



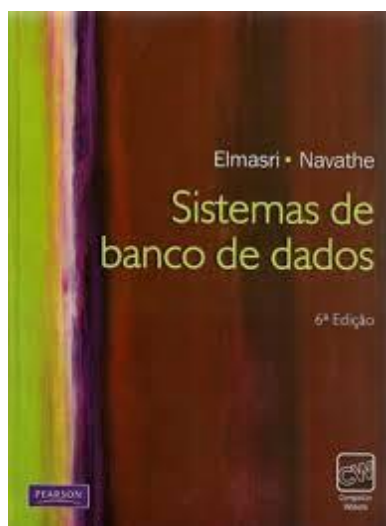
Unidade 17 – Programação SQL Embutida Estrutura de Programas Cobol



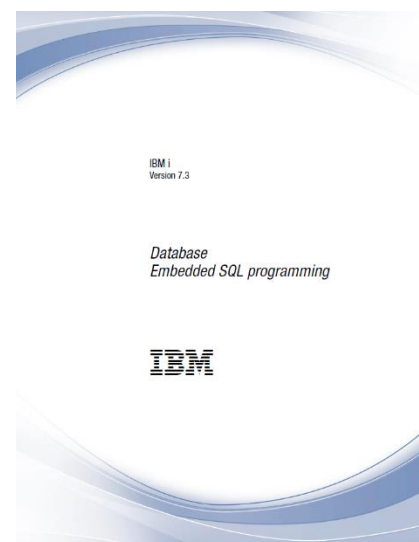
Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUVSP



Bibliografia



Sistemas de Banco de Dados
Elmasri / Navathe 6ª edição



Database Embedded SQL
Programming – IBM i



Bibliografia Adicional

- ✓ Programação Estruturada em COBOL – 9ª. Edição – Stern & Stern – LTC – 2002
- ✓ Manual de Cobol Estruturado – McCracken – Editora Campus
- ✓ Programming in Cobol/400 – Cooper, Stern & Stern – 2001- John Wiley & Sons
- ✓ Structured Cobol for the AS/400 – Cooper, Stern & Stern - 2003
- ✓ IBM - ILE Cobol for AS/400 – Reference Sumary
- ✓ IBM – ILE Cobol for AS/400 – Programmer's Guide
- ✓ IBM – ILE Cobol for AS/400 – Reference's Guide





Estrutura Básica de um programa Cobol

- ✓ Cada instrução Cobol é codificada em uma linha de 80 colunas;
- ✓ Há posições na linha reservadas para determinadas finalidades;
- ✓ Regras podem variar em diferentes compiladores.





Regras de Codificação

- ✓ Colunas de **1** a **6** e de **73** a **80** são opcionais e raramente usadas;
- ✓ Coluna **7** é usada para comentário e para continuação;
- ✓ Comandos da linguagem Cobol são codificados entre as colunas **8** e **72**.





Coluna 7

| | |
|---|---|
| * | Para a definição de comentários |
| / | Força quebra de página durante a impressão da listagem do código fonte. |
| - | Indica a continuação de um literal não-numérico. |





Regras de Margem

✓ Colunas **8-72** divididas em 2 áreas:

=> AREA **A** => Colunas 8,9,10 e 11

=> AREA **B** => Colunas 12 a 72





Margem A

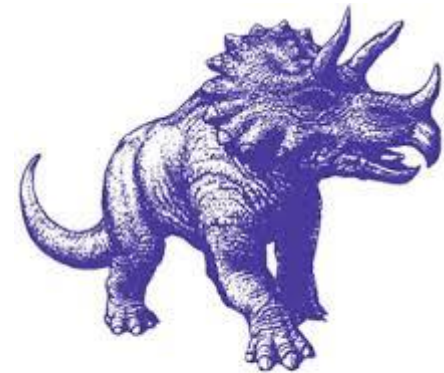
- ✓ **DIVISIONS, SECTIONS** e **PARAGRAPH-NAMES** devem iniciar na AREA **A**.
- ✓ Primeira letra do nome deve iniciar nas colunas **8,9, 10** ou **11**.
- ✓ Entrada pode se estender na AREA B.





Margem B

- ✓ Todos os outros comandos, cláusulas e sentenças devem se iniciar na AREA **B** (colunas 12, 13, 14, etc.)
- ✓ Entradas de **SELECT** da **ENVIRONMENT DIVISION**;
- ✓ Entradas de descrição de dados na **DATA DIVISION**;
- ✓ Todas as instruções da **PROCEDURE DIVISION**.

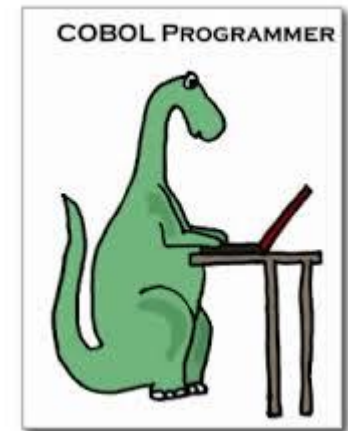


COBOL



IDENTIFICATION DIVISION

- ✓ Provê informação de identificação do programa;
- ✓ Dividida em parágrafos;
- ✓ Único parágrafo requerido **PROGRAM-ID**;
- ✓ Outros parágrafos são opcionais.





ENVIRONMENT DIVISION

- ✓ Descreve arquivos e dispositivos para processá-los;
- ✓ Requerido para programas que processam arquivos;
- ✓ Esta divisão é dependente de plataforma uma vez que dispositivos diferem de computador para computador.





Seções da Environment Division

✓ CONFIGURATION SECTION

- ✓ Descreve o computador usado para compilar/executar o programa;
- ✓ Opcional e recomendado que você a omita;

✓ INPUT-OUTPUT SECTION

- ✓ Descreve arquivos de entrada e saída e dispositivos usados pelo programa;
- ✓ Requerido para todos os programas que utilizam arquivos;



INPUT-OUTPUT SECTION

- ✓ Segue a **CONFIGURATION SECTION** (se codificada);
- ✓ Inclui o parágrafo **FILE-CONTROL**;
- ✓ Contém um statemet **SELECT** para cada arquivo usado pelo programa;
- ✓ Cada **SELECT** define um file-name e assinala um nome de dispositivo para aquele arquivo





Regras para palavras definidas pelo usuário

- ✓ 1 to 30 caracteres;
- ✓ Somente letras, dígitos e hífen (-);
- ✓ Sem espaços embutidos;
- ✓ Ao menos um caractere alfabético;
- ✓ Não pode começar ou encerrar com um hífen;
- ✓ Não pode ser uma palavra reservada COBOL.





DATA DIVISION

- ✓ Define e descreve dados em memória;

- ✓ Duas principais seções:

- **FILE SECTION**

- ✓ Define todos os arquivos de input e output, registros e campos;
- ✓ Requerido para todos os programas que usam arquivos;

- **WORKING-STORAGE SECTION**

- ✓ Define constantes, indicadores de fim-de-arquivo e áreas de trabalho;
- ✓ Define campos que não fazem parte dos arquivos de input e de output.





Entradas da File Description

- ✓ Cada arquivo é definido com uma sentença **FD** (File Descriptor);
- ✓ Um **FD** para cada statement SELECT na ENVIRONMENT DIVISION;
- ✓ **FD** seguido por
 - File-name
 - Cláusulas opcionais para descrever o arquivo e o formato de seus registros





Definindo um Registro

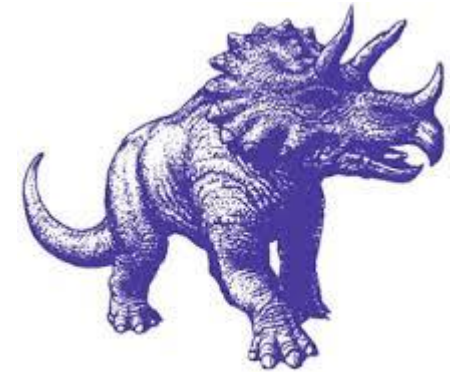
- ✓ Cada **FD** seguido por entradas de descrição do registro do arquivo;
- ✓ Dados agrupados no COBOL por níveis;
- ✓ Record-name definido no nível **01**. Considerado o mais alto nível de dado;
- ✓ Campos dentro de um registro definido em sub-níveis com números de nível de **02** a **49**.





Exemplo – Record Description

01 Empregado-Reg-In.
 05 Nome-In ...
 05 Salario-anual-In ...
 05 Descricao_Funcao-In ...



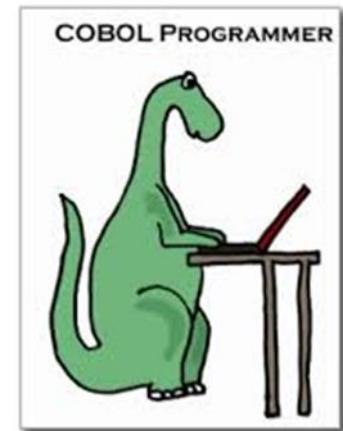
COBOL

- ✓ Campos no nível 05 estão subordinados à entrada de nível 01;
- ✓ Todos os campos ao mesmo nível (05), são independentes e não subordinados entre si.



SELECT, FD, 01

- ✓ **SELECT** atribue nomes a arquivos e os assinala à dispositivos de hardware;
- ✓ **FD** descreve arquivos;
- ✓ **01** define registros;
- ✓ **02 - 49** descreve campos dentro de um registro.





Ítems Elementares e ítems de Grupo

- ✓ Ítems definidos dentro de um número de nível podem ser de dois tipos:
 - Ítem Elementar – campo que não apresenta subdivisões
 - Deve incluir uma cláusula PICTURE
 - Ítem de Grupo – campo que apresenta sub-divisões
 - Não tem cláusula PICTURE



COBOL
ENGINEERING



Ítems Elementares e ítems de Grupo

01 Empregado-Rec-In.
 05 Nome-In.
 10 Primeiro-Nome-In (Picture clause)
 10 Ultimo-Nome-In (Picture clause)
 05 Salario-anual-In (Picture clause)

- ✓ **Nome-In** é ítem de grupo uma vez que é subdividido em primeiro e ultimo nome;
- ✓ **Empregado-Rec-In** é também ítem de grupo;
- ✓ **Primeiro-Nome-In** é ítem elementar uma vez que **não** apresenta sub-divisões.





Cláusulas PICTURE (PIC)

- ✓ Especificam o tipo de dado armazenado no campo;
- ✓ Indicam o tamanho do campo.





Tipos de campos de dados

✓ Alphabetic

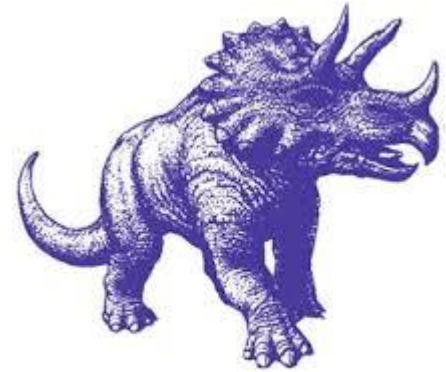
- Somente letras ou espaços;
- Para nomes, itens de descrição, etc.

✓ Alphanumeric

- Somente caracteres – letras, dígitos, caracteres especiais;
- Para um endereço como “Rua Brasil, 567”.

✓ Numeric

- Somente dígitos;
- Para campos empregados em operações aritméticas.

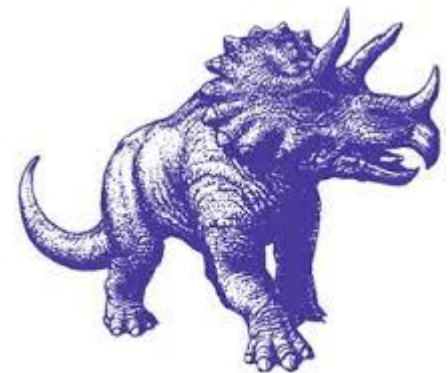


COBOL



Tipos de dados na cláusula PICTURE

- ✓ A para alphabetic
- ✓ X para alphanumeric
- ✓ 9 para numeric



COBOL



Tamanho dos campos de dados

O tamanho do campo é denotado por:

✓ Número de A's, X's or 9's usados na **PICTURE**

01 Custo-Reg-Entrada.

| | | |
|----|---------------|---------------|
| 05 | Custo-Entrada | PICTURE XXXX. |
|----|---------------|---------------|

| | | |
|----|--------------|----------------|
| 05 | Taxa-Entrada | PICTURE 99999. |
|----|--------------|----------------|



Tamanho dos campos de dados

Pode também ser denotado por:

- ✓ A, X or 9 seguido por um número entre parênteses:

01 Custo-Reg-Entrada.

05 Custo-Entrada PICTURE X(4).

05 Taxa-Entrada PICTURE 9(5).





Definindo Campos no Registro

- ✓ Deve contabilizar todas as posições definidas do layout do registro;
- ✓ Deve descrever os campos na ordem em que aparecem no registro;
- ✓ Nomes de campos devem ser únicos;
- ✓ Para campos não usados no programa:
 - Data-name pode ser deixado em branco (recomendável);
 - Pode-se usar a palavra reservada FILLER como um data-name;
 - Posições devem ainda ser definidas com uma cláusula PIC.





Alocação de memória

- ✓ Entradas na **DATA DIVISION** alocam memória para os dados;
- ✓ Entradas na **FILE SECTION** alocam memória para os dados dos registros em arquivos de entrada/saída.





Tipos de Constantes

- ✓ Literal Numérico
 - Exemplos: .05 5280 199.99
- ✓ Literal Não-numérico (alphanumeric)
 - Exemplos: “**INVALIDO**” “Entre com sua idade”
- ✓ Constante Figurativa
 - SPACES ZEROS





Regras para Literais Numéricos

- ✓ 1 a 18 dígitos.
- ✓ Sinal de + or – pode ser incluído a esquerda do primeiro dígito.
- ✓ Ponto Decimal permitido dentro do literal.

Literais numéricos válidos



23 +2359.4 .125 –68734





Regras para Literais Não-numéricos

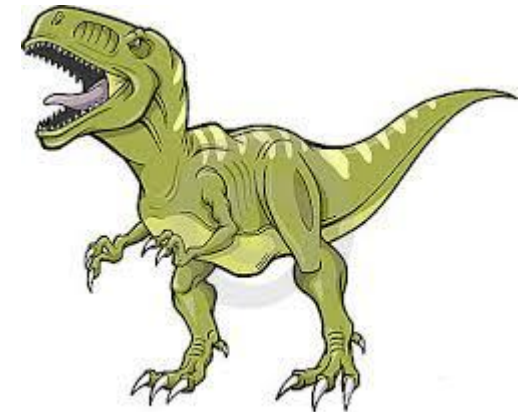
- ✓ Deve estar delimitador por aspas (“”).
- ✓ De 1 a 160 caracteres, incluindo espaço.

Literais Não-numéricos válidos



“Rua Chile, 70” “\$14.99”

“12,342” “Entre com um valor entre 1 e 5”





Constantes Figurativas – ZERO

- ✓ ZERO, ZEROS or ZEROES significam todos zeros

Exemplo:

MOVE ZEROS TO TOTAL-OUT



- ✓ Preenche cada posição em **TOTAL-OUT** com um zero;
- ✓ Pode ser usado tanto quanto campos numéricos quanto alfanuméricos.



Constantes Figurativas – SPACE

- ✓ SPACE ou SPACES significa espaços ou brancos;

Exemplo:

MOVE SPACES TO CODIGO



- ✓ Preenche cada posição em **CODIGO** com um espaço ou branco;
- ✓ Empregado somente com campos alfanuméricos uma vez que branco é um caractere numérico inválido.



WORKING-STORAGE SECTION

- ✓ Segue a FILE SECTION;
- ✓ Inicia na margem **A**, encerra com ponto;
- ✓ Todos os itens devem ser definidos no nível 01 ou em entradas subordinadas ao nível de entrada 01.





WORKING-STORAGE SECTION

- ✓ São aplicadas as regras para dados definidos pelo usuário;
- ✓ Ítems Elementares:
 - Devem incluir a cláusula **PICTURE**;
 - Podem ter um valor inicial definido pela cláusula **VALUE**.





Usos da WORKING-STORAGE



Para definir campos usados para:

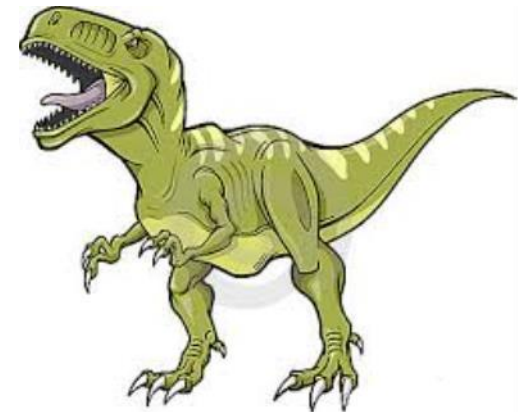
- Resultados aritméticos intermediários;
- Contadores e totalizações;





Cláusula VALUE

- ✓ Para definir valores iniciais de campos;
- ✓ Se omitidos, valores dos campos são indefinidos quando do início do programa;
- ✓ Podem somente ser usados na WORKING-STORAGE SECTION.





Cláusula VALUE

- ✓ Contém literal ou constante figurative;
- ✓ Tipo de dado deve ser compatível com PICTURE;
- ✓ Literais Numéricos ou ZEROS usados com campos PIC 9



| | |
|-------------|-----------------------------|
| 01 WS-TAXA | PIC V99 VALUE .06. |
| 01 WS-TOTAL | PIC 999 VALUE ZEROS. |



Cláusula VALUE

- ✓ Literais Não-numéricos, ZEROS or SPACES usados com campos PIC X:

```
01 WS-EOF      PIC X(3) VALUE "SIM".
```

```
01 WS-DESC     PIC X(8) VALUE SPACES.
```






Laboratório – 3

```
IDENTIFICATION DIVISION.  
*CURSO DE LINGUAGEM COBOL...  
PROGRAM-ID.  PGMCOB03.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
77  CAMPO-1  PIC X(30) VALUE "MENSAGEM PGMCOB03".  
PROCEDURE DIVISION.  
100-INICIO.  
      DISPLAY CAMPO-1.
```


Laboratório - 3



```

Session A - [24 x 80]
File Edit View Communication Actions Window Help
Columns . . . : 1 71 Edit AVFREITAS1/COBFONTES
SEU==> PGMCOB03
FMT CB .....-A+++B+++++*****
***** Beginning of data *****
0001.00 IDENTIFICATION DIVISION.
0002.00 PROGRAM-ID. PGMCOB03.
0003.00 DATA DIVISION.
0004.00 WORKING-STORAGE SECTION.
0005.00 77 CAMPO-1 PIC X(30) VALUE 'MENSAGEM PGMCOB03'.
0006.00 PROCEDURE DIVISION.
0007.00 100-INICIO.
0008.00 DISPLAY CAMPO-1.
***** End of data *****

F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle
F16=Repeat find F17=Repeat change F24=More keys
(C) COPYRIGHT IBM CORP. 1981, 2013.

MA a A 02/009
I902 - Session successfully started

```



Laboratório - 4

```
IDENTIFICATION DIVISION.  
*CURSO DE LINGUAGEM COBOL...  
PROGRAM-ID.  PGMCOB04.  
DATA DIVISION.  
WORKING-STORAGE SECTION.  
01  REGISTRO.  
    05  CAMPO1  PIC X(10)  VALUE  "MENSAGEM  ".  
    05  CAMPO2  PIC X(10)  VALUE  "  PGMCOB04  ".  
PROCEDURE DIVISION.  
100-INICIO.  
    DISPLAY  REGISTRO.
```



Laboratório - 4

```

Session A - [24 x 80]
File Edit View Communication Actions Window Help
Columns . . . : 1 71 Edit AVFREITAS1/COBFONTES
SEU==> PGMCOB04
FMT CB .....-A+++B+++++*****
***** Beginning of data *****
0001.00 IDENTIFICATION DIVISION.
0002.00 PROGRAM-ID. PGMCOB04.
0003.00 DATA DIVISION.
0004.00 WORKING-STORAGE SECTION.
0005.00 01 REGISTRO.
0006.00 05 CAMPO1 PIC X(10) VALUE 'MENSAGEM'.
0007.00 05 CAMPO2 PIC X(10) VALUE 'PGMCOB04'.
0008.00 PROCEDURE DIVISION.
0009.00 100-INICIO.
0010.00 DISPLAY REGISTRO.
***** End of data *****

F3=Exit F4=Prompt F5=Refresh F9=Retrieve F10=Cursor F11=Toggle
F16=Repeat find F17=Repeat change F24=More keys
(C) COPYRIGHT IBM CORP. 1981, 2013.

MA a A 02/009
I902 - Session successfully started

```



PROCEDURE DIVISION

- ✓ Contém instruções para:
 - Ler dados;
 - Processar dados;
 - Produzir saída;
 - Executar operações de fim-de-job.

- ✓ É dividida em parágrafos;

- ✓ Consiste de uma série de instruções para executar um conjunto específico de operações.





Regras para nome de parágrafo

- ✓ Codificado na Area **A**, seguido por um ponto;
- ✓ Seguem as regras para a formação de data-names exceto que podem ser formados unicamente por dígitos:
 - **1010, 1020, 1030**, etc. são nomes de parágrafos válidos;
- ✓ Devem ser únicos.





Comandos da Procedure Division

- ✓ Todos comandos devem ser codificados na Area **B**;
- ✓ Comandos começam com um verbo (**READ**, **MOVE**);
- ✓ **OPEN** - para abrir arquivos a serem processados;
- ✓ **PERFORM UNTIL ... END-PERFORM**
- ✓ **CLOSE** – para fechar arquivos ao encerrar o processamento;
- ✓ **STOP RUN** – para encerrar o programa.





Comando OPEN

- ✓ Torna arquivos disponíveis para processamento;
- ✓ Identifica quando os arquivos serão usados para input ou output.

FORMATO

OPEN { INPUT file-name-1 ...
OUTPUT file-name-2 ... }

- ✓ File-names usados devem aparecer em statement SELECT;
- ✓ Arquivo deve ser acessado com OPEN antes da leitura ou da gravação.



PERFORM UNTIL ... END-PERFORM

- **FORMATO**

PERFORM UNTIL condicao-1
comando-1 ...
[END-PERFORM]



- ✓ Repetidamente executa o(s) comando(s) entre **PERFORM UNTIL ... END-PERFORM** até a condição especificada na cláusula UNTIL for atendida;
- ✓ Num típico programa batch, as instruções se repetem até que não existam mais registros para serem processados;
- ✓ Quando a condição é atendida, o programa continua com o comando após **END-PERFORM**.



PERFORM UNTIL ... END-PERFORM

EXEMPLO



```
PERFORM UNTIL WS-More-Data = "NO"  
  READ Payroll-File  
  AT END  
    MOVE 'NO' To WS-More-Data  
  NOT AT END  
    PERFORM 200-Process-Record  
  END-READ  
END-PERFORM
```



Comando READ

- ✓ Lê um registro de um arquivo aberto para input;
- ✓ Transfere o registro lido para a área de memória de entrada;
- ✓ Torna um registro disponível por vez, não o arquivo inteiro;



READ file-name-1
AT END statement-1 ...
[NOT AT END statement-2 ...]
[END-READ]

- ✓ File-name deve aparecer no statement **SELECT**, na entrada **FD** e deve ser aberto previamente com um **OPEN**;
- ✓ **AT END** testa se há mais registros.



Comando READ

- ✓ Se não houver mais registros:
 - Executa statement(s) após **AT END**;
- ✓ Se houver mais registros:
 - Lê o próximo registro
 - Executa statement(s) após **NOT AT END**.





PERFORM Simples

FORMATO

PERFORM nome-de-parágrafo

- ✓ Para executar instruções em parágrafo separado ou módulo uma vez;
- ✓ Transfere o controle para o parágrafo especificado;
- ✓ Executa todas as instruções no parágrafo;
- ✓ Controle retorna para o statement que segue o PERFORM.



PERFORM Out-of-Line

FORMATO

PERFORM paragraph-name UNTIL condition

- ✓ Repete parágrafo até que a condição seja atendida;
- ✓ O controle é transferido para o parágrafo especificado;
- ✓ Executa as instruções no parágrafo e retorna.



PERFORM In-Line

FORMATO

PERFORM UNTIL condition
statement(s)
END-PERFORM

- ✓ Repete statement(s) até que a condição seja atendida;
- ✓ Statement(s) a serem executados estão in-line , e não em um parágrafo separado.



Comando CLOSE

FORMATO

CLOSE file-name-1 ...

- ✓ Fecha todos os arquivos abertos;
- ✓ Indica que os arquivos não são mais necessários para processamento;
- ✓ Libera arquivos e desativa dispositivos.



STOP RUN

- ✓ Termina o programa;
- ✓ Usualmente é a última instrução no módulo principal;
- ✓ Execução continua com próximo parágrafo se **STOP RUN** for omitido.



Estrutura de Seleção Pseudocódigo

IF condição

THEN

instruções para serem
executadas se a
condição for atendida

ELSE

Instruções para
serem executadas se
a condição não for
atendida

END-IF

Exemplo

IF X IS LESS THAN Y

THEN

ADD X TO Y

ELSE

SUBTRACT X FROM
Y

END-IF



Comando MOVE

FORMAT 1

MOVE identifier-1 TO identifier-2

- ✓ Copia o conteúdo de identifier-1 para identifier-2;
- ✓ identifier-1 é o campo emissor;
- ✓ identifier-2 é o campo receptor.



Comando MOVE

Exemplo:

MOVE A TO B

- ✓ Copia o conteúdo de A para B;
- ✓ A é o campo emissor;
- ✓ B é o campo receptor;
- ✓ Conteúdo de A permanece inalterado.



Verbos Aritméticos Básicos

- ✓ **ADD, SUBTRACT, MULTIPLY, DIVIDE**
- ✓ Todos requerem campos com
 - Cláusulas PICTURE numéricas;
 - Dados numéricos no momento da execução.



Comando COMPUTE

- ✓ Comando geral aritmético usando símbolos no lugar de verbos aritméticos

Formato

COMPUTE identifier-1 ... = $\left\{ \begin{array}{l} \text{arithmetic-exp-1} \\ \text{literal-1} \\ \text{identifier-2} \end{array} \right\}$

- ✓ Identificador à esquerda do sinal de igual recebe o valor da operação efetuada à direita do sinal de igual;



Exemplos COMPUTE

Assuma que X, Y e Z são campos numéricos

$$X = 9, Y = 4 \text{ e } Z = 12$$

Comando COMPUTE

Resultado

| | |
|-----------------------|--------|
| COMPUTE Z = X * Y | Z = 36 |
| COMPUTE X = Z - Y + 2 | X = 10 |
| COMPUTE X = Y | X = 4 |
| COMPUTE Z = Y ** 2 | Z = 16 |



Exemplo de COMPUTE

Assuma que X, Y e Z são campos numéricos

$$X = 6, Y = 18 \text{ e } Z = 5$$

Comando COMPUTE

Resultado

COMPUTE Z = Y / X + 3

Z = 6

COMPUTE Z = Y / (X + 3)

Z = 2

COMPUTE Y = Z + X * 10

Y = 65

COMPUTE Y = Z * X / 10

Y = 3



Comando IF

Formato

IF condition-1

[THEN

comando-imperativo-1 ...

[ELSE

comando-imperativo-2 ...]

[END-IF]



Comando IF – Exemplo

```
IF DISC-CODE = 1 THEN  
    MULTIPLY AMT BY .15 GIVING WS-DISCOUNT  
ELSE  
    MOVE 0 TO WS-DISCOUNT  
END-IF
```



Operadores Relacionais

Símbolos para condições relacionais:

| <u>Símbolo</u> | <u>Significado</u> |
|----------------|--------------------|
| < | é menor que |
| > | é maior que |
| = | é igual a |
| <= | menor ou igual a |
| >= | maior ou igual a |



Cláusula CONTINUE

- ✓ Usada para indicar que nenhuma operação deve ser executada quando uma condição for atendida.

IF AMT1 = AMT2

THEN

CONTINUE

ELSE

ADD 1 TO TOTAL

END-IF

{ Nenhuma operação executada se AMT1 = AMT2, continua com comandos após END-IF