PENANGANAN STRING PADA COBOL

OBJEKTIF :

1. Mahasiswa Mengetahui Penanganan String pada Bahasa COBOL.
2. Mahasiswa Mampu Membuat dan Menjalankan Perintah Penanganan String pada Bahasa COBOL.
3. Mahasiswa Mampu Memahami Struktur Perintah Penanganan String pada Bahasa COBOL.

8.1 PENANGANAN STRING PADA COBOL

Dalam kebanyakan Bahasa Pemrograman, manipulasi string dilakukan dengan menggunakan fungsi dalam library seperti pada Java dengan metode *Java String Class*. Pemrograman Bahasa COBOL juga menggunakan fungsi library manipulasi string, akan tetapi sebagain besar manipulasi string di selesaikan menggunakan *reference modification* dan juga menggunakan 3 penanganan string *(string-handling)* yaitu *INSPECT*, *STRING* dan *UNSTRING*. Ketiga penanganan string tersebut akan digunakan untuk menghitung dan mengganti karakter serta menyatukan dan memisahkan string. Pernyataan penangan string pada cobol juga digunakan untuk melakukan beberapa operasi fungsional pada strings.

8.2 PERNYATAAN INSPECT

Pernyataan inspect digunakan untuk menghitung atau menggantikan karakter. Operasi string dapat dilakukan pada nilai alfanumerik, numerik atau alfabetik, dengan operasi dilakukan dari kiri ke kanan.

INSPECT TALLYING

Digunakan untuk menghitung banyaknya karakter dalam sebuah string.

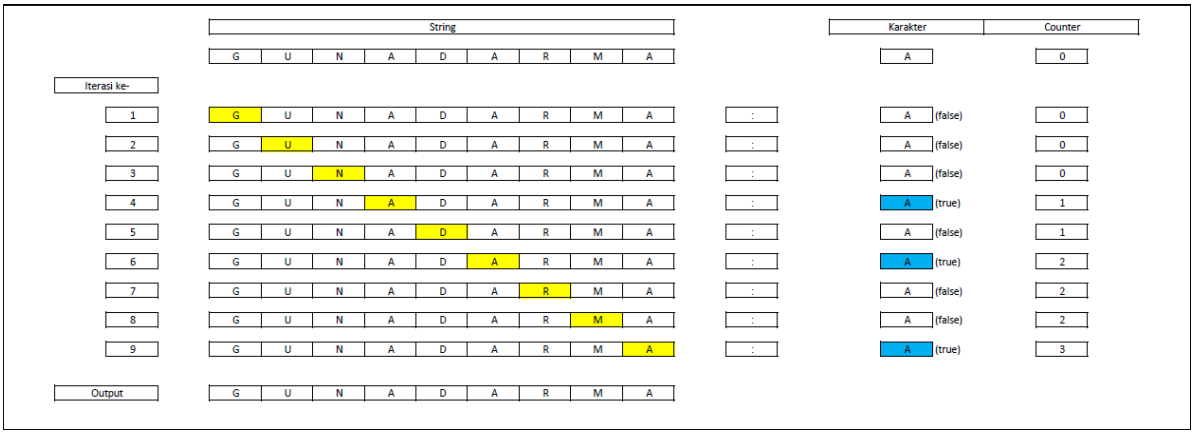
Bentuk umum :

```
        INSPECT input-string
        TALLYING output-count FOR ALL CHARACTERS
```

Dimana,

- Input-string : String yang karakternya akan dihitung
- Output-string : Data item untuk menampung jumlah karakter

Contoh penggunaan inspect tallying :



INSPECT REPLACING

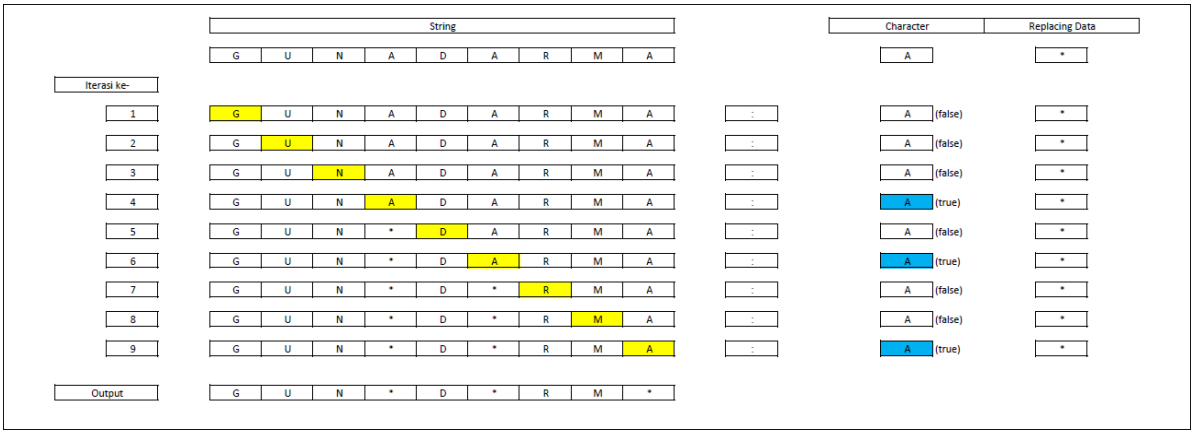
Digunakan untuk menggantikan sebuah karakter didalam string dengan karakter pengganti.

Bentuk umum :

```
INSPECT input-string
REPLACING ALL char1 BY char2
```

Dimana, input-string merupakan string yang karakternya akan diganti, dari char1 ke char2.

Contoh penggunaan inspect replacing:



INSPECT TALLYING REPLACING

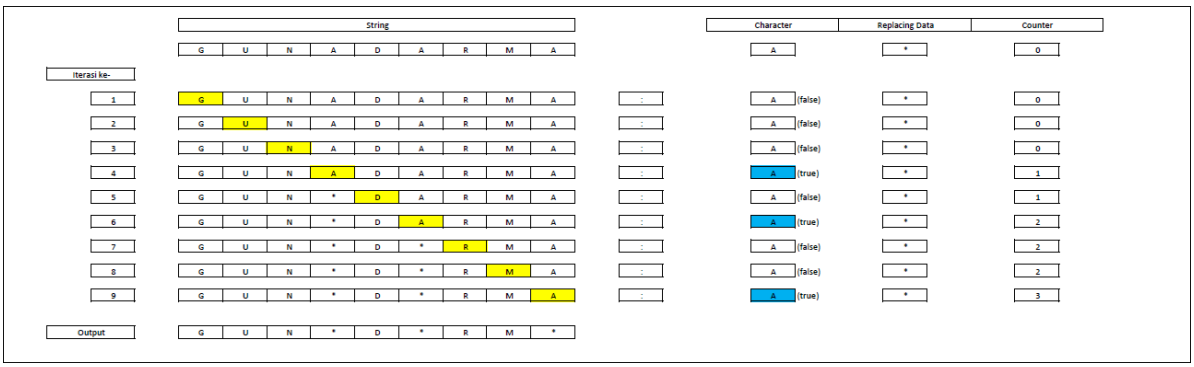
Digunakan untuk menghitung karakter dari string tertentu dan menggantikan suatu karakter dalam string dengan menggunakan karakter dalam string tertentu.

Bentuk umum :

```
INSPECT input-string
TALLYING output-count FOR ALL CHARACTERS
REPLACING ALL char1 BY char2
```

Dimana,

- Input-string : String yang karakternya akan dihitung dan diganti
- Output-count : Data item untuk menampung jumlah karakter



8.3 PERNYATAAN STRING

Pernyataan ini digunakan untuk menggabungkan *(concate)* dua atau lebih karakter string menjadi string yang lebih panjang. Pernyataan ini melakukan beberapan tugas dari pernyataan *MOVE* karena, penggunaan pernyataan ini sama saja memindahkan karakter string ke string lainnya dengan pergerakan dari kiri ke kanan. Maka dari itu, satu pernyataan string dapat ditulis

sebagai pengganti serangkaian pernyataan *MOVE*.

Bentuk umum :

```
        STRING string1 DELIMITED BY SPACE
        string2 DELIMITED BY SIZE
        INTO string-tujuan
        WITH POINTER counter-str
        ON OVERFLOW DISPLAY message1
        NOT ON OVERFLOW DISPLAY message2
        END-STRING.
```

Penjelasan :

String yang berada pada string-1 akan dipindahkan ke string tujuan yaitu string-tujuan. Dimulai dari karakter string-1 paling kiri dipindahkan ke posisi paling kiri dari string tujuan, begitu seterusnya. Perpindahan ini akan selesai jika string sudah habis atau dijumpainya pembatas (string-2) pada string tersebut. Dan operasi string akan selesai jika semua string sudah dipindahkan. Dan string tujuan penuh.

Perlu diketahui bahwa pernyatan ini hanya dapat menggabungkan item data berupa alfabetik dan alfanumerik. Dan tidak dapat melakukan operasi jika berupa numerik atau floating point. Terdapat beberapa penggunaan klausa yang bersifat optional yaitu D*ELIMITED BY, WITH POINTER, ON OVERFLOW, dan NOT ON OVERFLOW*. Sedangkan *END-STRING* digunakan sebagai terminator.

Terdapat beberapa peraturan dalam operasi string yaitu :

1. DELIMITED BY

   Digunakan untuk mengatur batas string selama operasi penggabungan. Bersifat optional dan terdapat tig acara penetapannya, yaitu :

   - SIZE

     DELIMITED BY SIZE digunakan dalam operasi penggabungan, dimana Klausa *DELIMITED BY SIZE* akan menyebabkan semua string sumber akan dipindahkan ke string tujuan

   - SPACE

     Sesudah penggunaan klausa *DELIMITED BY* dapat menggunakan literal, maka dapat menggunakan *figurative constant* (seperti *SPACES*) kecuali untuk *ALL literal figurative constant*. Ketika *figurative constant* digunakan berarti satu ukuran karakter.

   - Specified delimeter :

     String akan digabungkan dengan data sebagai pembatas ditemukan *(identifier)*

2. WITH POINTER

- Penggunaan optional ini untuk menunjukkan posisi pertama dari string tujuan.
- Klausa *ON OVERFLOW* dijalankan jika karakter pada string awal tetap harus di pindahkan ke string tujuan tetapi pada string tujuan sudah penuh.
- Ketika frasa *WITH POINTER* digunakan, nilainya menentukan posisi karakter awal untuk dimasukkan ke dalam string tujuan. Karena setiap karakter dimasukkan ke string tujuan, maka pointer bertambah. Ketika pointer menunjuk di luar ujung string tujuan, pernyataan string berhenti.
- Ketika frasa *WITH POINTER* digunakan, maka sebelum pernyataan string dijalankan, program harus mengatur nilai awal pointer yaitu pada counter-str, dimana nilainya lebih besar dari nol dan kurang dari panjang string tujuan.
- Jika frasa *WITH POINTER* tidak digunakan, maka operasi dimulai dari posisi paling kiri

- Nilai pointer harus berupa integer, dan nilainya harus lebih besar dari ukuran string tujuan. Misalnya, pointer dinyatakan dengan PIC 9, maka akan terlalu kecil jika string tujuan memiliki panjang 10 karakter (PIC (10)).

3. ON OVERFLOW

- Digunakan untuk menjalankan *imperative statement* ketika *overflow* terjadi pada selama operasi penggabungan string
- Dijalankan jika pointer bernilai (implisit atau eksplisit) :
    - Lebih kecil dari 1
    - Melebihi nilai ukuran string penerima

4. NOT ON OVERFLOW

   Digunakan untuk menjalankan pernyataan imperative statement ketika operasi penggabungan string berhasil. Dan merupakan kebalikan dari *ON OVERFLOW*

Pernyataan STRING berakhir jika memenuhi salah satu kondisi berikut, yaitu :

- Semua string sumber sudah diproses
- String tujuan sudah penuh
- Pointer menunjuk keluar string

8.4 PERNYATAAN UNSTRING

Pernyataan unstring digunakan untuk memisahkan sebuah string menjadi sub strings. Unstring menyalin karakter dari string sumber ke string tujuan hingaa terjadi kondisi yang menghentikan perpindahan data. Ketika pergerakan data berakhir untuk string tujuan tertentu, string tujuan berikutnya menjadi area penerima, dan karakter disalin ke dalamnya sampai sekali lagi kondisi penghentian ditemui. Karakter disalin dari string sumber ke string tujuan sesuai dengan aturan pernyataan *MOVE*, dengan mengisi ruang atau pemotongan yang diperlukan.

Bentuk umum inline perform until :

```
        UNSTRING string1 DELIMITED BY SPACE
        INTO sub-string2, sub-string2
        WITH POINTER counter-str
        ON OVERFLOW DISPLAY message
        NOT ON OVERFLOW DISPLAY message
        END-UNSTRING.
```

Dalam pengoperasiannya pernyataan ini membutuhkan minimal dua tempat *(data name)* tujuan untuk melakukan operasi pemisahan string. Dan memisahkan data yang berupa alfanumerik dan alfabetik. Untuk *figurative constant* hanya mengizinkan *SPACE*. Terdapat beberapa klausa optional dalam pendeklarasian pernyataan unstring diantaranya *DELIMITED BY, TALLYING, WITH POINTER, ON OVERFLOW,* dan *NOT ON OVERFLOW*. Dan *END-UNSTRING* sebagai terminator.

Pernyataan unstring dapat dipengaruhi penggunaan dari sejumlah klausa optional. Pengaruh penggunaannya sebagai berikut :

- DELIMITED BY :

  DELIMITED BY digunakan untuk mengontrol perpindahan data, dan bersifat optional dalam pernyataan unstring. Delimeter dapat berupa dua atau lebih karakter tetapi karakter tersebut harus berdekata *(contagious)* atau harus berurutan *(sequence)*. Ketika dua atau lebih delimeter digunakan, akan terjadi kondisi *OR*. Misalnya :

```
        DELIMITED BY "AB" or "BC"
```

Baik AB atau BC disebut sebagai delimeter.

- WITH POINTER :

  Penggunaan klausa ini untuk menunjukkan delimeter karakter berikutnya dalam string sumber. Jika menggunakannya maka harus memberikan nilai yang lebih besar dari nol atau harus cukup besar untuk menampung nilai yang lebih besar dari ukuran string sumber, karena ketika unstring berakhir, maka akan menunjuk ke satu posisi karakter di luar ujung string.

- ON OVERFLOW :

  Digunakan untuk mengeksekusi pernyataan *imperative* ketika terjadinya *overflow* selama operasi unstring. Ketika kondisi ini terjadi, tidak ada lagi data yang dapat dikirim ke string tujuan dan operasi unstring dihentikan. *Overflow* terjadi jika :

  - Nilai pointer (implisit maupun ekspilisit) kurang dari 1
  - Nilai pointer (implisit maupun eksplisit) melebihi nilai yang sama dengan Panjang string sumber
  - Semua string tujuan sudah diproses, tetapi masih terdapat karakter yang belum diperiksa dalam string sumber

- NOT ON OVERFLOW :

  Pernyataan ini dieksekusi ketika pernyataan unstring berhasil dijalankan. Dan merupakan kebalikan dari pernyataan *ON OVERFLOW*.

Pernyataan unstring akan berhenti ketika terjadi keadaan berikut :

- Semua karakter pada string sumber sudah diujikan
- Semua string tujuan sudah diproses
- Terjadinya error (pinter menunjuk ke luaran dari string sumber)