PEMROGRAMAN COBOL

LABORATORIUM SISTEM INFORMASI

2009

DAFTAR ISI

KATA PENGANTAR

COBOL atau Common Buisness Oriented Language adalah suatu bahasa komputer tingkat tinggi yang berorientasi langsung pada permasalahan bisnis.

COBOL hampir dapat digunakan pada semua komputer. Diciptakan pada tahun 1959. Pengembangan bahasa COBOL selanjutnya dilakukan oleh suatu grup yang disebut CODASYL, singkatan dari Conference on Data System Language.

Bahasa COBOL pertama diperkenalkan secara formal pada bulan Januari 1960, dan disebut COBOL-60. Karena orientasinya pada suatu masalah, bahasa ini disebut juga dengan istilah Problem Oriented Language.

COBOL dibuat untuk operasi yang mencakup langkah dasar pengolahan data, yaitu membaca data, memproses data, dan kemudian menghasilkan output berupa informasi.

COBOL merupakan bahasa terstruktur, mudah dibaca, dan mudah dipelajari. COBOL terdiri dari empat divisi yaitu IDENTIFICATION DIVISION, ENVIRONMENT DIVISION, DATA DIVISION, dan PROCEDURE DIVISION.

Ada dua bagian utama dalam bahasa COBOL, DATA DIVISION dan PROCEDURE DIVISION. DATA DIVISION memuat dan menggambarkan bentuk dan jenis dari data input dan outputnya. PROCEDURE DIVISION menggambarkan prosedur yang digunakan untuk menyelesaikan permasalahan dengan bentuk dan jenis data yang ada pada DATA DIVISION.

Dari apa yang dapat dikerjakan COBOL, konsep COBOL orientasinya pada permasalahan bisnis sebenarnya dapat untuk semua permasalahan yang berhubungan dengan pengolahan data, misalnya pengolahan data mahasiswa.

PENGENALAN DASAR PEMROGRAMAN COBOL

1

Obyektif :

1. **Mengetahui tentang bahasa COBOL**
2. **Mengerti tentang dasar-dasar bahasa COBOL**
3. **Dapat menjalankan program COBOL**

A. Pengenalan COBOL

Sekilas tentang program COBOL.

COBOL adalah kepanjangan dari Common Bussiness Oriented Language. Bahasa COBOL digolongkan sebagai High Level Language (bahasa pemrograman tingkat tinggi) yang berorientasi pada masalah bisnis. Diciptakan pada tahun 1959.

COBOL pertama kali diperkenalkan secara formal bulan Januari 1960 dan disebut dengan COBOL-60. Dan diperbaharui tahun 1965. Pada tahun 1968 dan 1974, bahasa COBOL dikembangkan dan distandarisasikan dengan nama ANSI COBOL (American National Standarts Institute).

Bahasa pemrograman COBOL sangat terstruktur, karena mudah dibaca dan memiliki struktur yang jelas.

COBOL dibuat untuk operasi pengolahan data, yaitu membaca data, memproses data dan menghasilkan output berupa informasi.

Dasar-dasar COBOL.

Untuk membuat program COBOL yang baik dan benar, minimal anda harus menguasai atau mengetahui beberapa hal berikut :

- **Cara penulisan.**

Kolom maksimal yang disediakan program COBOL adalah kolom 1 sampai 80, dengan ketentuan sebagai berikut :

Kolom 1-6 : digunakan untuk line number (optional) ditulis dari kecil ke besar.

Kolom 7 : digunakan untuk melanjutkan baris sebelumnya dengan menambahkan tanda (-) dan baris sambungannya ditulis di area B.

Bila diisi (*) maka yang ditulis pada baris ini dianggap komentar.

Kolom 8-11 : disebut dengan area A. untuk menulis divisi, section, nama paragraph, judul file description (FD), level number 01 dan level number 77.

Kolom 12-72 : disebut area B. untuk menulis elemen program selain yang ditulis di area A.

Kolom 73-80 : kolom ini tidak akan diproses oleh program jadi data diisi catatan atau curahan hati untuk dokumentasi.

Beberapa hal lagi yang anda harus perhatikan tentang penulisan yaitu penggunaan titik dan spasi. Bahasa pemrogram COBOL sangat sangat ketat tentang peraturan penulisan, pastikan anda memberikan tanda titik (.) setelah nama divisi, section, paragraph atau baris instruksi dan spasi (space) untuk pemisah antara COBOL RESERVED WORDS dengan variable atau string.

Kedua hal tadi kelihatan sepele tapi jika program yang anda buat panjang nya sampai puluhan baris hal ini mungkin dapat membuat anda frustasi. Karena akan menyebabkan pesan kesalahan yang memusingkan kepala.

Cara menjalankan program COBOL.

Program COBOL ini tidak memiliki built-in editor seperti Pascal atau Basic sehingga anda harus mengetik dan mengkompile secara terpisah. Untuk mengetik source program COBOL anda dapat menggunakan sembarang text editor apakah itu MS-Editor, Notepad, SideKick dan lain-lain sesuai keinginan anda. Tetapi disarankan menggunakan SideKick karena lebih cepat terutama pada saat proses debugging source program.

Cara penggunaan SideKick.

SideKick adalah suatu editor tool yang resident di memori, untuk menjalankan SideKick, ketik :

C:\>SK

Setelah tampil jendela berwarna biru tekan CTRL+ALT maka akan keluar menu pop-up pilih notepad.

Tekan F3 untuk membuat file baru (untuk cobol dengan extensi .COB) , F2 untuk menyimpan source program dan Ecs untuk ke DOS prompt.

Untuk mengkompile source program COBOL yang anda buat tekan F2 lalu Ecs , ketikan:

C:\>COBOL nama_program.cob atau C:\>COBOL nama_program;

Untuk menjalankan (running) ketikan:

C:\>RUNCOBOL nama_program

Contoh Program Sederhana.

```
PRAK.COB        Sun Aug 20 20:55:58 2006    Page   1
line number source line   Microsoft COBOL         Version 2.20

  1        IDENTIFICATION DIVISION.
  2        PROGRAM-ID. CONTOH.
  3        AUTHOR. SAYA.
  4        ENVIRONMENT DIVISION.
  5        CONFIGURATION SECTION.
  6        SOURCE-COMPUTER. IBM-PC.
  7        OBJECT-COMPUTER. IBM-PC.
  8        DATA DIVISION.
  9        WORKING-STORAGE SECTION.
 10         01 MASUKAN.
 11            02 NAMA PIC X(20).
 12            02 NPM PIC X(8).
 13         SCREEN SECTION.
 14         01 HAPUS-LAYAR.
 15            02 BLANK SCREEN.
 16         PROCEDURE DIVISION.
 17         MASUKKAN-DATA.
 18            DISPLAY HAPUS-LAYAR.
 19            DISPLAY 'NAMA : '.
 20            ACCEPT NAMA.
 21            DISPLAY 'NPM  : '.
 22            ACCEPT NPM.
 23         SELESAI.
 24            STOP RUN.
 25
```

STRUKTUR COBOL

2

Obyektif :

4. **Mengetahui struktur bahasa COBOL**
5. **Mengetahui pambagian divisi pada COBOL**
6. **Dapat mengetahui section yang ada tiap divisi**

- **Struktur Program Cobol.**

  Struktur utama program COBOL terdiri dari 4 divisi utama yaitu :

  IDENTIFICATION DIVISION.

  ENVIRONMENT DIVISION.

  DATA DIVISION.

  PROCEDURE DIVISION.

  Berikut ini adalah penjelasan singkat untuk setiap divisi :

  - **IDENTIFICATION DIVISION.**

    Dari nama divisi-nya kita dapat menyimpulkan kegunaan dari divisi ini yaitu sebagai identifikasi program COBOL yang kita buat misalnya nama pembuat (AUTHOR) dan nama program (PROGRAM-ID). Contoh :

    ```
    IDENTIFICATION DIVISION.
    PROGRAM-ID. CONTOH.
    AUTHOR. SAYA.
    ```

  - **ENVIRONMENT DIVISION.**

    Divisi ini berguna untuk memberikan informasi peralatan yang digunakan dalam program, dibagi menjadi 2 section yaitu CONFIGURATION SECTION dan INPUT-OUTPUT SECTION.

    CONFIGURATION SECTION bersifat optional (boleh tidak ditulis) yang berisi tentang peralatan hardware yang digunakan program.

INPUT-OUTPUT SECTION, section ini digunakan untuk operasi file (akan dijelaskan kemudian). Contoh :

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. IBM-PC.
OBJECT-COMPUTER. IBM-PC.
```

- **DATA DIVISION.**

Divisi ini berguna untuk mendeskripsikan variable-variabel dan jenis tipe data yang digunakan dalam program COBOL. Divisi ini memiliki 5 section yaitu:

1. FILE SECTION.
2. WORKING-STORAGE SECTION.
3. SCREEN SECTION.
4. LINKAGE SECTION.
5. REPORT SECTION.

Diantara 5 section yang disebutkan diatas yang paling sering digunakan adalah WORKING-STORAGE SECTION, SCREEN SECTION dan FILE SECTION.

WORKING-STORAGE SECTION digunakan untuk mendeklarasikan variable dan tipe data yang digunakan dalam program COBOL. SCREEN SECTION digunakan untuk menentukan format layar tampilan baik input atau output. Dan FILE SECTION berguna untuk operasi file (akan dijelaskan kemudian). Contoh :

```
DATA DIVISION.
WORKING-STORAGE SECTION.
 01 MASUKAN.
      02 NAMA PIC X(20).
      02 NPM PIC X(8).
SCREEN SECTION.
01 HAPUS-LAYAR.
     02 BLANK SCREEN.
```

- **PROCEDURE DIVISION.**

  Divisi ini merupakan inti dari bahasa pemrograman COBOL karena pada divisi inilah semua statement instruksi dibuat. Misalnya DISPLAY, ACCEPT dan STOP RUN. DISPLAY di gunakan untuk output, ACCEPT untuk menerima input dan STOP RUN untuk menghentikan proses program.

Contoh Program Sederhana.

```
PRAK.COB        Sun Aug 20 20:55:58 2006    Page   1
line number source line   Microsoft COBOL         Version 2.20

 1       IDENTIFICATION DIVISION.
 2       PROGRAM-ID. CONTOH.
 3       AUTHOR. SAYA.
 4       ENVIRONMENT DIVISION.
 5       CONFIGURATION SECTION.
 6       SOURCE-COMPUTER. IBM-PC.
 7       OBJECT-COMPUTER. IBM-PC.
 8       DATA DIVISION.
 9       WORKING-STORAGE SECTION.
10        01 MASUKAN.
11           02 NAMA PIC X(20).
12           02 NPM PIC X(8).
13        SCREEN SECTION.
14        01 HAPUS-LAYAR.
15           02 BLANK SCREEN.
16        PROCEDURE DIVISION.
17        MASUKKAN-DATA.
18           DISPLAY HAPUS-LAYAR.
19           DISPLAY 'NAMA : '.
20           ACCEPT NAMA.
21           DISPLAY 'NPM  : '.
22           ACCEPT NPM.
23        SELESAI.
24           STOP RUN.
25
```

Section-section pada masing-masing Division

Section pada program COBOL dimulai pada Environtment Division

1. Section pada Environment Division.

Adalah divisi kedua dari program COBOL yang berguna untuk menyediakan informasi tentang peralatan yang dipergunakan oleh program COBOL yang dbuat.. Dibagi menjadi 2 divisi yaitu :

1. CONFIGURATION SECTION
2. INPUT-OUTPUT SECTION.

Sedangkan bentuk umumnya adalah :

EVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. *nama - komputer*.
OBJECT-COMPUTER. *nama – komputer*.
SPECIAL-NAMES.
PRINTER IS
CURRENCY SIGN IS *karakter*.
DECIMAL-POINTS IS COMMA.
INPUT-OUTPUT SECTION.
FILE-CONTROL
{file control entry}

1.1. Configuration Section.

Pada seksi ini digunakan untuk menuliskan informasi tentang jenis komputer yang digunakan dalam pembuatan program.

- SOURCE-COMPUTER adalah nama komputer yang digunakan untuk mengkompilasi program COBOL yang anda buat. Ditulis di area A.
- OBJECT-COMPUTER adalah nama komputer yang digunakan untuk menjalankan atau mengeksekusi program COBOL yang anda buat. Ditulis di area A.

- SPECIAL-NAMES.

  Bersifat optional (bisa dicamtumkan atau tidak), digunakan untuk membuat nama pengganti menurut programmer. Contoh.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    PRINTER IS PENCETAK.
    CURRENCY SIGN IS Rp.
    DECIMAL-POINTS IS COMMA.
```

Keterangan :

- PRINTER IS pencetak, artinya adalah memberikan nama pengganti PRINTER menjadi “pencetak”. Jadi jika anda ingin menggunakan printer maka anda dapat menulis di PROCEDURE DIVISION :

```
DISPLAY ‘INI AKAN DICETAK !’ UPON PENCETAK.
```

- CURRENCY SIGN IS Rp, artinya adalah mengganti nilai default mata uang ($) menjadi “Rp”, atau symbol yang lain sesuai keinginan anda.
- DECIMAL POINT IS COMMA, adalah untuk menentukan karakter (,) sebagai pemisah pada bilangan desimal. Nilai default-nya adalah titik (.).

1.2. Input-output Section.

Seksi ini digunakan bila anda akan menggunakan file (akan dibahas lebih lanjut dalam operasi file). Bentuk umum adalah :

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
{ file-control entry }.
```

Contoh Program 1.

```
TEST.COB                    Mon Aug 21 18:46:47 2006    Page   1
line number source line   Microsoft COBOL              Version 2.20

 1        IDENTIFICATION DIVISION.
 2        PROGRAM-ID. COBA.
 3        AUTHOR. SAYA.
 4        ENVIRONMENT DIVISION.
 5        CONFIGURATION SECTION.
 6        SOURCE-COMPUTER. IBM-PC.
 7        OBJECT-COMPUTER. IBM-PC.
 8        SPECIAL-NAMES.
 9           DECIMAL-POINT IS COMMA.
10        DATA DIVISION.
11        WORKING-STORAGE SECTION.
12        01  HASIL.
13            02 NILAI PIC 9999,99.
14        SCREEN SECTION.
15        01  HAPUS-LAYAR.
16            02 BLANK SCREEN.
17        PROCEDURE DIVISION.
18        MULAI.
19            MOVE 4612,34 TO NILAI.
20            DISPLAY 'ANGKA = ', NILAI.
21        SELESAI.
22            STOP RUN.
23
24
```

2. Section pada Data Division.

Adalah divisi yang ketiga dari empat divisi di dalam bahasa pemrograman COBOL. Divisi ini berguna untuk mendeklarasikan data input maupun output, tentang bagaimana jenis data yang digunakan dan sifat dari data tersebut. Dibagi menjadi 5 divisi yaitu :

1. FILE SECTION.
2. WORKING-STORAGE SECTION.
3. SCREEN SECTION.
4. LINKAGE SECTION.
5. REPORT SECTION.

Sedangkan bentuk umumnya (singkat) adalah :

DATA DIVISION.

FILE SECTION.

FD nama-file

WORKING-STORAGE SECTION.

{Level-number deskripsi data.}

{deskripsi record}.

SCREEN SECTION.

{level-number nama-layar}

{BLANK SCREEN}.

DLL.

2.1. FILE SECTION.

Section ini digunakan untuk menjelaskan semua keterangan mengenai file-file yang dipakai didalam program (akan dijelaskan di operasi file).

2.2. WORKING-STORAGE SECTION.

Section ini adalah section yang paling penting dari semua section pada DATA DIVISION karena pada section ini kita mendeklarasikan semua variable dan jenis tipe data yang nantinya akan kita gunakan pada PROCEDURE DIVISION. Untuk sekedar review, anda dapat menulis level number dan picture karakter pada section ini.

Terdapat 2 jenis data yaitu data item individu ditulis dengan level number 77 sedangkan group data item dinyatakan dengan level number 01 untuk nama record dan 02 sampai 49 untuk deskripsi record.

2.3. SCREEN SECTION.

Section ini berguna untuk mengatur bentuk format dari layar input maupun output. Pada section ini juga anda dapat menuliskan level number, tetapi hanya level number 01 sampai 49 saja yang dapat digunakan, sedangkan level number 77 hanya dapat ditulis di WORKING-STORAGE SECTION. SCREEN SECTION berhubungan dengan statement DISPLAY dan ACCEPT pada PROCEDURE DIVISION. Berikut ini beberapa perintah display formatting pada SCREEN SECTION.

LINE clause untuk menempatkan kursor pada baris yang ditentukan.

COLUMN clause untuk menempatkan kursor pada kolom yang ditentukan.

Contoh :

LINE 1 COLUMN 5 VALUE ‘ NAMA ANDA : ‘.

COLUMN PLUS 1 PIC 9(6) TO HARGA.

UNDERLINE clause, untuk memberi garis bawah.

REVERSE-VIDEO clause, membalik warna layar.

HIGLIGHT clause, untuk menampilkan tampilan yang lebih terang.

BLINK clause, digunakan untuk membuat blinking layar.

Contoh Program :

```
M2A.COB                  Mon Aug 21 20:04:00 2006    Page   1
 line number source line   Microsoft COBOL            Version 2.20

  1        IDENTIFICATION DIVISION.
  2        PROGRAM-ID. LAYAR.
  3        AUTHOR. SAYA.
  4        ENVIRONMENT DIVISION.
  5        DATA DIVISION.
  6        SCREEN SECTION.
  7        01 HAPUS-LAYAR.
  8           02 BLANK SCREEN.
  9        01 LAYAR.
 10           02 LINE 2 COLUMN 25 'UNIVERSITAS GUNADARMA' UNDERLINE.
 11           02 LINE 5 COLUMN 25 'LAB SISTEM INFORMASI' HIGHLIGHT.
 12           02 LINE 7 COLUMN 28 'PRAKTIKUM COBOL' REVERSE-VIDEO.
 13           02 LINE 9 COLUMN 28 'SELAMAT DATANG' BLINK.
 14        PROCEDURE DIVISION.
 15        MULAI.
 16           DISPLAY HAPUS-LAYAR.
 17           DISPLAY LAYAR.
 18        SELESAI.
 19           STOP RUN.
```

FROM, TO dan USING

3

Obyektif :

7. **Mengetahui penggunaan perintah FROM, USING, TO clause**
8. **Mengetahui penggunaan level number**
9. **Mengetahui penggunaan Picture Editing**

---

1. Penggunaan Perintah FROM, TO dan USING clause.

Perintah FROM, TO, USING clause digunakan untuk menampilkan atau menampilkan data yang bentuk data-itemnya berhubungan dengan data (nilai dari PIC clause) pada WORKING-STORAGE SECTION.

Perintah FORM digunakan untuk menampilkan / menerima data dari suatu variable.

Perintah TO digunakan untuk memasukan nilai / menyimpan nilai kedalam variable.

Perintah USING adalah pengganti perintah FORM dan TO artinya perintah ini dapat melakukan kedua hal diatas.

Contoh Program 1.

```
M2.COB                          Mon Aug 21 20:24:20 2006
Page   1
 line number source line   Microsoft COBOL                Version
2.20

   1         IDENTIFICATION DIVISION.
   2         PROGRAM-ID. HARGA.
   3         AUTHOR. SAYA.
   4         ENVIRONMENT DIVISION.
   5         DATA DIVISION.
   6         WORKING-STORAGE SECTION.
   7         01  DATA-MASUK.
   8             02 NAMABRG PIC A(20) .
   9             02 HARGA   PIC 9(6)  VALUE 0.
  10             02 JUMLAH  PIC 9(3)  VALUE 0.
  11             02 TOTAL   PIC 9(6)  VALUE 0.
  12         SCREEN SECTION.
  13         01  HAPUS-LAYAR.
```

```
14              02 BLANK SCREEN.
15          01  LAYAR-MASUK.
16              02 LINE 5 COLUMN 3 VALUE 'NAMA BARANG : '.
17              02 COLUMN PLUS 1 PIC A(20) TO NAMABRG.
18              02 LINE 7 COLUMN 3 VALUE 'HARGA : '.
19              02 COLUMN PLUS 1 PIC 9(6) TO HARGA.
20              02 LINE 9 COLUMN 3 VALUE 'JUMLAH : '.
21              02 COLUMN PLUS 1 PIC 9(3) TO JUMLAH.
22          01 LAYAR-TAMPIL.
23              02 LINE 13 COLUMN 3 VALUE 'TOTAL : '.
24              02 COLUMN PLUS 1 PIC 9(6) FROM TOTAL.
25          PROCEDURE DIVISION.
26          MULAI.
27              DISPLAY HAPUS-LAYAR.
28              DISPLAY LAYAR-MASUK.
29              ACCEPT LAYAR-MASUK.
30              COMPUTE TOTAL = JUMLAH * HARGA.
31              DISPLAY LAYAR-TAMPIL.
32          SELESAI.
33              STOP RUN.
34
```

2. Fungsi dan Penggunaan Level Number.

Level number suatu nilai integer yang menunjukan jenjang dari data item dalam suatu record, Makin besar nilai integer, makin rendah tingkatannya. Berikut ini level number pada bahasa pemrograman COBOL :

a. Level number 01 digunakan sebagai awal dari record (nama record).
b. Level number 02 sampai dengan 49 digunakan untuk mengisi keterangan dari record, anda memilih bebas memilih salah satu angka antara 02 sampai 49, tetapi sebaiknya memilihnya secara berurut agar program lebih mudah dibaca.
c. Level number 66 digunakan untuk untuk RENAMES
d. Level number 77 digunakan untuk menyatakan variable independent (berdiri sendiri), hanya ada WORKING-STORAGE SECTION.
e. Level number 88 digunakan untuk variable pilihan atau kondisi.

3. Picture Clause Dan Editing.

3.1 Picture Clause

BU: variable PIC / PICTURE karater

Berguna untuk menentukan jenis tipe data untuk variable saja, tetapi dapat juga digunakan untuk menampilkan bentuk data output. Berikut ini Picture Clause yang digunakan dalam bahasa COBOL :

- **Picture karakter 9**

  Digunakan untuk menyimpan data dalam bentuk numeric, sedangkan untuk menentukan jumlah data yang disimpan ditentukan oleh banyaknya 9 yang anda ketik setelah perintah PIC , perhatikan contoh !

  HARGA PIC 9999. → berarti variable harga memiliki 4 digit (posisi) angka numeric, tetapi anda dapat juga menyingkatnya dengan menggunakan tanda kurung “( )”, sehingga menjadi :

  HARGA PIC 9(4).

- **Picture karakter A.**

  Digunakan untuk menyimpan data dalam bentuk alphabetic dan bersifat rata kiri.. Ketentuan jumlah karakter untuk Picture karakter A sama dengan ketentuan jumlah karakter Picture karakter 9.

- **Picture karakter X.**

  Digunakan untuk menyimpan data dalam bentuk alphanumeric atau campuran huruf, angka dan special karakter. Bersifat rata kiri.

- **Picture karakter V.**

  Digunakan untuk jumlah digit dibelakang koma untuk bilangan decimal, jadi picture karakter jenis ini hanya bisa digunakan bersama picture karakter 9.

  Contoh :

  02 HARGA PIC 999V99 → 234.50 →5 digit dan 2 digit belakang koma.

  02 HARGA PIC 9(4)V9(3) →4654.125 → 7 digit dan 3 digit belakang koma.

- **Picture Karakter P.**

  Digunakan bersama dengan picture karakter V, digunakan untuk menimbulkan angka 0, picture karakter ini jarang digunakan.

  Contoh :

02 HARGA PIC 9PPV → 500.

- **Picture Karakter S.**

  Digunakan untuk memberi tanda minus (-), karena bila digunakan picture karakter 9 nilai default yang ditampilkan hanya nilai positif walaupun hasil dari perhitungan adalah negatif. Jadi dengan picture karakter ini anda dapat menampilkan nilai negatif. Perlu diingat bahwa penggunaan pic karakter s tidak menambah jumlah digit.

  Contoh :

  02 SALDO PIC S99 →-25 → tetap 2 digit.

3.2. Picture Editing.

Sebelum membahas picture editing, ada baiknya kita tahu perbedaan picture clause dan picture editing. Pebedaan picture clause dan picture editing adalah :

Pada picture clause digunakan untuk memberikan informasi pada compiler COBOL mengatur memori untuk data yang disimpan dan akan digunakan kemudian, sedangkan picture editing digunakan untuk melakukan perubahan bentuk data yang akan dicetak untuk output, agar lebih mudah dibaca.

Catatan :

Picture editing tidak dapat digunakan langsung sebagai variable penerima, seperti halnya picture clause.

Berikut ini picture editing pada bahasa pemrograman COBOL.

- **Picture editing karakter Z.**

  Digunakan untuk menghilangkan nilai 0 didepan bilangan, digunakan bersama picture clause 9.

  Contoh :

  12345 → PIC ZZZZZ → 12345.

  0123 → PIC ZZZZZ → 123.

  00123 → PIC Z9999 → 0123.

- **Picture editing karakter $.**

Untuk memberikan tanda $ pada output, letaknya diujung kiri.

Contoh :

12345 → PIC $ 99999 → $12345.

- **Picture editing karakter (.) dan (,).**

  Karakter (.) digunakan untuk menunjukan letak posisi koma pada bilangan desimal. Karakter (,) digunakan untuk memberi bentuk koma, biasanya digunakan untuk menyatakan nilai uang.

  Contoh :

  123.45 → PIC 999.99 → 123.45

❖Picture editing karakter “-“

Jika anda menggunakan picture editing jenis ini maka bilangan akan dicetak minus, bila bernilai negatif , tetapi bila bernilai positif akan diganti dengan blank.

Contoh :

12345 → PIC –9(5) → 12345

❖Picture editing karakter “+”

Digunakan untuk memberikan tanda plus pada permulaan atau pada bagian akhir dari suatu bilangan. Jika nilai yang diberikan bernilai negatif (-) maka akan ditampilkan minus, tetapi jika nilai yang diterima adalah positif maka akan ditampilkan tanda positif (+).

- **Picture editing karakter DB dan CR.**

  Untuk memberikan akhiran DB atau CR pada suatu bilangan, biasanya digunakan untuk aplikasi akuntansi.

  Contoh :

  12345 → PIC 9(5)CR → 12345CR

- **Picture editing karakter B.**

  Digunakan untuk menyisipkan blank pada posisi karakter B itu ada, untuk mengedit nilai data bukan numeric.

  Contoh:

123456 → PIC 99B9B999 → 12 3 456

❖Picture editing karakter “*”

Digunakan untuk mengganti nilai 0 pada suatu bilangan dengan tanda *

Contoh :

123→ PIC **999 →**123

- **Picture editing karakter 0.**

  Digunakan untuk menyisipkan karakter 0 dimana karakter 0 itu berada.

  Contoh :

  12345 → PIC 9(5)0 → 123450

- **Picture editing karakter “/”.**

  Digunakan untuk menyisipkan tanda “/” ada posisi karakter “/” berada.

  Contoh :180260 → PIC 99/99/99 → 18/02/60

MOVE, GO TO, PERFORM

4

Obyektif :

10. **Mengetahui fungsi MOVE**
11. **Mengetahui fungsi GO TO**
12. **Mengetahui fungsi PERFORM**

---

1. Pernyataan MOVE.

- Memindahkan data ke suatu item penerima, sehingga input data dapat dimanipulasi untuk menghasilkan output.
- Memungkinkan data yang belum diedit yang berguna dalam penyajian informasi.
- Bentuk umum :

MOVE { Identifier-1 Literal } TO Identifier-2 (Identifier-3)…

Contoh program :

```
*----------------------------------*
* Contoh Pemakaian MOVE LITERAL    *
*----------------------------------*
 IDENTIFICATION DIVISION.
 PROGRAM-ID. LATIHAN.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 TANGGAL-SISTEM.
 02. TGL PIC 99.
 02. BLN PIC 99.
 02. THN PIC 99.
 TANGGAL-KINI.
 02. TGL PIC 99.
 02. FILLER PIC X VALUE '-'.
 02. BLN PIC 99.
 02. FILLER PIC X VALUE VALUE '-'.
 02. THN PIC 99.
 PROCEDURE DIVISION.
 PROGRAM-UTAMA.
 ACCEPT TANGGAL-SISTEM FROM DATE.
 MOVE CORR TANGGAL-SISTEM TO TANGGAL-KINI.
 DISPLAY TANGGAL-SISTEM.
 DISPLAY TANGGAL-KINI.
 SELESAI.
```

```
               STOP “TEKAN ENTER UNTUK MENGAKHIRI PROGRAM”.
               STOP RUN.
```

Contoh Program :

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PRAK-1.
AUTHOR. OMAR.
ENVIRONMEMT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01.  NILAI-MASUKKAN.
     02.  NAMA PIC X(20).
     02.  NPM PIC 9(8).
     02.  KELAS PIC X(7).
     02.  MID PIC 9(2).
     02.  FINAL PIC 9(2)
     02.  RATA PIC 9(9V99).
01.  NILAI-HASIL.
     02.  RATA-RATA PIC Z(3).
01.  COBA-LAGI PIC X.
     88.  YA VALUE ‘Y’,’y’.
     88.  TIDAK VALUE ‘T’,’t’.
SCREEN SECTION.
01.  CLS.
   02.  BLANK SCREEN.
01.  TAMPILAN-INPUT.
   02. LINE 5 COLUMN 5 VALUE ‘MENGHITUNG NILAI MAHASISWA’.
   02. LINE 7 COLUMN 3 VALUE ‘NAMA MAHASISWA : ‘.
   02. COLUMN PLUS 1 PIC X(20) TO NAMA.
   02. LINE 8 COLUMN 3 VALUE ‘NPM : ‘.
   02. COLUMN PLUS 1 PIC 9(8) TO NPM.
   02. LINE 9 COLUMN 3 VALUE ‘KELAS: ‘.
   02. COLUMN PLUS 1 PIC X(7) TO KELAS.
   02. LINE 10 COLUMN 3 VALUE ‘NILAI MIDTEST : ‘.
   02. COLUMN PLUS 1 PIC 9(2) TO MID.
   02. LINE 11 COLUMN 3 VALUE ‘NILAI FINAL : ‘.
   02. COLUMN PLUS 1 PIC 9(2) TO FINAL.
01.  SELEKSI.
   02. LINE 15 COLUMN 3 VALUE ‘INGIN MENCOBA LAGI [Y/T]? ‘.
   02. COLUMN PLUS 1 PIC X TO COBA-LAGI.
PROCEDURE DIVISION.
MULAI.
   DISPLAY CLS.
   DISPLAY TAMPILAN-INPUT.
   ACCEPT TAMPILAN-INPUT.
   COMPUTE RATA = (MID + FINAL) / 2.
   MOVE RATA TO RATA-RATA.
   DISPLAY (13, 3) ‘NILAI RATA-RATA : ‘
   DISPLAY SELEKSI.
   ACCEPT SELEKSI.
   IF YA GO TO MULAI.
SELESAI.
   STOP RUN.
```

```
OUTPUTNYA :
                  MENGHITUNG NILAI MAHASISWA
         NAMA MAHASISWA  : Omar Pahlevi
         NPM             : 17105219
         KELAS           : 4 KA 49
         NILAI MIDTEST   : 70
         NILAI FINAL     : 70
         NILAI RATA-RATA : 70.00
         INGIN MENCOBA LAGI [Y/T] :
```

2. Prenyataan GO TO

GO TO membentuk loncatan tanpa syarat, yaitu program proses meloncat langsung tanpa syarat ke nama paragraph yang di tunjuk.

Bentuk Umum nya :

GO TO nama-paragraph

Statement GO TO harus selalu diikuti oleh nama-paragraph.

Contoh :

```
        MOVE A TO B
        GO TO ALINEA-5.
  ALINEA-4.
        ADD X TO Y.
  ALINEA-5.
        MULTIPLY X BY Y.
```

2.1. GO TO …….. DEPENDING verb

GO TO …. DEPENDING verb akan membentuk statement yang akan membawa proses meloncat (GO TO) ke suatu nama-paragraph tergantung dari (DEPENDING) nilai nama-data yang ditunjukkan dalam statement ini.

Bentuk Umumnya :

GO TO nama-paragraph-1, nama-paragraph-2,… nama-paragraph-n

DEPENDING ON nama-data.

Contoh :

GO TO TEMPAT-1, TEMPAT-2, HABIS DEPENDING ON KODE.

Sebenarnya statement ini menyederhanakan bentuk statement IF untuk suatu kondisi :

```
IF KODE = 1 GO TO TEMPAT-1.
IF KODE = 2 GO TO TEMPAT-2.
IF KODE KODE = 3 GO TO HABIS.
```

3. Pernyataan PERFORM.

3.1. Pernyataan PERFORM digunakan untuk :

1. Proses Pengulangan.

   Yaitu jika terdapat suatu proses atau beberapa proses yang dijalankan beberapa kali.

2. Pemrograman Terstruktur.
   - Suatu cara mengorganisir program untuk memudahkan pengembangan, pemahaman, dan pemodifikasian program.
   - Pendekatan pembuatan program menggunakan konsep TOP-DOWN

3.2. Perbedaan proses antar pernyataan PERFORM dan GO TO.

Setelah selesai mengerjakan seluruh isi paragraph yang diinginkan, maka :

Pada PERFORM => akan kembali lagi ke statement setelah statement PERFORM tersebut.

Pada GO TO => tidak akan kembali ke statement setelah GO TO, tetapi proses dilanjutkan ke paragraph berikutnya dari paragraph yang dituju.

3.3. Keuntungan Penggunaan PERFORM :

Suatu prosedur dapat dieksekusi pada berbagai tempat yang berbeda di dalam program, sehingga menghemat kode program.

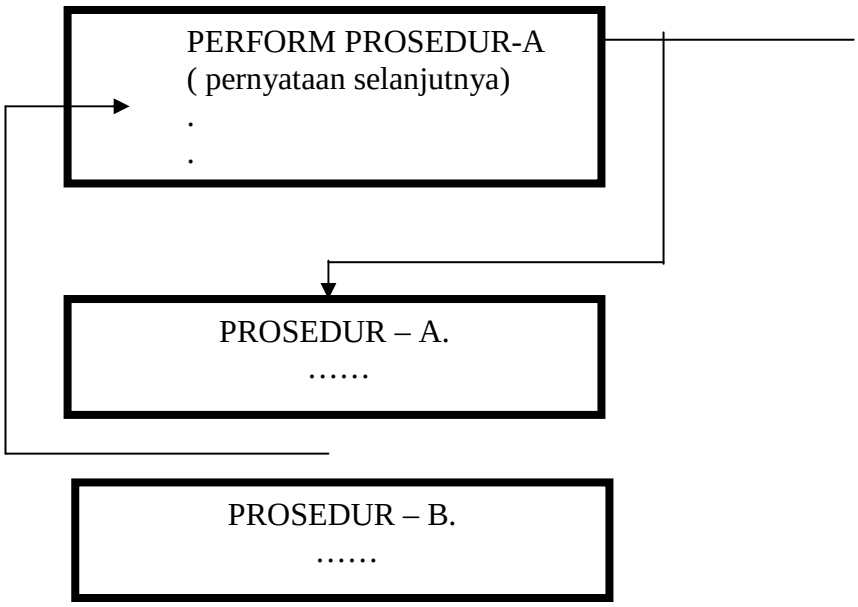
3.4. Bentuk dasar PERFORM :

PERFORM NAMA-PROSEDUR-1 { THROUGH THRU } NAMA PRO 2

Dapat di bagi menjadi 2, yaitu :

a. PERFORM NAMA-PROSEDUR-1.

Ilustrasi:



Contoh Program :

```
*---CONTOH PEMAKAIAN PERFORM---*
 IDENTIFICATION DIVISION.
 PROGRAM-ID. PERFORM.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 PROCEDURE DIVISION.
 MULAI.
      PERFORM PROCEDURE-A.
      DISPLAY ‘ SELAMAT BELAJAR ‘.
      PERFORM PROCEDURE-B.
      PERFORM PROCEDURE-A.
      STOP RUN.
 PROCEDURE-A.
      DISPLAY ‘**********************’.
      DISPLAY ‘^^^^^^^^^^^^^^^^^^^^^^’.
 PROCEDURE-B.
      DISPLAY ‘ DAN SEMOGA SUKSES’.

Hasilnya :
      **********************************
      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
             SELAMAT BELAJAR
            DAN SEMOGA SUKSES
       ********************************
       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

b. PERFORM NAMA PROCEDURE-1 THROUGH NAMA-PROCEDURE-2. atau

PERFORM NAMA-PROCEDURE-1 THROUGH NAMA-PROCEDURE-2.

Contoh Program :

```
*----CONTOH PEMAKAIAN PERFORM THRU----*
IDENTIFICATION DIVISION.
PROGRAM-ID. PERFORM2.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
MULAI.
  PERFORM GARIS.
  PERFORM PROCEDURE-A THRU PROCCEDURE-B.
  DISPLAY ‘ SEMOGA KITA JADI ORANG SUKSES ‘.
  PERFORM GARIS.
  STOP RUN.
PROCEDURE-A.
  DISPLAY ‘ AYO BELAJAR ‘.
PROCEDURE-B.
  DISPLAY ‘ JANGAN MALAS ‘.
GARIS.
  DISPLAY’~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~’.
```

Hasilnya :

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
             AYO BELAJAR
             JANGAN MALAS
       SEMOGA KITA JADI ORANG SUKSES
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

TABEL DIMENSI SATU,
TABEL MULTI DIMENSI

5

Obyektif :

13. **Mengetahui cara pembuatan Tabel satu dimensi**
14. **Mengetahui cara pembuatan Tabel dua dimensi**
15. **Dapat membuat program tabel sederhana**

---

1. Pendefinisian Tabel (Tabel 1 dimensi)

Tabel adalah suatu kumpulan data dg tipe sama yg diakses dg menggunakan nama yg sama, serta disimpan di memori secara berurutan (disebut juga array).

Pendefinisian Tabel ada pada DATA DIVISION di WORKING-STORAGE SECTION.

Contoh pendefinisian tabel :

```
01    tabel-nilai-siswa.
     02  nilai-siswa      PIC 999 OCCURS 5 TIMES.
```

Menyatakan array nilai-siswa yang menampung 10 data numerik.

Contoh tabel di atas identik dengan :

```
01    data-nilai-siswa.
     02  nilai-siswa-1    PIC 999.
     02  nilai-siswa-2    PIC 999.
     02  nilai-siswa-3    PIC 999.
     02  nilai-siswa-4    PIC 999.
     02  nilai-siswa-5    PIC 999.
```

2. Pengisian & Pengaksesan isi Tabel

Untuk menunjuk ke masing-masing elemen tabel diperlukan sebuah subscript yg dapat berupa literal numerik (bilangan bulat positif) ataupun suatu identifier yg berisi bilangan bulat positif.

Pada contoh tabel-nilai-siswa, nilai subscript adalah antara 1 s/d 5. Subscript 1 menunjukkan elemen tabel yg pertama, subscript 2 menunjukkan elemen tabel yg kedua, dst.

Untuk menunjuk atau mengakses elemen ke n pada suatu tabel dengan menyebutkan nama-tabel diikuti dengan nilai subscriptnya diapit tanda kurung.

Contoh untuk pengaksesan elemen ke 5 pd tabel-nilai-siswa :

```
nilai-siswa (5).
MOVE 75 TO nilai-siswa (5).
DISPLAY nilai-siswa (5).
```

Berikut ini adalah potongan program untuk pemasukan data tabel :

```
PERFORM pemasukan-nilai-siswa
        VARYING i  FROM 1 BY  1   UNTIL   i >  10.
---------------------------------------------------------------
pemasukan-nilai-siswa.
        DISPLAY (  ,   )  'Nilai  :  '.
        ACCEPT nilai-siswa ( i ).
```

Contoh program :

```
IDENTIFICATION DIVISION.
PROGRAM-ID. tabel1.
*Contoh penggunaan TABEL untuk pemasukan dan
*menampilkan sejumlah nilai siswa

ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77     n    PIC 99.
01   tabel-nilai-siswa.
     02 nilai-siswa           PIC 999 OCCURS 10 TIMES.
SCREEN SECTION.
```

```
01   hapus-layar.
     02 BLANK SCREEN.

PROCEDURE DIVISION.
program-utama.
     PERFORM pemasukan-nilai-siswa
          VARYING  n  FROM 1 BY 1 UNTIL   n  >  10.
     DISPLAY hapus-layar.
     PERFORM tampilan-nilai-siswa
          VARYING  n  FROM 1 BY 1 UNTIL  n  >   10.
     STOP RUN.

pemasukan-nilai-siswa.
     DISPLAY (  ,  )  'Nilai :  '.
     ACCEPT  (  ,  )  nilai-siswa (n).
     DISPLAY SPACE.

tampilan-nilai-siswa.
     DISPLAY (  ,  ) 'Nilai ke ', n , ' : '  nilai-siswa ( n
).
     DISPLAY SPACE.
```

3. Tabel 1 Dimensi dengan 2 buah kolom

Untuk pembuatan tabel yg tiap elemennya mengandung 2 buah data, seperti pd tabel berikut ini :

NAMA SALESMAN	JUMLAH PENJUALAN
Windy Arwindya	500.000
Afif Susanto	2.100.500
Miko Ariko	1.750.000
Lely Nurlela	5.700.000
Chika	760.000

Pendefinisian Tabel dg tiap elemen mengandung 2 data seperti di atas adl sbg berikut :

```
01   tabel-penjualan.
    02 data-penjualan-salesman  OCCURS  5  TIMES.
       03 nama-salesman             PIC  X(25).
       03 hasil-penjualan           PIC  9(6).
```

Untuk mengisikan kelima elemen tabel tersebut adalah sbg berikut :

```
PERFORM  pemasukan-data-tabel
    VARYING  i  FROM  1  BY  1  UNTIL  i  >  5.
-------------------------------------------------------------
-------
pemasukan-data-tabel.
      DISPLAY (  ,  )    'Nama salesman  :  '.
      ACCEPT   nama-salesman ( i ).
      DISPLAY SPACE.
```

```
       DISPLAY (  ,  )     'Hasil penjualan  :  '.
       ACCEPT  hasil-penjualan ( i ).
```

* Contoh Program :

```
IDENTIFICATION DIVISION.
PROGRAM-ID.       tabel.
*Contoh penggunaan tabel 1 dimensi yg berisi nama salesman
*     dan hasil penjualan
ENVIRONMENT DIVISION.
CONFIGURATIO SECTION.
SPECIAL-NAMES.
      DECIMAL-POINT IS COMMA.

DATA DIVISION.
WORKING-STORAGE SECTION.
77  n             PIC 9.
77  garis         PIC X(47) VALUE ALL '-'.

01  tabel-penjualan.
    02  data-penjualan-salesman  OCCURS  5  TIMES.
        03  nama-salesman     PIC X(25).
        03  hasil-penjualan   PIC 9(8).
01  judul-1       PIC X(20)  VALUE  '  Tabel Penjualan   '.
01  judul-2.
    02  FILLER    PIC X(2) VALUE '| '.
    02  FILL      PIC X(26) VALUE '    NAMA SALESMAN    '.
    02  FILL      PIC  X(2) VALIE '| '.
    02  FILL      PIC X(16) VALUE ' HASIL PENJUALAN '.
    02  FILL      PIC X(1) VALUE '|'.
01  isi-tabel.
    02  FILL      PIC X(2) VALUE '| '.
    02  lap-nama-salesman     PIC x(26).
    02  FILL      PIC X(2) VALUE '| '.
    02  lap-hasil-penjualan         PIC ZZ.ZZZ.ZZZ.ZZZBB.
    02  FILL      PIC X(1) VALUE '|'.

SCREEN SECTION.
01  hapus-layar.
    02  BLANK SCREEN.
PROCEDURE DIVISION.
program-utama.
    PERFORM pemasukan-data-tabel
        VARYING  n FROM 1 BY 1 UNTIL  n  >  5
    PERFORM tampilan-tabel-penjualan
        VARYING n FROM 1 BY 1 UNTIL  n  >  5.
    STOP RUN.

pemasukan-data-tabel.
    DISPLAY (  ,  )  n,  '. Nama salesman  :  '.
    ACCEPT (  ,  )  nama-salesman (n).
    DISPLAY SPACE.
    DISPLAY (  ,  )  SPACE,  ' Hasil penjualan  :  '.
    ACCEPT (  ,  ) hasil-penjualan (n).
    DISPLAY SPACE.
tampilan-tabel-penjualan.
    DISPLAY hapus-layar.
    DISPLAY judul-1.
```

```
    DISPLAY garis.
    DISPLAY judul-2.
    DISPLAY garis.
    PERFORM  tampilkan-isi-tabel
        VARYING  n  FROM  1  BY  1  UNTIL  n  >  5.
    DISPLAY garis.
tampilkan-isi-tabel.
    MOVE nama-salesman (n)  TO  lap-nama-salesman.
    MOVE hasil-penjualan (n)  TO  lap-hasil-penjualan.
    DISPLAY isi-tabel.
```

4. Tabel 2 dimensi

Pada tabel dimensi 2 mempunyai 2 buah subscript, berbeda dg tabel dimensi satu yg untuk pengaksesannya hanya diperlukan 1 buah subscript, sebagai contoh dapat dilihat pd tabel berikut :

Cabang	Jumlah kendaraan terjual		
	Bulan 1	Bulan 2	Bulan 3
Medan	21	15	8
Semarang	32	28	27
Solo	12	11	13
Yogya	4	6	10

Untuk tabel di atas, pendefinisiannya adl sbg berikut :

```
01  tabel-penjualan-mobil.
    02  cabang     OCCURS 4 TIMES.
        03   bulan       OCCURS 3 TIMES.
            04  jumlah-unit-terjual       PIC 99.
```

Definisi tabel di atas :

02 cabang OCCURS 4 TIMES. ⇒ Menyatakan bahwa ada 4 buah cabang

03 bulan OCCURS 3 TIMES. ⇒ Menyatakan bahwa setiap cabang ada data dalam 3 bulan

jumlah-unit-terjual PIC 99. ⇒ Menyatakan elemen dari tabel bulan berupa jumlah-unit-terjual , yg bertipe numerik (2 digit).

Untuk mengakses elemen jumlah-unit-terjual , diperlukan pengenal/penunjuk berupa :

jumlah-unit-terjual (subscript-cabang, subscript-bulan)

contoh : Untuk mengakses data penjualan cabang Medan pada bulan kedua, bentuknya adalah :

jumlah-unit-terjual (1, 2)

Contoh Program :

```
IDENTIFICATION DIVISION.
PROGRAM-ID.       tabel2.
*Contoh pemakaian tabel dua dimensi
ENVIRONMENT DIVISION.

DATA DIVISION.
WORKING-STORAGE SECTION.
01   ws-subscript.
     02 subscript-cabang      PIC 9.
     02 subscript-bulan       PIC 9.
01   tabel-penjualan-mobil.
     02 cabang       OCCURS 4 TIMES.
        03 bulan     OCCURS 3 TIMES.
           04 jumlah-unit-terjual   PIC 99.
77   total-penjualan-cabang   PIC 99 VALUE 0.

SCREEN SECTION.
01   hapus-layar.
     02    BLANK SCREEN.

PROCEDURE DIVISION.
program-utama.
     PERFORM entry-tabel.
     PERFORM tampilkan-tabel.
     STOP RUN.

entry-tabel.
     DISPLAY ‘ Pemasukan data ke dalam tabel dimensi dua :
‘.
     PERFORM isi-data-tabel
        VARYING subscript-cabang FROM 1 BY 1
           UNTIL subscript-cabang  >  4
                  AFTER subscript-bulan FROM 1 BY 1
                  UNTIL subscript-bulan  >  3.

isi-data-tabel.
    DISPLAY (  ,  )  ‘Cabang  ‘,  subscript-cabang,
                    ‘Bulan  ‘,  subscript-bulan, ‘  :  ‘.
    ACCEPT (  ,  )  jumlah-unit-terjual (subscript-cabang,
subscript-bulan).
    DISPLAY SPACE.

tampilkan-tabel.
    DISPLAY hapus-layar.
    DISPLAY ‘ Menampilkan isi tabel berdimensi dua :  ‘.

       PERFORM display-data-tabel
        VARYING  subscript-cabang FROM 1 BY 1
              UNTIL  subscript-cabang  >  4
                   AFTER  subscript-bulan  FROM  1 BY 1
                    UNTIL  subscript-bulan  >  3.

display-data-tabel.
    DISPLAY (  ,  )  ‘Cabang  ‘,  subscript-cabang,
                    ‘Bulan  ‘,  subscript-bulan,  ‘  :  ‘,
                    jumlah-unit-terjual (subscript-cabang,
subscript-bulan).
    DISPLAY SPACE.
```

```
    ADD  jumlah-unit-terjual (subscript-cabang, subscript-
bulan)
         TO  total-penjualan-cabang.

    IF  (subscript-bulan  =  3)
        DISPLAY  '*** Total penjualan per cabang  =  ',  total-
penjualan-cabang.
    MOVE  0  TO  total-penjualan-cabang.
```

SINTAKS-SINTAKS
PADA FILE SEQUENSIAL

6

Obyektif :

16. **Mengetahui sintaks-siantaks pada file sequensial**
17. **Mengerti sintaks-siantaks yang ada**
18. **Dapat membuat program file sequensial sederhana**

Organisasi file secara sequensial (urut) memungkinkan pengaksesan record file secara berurutan. Urutan data record yang direkam ke file sama dengan urutan sewaktu data tersebut direkamkan. Urutan data ini tidak berubah walaupun data baru ditambahkan. Data baru yang direkamkan akan menempati urutan record selanjutnya setelah record terakhir yang telah ada.

1. Sintaks ENVIRONMENT DIVISION pada file sequential.

Pada ENVIRONMENT DIVISION, untuk CONFIGURATION SECTION sama seperti yang lainnya, yang berbeda adalah pada INPUT-OUTPUT SECTION pada FILE CONTROL, karena informasi mengenai file sequensial disebutkan disini.

```
INPUT-OUTPUT SECTION.
FILE-CONTROL
SELECT nama-file ASSIGN TO { DISK    }
                           { PRINTER }
[ORGANIZATION IS [LINE] SEQUENTIAL]
ACCESS MODE IS SEQUENTIAL.
FILE STATUS IS nama-data.
```

SELECT clause harus ditulis pertama.

ASSIGN clause menunjukkan bentuk dari file yg akan dipergunakan , disk atau print file.

ORGANIZATIONAL IS LINE SEQUENTIAL atau ORGANIZATIONAL IS SEQUENTIAL, boleh pilih salah satu.

FILE STATUS clause dipergunakan untuk menunjukkan jenis kesalahan yang terjadi dari suatu hasil proses operasi file.

2. Sintaks DATA DIVISION pada file sequensial

DATA DIVISION pada file sequensial mengandung file description entry sbb:

```
LABEL { RECORD IS    } { STANDARD }
      { RECORDS ARE  } { OMMITED  }

[VALUE OF FILE-ID IS nama-file di label
```

LABEL RECORD IS OMMITED atau LABEL RECORDS ARE OMMITED digunakan untuk print file yang tidak mempunyai label. LABEL RECORD IS STANDARD atau LABEL RECORDS ARE STANDARD untuk yang mempunyai label.

3. Sintaks PROCEDURE DIVISION pada file sequential

Statemen khusus yang dipergunakan pada file sequensial di dalam PROCEDURE DIVISION dapat dibentuk dari OPEN, CLOSE, READ, WRITE, REWRITE, dan USE.

3.1. OPEN verb

Digunakan untuk membuka file yg akan diakses.

Bentuk umum :

```
                 { INPUT    nama-file1  [ , nama-file2 ] .... }
                 { OUPUT    nama-file3  [ , nama-file4 ] .... }
OPEN { I-O         nama-file5  [ , nama-file6 ] .... }
                 { EXTEND   nama-file7  [ , nama-file8 ] .... }
```

- OPEN INPUT

Menunjukkan file yg akan dibuka sbg file input, atau data yg akan dibaca (READ) dari file ini.

- OPEN OUTPUT

  Menunjukkan file yg akan dibuka sbg file output atau data yg akan direkam (WRITE) pd file ini. Jika sudah ada data sebelumnya, maka data baru akan direkam & data yg lama akan terhapus.

  Jika akan menambahkan data ke file yg sudah berisi data, maka file harus dibuka dengan OPEN EXTEND
- OPEN I-O

  Menunjukkan file yg dibuka sebagai file-input dan juga sebagai file-output, yaitu file yg dibuka untuk tujuan sbg input & output sekaligus dapat dibaca & dimodifikasi.

3.2. CLOSE verb

Setelah operasi file selesai, maka semua file yg digunakan atau file yg masih terbuka harus ditutup kembali guna mencegah rusaknya struktur dari file.

Bentuk umum :

CLOSE nama-file1 [WITH LOCK][, nama-file2 [WITH LOCK]]

LOCK digunakan bila file yg sudah ditutup tidak dapat dibuka pd saat proses berlangsung, tetapi bila proses sudah dihentikan dan program dijalankan lagi file dapat dibuka kembali.

3.3. WRITE verb

Digunakan untuk menampilkan output ke printer atau merekam data ke file yg ditunjuk.

Bentuk umum :

WRITE nama-record [FROM nama-data1]

({ BEFORE / AFTER } ADVANCING { { { *nama-data2 / int* } [LINE] / PAGE [LINES] } })

[; AT { END OF PAGE/EOP } statement imperatif]

- statement WRITE

  Penulisan statement WRITE harus diikuti oleh nama-record dg level-number 01 pd FD di dalam DATA DIVISION yg dihubungkan dg suatu file yg dibuka.

  Jika file yg dibuka berupa disk-file, maka statement WRITE berfungsi untuk merekam data ke dalam file tersebut. Jika file yg dibuka berupa print-file, maka data akan dicetak pd printer.

- BEFORE option

  Menunjukkan banyaknya spasi baris pencetakan antara baris yg sedang dicetak pd kertas printer dg baris pencetakan selanjutnya.

- AFTER option

  Menunjukkan banyaknya spasi baris pencetakan yg digunakan oleh BEFORE atau AFTER yg ditunjukkan oleh nilai *integer* atau *nama-data2*.

  integer 0 tanpa spasi
  integer 1 1 spasi baris
  integer 2 2 spasi baris
  integer 3 3 spasi baris

  Jika BEFORE atau AFTER tidak digunakan, maka pencetakan diprinter akan dianggap sebagai AFTER ADVANCING 1 LINES

- PAGE option

  Digunakan bila data yg dicetak sebelum atau sesudah printer menempati posisi pd halaman baru selanjutnya.

- END OF PAGE atau EOP phrase

  Menunjukkan baris yg terakhir dalam 1 halaman pencetakan diprinter.

3.4. READ verb

Digunakan untuk membaca record yg ada pd file.

Bentuk umum :

READ nama-file RECORD [INTO nama-data]

[; AT END statement imperatif].

- INTO option

Digunakan bila data record yg dibaca akan dipindahkan secara langsung ke nama-data tertentu.

- AT END clause

Digunakan untuk mengetahui apakah data yg dibaca sudah habis.

contoh :

```
PROCEDURE DIVISION.
     -----------------
     -----------------
     -----------------
BUKA-FILE.
OPEN INPUT NILAI.
BACA-DATA.
     READ NILAI INTO NILAI1 AT END GO TO SELESAI.
TAMPILKAN.
     DISPLAY NILAI.
     GO TO BACA-DATA.
SELESAI.
     CLOSE NILAI.
     STOP RUN.
```

3.5. REWRITE verb

Digunakan untuk merekam ulang record yg sudah pernah direkam. Biasanya digunakan untuk memodifikasi atau memperbaiki data. Record yg direkam ulang adl record terakhir yg dibaca.

Bentuk umum :

REWRITE nama-record [FROM nama-data]

-Pada waktu REWRITE dilaksanakan, maka file sudah harus dibuka sebagai I Input-output file (OPEN I-O)

-Nama record setelah perintah REWRITE adl nama record yg sudah dijelaskan dalam DATA DIVISION pd FILE SECTION

-Jika digunakan FROM, record akan direkam ulang dg nilai data yg ada pd nama data setelah perintah FROM tersebut

PROGRAM
FILE SEQUENSIAL SEDERHANA

7

Obyektif :

19. **Memahami program file sequensial**
20. **Dapat membuat program yang serupa**
21. **Dapat mengembangkan program yang ada**

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SEQUEN1.
AUTHOR.
SECURITY. PROGRAM MEMASUKKAN DATA SEQUENSIAL.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT PENJUALAN ASSIGN TO DISK.
   ORGANIZATION IS SEQUENTIAL.
   FILE STATUS IS STATUS-SALAH.
DATA DIVISION.
FILE SECTION.
FD PENJUALAN
   LABEL RECORD IS STANDARD.
   VALUE OF FILE-ID ID JUAL.DAT.
   DATA RECORD IS RECORD-RELASI.
01 DATA-PENJUALAN.
   02 NOMER-FAKTUR         PIC X(12).
   02 NAMA-LANGGANAN       PIC X(25).
   02 KODE-JUAL            PIC A.
      88 TUNAI             VALUE IS 'T'.
      88 KREDIT            VALUE IS 'K'.
   02 NILAI-JUAL           PIC 9(6).
WORKING-STORAGE SECTION.
77 STATUS-SALAH            PIC XX.
01 SUDAH-BENAR             PIC X.
   88 BENAR                VALUE 'Y', 'y'.
   88 BELUM                VALUE 'T', 't'.
01 MASUKKAN-LAGI-TIDAK     PIC X.
   88 LAGI                 VALUE 'Y','y'.
   88 TIDAK                VALUE 'T','t'.
SCREEN SECTION
01 HAPUS-LAYAR.
   02 BLANK SCREEN.
01 LAYAR-DATA.
   02 LINE 4 COLUMN 5 'NOMOR FAKTUR    :'.
   02 COLUMN PLUS 1 PIC X(22) TO NOMER-FAKTUR REQUIRED.
   02 LINE 6 COLUMN 5 'NAMA LANGGANAN  :'.
   COLUMN PLUS 1 PIC X(25) TO NAMA-LANGGANAN.
   02 LINE 8 COLUMN 5 'KODE TRANSAKSI  :'.
```

```
   02 COLUMN PLUS 1 PIC A TO KODE-JUAL.
   02 LINE 10 COLUMN 5 'NILAI PENJUALAN :'.
   02 COLUMN PLUS 1 PIC 9(6) TO NILAI-JUAL.
PROCEDURE DIVISION.
RYTIN-UTAMA SECTION.
BIKA-FILE.
   OPEN OUTPUT PENJUALAN.
MULAI.
   MOVE SPACE TO SUDAH-BENAR
   PERFORM MASUKKAN-DATA UNTIL BENAR
   PERFORM REKAM-DATA
   PERFORM ADA-LAGI-TIDAK.
SELEKSI-MASUKKAN-LAGI-TIDAK.
   IF LAGI
      GO TO MULAI.
SELESAI.
   CLOSE PENUALAN
   STOP RUN.
RUTIN-BAGIAN SECTION.
MASUKKAN-DATA.
   DISPLAY HAPUS-LAYAR
   DISPLAY LAYAR-DATA
   ACCEPT LAYA-DATA.
   DISPLAY (18,5) 'SUDAH BENAR (Y/T)?'.
   ACCEPT ( , ) SUDAH BENAR.
REKAM-DATA.
   MOVE '  ' TO STATUS-SALAH
   WRITE DATA-PENJUALAN
   PERFORM SELEKSI REKAMAN.
ADA-LAGI-TIDAK.
   DISPLAY (20,5) 'ADA LAGI DATA LAINNYA (Y/T)?'
   ACCEPT ( , ) MASUKKAN-LAGI-TIDAK.
SELEKSI-REKAMAN.
   IF STATUS-SALAH = '34'
      DISPLAY (18,5) 'TIDAK TEREKAM, DISK PENUH'.
   IF STATUS-SALAH = '91'
      DISPLAY (18,6) 'STURKTUR FILE RUSAK'.
```