

Konsep Dasar Program COBOL

Pendahuluan

COBOL (Common Business Oriented Language) adalah suatu bahasa komputer awam (High Level Language) yang berorientasi langsung pada permasalahan bisnis. COBOL hampir dapat digunakan pada semua komputer bila terdapat compiler COBOL-nya. Diciptakan pada tahun 1959 dan dikembangkan oleh suatu grup bernama CODASYL (Conference on Data System Language).

Diperkenalkan pertama kali secara formal pada bulan Januari 1960. COBOL mempunyai banyak versi, sehingga untuk mempermudah penggunaannya COBOL terus dikembangkan dan distandarisasi pada tahun 1968 dan 1974, diberi nama ANSI COBOL (American National Standard Institute COBOL)

Struktur Program COBOL

Program COBOL dibagi kedalam 4 bagian yang disebut dengan **Division**. Masing-masing divisi dimulai dengan judul divisi dengan urutan sbb :

IDENTIFICATION DIVISION
ENVIRONMENT DIVISION
DATA DIVISION
PROCEDURE DIVISION

Masing-masing divisi dapat terdiri dari urutan-urutan susunan bagian. Tersusun secara hirarki, yang secara umum sbb :

Division
Region
Section
Paragraph
Sentence/Entry
Statement/Clause
Phrase/Option

Istilah *Region*, *sentence*, *statement* dan *phrase/option* digunakan pada PROCEDURE DIVISION, sedangkan istilah *Entry* dan *clause* digunakan pada ke-3 divisi lainnya.

1. *Division*, merupakan bagian utama dari suatu program COBOL dan selalu diawali dengan judul divisi.
2. *Region*, merupakan suatu kumpulan bagian tertentu dalam PROCEDURE DIVISION.
3. *Section*, merupakan suatu kumpulan dari *paragraph* atau *entry* dan selalu diawali dengan judul seksi.
4. *Paragraph*, merupakan suatu grup dari kalimat (*sentences*) didalam PROCEDURE DIVISION dan selalu diawali dengan nama/ judul Paragraph.
5. *Entry*, merupakan sesuatu yang harus dituliskan pada tempat-tempat tertentu didalam program COBOL. Suatu *entry* dapat juga dikatakan sebagai suatu set (kumpulan) dari *clause* (anak kalimat) yang diakhiri dengan titik
6. *Sentence*, merupakan kumpulan dari satu atau lebih *statement*, dan harus diakhiri dengan tanda titik.
7. *Clause*, merupakan kumpulan dari kata yang membentuk suatu arti. *Clause* adalah bagian dari *entry*.
8. *Statement*, merupakan perintah pengerjaan untuk komputer. *Statement* dalam bentuk kata kerja yang merupakan COBOL Reserved Words.
9. *Phrase*, suatu grup kata yang merupakan bagian dari *statement* atau *clause*.

10. *Option*, kebanyakan Phrase adalah optional (boleh disertakan/tidak) maka, sering disebut dengan *Option*.

User Defined Word

Programmer dapat membentuk kata-kata untuk membuat dan mendefinisikan tersendiri untuk pemberian nama pada :

1. **Nama-program** (program-name) adalah suatu nama yang diberikan untuk menunjukkan identitas dari program yang dibuat, dituliskan pada paragraph PROGRAM-ID dalam IDENTIFICATION DIVISION.
2. **Nama-alat** (mnemonic-name) adalah nama yang dibuat oleh programmer untuk menunjukkan suatu alat tertentu. Dibentuk pada paragraph SPECIAL-NAMES dalam ENVIRONMENTDIVISION.
3. **Nama-file** (file-name) adalah nama yang dibuat untuk menunjukkan suatu file tertentu yang dipergunakan dalam program. Dibentuk pada paragraph FD (File Description) dalam DATA DIVISION atau pada statement CLOSE< OPEN< READ dalam PROCEDURE DIVISION.
4. **Nama-record**, pada COBOL, record harus diberi nama-record (Record-name) pada DATA RECORD *clause* dan pada *record description entry* dalam DATA DIVISION.
5. **Nama-data** (data-name) adalah nama yang dibuat untuk menunjukkan suatu data item yang dipergunakan dalam program. Dibentuk pada FILE SECTION di *record description entry* dalam DATA DIVISION dan pada WORKING-STORAGE SECTION.
6. **Nama-indek dan nama-data-indek**, digunakan untuk data pada tabel yang di-indek. Dibentuk secara implisit dengan OCCURS dan INDEXED BY *clause*. Nama-data-indek didefinisikan dengan menggunakan USAGE IS INDEX dalam DATA DIVISION.
7. **Nama-kondisi** (condition-name) adalah nama data yang dihubungkan dengan suatu nilai tertentu. Didefinisikan dalam DATA DIVISION dengan level number 88 dan dioperasikan dalam PROCEDURE DIVISION pada statement IF.
8. **Nama-prosedur** (procedure-name) atau nama-paragraph (paragraph-name) adalah nama yang menunjukkan suatu paragraph dalam PROCEDURE DIVISION. Nama-paragraph diperlukan bila proses akan melompat ke paragraph tertentu dengan menggunakan statement GO TO dan PERFORM dalam PROCEDURE DIVISION.
9. **Nama-seksi** , pada PROCEDURE DIVISION, dapat dibuat menjadi beberapa seksi dan tiap seksi dapat dibentuk dengan dimulai judul seksinya yang disebut dengan Section-name.
10. **Nama-kualifikasi** bila nama-data atau nama-kondisi tidak unik (ada yang sama satu dengan yang lain), untuk menentukan yang mana yang akan digunakan, dapat digunakan *qualifier*.

Syarat Pemberian nama :

1. Gabungan dari huruf A-Z atau a-z, angka 0-9, Hyphen (-).
2. Panjang maksimum 30 karakter.
3. Paling sedikit harus mengandung 1 huruf.
4. tidak boleh mengandung karakter khusus kecuali hyphen yang diletakkan ditengah-tengah, tidak boleh diawal atau diakhir.
5. Tidak boleh mengandung COBOL reserved word.
6. Tidak boleh ada blank atau spasi.

Bentuk data

Dibagi menjadi 2 :
1. Data Variabel
2. Konstanta

Data variabel

Data variabel adalah data yang nilainya dapat berubah didalam program. Nilai data ini akan selalu berubah bila dibaca nilai data yang lain dengan nama data yang sama.

Konstanta

Konstanta atau data konstanta adalah bentuk dari data yang dibutuhkan untuk pengolahan dimana nilai datanya tidak tergantung dari input yang dibaca. Ada 3 bentuk dari konstanta yang dapat dipakai dalam PROCEDURE DIVISION pada program COBOL, yaitu Numeric Literal, Non Numeric Literal, dan Figurative Constant.

Numeric Literal

1. Aturan penggunaan literal numerik :
 1. Panjang maksimum 18 digit.
 2. Boleh ada tanda plus atau minus yang letaknya pada posisi terkiri tanpa ada spasi dengan angka pertama. Bila tidak bertanda berarti bernilai positif.
 3. Boleh ada titik desimal dimana saja kecuali pada posisi terkanan.

Non Numeric Literal

Adalah konstanta yang digunakan bukan untuk operasi arithmatika. Aturan penggunaan literal bukan numerik :

1. Panjang maksimum 120 karakter
2. Boleh terdiri dari kumpulan karakter (Character set) kecuali karakter petik.
3. Dibatasi tanda petik pada awal dan akhir.

Figurative Constant

Figurative constant termasuk dalam COBOL reserved word yang mempunyai maksud tertentu yang namanya sudah dikenal oleh compiler. Contoh :

ZERO, ZEROS, ZEROES berarti nilai nol
SPACE, SPACES berarti nilai 1 blank atau spasi
QUOTE, QUOTES berarti tanda petik

Operator Arithmatika

Operator	Fungsi	Jenjang
**	Pemangkatan	1
*	Perkalian	2
/	pembagian	2
+	penjumlahan	3
-	pengurangan	3

ungkapan arithmatika adalah kombinasi dari literal numerik, nama data, operator aritmatika, kurung buka dan kurung tutup. Penulisan ungkapan aritmatika harus dipisahkan paling sedikit dengan sebuah blank atau spasi.

Aturan Penulisan Source Program

Kolom yang tersedia untuk menuliskan program COBOL adalah kolom 1- 80, dengan ketentuan :

Kolom 1 – 6 : Digunakan untuk nomor urut bilamana diperlukan, sifatnya optional, dan nomor yang diberikan harus urut ascending.

Kolom 7 :

- a. Tanda baris sambung dari baris sebelumnya dengan memberikan tanda hyphen (-), baris sambungannya dimulai dari Area B.
- b. Bila kolom ini diisi dengan karakter “*”, maka apa yang ditulis pada baris ini akan dianggap sebagai komentar.
- c. Bila kolom ini diisi dengan slash (/), maka baris yang ada tanda ini dianggap sebagai komentar dan akan dicetak mulai halaman baru teratas, bila source program dicetak di printer.

- d. Bila kolom ini di isi dengan karakter “D” , baris ini juga dianggap sebagai komentar, tetapi bila pada ENVIRONMENT DIVISION dalam paragraph SOURCE-COMPUTER disebutkan WITH DEBUGGING MODE, maka akan berguna untuk tujuan *debugging*.

- Kolom 8 – 11 : Disebut dengan area A, semua judul divisi, judul seksi, nama paragraph, judul file description (FD), level number 01 dan level number 77 ditulis mulai kolom ke-8.
- Kolom 12 – 72: Disebut dengan area B, semua elemen program selain yang ditulis di area A ditulis pada area B.
- Kolom 73 – 80: Merupakan kolom yang tidak diproses oleh komputer, jadi jika diisi dengan catatan-catatan apa saja hanya untuk dokumentasi program.

Level Number

Level number atau nomor jenjang adalah suatu nomor yang menunjukkan jenjang dari data item dalam suatu record. Level number yang digunakan :

- a) 01, digunakan untuk mengawali keterangan dari record (record description).
- b) 02 – 49, digunakan untuk mengawali keterangan dari data item.
- c) 66, digunakan untuk RENAMES.
- d) 77, Digunakan untuk mengawali data item yang berdiri sendiri. Hanya boleh terdapat pada WORKING-STORAGE SECTION.
- e) 88, digunakan untuk mengawali suatu nama kondisi.

Picture clause

Picture clause digunakan untuk menerangkan masing-masing data-item yang digunakan, mengenai ukuran dari field dan memberikan informasi dari nilai data tersebut, juga mengatur bentuk data dimemori. Macam-macam picture clause :

1. Picture karakter 9
Menunjukkan bahwa posisi sebuah memori hanya dapat mengandung nilai-nilai numerik saja. Ukuran panjang data-item ditentukan oleh banyaknya karakter 9 yang digunakan. Bersifat *right justified*.
2. Picture karakter V
Menunjukkan lengkap anggapan dari titik desimal. Anggapan disini maksudnya adalah titik desimal tersebut tidak ditulis di memori, sehingga tidak termasuk sebagai panjang field.
3. Picture karakter P
Digunakan dengan gabungan Picture karakter V, yang digunakan untuk menimbulkan angka 0.
4. Picture karakter S
Digunakan untuk menyimpan tanda dari nilai data, dan tidak dihitung sebagai panjang field.
5. Picture karakter A
Digunakan untuk menyimpan nilai data huruf(alphabetic). Bersifat *left justified*.
6. Picture karakter X
Digunakan untuk menyimpan nilai data alphanumeric, dapat berupa gabungan angka, huruf, ataupun karakter khusus.

Picture Editing

Berguna untuk perubahan bentuk data dari data yang telah tersimpan distorage (memori). Dengan adanya editing (perubahan), data yang dicetak pada output akan tampak lebih mudah dibaca, mudah dimengerti dan mempunyai susunan yang lebih baik dibandingkan dengan bentuk data asli yang tersimpan di storage. Macam-macam picture editing :

1. Picture editing Z
Digunakan untuk menggantikan angka 0 di awal agar tidak tampak pada waktu pencetakan.
2. Picture editing \$
Pada saat pencetakan karakter \$ akan tampak pada ujung paling kiri.
3. Picture editing “.”
Digunakan untuk menunjukkan letak posisi dari titik desimal pada pencetakan, hanya boleh dipergunakan 1 saja.
4. Picture editing “,”
Digunakan untuk memberikan tanda koma pada tempat-tempat tertentu, dan boleh digunakan lebih dari 1.
5. Picture editing “-“
Jika data asli bernilai minus maka penggunaan tanda minus di awal atau diakhir akan menyebabkan tanda minus muncul pada saat pencetakan, sedangkan bila data positif akan digantikan dengan blank.
6. Picture editing “+”
Jika tanda plus digunakan diawal atau diakhir data maka akan tampak pada pencetakan bila data bernilai positif, jika data negatif akan muncul tanda minus, jika tidak bertanda dianggap positif.
7. Picture editing B
Digunakan untuk mengedit nilai data bukan numerik. Blank akan disisipkan pada posisi dimana karakter B ditempatkan.
8. Picture editing “*”
Digunakan untuk menggantikan nilai nol pada nilai data disebelah kanan dengan tanda *.
9. Picture editing 0(nol)
Digunakan untuk menyisipkan angka 0 pada posisi dimana karakter 0 ditempatkan.
10. Picture editing “/”
Digunakan untuk menyisipkan karakter “/” diposisi dimana karakter tersebut ditempatkan.
11. Picture editing DB dan CR
Sering digunakan pada aplikasi akuntansi. Ditulis mulai ujung sebelah kanan dari picture, dan hasilnya hanya tampak pada hasil edit bila nilai datanya negatif.

IDENTIFICATION DIVISION

Merupakan divisi yang pertama dalam COBOL dan yang paling sederhana. IDENTIFICATION DIVISION tidak dibagi dalam beberapa section tetapi langsung terdiri dari beberapa paragraph.

Bentuk umum dari IDENTIFICATION DIVISION :

```

IDENTIFICATION DIVISION.
PROGRAM-ID. nama program.
[ AUTHOR. nama programmer .]
[ INSTALLATION. nama instalasi. ]
[ DATE-WRITTEN. Tgl program dibuat. ]
[ DATE-COMPILED. Tgl program dicompile. ]
[ SECURITY. sifat program. ]

```

keterangan :

- Tulisan dalam tanda ‘ [] ’ bersifat optional, boleh disertakan boleh juga tidak.
- Ditulis persis seperti contoh diatas, diikuti titik dan diberi 1 spasi.
- Penulisan divisi dan semua paragraphnya dimulai pada area A atau kolom ke-8.
- Huruf kecil adalah nama yang dibuat oleh programmer.

- PROGRAM-ID panjang maks. 6 karakter, boleh lebih tetapi hanya 6 karakter pertama saja yang dibaca.

ENVIRONMENT DIVISION

Menyediakan informasi mengenai peralatan yang dipergunakan didalam program.

Bentuk umum ENVIRONMENT DIVISION :

```

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
[ SOURCE-COMPUTER. nama-komputer WITH DEBUGGING MODEL. ]
[ OBJECT-COMPUTER. nama-komputer. ]
SPECIAL-NAMES.
[ PRINTER IS nama-mnemonic ]
[ CURRENCY SIGN IS literal ]
[ DECIMAL-POINT IS COMMA. ]

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

```

CONFIGURATION SECTION

Diperlukan karena kemungkinan program dibuat pada sebuah komputer dan dijalankan pada komputer yang lain.

- Paragraph SOURCE-COMPUTER
Paragraph ini digunakan untuk menunjukkan nama komputer yang digunakan dalam pembuatan dan mengkompilasi program. Ditulis mulai kolom ke-8 atau area A. Bila Clause WITH DEBUGGING MODE disertakan dan pada kolom ke-7 diberi karakter 'D' akan dikompilasi untuk tujuan penelusuran kesalahan. Statement untuk penelusuran kesalahan adalah statement READY TRACE, RESET TRACE, EXHIBIT.
- Paragraph OBJECT_COMPUTER
Paragraph ini untuk menunjukkan nama komputer yang digunakan untuk menjalankan program yang telah dikompilasi.
- Paragraph SPECIAL_NAMES
Paragraph ini sifatnya optional, digunakan untuk membuat nama khusus yang menghubungkan nama-mnemonic implementor dengan nama mnemonic yang dibuat programmer.
 - a. PRINTER IS Clause
digunakan untuk menghubungkan nama mnemonic yang dibuat programmer dengan alat pencetak (PRINTER). Clause ini digunakan pada statement DISPLAY yang menggunakan UPON dalam PROCEDURE DIVISION.
 - b. CURRENCY SIGN IS Clause
Untuk menunjukkan tanda mata uang digunakan \$, jika ingin menggunakan simbol lain maka dituliskan pada SPECIAL_NAMES.
 - c. DECIMAL_POINT IS COMMA Clause
Pada bahasa COBOL titik desimal digunakan bentuk karakter '.' bila di inginkan bukan karakter '.' tetapi koma dapat dituliskan SPECIAL_NAMES. DECIMAL POINT IS COMMA.

INPUT-OUTPUT SECTION.

Bila akan digunakan file dengan media simpanan sekunder, maka paragraph FILE_CONTROL dalam seksi ini harus ditulis. FILE_CONTROL entry terdiri dari 3 bentuk, tergantung dari organisasi file yang dipergunakan, yaitu ; Sequential file, indexed file, dan relative file.

- INPUT-OUTPUT SECTION untuk SEQUENTIAL FILE

Organisasi file bersifat terurut, data direkamkan direcord dalam file secara urut dan urutannya tidak akan berubah. Organisasi file ini hanya pada DISK dan PRINTER.

Bentuk umum :

```
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 ].
```

- INPUT-OUTPUT SECTION untuk INDEXED FILE

Organisasi file yang datanya dapat diambil langsung diposisi record yang mempunyai nilai kunci tertentu yang unik. Organisasi seperti ini memungkinkan pencarian data secara cepat. Organisasi file ini hanya pada DISK. Bentuk umum :

```
INPUT-OUTPUT SECTION.  
FILE_CONTROL.  
SELECT nama-file ASSIGN TO DISK  
    ORGANIZATION IS INDEXED  
    ACCESS MODE IS SEQUENTIAL  
                    RANDOM  
                    DYNAMIC  
    RECORD KEY IS nama-data 1  
    [ FILE STATUS IS nama-data 2 ]
```

- INPUT-OUTPUT SECTION untuk RELATIVE FILE

Organisasi file yang tiap-tiap recordnya dibedakan dengan suatu nomor record relative. Organisasi ini memungkinkan pencarian data yang cepat, hanya terdapat pada DISK.

Bentuk umum :

```
INPUT-OUTPUT SECTION.  
FILE_CONTROL.  
SELECT nama-file ASSIGN TO DISK  
    ORGANIZATION IS RELATIVE  
    ACCESS MODE IS SEQUENTIAL[, RELATIVE KEY IS nama-data 1 ]  
                    RANDOM  
                    DYNAMIC , RELATIVE KEY IS nama-data 1  
    [ FILE STATUS IS nama-data 2 ]
```

DATA DIVISION

Memberikan penjelasan tentang input data dan output yang dipergunakan, atau berisi semua keterangan tentang file, record, nama-data serta bentuk / format yang akan dipergunakan didalam PROCEDURE DIVISION.

1. FILE SECTION

Berisi mengenai file-file yang dipakai didalam program, FILE SECTION ini ada bila dipergunakan file dalam bentuk simpanan sekunder yaitu disk atau printer yang mempunyai hubungan dengan INPUT_OUTPUT SECTION dalam ENVIRONMENT DIVISION.

Bentuk umumnya :
 DATA DIVISION.
 FILE SECTION.
 FD nama-file

```

    ;BLOCK CONTAINS [integer-1 TO ] integer-2      CHARACTERS/RECORDS
  [ ;RECORD CONTAINS [ integer-3 TO ] integer-4    CHARACTERS]
    ;LABEL      RECORD IS      } STANDARD
    ;           RECORDS ARE    } OMITTED
  [ ;VALUE OF FILE-ID IS nama-file di label      ]
    ;DATA RECORD IS      }
    ;           RECORDS ARE } nama-record-1 [, nama-record-2]
    ;LINKAGE IS  nama-data-1 } LINES ; WITH FOOTING AT } nama-data-2
    ;           integer-5   }                          } integer-6
    ;LINES AT TOP      } nama-data-3
    ;                 } integer-7
    ;LINES AT BOTTOM   } nama-data-4
    ;                 } integer-8
  
```

judul FD ditulis pada area A dan diikuti oleh nama filenya yang harus sama dengan yang disebutkan pada INPUT-OUTPUT SECTION dalam ENVIRONMENT DIVISION, nama-file ditulis mulai area B.

- **BLOCK CONTAINS Clause**
Menunjukkan ukuran dari record didalam file untuk tiap-tiap blocknya. Clause ini biasa digunakan pada file yang berupa tape magnetik.
- **RECORD CONTAINS Clause**
Menunjukkan banyaknya karakter tiap-tiap recordnya. Clause ini boleh tidak disertakan, karena nantinya akan termasuk dalam record description entry
- **LABEL RECORD dan LABEL RECORDS Clause**
Menunjukkan apakah file yang digunakan mempunyai label atau tidak.
LABEL RECORD IS OMITTED digunakan untuk card-file atau print-file yang tidak mempunyai label.
LABEL RECORD IS STANDARD digunakan untuk disk-file yang mempunyai label.
- **VALUE OF FILE-ID Clause**
Menunjukkan informasi mengenai file yang mempunyai label di disk. Label di disk ini berupa suatu nama-file yang berisi data yang direkamkan tersebut.
- **DATA RECORD atau DATA RECORDS Clause**
Menunjukkan nama-record dalam file, sifatnya optinal, hanya sebagai dokumentasi saja.
- **LINAGE Clause**
Pencetakan output di printer dapat dilakukan dengan 2 cara, yaitu:
 - a. Dengan menganggap printer sebagai print-file, yang berbentuk organisasi file secara sequential, dengan menggunakan statement WRITE pada procedure division.

- b. Dengan tanpa menggunakan print-file, tetapi langsung menghubungkan alat cetak printer sebagai nama-mnemonik pada paragraph SPECIAL-NAMES, yang kemudian menggunakan statement DISPLAY... UPON.

LINAGE Clause digunakan untuk mengatur pencetakan output diprinter yang mempergunakan cara pertama, yaitu sebagai print-file yang menggunakan statement WRITE, menunjukkan jumlah baris yang akan dicetak per halamannya, jumlah dari baris kosong paling atas (TOP MARGIN) dan jumlah dari baris kosong paling bawah (BOTTOM MARGIN).

LINAGE IS menunjukkan tubuh dari laporan, daerah laporan yang akan dicetak meliputi judul, isi dan footingnya.

WITH FOOTING AT menunjukkan daerah letak permukaan foot-note/footing (untuk pencetakan suatu total atau keterangan-keterangan untuk laporan).

LINES AT BOTTOM menunjukkan sejumlah baris yang tidak dipergunakan pada ujung bawah laporan.

2. WORKING-STORAGE SECTION

Mempunyai maksud pemesanan tempat di internal memori (STORAGE) yang diperlukan oleh pekerjaan (WORKING) proses program. Pemesanan tempat ini biasanya digunakan untuk :

- a. Bentuk, nilai dan nama data yang diperlukan didalam proses yang terpisah dan belum di sebutkan pada FILE SECTION. Data yang disebutkan pada FILE SECTION adalah data yang akan diambil dari disk-file, atau data yang akan direkamkan pada disk-file atau yang akan dicetak pada print-file.
- b. Persiapan penulisan judul.
- c. Pemesanan tempat untuk data output yang akan ditampilkan.

Bentuk umum :
 WORKING-STORAGE SECTION.
 77- level data description entry
 record description.

3. SCREEN SECTION

berguna untuk menunjukkan bentuk format dari layar terminal untuk menampilkan data atau memasukkan data. Data-item yang dipergunakan dapat berupa group data item atau data item individu. Bentuk umum :

```

SCREEN SECTION.
level-number [nama-layar]
[ BLANK SCREEN ]
[ LINE NUMBER IS [PLUS] integer-1 ]
[ COLUMN NUMBER IS [PLUS] integer-2 ]
[ BLANK LINE ]
[ BELL ]
UNDERLINE
REVERSE-VIDEO
HIGHLIGHT
BLINK
[ VALUE IS literal-1 ]
Picture literal-2
PIC is karakter-string FROM nama-data-1 TO nama-data-2
USING nama-data-3
  
```

```

[     BLANK WHEN ZERO ]
[     AUTO ]
[     SECURE   ]
[     REQUIRED  ]
[     FULL   ]

```

SCREEN berhubungan dengan statemet DISPLAY dan ACCEPT dalam PROCEDURE DIVISION. Statement DISPLAY dipergunakan untuk menampilkan format yang telah dibentuk di SCREEN SECTION pada layar terminal. Statement ACCEPT digunakan untuk memasukkan nilai data-item lewat layar terminal dengan bentuk format yang dibentuk pada SCREEN SECTION tersebut.

- BLANK SCREEN Clause, digunakan untuk membersihkan layar dan menempatkan kursor pada posisi kiri atas.
- LINE Clause, digunakan untuk menempatkan cursor pada posisi baris tertentu.
- COLUMN Clause, digunakan untuk menempatkan kursor pada posisi kolom tertentu.
- BLANK LINE Clause, digunakan untuk menghapus tampilan pada baris tertentu dilayar.
- UNDERLINE, REVERSE-VIDEO, HIGHLIGHT dan BLINK Clause, digunakan untuk menampilkan efek tertentu.
 1. UNDERLINE clause, digunakan untuk memberikan garis bawah pada tampilannya.
 2. REVERSE-VIDEO clause, digunakan untuk membalik warna dari tampilannya, warna dasar menjadi warna tampilannya dan sebaliknya.
 3. HIGHLIGHT clause, digunakan untuk menampilkan tampilan dengan bentuk yang lebih terang.
 4. BLINK clause, digunakan untuk membuat kedap-kedip bentuk tampilannya.
- FROM, TO dan USING Clause, digunakan untuk menampilkan atau menerima data yang bentuk data-itemnya dihubungkan ditempat lain diluar SCREEN SECTION. misalnya pada WORKING-STORAGE SECTION, FILE SECTION, atau LINKAGE SECTION.
 1. FROM clause digunakan untuk menampilkan isi data.
 2. TO clause digunakan untuk menerima isi data.
 3. USING clause digunakan untuk menggantikan FROM clause, bila hanya dipergunakan statement DISPLAY. atau digunakan untuk menggantikan FROM dan TO yang digunakan bersama-sama, bila digunakan statement DISPLAY dan ACCEPT.
- BLANK WHEN ZERO Clause, digunakan untuk menampilkan spasi / blank bila suatu data numerik mengandung nilai 0.
- AUTO, SECURE, REQUIRED, dan FULL Clause, digunakan untuk memberikan efek tertentu pada waktu memasukkan nilai suatu data.
 1. AUTO clause, digunakan untuk pergeseran cursor secara otomatis ke field berikutnya, bila field data item sudah penuh terisi.
 2. SECURE clause, digunakan untuk membuat supaya suatu nilai data yang dimasukkan pada field data item tertentu tidak tampak sewaktu diketik, sebagai gantinya akan muncul karakter '*'.
 3. FULL clause, digunakan untuk suatu field data item yang harus diisi dengan suatu data sampai penuh untuk tempat yang disediakan.

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 TO nama-data-2 [, nama-data-3] ...
literal

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 , nama-data .. UPON nama-mnemonic
literal , literal

c. Bentuk 3

DISPLAY (posisi tampilan) nama-data ..UPON nama-mnemonic
literal ERASE

- **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 : ZERO - FILL
SPACE - FILL
LEFT - JUSTIFY
RIGHT - JUSTIFY
TRAILING - SIGN
PROMPT

ACCEPT (posisi layar) nama data WITH

UPDATE
LENGTH - CHECK
EMPTY – CHECK
AUTO – SKIP
NO – ECHO
BEEP

1. ZERO-FILL phrase menyebabkan bila posisi-posisi field data-item penerima data tidak di isi dengan data (langsung menekan enter) akan terisis dengan nol.
2. 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.
3. LEFT-JUSTIFY phrase tidak berfungsi pada MS COBOL, tetapi boleh ditulis
4. RIGHT-JUSTIFY phrase menyebabkan setelah data dimasukkan, hasil akhir yang tampak dilayar akan rata sebelah kanan. Digunakan untuk jenis data-item alphabetik atau alphanumeric.
5. TRAILING-SIGN phrase menyebabkan tanda operasi + atau – tampak diposisi paling kanan dari field data input.
6. 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).
7. UPDATE phrase menyebabkan tampilan utuk field data-item penerima berbentuk nilai awal dari field penerima tersebut.
8. LENGTH-CHECK phrase menyebabkan penekanan tombol carriage return tidak berfungsi kalau semua posisi field penerima belum terisi semua.
9. 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.
10. 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.
11. NO-ECHO phrase menyebabkan data yang dimasukkan tidak tampak dilayar.
12. BEEP phrase menyebabkan bunyi bel sewaktu data di input.

d. Bentuk 4

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

1. DATE, akan mendapatkan 6 digit nilai standard dengan bentuk YYMMDD, diambil langsung dari “system-date”.(2 digit tahun, 2 digit bulan, 2 digit tanggal)
2. DAY, akan mendapatkan 5 digit nilai “julian date” dengan bentuk YYDDD (2 digit tahun, 3 digit jumlah hari untuk tanggal tersebut)
3. 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).
4. 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, nama-data-2 ... TO nama-data-m [ROUNDED]
     literal-1      literal-2
                                     [; ON SIZE ERROR statement imperative]
```

BU-2:

```
ADD  nama-data-1, nama-data-2 ...GIVING nama-data-m [ROUNDED]
     literal-1      literal-2
                                     [; ON SIZE ERROR statement imperative]
```

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 storsge dan program akan melanjutkan pada imperative statement yang mengikutinya.

- **SUBSTRACT Verb**

Digunakan untuk operasi pengurangan suatu nilai data numerik.

BU :

```
SUBSTRACT nama-data-1 ,      nama-data-2
           literal-1      literal2
           FROM nama-data-n      GIVING      nama-data-m
           literal-n
           [ROUNDED] [;ON SIZE ERROR imperative statement]
```

- **MULTIPLY Verb**

Digunakan untuk mengalikan 2 nilai numerik dan menyimpan hasilnya.

BU 1 :
MULTIPLY nama-data-1 BY nama-data-2
literal-1

BU 2 :
MULTIPLY nama-data-1 BY nama-data-2
literal-1 literal-2
GIVING nama-data-3 [ROUNDED]
[; ON SIZE ERROR imperative statement]

- **DIVIDE Verb**

Digunakan untuk membentuk statement operasi pembagian.

BU 1 :
DIVIDE nama-data-1 INTO nama-data-2 [ROUNDED]
literal-1 [; ON SIZE ERROR imperative statement]

BU 2 :
DIVIDE nama-data-1 INTO nama-data-2
literal-1 literal-2
GIVING nama-data-3[ROUNDED]
[; ON SIZE ERROR imperative statement]

BU 3 :
DIVIDE nama-data-1 BY nama-data-2
literal-1 literal-2
GIVING nama-data-3[ROUNDED]
[; ON SIZE ERROR imperative statement]

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

BU 5 :
DIVIDE nama-data-1 BY nama-data-2
literal-1 literal-2
GIVING nama-data-3[ROUNDED]
REMAINDER nama-data-4
[; ON SIZE ERROR imperative statement]

- **COMPUTE**

Digunakan untuk operasi yang lebih rumit, untuk menyederhanakan 4 arithmetic 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
- **ALTER Verb**
Digunakan untuk merubah arah tujuan proses dari statement GO TO yang telah ada di program.
BU :
ALTER nama-paragraph-1 TO [PROCEED TO] nama-paragraph-2
- **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.
BU :
PERFORM nama-paragrappg-1 THROUGH nama-paragraph-2
THRU

DEBUGGING

Debugging adalah prosese mencari sebab kesalahan program dan membetulkannya. Kesalahan-kesalahan tersebut diantaranya : kesalahan bahasa (language errors), kesalahan sewaktu proses (run-time errors), kesalahan logika program (logical errors).

Kesalahan Bahasa

Kesalahan bahasa atau kesalahan penulisan (syntax errors) adalah kesalahan didalam penulisan sorce program yang tidak sesuai dengan yang disyaratkan. Kesalahan in merupakan kesalahan yang relatif mudah dilacak dan dibetulkan, karena kompil COBOL akan mendeteksi dan memberitahukan letak serta sebab kesalahannya.

Kesalahan Sewaktu Proses

Kesalahan sewaktu proses (run-time errors) adalah kesalahan yang terjadi sewaktu *executable* program dijalankan. Kesalahan ini menyebabkan program terhenti sebelum saatnya. Bila terjadi run-time errors, COBOL akan menyajikan sebab dari kesalahan, nama-program dari PROGRAM-ID dan baris dari program yang salah.

Kesalahan Logika

Merupakan kesalahan yang tidak bisa dilacak oleh COBOL, karena kesalahan terdapat pada logika pemrogramannya.

Contoh-contoh program

Contoh 1: (Hello.cob)

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. HELLO.  
AUTHOR. AYU.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
SCREEN SECTION.  
01 HAPUS-LAYAR.  
    02 BLANK SCREEN.  
PROCEDURE DIVISION.  
MULAI.  
    DISPLAY HAPUS-LAYAR.  
    DISPLAY "HELLO WORLD".  
    STOP RUN.
```

Output 1 :

HELLO WORLD

Contoh 3 :

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. PERFORM.  
AUTHOR. AYU.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
77 Y PIC 99.  
PROCEDURE DIVISION.  
MULAI.  
    PERFORM TAMPILKAN VARYING Y  
        FROM 1 BY 1 UNTIL Y > 10.  
    STOP RUN.  
TAMPILKAN.  
    DISPLAY Y;
```

Contoh 4 :

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. PERFORM1.  
AUTHOR. AYU.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
PROCEDURE DIVISION.  
MULAI.  
    PERFORM BINTANG.  
    DISPLAY '  C O B O L'.  
    PERFORM BINTANG.  
    STOP RUN.  
BINTANG.  
    DISPLAY '*****'.
```

Contoh 2:

```
IDENTIFICATION DIVISION  
PROGRAM-ID. UJIAN.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
77 X PIC 9(7).  
77 Y PIC ++Z9(5).  
SCREEN SECTION.  
01 HAPUS-LAYAR.  
    02 BLANK SCREEN.  
PROCEDURE DIVISION.  
MULAI.  
    DISPLAY HAPUS-LAYAR.  
    MOVE 0001234 TO X.  
    MOVE X TO Y.  
    DISPLAY Y.  
    STOP RUN.
```

Output 2 : +01234

Output 3 :

01
02
03
04
05
06
07
08
09
10

Output 4 :

```
*****  
  C O B O L  
*****
```

Contoh 5 :

```

IDENTIFICATION DIVISION
PROGRAM-ID. GOTO.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
Prosedur1.
    GO TO Prosedur3.
    DISPLAY "Procedure 3".
Prosedur2.
    DISPLAY "Procedure 1".
Prosedur3.
    DISPLAY "Procedure 2".
Selesai.
STOP RUN.

```

Output 5 :

```

Procedure 2

```

Contoh 6 :

```

IDENTIFICATION DIVISION.
PROGRAM-ID. CONTOH6.
AUTHOR. AYU.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 JUMLAH PIC 99.
77 A PIC 999.
77 B PIC 9 VALUE 2.
77 C PIC 9 VALUE 1.
PROCEDURE DIVISION.
MULAI.
    DISPLAY 'MASUKAN NILAI A ???'
    ACCEPT A.
    DISPLAY 'BERAPA KALI PERULANGAN : '
    ACCEPT JUMLAH.
    PERFORM HITUNG JUMLAH TIMES.
    STOP RUN.
HITUNG.
    COMPUTE A = B * ( A + 2 ).
    DISPLAY 'PERULANGAN KE : ', C.
    COMPUTE C = C + 1.
    DISPLAY 'NILAI A = ', A.

```

Output 6:

```

MASUKAN NILAI A???
2
BERAPA KALI PERULANGAN :
3
PERULANGAN KE : 1
NILAI A : 008
PERULANGAN KE : 2
NILAI A : 020
PERULANGAN KE : 3
NILAI A : 044

```

Contoh 7 : (lfstat.cob)

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. IFSTAT.  
AUTHOR. AYU.  
ENVIRONMENT DIVISION.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
77 A PIC 99.  
77 B PIC 99.  
77 C PIC 99 VALUE 32.  
PROCEDURE DIVISION.  
MULAI.  
    DISPLAY 'INPUT NILAI A: ',  
    ACCEPT A.  
    DISPLAY 'INPUT NILAI B: ',  
    ACCEPT B.  
    IF ( A + B ) NOT= C PERFORM GARIS  
    IF ( C - A ) < 20 PERFORM BINTANG.  
    DISPLAY 'BAHASA COBOL'.  
    STOP RUN.  
BINTANG.  
    DISPLAY '*****'.  
GARIS.  
    DISPLAY '-----'.
```

Output 7 :

```
INPUT NILAI A :  
10  
INPUT NILAI B :  
5  
-----  
BAHASA COBOL
```