BAB 3. PROCEDURE DIVISION

Merupakan inti dari pemrograman COBOL. Statement yang ada pada PROCEDURE DIVISION dibentuk dari verb, diantaranya: MOVE, DISPLAY, ACCEPT, dan STOP.

- **MOVE verb**
  Digunakan untuk memindahkan data dari satu field ke lokasi field yang lain, sehingga input data dapat dimanipulasi untuk menghasilkan output.
  Bentuk umum :

  MOVE { nama-data-1 / literal } TO nama-data-2 [, nama-data-3] …

  Bentuk khusus dari MOVE adalah MOVE CORRESPONDING, yang berguna untuk memindahkan data dari group data item ke group lain.
  Bentuk umum ;

  MOVE CORRESPONDING nama-data-1 TO nama-data-2

- **DISPLAY verb**
  Digunakan untuk menampilkan hasil dilayar ataupun printer. Jika dipergunakan statement WRITE untuk menampilkan hasil di printer, maka print-file harus disebutkan terlebih dahulu di ENVIRONMENT DIVISION pada FILE-CONTROL. Ada 3 bentuk statement DISPLAY:
  a. Bentuk 1

     DISPLAY nama-layar

  b. Bentuk 2

     DISPLAY { nama-data / literal } [, nama-data / , literal] UPON nama-mnemonic

  c. Bentuk 3

     DISPLAY { (posisi tampilan) { nama-data / literal / ERASE } } …UPON nama-nemonic

♦ **ACCEPT verb**
Digunakan untuk memasukkan data lewat layar sewaktu program tersebut dijalankan (runtime). Ada 4 bentuk statement ACCEPT :

a. Bentuk 1

BU : ACCEPT nama-data

Data yang dimasukkan akan ditempatkan pada nama-data setelah ACCEPT, yang bentuk, jenis dan panjangnya sudah ditentukan dalam DATA DIVISION.

b. Bentuk 2

BU : ACCEPT nama-layar [ON ESCAPE statement-imperative]

Digunakan untuk menerima data dan mengirimkan data tersebut ke (TO) atau menggunakan (USING) field data item yang disebutkan pada nama-layar di SCREEN SECTION dalam DATA DIVISION.

c. Bentuk 3

BU :

ACCEPT (posisi layar) nama data WITH { ZERO - FILL | SPACE - FILL | LEFT - JUSTIFY | RIGHT - JUDTIFY | TRAILING - SIGN | PROMPT | UPDATE | LENGTH - CHECK | EMPTY – CHECK | AUTO – SKIP | NO – ECHO | BEEP }

- ZERO-FILL phrase menyebabkan bila posisi-posisi field data-item penerima data tidak di isi dengan data (langsung menekan enter) akan terisis dengan nol.
- SPACE-FILL phrase menyababkan bila posisi-posisi fieldd data-item dilayar tidak di isi dengan data (langsung menekan enter) akan terisi blank pada layar tetapi field data-item penerima tetap berisi nilai nol atau nilai sebelumnya, biasanya untuk jenis data numerik.
- LEFT-JUSTIFY phrase tidak berfungsi pada MS COBOL, tetapi boleh ditulis
- RIGHT-JUSTIFY phrase menyebabkan setelah data dimasukkan, hasil akhir yang tampak dilayar akan rata sebelah kanan. Digunakan untuk jenis data-item alphabetik atau alphanumerik.
- TRAILING-SIGN phrase menyebabkan tanda operasi + atau – tampak diposisi paling kanan dari field data input.

- PROMPT phrase menyebabkan tampilan untuk field data-item penerima berbentuk nol untuk posisi digit, titik untuk desimal point dan spasi untuk tanda operasi + (plus) atau – (minus).
- UPDATE phrase menyebabkan tampilan utuk field data-item penerima berbentuk nilai awal dari field penerima tersebut.
- LENGTH-CHECK phrase menyebabkan penekanan tombol carriage return tidak berfungsi kalau semua posisi field penerima belum terisi semua.
- EMPTY-CHECK phrase menyebabkan penekanan tombol carriage return tidak berfungsi jika tidak paling sedikit sebuah karakter atau angka yang bukan sifatnya terminator sudah di input.
- AUTO-SKIP phrase menyebabkan proses pemasukan data bergeser ke field penerima data lain berikutnya, bila posisi field penerima sudah penuh terisi tanpa harus menekan tombol carriage return atau tombol terminator yang lainnya.
- NO-ECHO phrase menyebabkan data yang dimasukkan tidak tampak dilayar.
- BEEP phrase menyebabkan bunyi bel sewaktu data di input.

d. Bentuk 4

bu : ACCEPT nama-data FROM { DATE | DAY | TIME | ESCAPE-KEY }

- DATE, akan mendapatkan 6 digit nilai standard dengan bentuk YYMMDD, diambil langsung dari "system-date".(2 digit tahun, 2 digit bulan, 2 digit tanggal)
- DAY, akan mendapatkan 5 digit nilai "julian date" dengan bentuk YYDDD (2 digit tahun, 3 digit jumlah hari untuk tanggal tersebut)
- TIME, akan mendapatkan 8 digit nilai dengan bentuk JJMMDDSS (2 digit jam 00-23, 2 digit menit 00-59, 2 digit detik 00-59, 2 digit seperseratus detik 00-99).
- ESCAPE-KEY, akan mendapatkan 2 digit kode yang dihasilkan dari penekanan tombol-tombol terminator. yaitu : Backtab = 99, Escape = 01, Carriage-return = 00, Function key 1 – 10 = 02 – 11.

- **STOP verb**

Digunakan untuk menghentikan program baik secara permanen maupun sementara.

BU:

STOP { literal | RUN }

STOP literal, akan menyebabkan proses program terhenti sementara dan literal akan ditampilkan dilayar. Jika operator menekan sembarang tombol maka program akan dilanjutkan mulai statement setelah STOP literal tersebut.
STOP RUN, akan menyebabkan program berhenti secara permanen.

- **ADD Verb**

Digunakan untuk menambahkan 2 atau lebih operand numerik dan menyimpan hasilnya.

BU-1:

ADD { nama-data-1, / literal-1 } { nama-data-2 / literal-2 } ….TO nama-data-m [ROUNDED]
[; ON SIZE ERROR statement imperative]

contoh : ADD 8 to B. atau ADD A, 15 to B

BU-2:

ADD { nama-data-1, / literal-1 } { nama-data-2 … / literal-2 } GIVING nama-data-m [ROUNDED]
[; ON SIZE ERROR statement imperative]

contoh : **ADD A, B GIVING C atau ADD A, 15 GIVING C Rounded.**

Keterangan :

1. **TO** digunakan bila beberapa nilai akan dijumlahkan dan hasilnya akan disimpan pada salah satu operand.
2. **GIVING** digunakan bila beberapa nilai dijumlahkan dan hasilnya disimpan pada nama data yang lain.
3. Field penerima harus merupakan nama data, bukan literal.
4. Bentuk **TO** dan **GIVING** harus ada dan salah satu diantaranya, tidak boleh dipergunakan keduanya.
5. Semua nama-data yang dipergunakan di dalam operasi aritmatika harus berbentuk data numerik dengan picture yang belum diedit kecuali operand dari field penerima.
6. **ROUNDED** option digunakan bila diinginkan hasil perhitungan dibulatkan
7. **ON SIZE ERROR** digunakan bila hasil perhitungan untuk digit-digit bilangan utuh (high order digits) tidak bisa masuk seluruhnya pada field penerima, nilai tidak akan disimpan di storage dan program akan melanjutkan pada imperative statement yang mengikutinya.

♦ **SUBSTRACT Verb**

Digunakan untuk operasi pengurangan suatu nilai data numerik.

BU :

SUBSTRACT { nama-data-1 / literal-1 }, { nama-data-2 / literal2 } **FROM** { Nama-data-n / literal-n }
GIVING nama-data-m **[ROUNDED]**
[;ON SIZE ERROR imperative statement]

Contoh : SUBSTRACT A FROM B
SUBSTRACT 2 FROM B
SUBSTRACT A, B FROM C GIVING D ROUNDED

- **MULTIPLY Verb**

Digunakan untuk mengalikan 2 nilai numerik dan menyimpan hasilnya.

BU 1 :

MULTIPLY { nama-data-1 / literal-1 } BY nama-data-2

Contoh : MULTIPLY A BY B
MULTIPLY 5 BY A

BU 2 :

MULTIPLY { nama-data-1 / literal-1 } BY { nama-data-2 / literal-2 }
GIVING nama-data-3 [ROUNDED]
[; ON SIZE ERROR imperative statement]

Contoh : MULTIPLY A BY B GIVING C ROUNDED

- **DIVIDE Verb**

Digunakan untuk membentuk statement operasi pembagian.

BU 1 :

DIVIDE { nama-data-1 / literal-1 } INTO nama-data-2 [ROUNDED]
[; ON SIZE ERROR imperative statement]

BU 2 :

DIVIDE { nama-data-1 / literal-1 } INTO { nama-data-2 / literal-2 }
GIVING nama-data-3 [ROUNDED]
[; ON SIZE ERROR imperative statement]

BU 3 :

DIVIDE { nama-data-1 / literal-1 } BY { nama-data-2 / literal-2 }
GIVING nama-data-3 [ROUNDED]
[; ON SIZE ERROR imperative statement]

BU 4 :

DIVIDE { nama-data-1 / literal-1 } INTO { nama-data-2 / literal-2 }
GIVING nama-data-3 [ROUNDED]
REMAINDER nama-data-4
[; ON SIZE ERROR imperative statement]

BU 5 :

DIVIDE { nama-data-1 / literal-1 } BY { nama-data-2 / literal-2 }
GIVING nama-data-3 [ROUNDED]
REMAINDER nama-data-4
[; ON SIZE ERROR imperative statement]

CONTOH :
DIVIDE A INTO B
DIVIDE 5 BY A GIVING C ROUNDED
DIVIDE B INTO A GIVING C

- **COMPUTE**

  Digunakan untuk operasi yang lebih rumit, untuk menyederhanakan 4 arithmatic verb sebelumnya.

  BU :

  COMPUTE nama-data-1 [ROUNDED] = ungkapan aritmatika
  [; ON SIZE ERROR imperative statement]

- **GO TO Verb**

  Digunakan untuk alih kontrol tanpa syarat ke paragraph tertentu.

  BU :

  GO TO nama-paragraph

- **GO TO … DEPENDING Verb**

  Digunakan untuk alih kontrol bersyarat. Beralih pada paragraph tertentu dengan kondisi tertentu.

BU :

GO TO nama-paragraph-1, nama-paragraph-2, … nama-paragraph-n
DEPENDING ON nama-data

Contoh program :

```
IDENTIFICATION DIVISION.
PROGRAM-ID. GOT0.
AUTHOR. JOGIYANTO HM .
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 JAWAB PIC X(5l).
PROCEDURE DIVISION.
TANYAKAN.
    DISPLAY 'MAKAN atau MlNUM ?'
    ACCEPT JAWAB.
SELEKSI.
    IF JAWAB = 'MINUM'  GO TO MlNUM.
MAKAN.
    DISPLAY 'ANDA HARUS MEMBAYAR Rp 2000,-'
    GO TO SELESAI.
MlNUM.
    DISPLAY 'HANYA MEMBAYAR SEBESAR Rp 500,-'
    DISPLAY 'UNTUK MINUM SEPUASNYA' .
SELESAI.
    STOP RUN.
```

*menggunakan GO TO..DEPENDING

……

```
WORKING-STORAGE SECTION.
 77 JABATAN PIC 9.
PROCEDURE DIVISIOK.
TANYA-JABATAN.
        DISPLAY 'JABATAN ANDA (1. 2. 3 ) ?'
        ACCEPT JABATAN
        DISPLAY SPACE
        GO TO TUNJ1, TUNJ2, TUNJ3  DEPENDING ON JABATAN
        DISPLAY 'TIDAK ADA JABATAN TERSEBUT, ULANGI ‘
        GO TO TANYA-JABATAN
    TUNJ1.
```

```
        DISPLAY 'TUNJANGAN ANDA SEBESAR Rp 100000,-'
        GO TO SELESAI.
    TUNJ2.
        DISPLAY ‘TUNJANGAN ANDA SEBESAR Rp 75000,’
        GO TO SELESAI.
    TUNJ3.
        DISPLAY 'TUNJANGAN ANDA SEBESAR Rp 5000C,-'.
    SELESAI.
        STOP RUN.
```

- **ALTER Verb**

  Digunakan untuk merubah arah tujuan proses dari statement GO TO yang telah ada di **prgram**.

  BU :

  ALTER nama-paragraph-1 **TO** [**PROCEED TO**] nama-paragraph-2

  CONTOH :

```
        PROCEDURE DIVISION.
        TAMPILKAN.
            DISPLAY ‘KOMPUTER' .
        PARA-1.
            GO TO PARA-:2.
        PARA-2.
            ALTER PARA-1 TO PROCEED TO SELESAI
            GO TO TAMPIL.
        SELESAI.
            STOP RUN.
```

- **PERFORM Verb**

  Digunakan untuk membuat suatu statement yang akan membawa proses dari program meloncat kesuatu paragraph, kemudian menjalankan seluruh statetement yang ada pada paragraph tersebut. Jika telah selesai akan kembali ke statement setelah PERFORM.

  BU1 :

  PERFORM nama-paragrapg-1 { **THROUGH** / **THRU** } nama-paragraph-2

BU2 :

PERFORM nama-paragrapg-1 { **THROUGH** / **THRU** } nama-paragraph-2

{ Nama-data / Numeric-integer } **TIMES**

BU3 :

PERFORM nama-paragrapg-1 { **THROUGH** / **THRU** } nama-paragraph-2

UNTIL Kondisi

BU4 :

PERFORM nama-paragraph-1 **THROUGH** nama-paragraph-2

VARYING { Nama-data-1 / Nama-index-1 } **FROM** { nama-data-2 / integer-1 / Nama-index-2 }

BY { nama-data-3 / Integer-2 } **UNTIL** kondisi-1

[**AFTER** { Nama-data-4 / nama-index-3 } **FROM** { nama-data-5 / integer-3 / Nama-index-4 }

BY { nama-data-6 / Integer-4 } **UNTIL** kondisi-2

[**AFTER**…….

Contoh1 :

```
DATA DIVISION.
PROCEDlJRE DIVISION.
MULAl.
        PERFORM MENCETAK 5 TIMES
        STOP RUN.
MENCETAK .
        DISPLAY 'COBOL' .
```

Contoh2 :

```
    IDENTIFICATION DIVISION.
    PROGRAM-ID. CONTOH-PERFORM.
    ENVIRONMENT DIVISION.
    WORKING-STORAGE SECTION.
    77 NILAIUTS PIC 9(3)V99.
    77 NILAIUAS PIC 9(3)V99.
    77 NILAI PIC Z(3).99.
    77 BENAR PIC X VALUE SPACE.
    SCREEN SECTION.
    01 HAPUS-LAYAR.
            02 BLANK SCREEN.
    PROCEDURE DIVISION.
    MULAI.
            DISPLAY HAPUS-LAYAR
            PERFORM MASUKKAN-DATA UNTIL BENAR = 'Y'
            PERFORM HITUNG
            PERFORM TAMPILKAN
            STOP RUN.
MASUKKAN-DATA.
    ……..
    DISPLAY 'SUDAH BENAR DATA INI (Y/T) ? '
    ACCEPT BENAR.
HITUNG.
    ……
```

Contoh3 :

```
DATA DIVISION.
WORKING-STORAGE SECTION.
77  X  PIC 9.
77 Y   PIC 9.
PROCEDURE DIVISION.
MULAI.
        DISPLAY 'X', 'Y'
        PERFORM TAMPILKAN
                VARYING X FROM 1 BY 1 UNTIL X = 5
                AFTER Y FROM 1 BY 1 UNTIL Y = 4
        STOP RUN.
TAMPILKAN.
        DISPLAY X, Y.
```

- Perintah OCCURS

  Perintah ini digunakan untuk mengulang data item di dalam suatu record beberapa kali membentuk suatu table.

  Bu : OCCURS integer TIMES

  { ASCENDING / DESCENDING } KEY ID namadata1, [namadata2,…]

  [INDEXED BY namaindek1, [namaindek2,…]

  - OCCURS clause tidak boleh dipakai pada level number 01 atau level number77. Digunakan pada file section, workingstorage-section.
  - Integer TIMES menunjukkan berapa kali data item akan diulang dalam suatu record.
  - ASCENDING atau DESCENDING menunjukkan bagaimana data diatur nilainya
  - KEY IS dan INDEXED BY menunjukkan nama kunci yang akan dipakai pada statement SEARCH untuk mencari data yang ada pada table.
  - OCCURS ini dipakai untuk membentuk table larik atau array

- Tabel Berdimensi Satu

  Tabel berdimensi Satu berisi nilai-nilai data berbentuk larik /array.

  Contoh :

  01 Tabel-Gaji.
  02 Gaji OCCURS 5 TIMES PIC 9(7).
  01 Nilai-data.
  02 Nilai OCCURS 10 TIMES PIC 9(3).

Value Clause tidak dapat digunakan bersama-sama dengan OCCURS clause, sehingga bila ingin menggunakan konstanta maka konstantan tersebut harus dimasukkan dahulu ke field2 tersendiri dengan nama data tersendiri. Baru setelah itu dibuat tabel dengan REDEFINES clause.
Contoh :

```
01    hari-hari.
      02    hari-1 pix x(6) value ‘senin’.
      02    hari-2 pic x(6) value ‘selasa.
01    tabel-hari redefines hari-hari
      02    hari Occurs 2 times Pic x(6).
```

Maka hari(1) = senin dan hari(2) = selasa

Tabel dimensi Dua

Tabel ini berbentuk matriks yang terdiri baris dan kolom.

Contoh :

```
01      Penjualan
        02      Salesmen Occurs 2 Times.
                03      Bulan Occurs 3 Times Pic 9(5).
```

Maka table tersebut terdiri dari 2 baris dan 3 kolom.

Variabel Array : Nama-Var(index1, index2,..)

Contoh :

Buatlah program untuk memasukan 5 data-mahasiswa yang terdiri dari nama dan npm, kemudian ditampilkan di layar dengan tampilan

…….....

NPM Nama

……