding positions till the end is reached. The operand containing greater number of characters is declared greater.

If two non-numeric operands of unequal size are compared, then the shorter data item is appended with spaces at the end till the size of the operands becomes equal and then compared according to the rules mentioned in the previous point.

Syntax

Given below following is the syntax of Relation condition statements:

```
[Data Name/Arithmetic Operation]

          [IS] [NOT]

[Equal to (=),Greater than (>), Less than (<),
 Greater than or Equal (>=), Less than or equal (<=) ]

[Data Name/Arithmetic Operation]
```

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM1 PIC 9(9).
   01 WS-NUM2 PIC 9(9).

PROCEDURE DIVISION.
   A000-FIRST-PARA.
   MOVE 25 TO WS-NUM1.
   MOVE 15 TO WS-NUM2.

   IF WS-NUM1 IS GREATER THAN OR EQUAL TO WS-NUM2 THEN
      DISPLAY 'WS-NUM1 IS GREATER THAN WS-NUM2'
   ELSE
      DISPLAY 'WS-NUM1 IS LESS THAN WS-NUM2'
   END-IF.

STOP RUN.
```

JCL to execute the above COBOL program.

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program it produces the following result:

```
WS-NUM1 IS GREATER THAN WS-NUM2
```

Sign Condition

Sign condition is used to check the sign of a numeric operand. It determines whether a given numeric value is greater than, less than, or equal to ZERO.

Syntax

Following is the syntax of Sign condition statements:

```
[Data Name/Arithmetic Operation]

          [IS] [NOT]

[Positive, Negative or Zero]

[Data Name/Arithmetic Operation]
```

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM1 PIC S9(9) VALUE -1234.
   01 WS-NUM2 PIC S9(9) VALUE 123456.

PROCEDURE DIVISION.
   A000-FIRST-PARA.
   IF WS-NUM1 IS POSITIVE THEN
      DISPLAY 'WS-NUM1 IS POSITIVE'.

   IF WS-NUM1 IS NEGATIVE THEN
      DISPLAY 'WS-NUM1 IS NEGATIVE'.

   IF WS-NUM1 IS ZERO THEN
      DISPLAY 'WS-NUM1 IS ZERO'.

   IF WS-NUM2 IS POSITIVE THEN
      DISPLAY 'WS-NUM2 IS POSITIVE'.

STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program it produces the following result:

```
WS-NUM1 IS NEGATIVE
WS-NUM2 IS POSITIVE
```

Class Condition

Class condition is used to check if an operand contains only alphabets or numeric data. Spaces are considered in ALPHABETIC, ALPHABETIC-LOWER, and ALPHABETIC-UPPER.

Syntax

Following is the syntax of Class condition statements:

```
[Data Name/Arithmetic Operation>]

          [IS] [NOT]

[NUMERIC, ALPHABETIC, ALPHABETIC-LOWER, ALPHABETIC-UPPER]

[Data Name/Arithmetic Operation]
```

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM1 PIC X(9) VALUE 'ABCD '.
   01 WS-NUM2 PIC 9(9) VALUE 123456789.

PROCEDURE DIVISION.
   A000-FIRST-PARA.

   IF WS-NUM1 IS ALPHABETIC THEN
      DISPLAY 'WS-NUM1 IS ALPHABETIC'.

   IF WS-NUM1 IS NUMERIC THEN
      DISPLAY 'WS-NUM1 IS NUMERIC'.

   IF WS-NUM2 IS NUMERIC THEN
      DISPLAY 'WS-NUM1 IS NUMERIC'.

STOP RUN.
```

JCL to execute the above COBOL program.

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
WS-NUM1 IS ALPHABETIC
WS-NUM1 IS NUMERIC
```

Condition-name Condition

A condition name is a user-defined name. It contains a set of values specified by the user. It behaves like Boolean variables. They are defined with level number 88. It will not have a PIC clause.

Syntax

Following is the syntax of user-defined condition statements:

```
88 [Condition-Name] VALUE [IS, ARE] [LITERAL] [THRU LITERAL].
```

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM PIC 9(3).
   88 PASS VALUES ARE 041 THRU 100.
   88 FAIL VALUES ARE 000 THRU 40.

PROCEDURE DIVISION.
   A000-FIRST-PARA.
   MOVE 65 TO WS-NUM.

   IF PASS
      DISPLAY 'Passed with ' WS-NUM ' marks'.

   IF FAIL
      DISPLAY 'FAILED with ' WS-NUM 'marks'.

STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
Passed with 065 marks
```

Negated Condition

Negated condition is given by using the NOT keyword. If a condition is true and we have given NOT in front of it, then its final value will be false.

Syntax

Following is the syntax for Negated condition statements:

```
IF NOT [CONDITION]
   COBOL Statements
END-IF.
```

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM1 PIC 9(2) VALUE 20.
   01 WS-NUM2 PIC 9(9) VALUE 25.

PROCEDURE DIVISION.
   A000-FIRST-PARA.

   IF NOT WS-NUM1 IS LESS THAN WS-NUM2 THEN
      DISPLAY 'IF-BLOCK'
   ELSE
      DISPLAY 'ELSE-BLOCK'
   END-IF.
```

```
STOP RUN.
```

JCL to execute the above COBOL program.

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
ELSE-BLOCK
```

Combined Condition

A combined condition contains two or more conditions connected using logical operators AND or OR.

Syntax

Following is the syntax of combined condition statements:

```
IF [CONDITION] AND [CONDITION]
   COBOL Statements
END-IF.
```

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM1 PIC 9(2) VALUE 20.
   01 WS-NUM2 PIC 9(2) VALUE 25.
   01 WS-NUM3 PIC 9(2) VALUE 20.

PROCEDURE DIVISION.
   A000-FIRST-PARA.

   IF WS-NUM1 IS LESS THAN WS-NUM2 AND WS-NUM1=WS-NUM3 THEN
      DISPLAY 'Both condition OK'
   ELSE
      DISPLAY 'Error'
   END-IF.

STOP RUN.
```

JCL to execute the above COBOL program.

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
Both condition OK
```

Evaluate Verb

Evaluate verb is a replacement of series of IF-ELSE statement. It can be used to evaluate more than one condition. It is similar to SWITCH statement in C programs.

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.

DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-A PIC 9 VALUE 0.

PROCEDURE DIVISION.
   MOVE 3 TO WS-A.

   EVALUATE TRUE
      WHEN WS-A > 2
         DISPLAY 'WS-A GREATER THAN 2'

      WHEN WS-A < 0
         DISPLAY 'WS-A LESS THAN 0'

      WHEN OTHER
         DISPLAY 'INVALID VALUE OF WS-A'
   END-EVALUATE.

STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
WS-A GREATER THAN 2
```