PERNYATAAN KONDISI PADA COBOL

OBJEKTIF :

1. Mahasiswa Mengetahui Pengetahuan Dasar Tentang Pernyataan Kondisi pada COBOL.
2. Mahasiswa Mampu Membuat dan Menjalankan Program Kondisi pada OpenCobolIDE.
3. Mahasiswa Mampu Memahami Struktur Pernyataan Kondisi pada COBOL.

Percabangan di dalam pemrograman digunakan komputer untuk menentukan Langkah kerja. Pernyataan kondisi digunakan untuk mengubah aliran eksekusi program tergantung dengan kondisi tertentu yang ditentukan oleh pemrogram. Pernyataan ini akan menghasilkan nilai benar (true) atau salah (false). Jika nilai yang dihasilkan benar, maka perintah akan dieksekusi. Sedangkan jika nilai yang dihasilkan salah, maka instruksi tidak akan dilaksanakan. Pernyataan kondisi dalam pemrograman COBOL antara lain pernyataan IF dan pernyataan Evaluate.

6.1 Pernyataan IF

Ketika suatu program berjalan, pernyataan program dieksekusi satu demi satu, secara berurutan, kecuali sebuah pernyataan ditemui yang mengubah urutan eksekusi.

Pernyataan IF adalah salah satu jenis pernyataan yang dapat mengubah urutan eksekusi dalam suatu program. Ini memungkinkan Anda untuk menentukan bahwa blok kode akan dieksekusi hanya jika kondisinya terlampir pernyataan IF terpenuhi. Pernyataan IF diakhiri dengan statement END IF.

Bentuk umum Penyataan IF :

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

Pada pernyataan IF, terdapat beberapa jenis kondisi yang dapat digunakan, antara lain Relation Condition, Sign Condition, Class Condition, Condition Name, Negates Condition, Combined Condition.

RELATION CONDITION

Relation Condition membandingkan dua operand, yang keduanya dapat berupa identifier, literal ataupun ekspresi aritmatika.

Jika dua operan non-numerik dengan ukuran yang sama dibandingkan, maka karakter dibandingkan dari kiri dengan posisi yang sesuai sampai akhir tercapai. Operand yang mengandung lebih banyak karakter dinyatakan lebih besar.

Jika dua operan non-numerik dengan ukuran yang tidak sama dibandingkan, maka item data yang lebih pendek ditambahkan dengan spasi di akhir sampai ukuran operan menjadi sama dan kemudian dibandingkan sesuai dengan aturan yang disebutkan pada poin sebelumnya.

Bentuk umum Relation Condition :

```
[Data Name/Arithmetic Operation]
  [IS] [NOT]
[Equal to (=),Greater than (>), Less than (<),
Greater than or Equal (>=), Less than or equal (<=)]
[Data Name/Arithmetic Operation]
```

Contoh program menggunakan relation condition :

```
IDENTIFICATION DIVISION.
PROGRAM-ID.PERNYATAAN-IF.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 NILAI PIC 9(5).

PROCEDURE DIVISION.
MULAI.
    DISPLAY 'MASUKKAN NILAI UJIAN : '.
    ACCEPT NILAI.
    IF NILAI GREATER THAN OR EQUAL 70 THEN
        DISPLAY 'ANDA DINYATAKAN LULUS'
    ELSE
      DISPLAY 'ANDA DINYATAKAN TIDAK LULUS'
    END-IF.
SELESAI.
    STOP RUN.
```

Pada program di atas, akan menghasilkan nilai berupa inputan untuk memasukkan nilai ujian. Jika nilai yang dimasukkan lebih besar sama dengan 70, maka program akan mencetak ANDA DINYATAKAN LULUS. Namun jika nilai yang dimasukan lebih kecil dari 70, maka program akan mencetak ANDA DINYATAKAN TIDAK LULUS.

SIGN CONDITION

Sign Condition digunakan untuk memeriksa tanda operan numerik. Ini menentukan apakah nilai numerik yang diberikan lebih besar dari, kurang dari, atau sama dengan NOL.

Bentuk umum Sign Condition :

```
[Data Name/Arithmetic Operation]
  [IS] [NOT]
[Positive, Negative or Zero]
[Data Name/Arithmetic Operation]
```

Contoh program menggunakan sign condition :

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SIGN-CONDITION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 ANGKA-1 PIC S9(6) VALUE -12345.
01 ANGKA-2 PIC S9(6) VALUE 0.
PROCEDURE DIVISION.
```

```
MULAI.
    IF ANGKA-1 IS POSITIVE THEN
      DISPLAY 'ANGKA-1 MERUPAKAN ANGKA POSITIF'.
    IF ANGKA-1 IS NEGATIVE THEN
      DISPLAY 'ANGKA-1 MERUPAKAN ANGKA NEGATIF'.
    IF ANGKA-2 IS POSITIVE THEN
      DISPLAY 'ANGKA-2 MERUPAKAN ANGKA POSITIF'.
    IF ANGKA-2 IS ZERO THEN
      DISPLAY 'ANGKA-2 MERUPAKAN ANGKA NOL'.
SELESAI.
    STOP RUN.
```

Pada program di atas, menggunakan picture clause S9 untuk bilangan numerik bertanda. Berdasarkan nilai yang ditentukan pada VALUE di Working-Storage Section, output yang dihasilkan adalah ANGKA-1 MERUPAKAN ANGKA NEGATIF dan ANGKA-2 MERUPAKAN ANGKA NOL.

CLASS CONDITION

Class Condition digunakan untuk memeriksa apakah operan hanya berisi huruf atau data numerik. Spasi dipertimbangkan dalam ALPHABETIC, ALPHABETIC-LOWER, dan ALPHABETIC-UPPER.

Bentuk Penulisan :

```
[Data Name/Arithmetic Operation>]
  [IS] [NOT]
[NUMERIC, ALPHABETIC, ALPHABETIC-LOWER, ALPHABETIC-UPPER]
[Data Name/Arithmetic Operation]
```

Contoh program menggunakan class condition :

```
IDENTIFICATION DIVISION.
PROGRAM-ID. CLASS-CONDITION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 LITERAL-1 PIC X(5) VALUE 'ABCD'.
01 LITERAL-2 PIC 9(5) VALUE 12345.

PROCEDURE DIVISION.
MULAI.
    IF LITERAL-1 IS ALPHABETIC THEN
      DISPLAY 'LITERAL-1 MERUPAKAN LITERAL ALPHABETIC'.
    IF LITERAL-1 IS NUMERIC THEN
      DISPLAY 'LITERAL-1 MERUPAKAN LITERAL NUMERIC'.
    IF LITERAL-2 IS ALPHABETIC THEN
      DISPLAY 'LITERAL-2 MERUPAKAN LITERAL ALPHABETIC'.
    IF LITERAL-2 IS NUMERIC THEN
      DISPLAY 'LITERAL-2 MERUPAKAN LITERAL NUMERIC'.
SELESAI.
    STOP RUN.
```

Berdasarkan program di atas, compiler akan memproses dan menyeleksi nilai pada working-storage dengan kondisi-kondisi yang ada pada Procedure Division. Jika nilai yang didefinisikan pada literal-1 adalah ABCD dan literal-2 adalah 12345, maka output yang dihasilkan adalah LITERAL-1 ADALAH LITERAL ALPHABETIC dan LITERAL-2 ADALAH LITERAL NUMERIC.

CONDITION NAME

Condition Name adalah nama yang ditentukan pengguna. Condition name berisi serangkaian nilai yang ditentukan oleh pengguna. Berperilaku seperti variabel Boolean. Condition name didefinisikan dengan nomor level 88 dan tidak akan memiliki PIC Clause.

Bentuk penulisan :

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

Contoh program menggunakan condition name :

```
        IDENTIFICATION DIVISION.
        PROGRAM-ID. CONDITION-NAME.
        DATA DIVISION.
        WORKING-STORAGE SECTION.
        01 TEST-NUMBER PIC 9(3).
        88 PASS VALUES ARE 041 THRU 100.
        88 FAIL VALUES ARE 000 THRU 040.

        PROCEDURE DIVISION.
        MULAI.
        MOVE 65 TO TEST-NUMBER.
            IF PASS
                DISPLAY 'PASSED WITH ' TEST-NUMBER ' MARKS'.
            IF FAIL
                DISPLAY 'FAILED WITH ' TEST-NUMBER ' MARKS'.
        SELESAI.
            STOP RUN.
```

Jika program dijalankan maka output yang dihasilkan yaitu PASSED WITH 065 MARKS.

NEGATED CONDITION

Negated Condition merupakan kondisi yang dinegasikan dengan memberikan kata kunci NOT. Jika suatu kondisi benar dan programmer telah memberikan NOT di depannya, maka nilai akhirnya akan bernilai salah.

Bentuk penulisan :

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

Contoh program menggunakan negated condition :

```
        IDENTIFICATION DIVISION.
        PROGRAM-ID. NEGATED-CONDITION.
        DATA DIVISION.
        WORKING-STORAGE SECTION.
        01 LITERAL-1 PIC 9(2) VALUE 20.
        01 LITERAL-2 PIC 9(2) VALUE 25.
```

```
        PROCEDURE DIVISION.
        MULAI.
            IF NOT LITERAL-1 LESS THAN LITERAL-2
              DISPLAY 'LITERAL-1 LEBIH KECIL DARI LITERAL-2'
            ELSE
              DISPLAY 'LITERAL-1 LEBIH BESAR DARI LITERAL-2'
            END-IF.

        SELESAI.
            STOP RUN.
```

Jika program dijalankan, maka akan menghasilkan nilai pada ELSE-BLOCK. Karena nilai pada statement IF dianggap salah dengan menggunakan NOT (Negated Condition). Sehingga output yang dihasilkan yaitu pada statement ELSE karena statement IF tidak bernilai benar.

COMBINED CONDITION

Combined condition berisi dua atau lebih kondisi yang terhubung menggunakan operator logika AND atau OR.

Bentuk umum penulisan :

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

Contoh program menggunakan combined condition :

```
        IDENTIFICATION DIVISION.
        PROGRAM-ID. COMBINED-CONDITION.
        DATA DIVISION.
        WORKING-STORAGE SECTION.
        01 LITERAL-1 PIC 9(2) VALUE 20.
        01 LITERAL-2 PIC 9(2) VALUE 25.
        01 LITERAL-3 PIC 9(2) VALUE 20.

        PROCEDURE DIVISION.
        MULAI.
            IF LITERAL-1 LESS THAN LITERAL-2 AND LITERAL-1 = LITERAL-3 THEN
                DISPLAY 'HASIL DARI KEDUA KONDISI TERSEBUT ADALAH TRUE'
            ELSE
                DISPLAY 'ERROR'
            END-IF.
        SELESAI.
            STOP RUN.
```

Pada progam di atas, compiler akan membandingkan 2 kondisi pada statement IF. Statement IF akan dieksekusi apabila dua kondisi tersebut terpenuhi. Jika tidak terpenuhi, maka blok statement ELSE akan di eksekusi. Hasil output dari program tersebut yaitu HASIL DARI KEDUA KONDISI TERSEBUT ADALAH TRUE.

6.2 Pernyataan EVALUATE

Pernyataan EVALUATE merupakan serangkaian pernyataan kondisi yang dapat digunakan untuk menggantikan pernyataan IF-ELSE. Pernyataan ini dapat digunakan untuk mengevaluasi lebih dari satu kondisi.

Jika program memiliki beberapa kondisi yang harus diperiksa, maka penggunaan EVALUATE ini lebih baik daripada penggunaan statement IF-ELSE. Dengan menggunakan satu kondisi EVALUATE, dapat memeriksa beberapa kondisi sekaligus. Pernyataan ini diakhiri dengan END-EVALUATE.

Jika tidak ada EVALUATE yang memenuhi kondisi, maka secara default pernyataan yang dikodekan dalam WHEN OTHER akan dieksekusi dan mengontrol transfer ke pernyataan yang dapat dieksekusi berikutnya setelah berakhirnya EVALUATE.

Bentuk umum pernyataan EVALUATE :

```
EVALUATE {condition-for-evaluate}
  WHEN value
     imperative-statements
  WHEN value
     imperative statements
 .....
  WHEN OTHER
     imperative statements
 END-EVALUATE
```

Pada {condition-for-evaluate} dapat berupa :

1. Constant/literals
2. Expression/identifier
3. Reference
4. TRUE
5. FALSE

Contoh program menggunakan pernyataan evaluate :

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PERNYATAAN-EVALUATE.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 BILANGAN PIC 9(2).

PROCEDURE DIVISION.
MULAI.
    DISPLAY 'MASUKKAN BILANGAN : '.
    ACCEPT BILANGAN.

    EVALUATE TRUE
        WHEN BILANGAN > 5
            DISPLAY 'BILANGAN INI LEBIH BESAR DARI 5'
        WHEN BILANGAN < 3
            DISPLAY 'BILANGAN INI LEBIH KECIL DARI 3'
        WHEN OTHER
            DISPLAY 'BILANGAN INI BERNILAI 4'
    END-EVALUATE.
SELESAI.
```

```
            STOP RUN.
```

Pada program di atas, akan menghasilkan nilai berupa inputan untuk memasukkan bilangan. Pada pernyataan evaluate, terdapat beberapa kondisi yang akan diseleksi. Jika bilangan memenuhi kondisi pada WHEN BILANGAN > 5 maka output yang tercetak adalah BILANGAN INI LEBIH BESAR DARI 5. Jika kondisi tersebut tidak terpenuhi, compiler akan menyeleksi ke kondisi kedua pada klausa WHEN BILANGAN < 3. Jika terpenuhi, maka akan mencetak output BILANGAN INI LEBIH KECIL DARI 3. Jika semua kondisi pada klausa WHEN tidak terpenuhi, maka statement pada klausa WHEN OTHER akan dieksekusi, yaitu BILANGAN INI BERNILAI 4. Apabila kita memasukkan nilai input 7, maka output yang dihasilkan adalah BILANGAN INI LEBIH DARI 5.