

***COBOL***  
***ANSI-85 e X/Open***

## A Convenção ANSI de 1985

Assim como a linguagem **SQL** para gerenciadores de bancos de dados e da "**C**" para software básico, o **COBOL** não é propriedade de nenhuma empresa por isso as suas regras de sintaxe foram normalizadas pelo **ANSI** (**American National Standards Institute**) em 1985 e foi liberado um novo padrão sintático pouco conhecido em micros no Brasil por motivos já mencionados.

Desde o começo, em 1959 o **ANSI**, determinou muitas restrições ao **COBOL**, com a intenção de obter programas legíveis por administradores temerosos de ter todos os processos de suas empresas dominados por cientistas de computação eletrônica da época e pelos grandes fabricantes de hardware.

Estas restrições são baseadas no princípio de que para maior legibilidade só deverão ser admitidos como comandos, palavras oriundas da língua inglesa, mas isso gerou dificuldades, já que para simplificar a lógica de programação, seriam necessárias palavras como **END-IF** e **END-PERFORM** que não existem na língua inglesa. Assim, tanto os críticos como os defensores da linguagem compartilham da opinião que isso é um exagero pois o mercado de programadores profissionais está suficientemente maduro para tranquilizar os administradores.

O comitê do **ANSI** de 1985 foi sensível a estes fatos reduzindo as restrições e permitindo que sentenças **COBOL** possuam palavras compostas separadas por hífen "-", tornando a linguagem **COBOL** mais amigável aos programadores sem perder a legibilidade.

O **ANSI** esta transferiu a responsabilidade de normatizar a linguagem **COBOL** para o **ISO** (**International Standards Organization**) isso significa que a linguagem se difundiu tanto pelo mundo que esta se transformando em padrão mundial, esperamos para breve o novo padrão **COBOL ISO 97** com normatização para interface gráfica, tratamento de mouse e programação orientada à objetos. Podemos esperar uma vida bem longa para o **COBOL**.

# Redução de restrições

O Comitê passou a aceitar palavras inglesas compostas com o prefixo **END** para fechar o escopo de instruções em alternativa ao ponto que continua fechando o escopo de todas as instruções.

## Formato genérico;

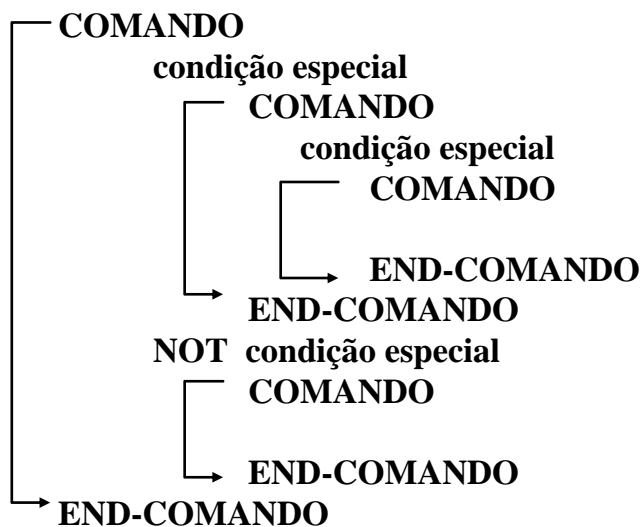


Condições especiais passam a aceitar opção negativa e comandos condicionais desta forma podemos encapsular comandos uns nos outros.

## Formato genérico;



## Ilustração genérica;



Com estes novos recursos podemos eliminar a necessidade de criar parágrafos inconvenientes sem perder legibilidade, a utilização de parágrafos passa a ter efeito de documentação ou para declarar rotinas reutilizáveis.

# Novas opções de tratamento de dados

## 1 - Manipulação de literais em hexadecimal;

Foi incluído um delimitador de literais adicional, se a literal for especificada com a letra **X** precedendo a primeira aspa o conteúdo entre as aspas será tratado em modo hexadecimal.

### Exemplos:

```
05 SALTA-LINHA                                PIC X(002) VALUE X"0D0A".

IF  BYTE-CONTROLE = X"0C"
   PERFORM SALTAR-PAGINA.

MOVE  X"0F"          TO COMPRIME-IMPRESSORA
WRITE PRINTER-REG FROM X"12"
```

## 2 - Campos usage decimal-ASCII (COMP-X);

Com este novo tipo de **USAGE** é possível criar variáveis que tratam caracteres da cadeia ASCII como valores numéricos inteiros, será necessário criar dois dígitos numéricos para cada byte, como um byte pode assumir valores de 0 a 255 o truncamento de dois dígitos só irá ocorrer com a tentativa de atribuição a partir de 256 enquanto um campo numérico de 2 dígitos em **USAGE DISPLAY** truncaria em 100.

### Exemplo:

```
05 LETRA-A-MAIUSCULA          PIC X(001) VALUE "A".
05 A REDEFINES LETRA-A-MAIUSCULA PIC 9(002) COMP-X.
05 LETRA-A-MINUSCULA          PIC 9(002) COMP-X.

ADD 32 TO LETRA-A-MAIUSCULA
   GIVING LETRA-A-MINUSCULA
```

Como no código ASCII a letra A maiúscula é representada pelo valor decimal 65, ao se somar 32 em sua redefinição em **USAGE COMP-X**, o resultado obtido será 97 que representa a letra a em minúsculas.

### 3 - Referência posicional;

É como se fosse uma redefinição de referências em tempo de execução, para isso basta declarar entre parênteses a posição inicial e o número de bytes que desejamos mover (origem ou destino).

```
MOVE ORIGEM (INICIO-1: TAMANHO-1)
  TO DESTINO (INICIO-2: TAMANHO-2)
```

Sendo que INICIO-1, INICIO-2, TAMANHO-1 e TAMANHO-2 podem ser variáveis inteiras ou constantes numéricas também inteiras.

#### Exemplo:

Supondo que precisamos extrair o mês de uma data:

##### ANSI 74:

```
01 EMISSAO PIC 9(006)
01 FILLER REDEFINES EMISSAO.
   05 DIA PIC 9(002).
   05 MES PIC 9(002).
   05 ANO PIC 9(002).

MOVE MES TO MES-E
```

##### ANSI 85:

```
01 EMISSAO PIC 9(006)

MOVE EMISSAO (3: 2) TO MES-E
```

### 4 - Movimentação reversa de campos editados;

Campos editados continuam só podendo participar de operações aritméticas como resultado, mas a restrição de só poder ser referenciada como campo de destino, no comando **MOVE**, foi removida.

#### Exemplo:

Se for declarada uma movimentação de um campo de valor editado (numérico, com decimais, sinalizado e com supressão de zeros não significativos) para um campo de valor decimal compactado, será realizada uma “de-edição”.

```
05 VALOR-E PIC ZZ.ZZ9,99+.
05 VALOR PIC S9(5)V99 COMP-3.

MOVE VALOR-E TO VALOR.
```

# Comandos estendidos

Neste módulo, vamos estudar os principais comandos modificados pelo comitê **ANSI** de 1985 na linguagem **COBOL**. A forma **ANSI** de 1974 continua sendo válida, porém opcional.

## 1 - Operações de leitura e gravação

(**DELETE**, **READ**, **REWRITE**, **START** e **WRITE**);

O comando **READ** incorpora a negativa da condição de fim de arquivo (**NOT AT END**) e todos passam a aceitar a negativa da condição de chave inválida (**NOT INVALID KEY**) e a ter o escopo delimitado por **END-DELETE**, **END-READ**, **END-REWRITE**, **END-START** e **END-WRITE** respectivamente.

### Exemplo:

#### ANSI 74:

```
WRITE REGISTRO
      INVALID KEY
          PERFORM ERRO-DE-GRAVACAO
          GO TO NAO-GRAVA.

      PERFORM PROCESSOS.

NAO-GRAVA.
```

#### ANSI 85:

```
WRITE REGISTRO
      INVALID KEY
          PERFORM ERRO-DE-GRAVACAO
      NOT INVALID KEY
          PERFORM PROCESSOS
END-WRITE
```

## 2 - Operações aritméticas

(**ADD**, **COMPUTE**, **DIVIDE**, **SUBTRACT**, e **MULTIPLY**);

Todos passam a aceitar a negativa da condição de estouro de campo (**NOT ON SIZE ERROR**) e a ter o escopo delimitado por **END-ADD**, **END-COMPUTE**, **END-DIVIDE**, **END-SUBTRACT** e **END-MULTIPLY** respectivamente.

## Exemplo:

### ANSI 74:

```
ADD 1 TO CAMPO
ON SIZE ERROR
    PERFORM ERRO-DE-SOMA
    GO TO NAO-SOMOU.
PERFORM PROCESSOS.

NAO-SOMOU.
```

### ANSI 85:

```
┌ ADD 1 TO CAMPO
│ ON SIZE ERROR
│     PERFORM ERRO-DE-SOMA
│ NOT ON SIZE ERROR
│     PERFORM PROCESSOS
└→ END-ADD
```

## 3 - IF

Passa a ter o escopo de instruções condicionais delimitado não apenas por um ponto, mas também pela nova cláusula **END-IF**.

## Exemplo:

```
┌ IF CONDIÇÃO-1
│ INSTRUÇÕES-1
│ ┌ IF CONDIÇÃO-2
│ │ INSTRUÇÕES-2
│ │ ELSE
│ │ ┌ IF CONDIÇÃO-3
│ │ │ INSTRUÇÕES-
│ │ │ ELSE
│ │ │ INSTRUÇÕES-
│ │ └→ END-IF
│ └→ INSTRUÇÕES-5
└→ END-IF
INSTRUÇÕES-6
└→ END-IF
```

## 4 - CALL

Incorpora em complemento a cláusula **USING**, novas cláusulas opcionais:

### 4.1 - BY REFERENCE

Libera a subrotina chamada para manipulação total do item referenciado, sendo a especificação default, assim como no ANSI 74 quando era opção imperativa não declarada.

### 4.2 - BY CONTENT

Não permite que a subrotina chamada mude o conteúdo do item referenciado.

### 4.3 - BY VALUE

Trata o item referenciado como uma literal.

### 4.4 - (NOT) ON [EXCEPTION|OVERFLOW]

Permite a declaração de procedimentos condicionados ao sucesso ou não na chamada da subrotina.

### 4.5 - END-CALL

Delimita o fim do escopo da cláusula **(NOT) ON EXCEPTION**.

### Exemplo:

```
CALL "EXTENSO" USING BY REFERENCE VALOR-FRASE
                     BY CONTENT   VALOR-NUMERICO
                     BY VALUE     "REAL"
                     BY VALUE     "REAIS"
    ON EXCEPTION
        DISPLAY "Rotina de extenso não disponível"
                LINE 23 COLUMN 3 WITH BEEP SIZE 70
        GOBACK
    NOT ON EXCEPTION
        MOVE VALOR-FRASE TO VALOR-FRASE-NOTA-FISCAL
    END-CALL
```



## 5 - PERFORM

Recebe duas novas cláusulas:

### 5.1 - END-PERFORM

Elimina a necessidade da definição de um parágrafo para criar um bloco de instruções repetitivas. A cláusula **END-PERFORM** delimita o escopo de instruções sob controle do último **PERFORM**.

**Exemplo:**

#### ANSI 74:

```
PERFORM LEITURA THRU FIM-LEITURA
                        UNTIL FS-ARQUIVO = "10"

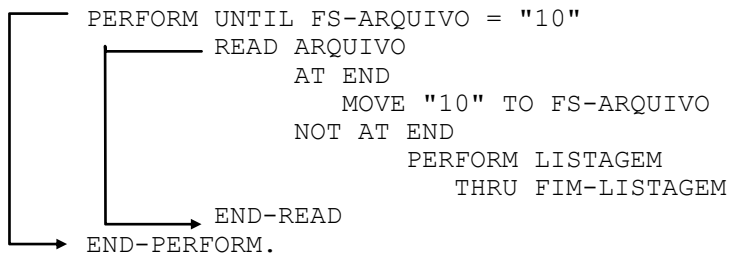
LEITURA.

READ ARQUIVO
  AT END MOVE "10" TO FS-ARQUIVO.

PERFORM LISTAGEM THRU FIM-LISTAGEM.

FIM-LEITURA. EXIT.
```

#### ANSI 85:



```
graph TD
    Start(( )) --> Perform[PERFORM UNTIL FS-ARQUIVO = "10"]
    Perform --> Read[READ ARQUIVO]
    Read --> AtEnd[AT END]
    AtEnd --> Move[MOVE "10" TO FS-ARQUIVO]
    Move --> NotAtEnd[NOT AT END]
    NotAtEnd --> PerformList[PERFORM LISTAGEM  
THRU FIM-LISTAGEM]
    PerformList --> EndRead[END-READ]
    EndRead --> EndPerform[END-PERFORM.]
    EndPerform --> Start
```

```
PERFORM UNTIL FS-ARQUIVO = "10"
  READ ARQUIVO
  AT END
    MOVE "10" TO FS-ARQUIVO
  NOT AT END
    PERFORM LISTAGEM
    THRU FIM-LISTAGEM
  END-READ
END-PERFORM.
```

### 5.2 - WITH TEST AFTER UNTIL condição (BEFORE)

Deve ser usado para alterar a seqüência lógica durante os testes de validade das condições que limitam a repetição do bloco de instruções ou parágrafo.

## 6 - SET

Permite que uma condição (Nível 88) seja forçada para verdadeira durante o processamento, de forma a tornar a **PROCEDURE DIVISION** independente da **DATA DIVISION**.

### Exemplo:

#### Definição da condição:

```
05 ESTADO-CIVIL PIC 9(002).  
88 SOLTEIRO VALUE 0.  
88 CASADO VALUE 1.  
88 VIUVO VALUE 2.  
88 DESQUITADO VALUE 3.  
88 DIVORCIADO VALUE 4.
```

#### ANSI 74:

```
MOVE 1 TO ESTADO-CIVIL.
```

#### ANSI 85:

```
SET CASADO TO TRUE.
```

# Novos comandos

Neste módulo vamos estudar os principais comandos homologados pelo comitê ANSI de 1985 para a linguagem **COBOL**.

## 1 - CANCEL

Eliminar módulos externos desnecessários para a continuidade do processamento.

### Exemplo:

```
CALL    "ROTINA"  
CANCEL "ROTINA"
```

## 2 - CONTINUE

Preencher lacunas de codificação "vazia".

### Exemplo:

Supondo que desejamos descobrir o tamanho da string contida na variável **CAMPO**.

```
01  CAMPO    PIC X(80) VALUE "COBOL ANSI".  
01  TAMANHO  PIC 9(02) VALUE 0.  
  
  PERFORM VARYING TAMANHO  
    FROM 80 BY -1  
    UNTIL CAMPO (TAMANHO: 1) NOT = SPACES  
      CONTINUE  
  END-PERFORM
```

Como o valor final da variável `TAMANHO` atende a nossa necessidade sem nenhum comando, já que tanto a comparação quanto o decremento estão codificados através de cláusulas do próprio comando **PERFORM** deveremos preencher esta lacuna com o comando **CONTINUE**.

### 3 - EVALUATE

Criado para simplificar a codificação e leitura de múltiplas decisões, resolvidas até então, com complexos ninhos de **IFs**.

### Exemplo:

## ANSI 74:

```
IF      OPCA0 = 1  
        PERFORM LISTA-PEDIDO THRU FIM-LISTA-PEDIDO  
ELSE  
    IF      OPCA0 = 2  
        PERFORM LISTA-NOTA THRU FIM-LISTA-NOTA  
    ELSE  
        IF      OPCA0 = 3  
            PERFORM LISTA-CLIENTE  
                THRU FIM-LISTA-CLIENTE  
        ELSE  
            IF      OPCA0 = 4  
                PERFORM LISTA-PRODUTO  
                    THRU FIM-LISTA-PRODUTO.
```

## ANSI 85:

```

EVALUATE OPCAO
  WHEN 1 PERFORM LISTA-PEDIDO THRU FIM-LISTA-PEDIDO
  WHEN 2 PERFORM LISTA-NOTA THRU FIM-LISTA-NOTA
  WHEN 3 PERFORM LISTA-CLIENTE THRU FIM-LISTA-CLIENTE
  WHEN 4 PERFORM LISTA-PRODUTO THRU FIM-LISTA-PRODUTO
END-EVALUATE

EVALUATE TRUE
  WHEN OPCAO = 1 PERFORM LISTA-PEDIDO THRU FIM-LISTA-PEDIDO
  WHEN OPCAO = 2 PERFORM LISTA-NOTA THRU FIM-LISTA-NOTA
  WHEN OPCAO = 3
    AND TESTE = "OK"
    PERFORM LISTA-CLIENTE THRU FIM-LISTA-CLIENTE
  WHEN OPCAO = 4 PERFORM LISTA-PRODUTO THRU FIM-LISTA-PRODUTO
  WHEN OPCAO = 5
    CONTINUE
  WHEN OTHER
    DISPLAY "Opção inválida" LINE 23 COLUMN 03
END-EVALUATE

```

## 4 - INITIALIZE

Elimina as freqüentes rotinas de inicialização de registros.

### Exemplo:

#### Área a inicializar:

```
01  REGISTRO.  
   05  CODIGO          PIC  9(006) .  
   05  DESCRICAO       PIC  X(030) .  
   05  PRECO           PIC  9(007)V99 COMP-3.  
   05  PRECO-EM-DOLAR  PIC  9(005)V99 .  
   05  ESTOQUES-NO-ANO.  
       10  QUANTIDADE   PIC  S9(005)V9(4) COMP-3  
                           OCCURS 12.
```

#### ANSI 74:

```
MOVE 0      TO  CODIGO  
                PRECO  
                PRECO-EM-DOLAR  
                QUANTIDADE (1)  
                QUANTIDADE (2)  
                QUANTIDADE (3)  
                QUANTIDADE (4)  
                QUANTIDADE (5)  
                QUANTIDADE (6)  
                QUANTIDADE (7)  
                QUANTIDADE (8)  
                QUANTIDADE (9)  
                QUANTIDADE (10)  
                QUANTIDADE (11)  
                QUANTIDADE (12)  
MOVE SPACES TO DESCRICAO
```

#### ANSI 85:

```
INITIALIZE REGISTRO.
```

# Considerações X/Open

Os comandos **X/Open** não são homologados pelo **ANSI**, contudo funcionam plenamente em todos os compiladores **COBOL** para arquiteturas abertas, pois a **X/Open Company Limited** é uma entidade privada de padronização do **UNIX** e suas especificações são implementadas por todas as companhias a produtoras de compiladores **COBOL** para **UNIX** que também os produzem para **DOS**, **Windows** e **OS/2**, logo, podem ser utilizados com um excelente nível de portabilidade.

## 1 - GOBACK

Serve de sinônimo aos comandos **EXIT PROGRAM** e **STOP RUN**.

Este comando é válido como:

**EXIT PROGRAM** quando executado em um sub-programa.

**STOP RUN** quando executado por um programa principal.

## 2 - “Split-keys” (Chaves concatenadas)

Permite a declaração de chaves alternativas compostas por campos descontinuos da **FILE DESCRIPTION (FD)**.

**Exemplo:**

```
SELECT TITULOS ASSIGN          TO DISK
      ORGANIZATION             IS INDEXED
      RECORD KEY               IS TITULOS-CHAVE
      ALTERNATE RECORD KEY IS TITULOS-BANCO-VENCIMENTO =
                              TITULOS-BANCO
                              TITULOS-VENCIMENTO
                              WITH DUPLICATES
      FILE STATUS              IS FS-TITULOS.

FD  TITULOS
   VALUE OF FILE-ID LB-TITULOS.

01  TITULOS-REG.
    05 TITULOS-CHAVE.
        10 TITULOS-DUPLICATA COMP-3 PIC 9(005).
        10 TITULOS-SERIE      PIC X(002).
    05 TITULOS-VALOR          PIC S9(009)V99.
    05 TITULOS-VENCIMENTO     COMP-3 PIC 9(008).
    05 TITULOS-BANCO          PIC 9(003).

MOVE 237          TO TITULOS-BANCO
MOVE 19961006     TO TITULOS-VENCIMENTO
START TITULOS KEY NOT LESS TITULOS-BANCO-VENCIMENTO.
```

### 3 - Ambiente multi-usuário

Existem três tipos de tratamentos para travamento de arquivos ou registros em ambiente multi-usuário, **AUTOMATIC**, **EXCLUSIVE** e **MANUAL**.

#### Formato genérico:

```
SELECT FILENAME ASSIGN TO DISK
      ORGANIZATION      IS INDEXED
      RECORD KEY        IS FILENAME-CHAVE
      LOCK MODE          IS [AUTOMATIC|EXCLUSIVE|MANUAL]
                        WITH [LOCK ON MULTIPLE RECORD(S) | ROLLBACK]
      FILE STATUS        IS FS-FILENAME.
```

#### **AUTOMATIC:**

Opção default para arquivos abertos como **I-O**, todo o registro lido será travado.

#### **EXCLUSIVE:**

Opção default para arquivos abertos como **OUTPUT**, o arquivo inteiro será de uso exclusivo para o primeiro programa que o abrir.

#### **MANUAL:**

O registro será travado apenas quando necessário através da cláusula **WITH LOCK** do comando **READ**.

#### **LOCK ON MULTIPLE RECORD(S):**

O registro continuara travado mesmo quando outro for lido, o comando **UNLOCK FileName** destravará todos os registros ao mesmo tempo.

#### **ROLLBACK:**

Permite processamento transacional para o arquivo, os pontos de integridade ou fim de transação, podem ser marcados com o comando **COMMIT**, caso o sistema caia entre um **COMMIT** e outro, todas as alterações no arquivo serão desconsideradas, para o uso efetivo deste recurso, será necessária a instalação de produtos de apoio como o **ACUSERVER** para o **ACUCOBOL**, o **FILESHARE** para o **MICRO FOCUS COBOL** e o **RM/Info Express** para o **RM/COBOL-85**.

#### **FILE STATUS:**

Se durante a abertura ou leitura o arquivo ou registro estiver travado, o **FILE STATUS** retornará o código correspondente.

## 4 - SCREEN SECTION

A **SCREEN SECTION** é uma excelente opção para o tratamento de telas, ao utiliza-la evitamos que a **PROCEDURE DIVISION** fique repleta de endereços de **DISPLAY** e **ACCEPT** que aumentam os custos de manutenção pois dificultam a modificação da apresentação.

Estudaremos a definição de telas no módulo de utilitários do **COBOLware**, utilitário **CTAC** (Codificador de Telas de Aplicação para COBOL).