

OBJETIVO DO CURSO

Oferecer um treinamento que possibilite aos interessados um conhecimento suficiente para desenvolver sistemas com base na linguagem COBOL.

Este treinamento destina-se aos interesses:

- Formação de programadores em linguagem COBOL;
- Visão de análise de sistemas baseados em COBOL;
- Base didática para aprofundar conhecimentos em programação;

Para tanto, é necessários que aos integrantes atendam aos seguintes pré-requisitos:

- Lógica de programação
- Operação de micro em ambiente Windows
- Visão crítica/interpretativa

CONTEÚDO PROGRAMÁTICO

- **CAPÍTULO 1:** Ambientação Mainframe
- **CAPÍTULO 2:** Visão Geral do COBOL
- **CAPÍTULO 3:** Estrutura do COBOL
- **CAPÍTULO 4:** DATA DIVISION + Prática I
- **CAPÍTULO 5:** Manipulação de Dados
- **CAPÍTULO 6:** O Comando PERFORM
- **CAPÍTULO 7:** Prática II (4 Aulas)
- **CAPÍTULO 8:** Matemática Básica
- **CAPÍTULO 9:** Equações
- **CAPÍTULO 10:** Elementos de Decisão
- **CAPÍTULO 11:** Chamando Programas
- **CAPÍTULO 12:** Arquivos (Parte I)
- **CAPÍTULO 13:** Arquivos (Parte II)
- **PORJETO FINAL**

CAPÍTULO I - AMBIENTAÇÃO MAINFRAME

OBJETIVO

Fornecer uma visão sistêmica e operacional no mainframe, através do ambiente TSO (Time Sharing Option – Opção de Compartilhamento em Tempo) e das ferramentas de operação ISPF (Interactive System Productivity Facility) e SDSF (Spool Display and Search Facility).

CONCEITOS: TSO

O TSO (Time Sharing Option – Opção de Compartilhamento em Tempo) foi criado para que usuários possam compartilhar um sistema ao mesmo tempo. Uma vez no TSO, o usuário tem acesso a Datasets, ferramentas Batch, ferramentas de monitoração de sistema, ferramentas de Sysout, ferramentas do JES, envia mensagens para outros usuários de TSO, etc.

CONCEITOS: ISPF

O ISPF (Interactive System Productivity Facility) funciona como um facilitador para o usuário de TSO. Disponibilizando em seu menu as opções para trabalhar com Datasets, ou até XMIT, tal qual o TSO, só que mais amistoso. O ISPF dá ainda opção para utilizar outras ferramentas de sistema.

CONCEITOS: SDSF

SDSF (Spool Display and Search Facility) sistema de visualização e de pesquisa herda a característica dos mainframes IBM, executando em sistema operacional z / OS, permite que os usuários e administradores possam visualizar e controlar vários aspectos do funcionamento do mainframe (monitoramento).

AMBIENTAÇÃO MAINFRAME

Ambiente TSO

- Logon TSO (Usuário e Senha)
- Ambiente das PF's de Navegação mais utilizadas
- PF1
- PF3
- PF7 e PF8
- ESC
- ENTER

Editor de Textos

- Inserir uma linha (e várias linhas)
- Repetir uma linha (e várias linhas)
- Copiar uma linha (e um bloco de linhas)
- Mover uma linha (e um bloco de linhas)
- Apagar uma linha (e um bloco de linhas)

AMBIENTAÇÃO MAINFRAME

Ambiente ISPF

- DataSets (Particionados e Fixo)
- Acessando um DataSet
- Acessando um membro do DataSet
- Editando um membro do DataSet
- Comandos de edição de um membro (Copiar, Inserir, Deletar, Mover, etc.).

Ambiente SDSF

- Visualização de Jobs de Compilação
- Visualização de Jobs de Execução

CAPÍTULO II – VISÃO GERAL DO COBOL

OBJETIVO

Dar uma visão geral do COBOL, da sua estrutura e da análise de um programa básico.

CONCEITOS: COBOL

Em 1959, foi criada a linguagem COBOL (Common Business Oriented Language) – Linguagem Comum Orientada a Negócios. Trata-se de uma linguagem de programação estruturada, que prima pela simplicidade, objetividade e organização de código-fonte. É uma linguagem de programação de alto nível, principalmente por seu código escrito assemelhar-se muito com comandos humanamente ditados à própria máquina, em inglês.

Estrutura do COBOL

As divisões do código fonte são:

IDENTIFICATION DIVISION.

ENVIRONMENT DIVISION.

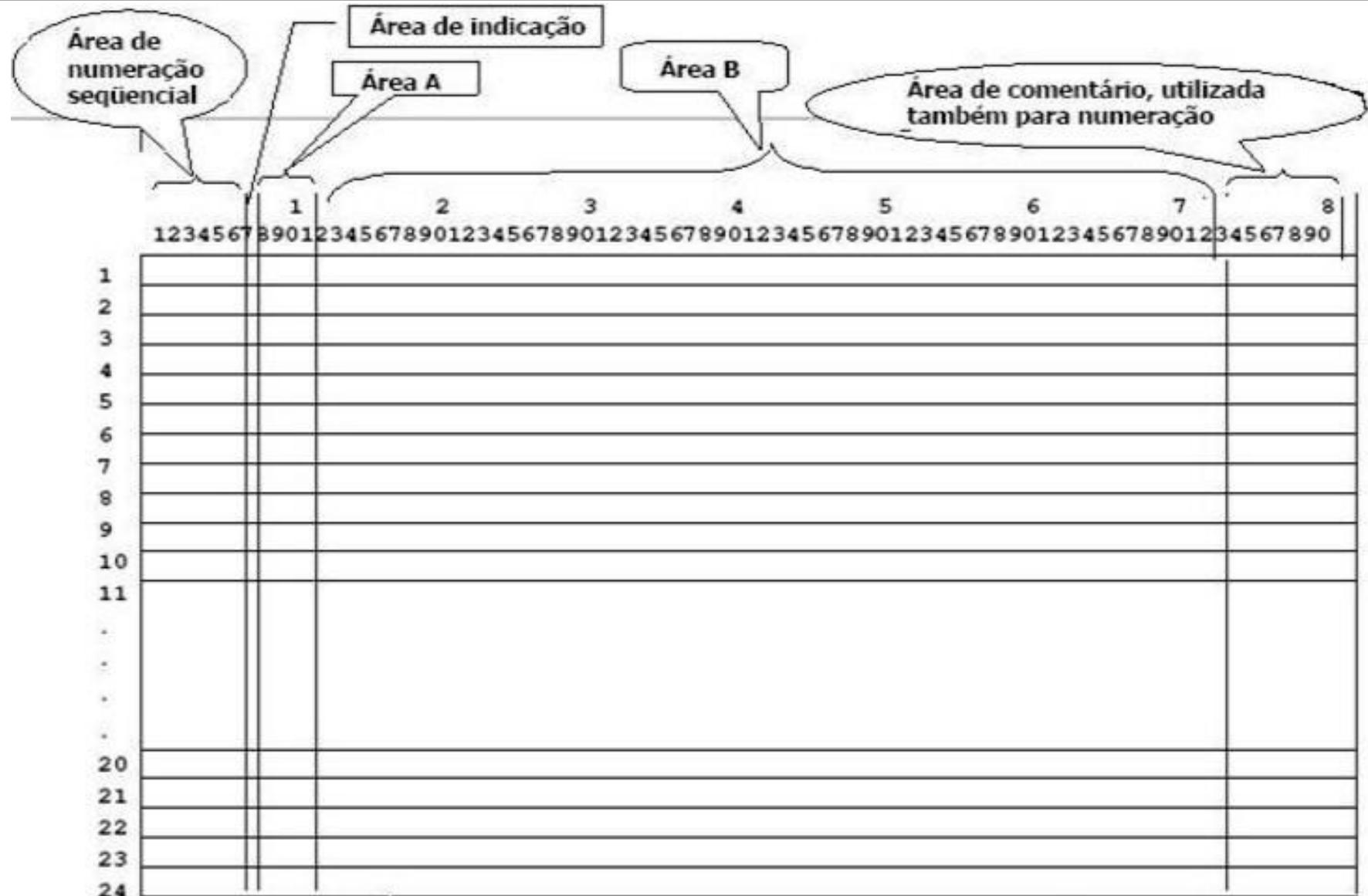
DATA DIVISION.

PROCEDURE DIVISION.

Regras de Codificação no Editor de Texto

- Colunas de 1 a 6** : Área de Numeração
- Coluna 7** : Área de Indicação ("-", "*", "/")
"-" Significa Continuação de uma Literal;
"*" Significa Linha de Comentário;
"/" Determina que Ocorrerá um Salto de Página no Relatório de Compilação.
- Colunas de 8 a 11** : Área A (Nomes de Divisões, Seções e Parágrafos)
- Colunas de 12 a 72** : Área B (Instruções Cobol)
- Colunas de 73 a 80** : Área de Comentário

CAPÍTULO II – VISÃO GERAL DO COBOL



CAPÍTULO II – VISÃO GERAL DO COBOL

Palavras Reservadas **COBOL**

- Significado próprio para o compilador, de uso exclusivo do compilador.
- Ex: DATA (dados) , TIME (hora), etc.

Nomes do **COBOL**

Nome-de-Dados (data-Names)

São nomes atribuídos aos dados utilizados no programa.
Ex: Nome de arquivos, nome campos, etc.

Nomes-de-Procedimento (Procedure-Names)

Identificam os nomes de parágrafos, seções e divisões.

Nomes-de-Condição (Condition-Names)

Designação de valores assumidos por um item de dados.

Nomes-Externos (External-Names)

São os atribuídos às partes físicas do computador.



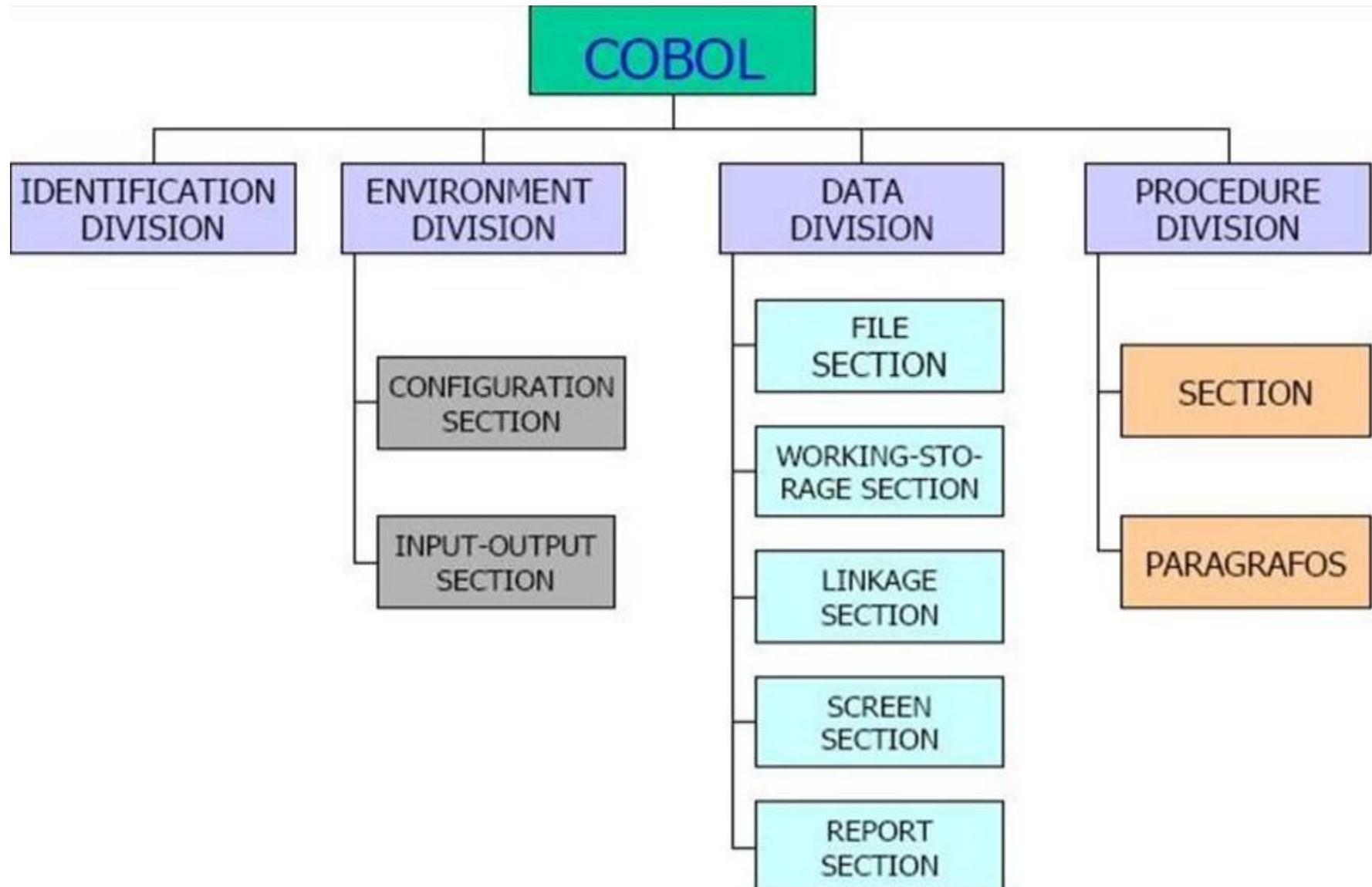
Conhecer a Estrutura das Divisões do Cobol

Conhecer as Divisões:

IDENTIFICATION DIVISION

ENVIRONMENT DIVISION

CAPÍTULO III - ESTRUTURA DO COBOL



CAPÍTULO III – ESTRUTURA DO COBOL

IDENTIFICATION DIVISION

PROGRAM-ID.	Nome- programa .
AUTHOR.	Nome- programador .
DATE-WRITTEN.	“Data da codificação” .
DATE-COMPILED.	Comentário .
SECURITY.	Comentário (Objetivo do programa) .

Objetivo:

- Identificar o programa no computador;
- Proporciona informações documentais, importante para análise superficial do programa.

CAPÍTULO III – ESTRUTURA DO COBOL

ENVIRONMENT DIVISION

CONFIGURATION SECTION

SPECIAL-NAMES. DECIMAL-POINT IS COMMA.

INPUT-OUTPUT SECTION

FILE-CONTROL.

SELECT file-name ASSIGN TO {DISK,PRINTER}

[ORGANIZATION IS { SEQUENTIAL, INDEXED, LINE SEQUENTIAL, RELATIVE}

[ACCESS MODE IS { SEQUENTIAL,RANDOM, DYNAMIC}

[RECORD KEY IS nome-chave-primaria]

[ALTERNATE RECORD KEY IS nome-chave-secundaria [WITH DUPLICATES]]

[FILE STATUS IS nome-campo-file-status] .

Objetivo:

Define os Arquivos a Serem Utilizados na Programação, sua Organização, meio de Acesso, Chaves Primárias e/ou Secundárias.

CAPÍTULO IV – DADOS NO COBOL

Conhecer como os Dados são Manipulados na Data Division na FILE SECTION e WORKING-STORAGE SECTION.

CAPÍTULO IV – DADOS NO COBOL

DATA DIVISION

FILE SECTION

FD nome-arquivo

[RECORD CONTAINS nn CHARACTERS]
[BLOCK CONTAINS nn RECORDS]
[LABEL RECORD IS { OMITTED, STANDARD }]
[VALUE OF FILE-ID valor-identificação-arquivo].

01 REG-ARQUIVO.

 03 CAMPO-01 PIC X(004) .

 03 FILLER PIC X(005) .

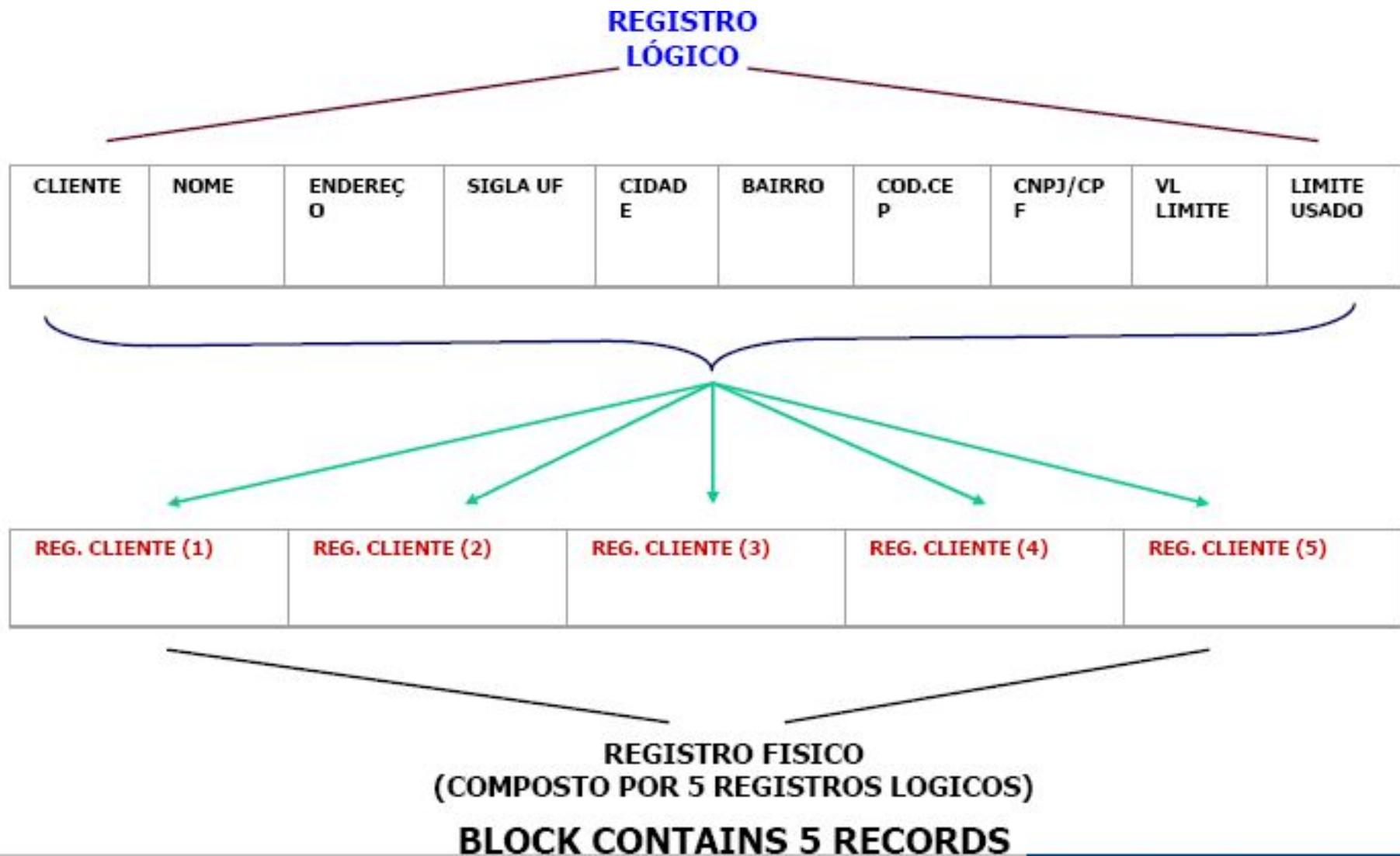
 03 DATA-NASC.

 05 DIA-NASC PIC 9(002) .

 05 MES-NASC PIC 9(002) .

 05 ANO-NASC PIC 9(002) .

CAPÍTULO IV – DADOS NO COBOL



CAPÍTULO IV – DADOS NO COBOL

Regras para Formação de Nomes:

Nome de Arquivos:

- De 1 até 30 caracteres;
- Nenhum caractere especial;
- Nenhum caractere branco no meio;
- Pelo menos um caractere alfabético.

Dados (registros, campos, etc):

- De 1 até 30 caracteres;
- Não podem começar nem terminar com hífen;
- Pelo menos um caractere alfabético;
- Não podem ser palavra reservada do COBOL;
- Podem conter letras, números ou hífens e mais nenhum caractere especial.

Regras para Formação de Nomes:

Literais Numéricos:

- Máximo de 18 dígitos;
- Sinal ("+" ou "-") à esquerda do número;
- Ponto decimal, que não pode ser o último caractere.

Literais Não Numéricos:

- Máximo de 120 caracteres, incluindo espaços branco,
- Qualquer caractere especial;
- Devem estar entre aspas, normalmente simples.

WORKING-STORAGE SECTION

Seção Utilizada Para Definir Itens de Dados de Trabalho Utilizado pelo Programa, que Podem Ser:

ITENS DE GRUPO – Itens que Podem Ser SubDivididos em Outros Itens de Grupo ou em Conjunto de Itens Elementares

ITENS ELEMENTARES – São Itens Que Não São SubDivididos

Nível ou Indicador de Nível:

Nível 01 – Normalmente Utilizado Para Definir Itens de Grupo

Nível 02 a 49 – Define Itens de Grupo Subordinado a um
Outro Item de Grupo ou Itens Elementares
do Item de Grupo

Nível 77 – Item Independente, Não Possue SubItem

CAPÍTULO IV – DADOS NO COBOL

Nível ou Indicador de Nível:

Níveis 66 - Utilizado Para Renomear Outras Variáveis do Programa

Exemplo:

```
01 DATA-SISTEMA.  
  10 ANO-SISTEMA          PIC 9(004) VALUE ZEROS.  
  10 MES-SISTEMA          PIC 9(002) VALUE ZEROS.  
  10 DIA-SISTEMA          PIC 9(002) VALUE ZEROS.  
  66 MES-DIA-SISTEMA    RENAMES MES-SISTEMA THRU DIA-SISTEMA.
```

CAPÍTULO IV – DADOS NO COBOL

Nível ou Indicador de Nível:

Níveis 88 - Especifica Condições que Devem ser Associadas a Valores Particulares

```
01  CODIGO-ENCARGO      PIC 9(003) .
    88  CPMF             VALUE 100, 101, 120, 400.
    88  IOF              VALUE 200 THRU 299, 301, 302.
    88  JUROS            VALUE 600, 610, 611.
```

PROCEDURE DIVISION.

```
MOVE  '100'          TO  CODIGO-ENCARGO
```

...

```
IF  CPMF
```

```
    MOVE  'CPMF'        TO  ARQ-SAIDA-MENSAGEM
```

```
END-IF
```

CAPÍTULO IV – DADOS NO COBOL

Nível ou Indicador de Nível:

Cláusula Picture: Descreve o Tamanho, Sinal, Tipo do Formato do Item de dado.

Formato do Item

Alfabético	- 77	NOME	PIC	A (10) .
Alfanumérico	- 03	ENDERECO	PIC	X (60) .
Numerico	- 77	VALOR	PIC	9 (13)V99.
Numerico c/sinal	- 77	VALOR	PIC	S9 (13)V99.
Decimal compactado	- 77	VALOR	PIC	S9 (13)V99 COMP-3.
Binario	- 77	VALOR	PIC	S9 (04) COMP.

Nível ou Indicador de Nível: Cláusula Picture: Formato de Edição

Formato dos Itens:

PIC 999.999.999, 99

PIC ZZZ.ZZZ.ZZ9, 99

PIC ---.---.--9, 99

PIC \$\$\$.\$\$\$.\$\$9, 99

PIC ***,***, **9, 99

PIC 999.999.999, 99CR

PIC 999.999.999, 99DB

PIC 999B999B999

CAPÍTULO IV – DADOS NO COBOL

Constantes Figurativas: Valores são Definidos pelo Compilador COBOL

Constante	Significado
ZERO, ZEROS, ZEROES	Igual a 0 (zero)
HIGH-VALUE, HIGH-VALUES	Maior valor atribuido a um campo
LOW-VALUE, LOW-VALUES	Menor valor atribuido a um campo
SPACE, SPACES	Igual a espaço (brancos)
Etc.	

CAPÍTULO IV – DADOS NO COBOL

Cláusula Redefines:

Redescreve uma Área ou de um Arquivo ou da Working-Storage, de Modo que os Caracteres Pertencentes à área Possam ser Referenciados por Outros Nomes.

```
01 DATA-SISTEMA.  
    10 ANO-SISTEMA          PIC 9(004) VALUE ZEROS.  
    10 MES-SISTEMA          PIC 9(002) VALUE ZEROS.  
    10 DIA-SISTEMA          PIC 9(002) VALUE ZEROS.  
01 DATA-SISTEMA-R          REDEFINES DATA-SISTEMA.  
    10 ANO-MES-SISTEMA      PIC 9(006).  
    10 FILLER                PIC 9(002).  
} } Área redefinida  
} de  
} DATA-SISTEMA
```

CAPÍTULO IV – DADOS NO COBOL

Tabelas – Occurs

Uma área de Tabela Pode ser Definida na Data Division, ou Seja na File Section ou na Working-Storage Section, Utilizando a Cláusula OCCURS.

01 REGISTRO-TRABALHO.

```
 03 IMPOSTO          PIC 9(05)V99.  
 03 INDICE          PIC 9(02)V99 OCCURS 20 TIMES.  
 03 VARIANCA        PIC 9(03)V99.
```

01 TB-ESTADOS.

```
 03 FILLER          PIC X(20) VALUE 'SPSAO PAULO'.  
 03 FILLER          PIC X(20) VALUE 'RJRIO DE JANEIRO'.  
 03 FILLER          PIC X(20) VALUE 'MGMINAS GERAIS'.  
 01 FILLER          REDEFINES TB-ESTADOS OCCURS 3 TIMES.
```

```
 03 SG-ESTADO       PIC X(02).  
 03 NM-ESTADO       PIC X(18).
```

CAPÍTULO IV - PRÁTICAS

Prática I- Aula 5 (Módulos 1 a 4)

Aula 6 (Módulos 1 a 4)

CAPÍTULO V – MANIPULAÇÃO DE DADOS

**Conhecer Como Obter Dados do Sistema ou
Visualizar Dados ou Informações.**

Cláusula **ACCEPT** - Aceitar.

Obtém dados de fora do Programa.

Formato:

ACCEPT <dados> FROM

{
 DATE
 TIME
 DAY
 SCAPE KEY}

ACCEPT (L, C) <dados>

Onde: L = linha

C = Coluna

Cláusula **DISPLAY** - Visualizar/Mostrar.

Mostrar/visualizar dados para fora do programa.

Formato:

DISPLAY <literal>

DISPLAY <dados>

DISPLAY <literal> <dados>

DISPLAY (L, C) <literal> <dados>

CAPÍTULO V – MANIPULAÇÃO DE DADOS

Instrução MOVE

Transfere Dados de uma Área de Memória para Uma ou mais Áreas.

Formato-1:

```
MOVE {Identifier-1} TO Identifier-2 [Identifier-3 . . .]  
      |  
      LITERAL
```

Formato-2:

```
MOVE {CORRESPONDING} Identifier-1 TO Identifier-2  
      |  
      CORR
```

CAPÍTULO V – MANIPULAÇÃO DE DADOS

Instrução MOVE

Formato-3:

```
MOVE { Identifier-1 (index-1:index-2)
      { LITERAL
        TO { Identifier-2
              { Identifier-3 (index-3:index-4) } }
```

Exemplo:

CAMPO1 = 'CONTABILIDADE E ADVOCACIA'

MOVE CAMPO1 (1:13) TO CAMPO2

MOVE CAMPO1 (17:9) TO CAMPO3

Resultado: CAMPO2 = 'CONTABILIDADE'
CAMPO3 = 'ADVOCACIA'

Instrução INSPECT

A Instrução **INSPECT** Especifica que Caracteres em um Item de Dados Serão Contados (**Tallying**) ou Substituídos (**Replacing**) ou Ambos.

Instrução INSPECT

INSPECT Identifier-1 TALLYING

```
{ Identifier-2 FOR { { ALL  
          LEADING { Identifier-3  
          CHARACTERS Literal-1 } }  
          { BEFORE } { INITIAL { Identifier-4  
          AFTER { Literal-2 } } } ... }
```

CAPÍTULO V – MANIPULAÇÃO DE DADOS

Instrução INSPECT

INSPECT Identifier-1 REPLACE

CHARACTERS BY { Identifier-5 } [{ BEFORE } | { AFTER }] INITIAL { Identifier-6 }
 { Identifier-5 } { Literal-3 } { BEFORE } { Identifier-6 } { Literal-4 }

{ { { ALL } | { LEADING } } { Identifier-7 } { Literal-5 } BY { Identifier-8 } { Literal-6 } [{ BEFORE } | { AFTER }] }
 { ALL } { Identifier-7 } { Literal-5 } { Identifier-8 } { Literal-6 } { BEFORE } { AFTER }

INITIAL { Identifier-9 } ... }

{ Identifier-9 }

...

CAPÍTULO V – MANIPULAÇÃO DE DADOS

Instrução STRING

```
STRING { Identifier-1 }  
        { Literal-1 }      DELIMITED BY { Identifier-2 }  
                           { Literal-2 }  
                           SIZE  
  
           INTO Identifier-3  
  
           [ WITH POINTER Identifier-4 ]  
  
           [ ON OVERFLOW Imperative-statement-1 ]  
  
           [ NOT ON OVERFLOW Imperative-statement-2 ]  
  
END-STRING
```

CAPÍTULO V – MANIPULAÇÃO DE DADOS

Instrução **UNSTRING**

```
UNSTRING { Identifier-1 } DELIMITED BY [ ALL ] { Identifier-2 }
          Literal-1                                         Literal-2
                                                       OR      [ ALL ] { Identifier-3 }
                                                       Literal-3

          INTO Identifier-4

          [ DELIMITED IN Identifier-5 ]
          [ COUNT IN Identifier-6 ]
          [ WITH POINTER Identifier-7 ]
          [ TALLYING IN Identifier-8 ]
          [ ON OVERFLOW Imperative-statement-1 ]
          [ NOT ON OVERFLOW Imperative-statement-2 ]
```

CAPÍTULO VI – O COMANDO PERFORM

Conhecer o Comando **PERFORM** do COBOL.

Instrução **PERFORM** - Executar

Executa os Comandos de um Parágrafo.
Quando Todas as Instruções são Executadas,
o Controle é Transferido para Instrução que
Segue Imediatamente a Cláusula **PERFORM**.

CAPÍTULO VI – O COMANDO PERFORM

Formato 1 – **PERFORM** Básico

OUT-LINE

```
PERFORM PROCEDURE-NAME-1 { THROUGH  
                           THRU } PROCEDURE-NAME-2 }
```

IN-LINE

```
PERFORM PROCEDURE-NAME-1 { THROUGH  
                           THRU } PROCEDURE-NAME-2  
END-PERFORM
```

CAPÍTULO VI – O COMANDO PERFORM

Formato 2 – **PERFORM** Com Opção **TIMES**

OUT-LINE

```
PERFORM PROCEDURE-NAME-1 { [ THROUGH ] PROCEDURE-NAME-2 ] }  
      { Identifier-1 TIMES }  
      { Integer-1 }
```

```
PERFORM PROCEDURE-NAME-1 { [ THROUGH ] PROCEDURE-NAME-2 ] }  
END-PERFORM
```

CAPÍTULO VI – O COMANDO PERFORM

Formato 2 – **PERFORM** Com Opção **TIMES**

IN-LINE

```
PERFORM PROCEDURE-NAME-1 { THROUGH  
                           THRU } PROCEDURE-NAME-2  
{ Identifier-1   TIMES }  
{ Integer-1      }  
  
END-PERFORM  
  
PERFORM { Identifier-1 } TIMES  
{ Integer-1 }  
  
ADD    1           TO    IND  
MOVE  ....  
IF    ....  
END-IF  
END-PERFORM
```

CAPÍTULO VI – O COMANDO PERFORM

Formato 3 – **PERFORM** Com Opção **UNTIL**

OUT-LINE

```
PERFORM PROCEDURE-NAME-1 { [ THROUGH ]  
                           [ THRU ] }  
PROCEDURE-NAME-2 }  
  
[ WITH TEST { BEFORE } ] UNTIL Condition-1.  
                           { AFTER }
```

CAPÍTULO VI – O COMANDO PERFORM

Formato 3 – **PERFORM** Com Opção **UNTIL**

IN-LINE

```
PERFORM PROCEDURE-NAME-1 { THROUGH  
                           THRU } PROCEDURE-NAME-2 }  
[ WITH TEST { BEFORE } ] UNTIL Condition-1.
```

END-PERFORM

```
PERFORM [ WITH TEST { BEFORE } ] UNTIL Condition-1.
```

```
MOVE .....  
ADD .....
```

CAPÍTULO VI – O COMANDO PERFORM

Formato 4 – **PERFORM** Com Opção **VARYING** **OUT-LINE**

```
PERFORM PROCEDURE-NAME-1 { THROUGH  
                           THRU } PROCEDURE-NAME-2  
[ WITH TEST { BEFORE }  
           { AFTER } ]  
VARYING { Identifier-2 } FROM { Literal-1  
                                Identifier-3  
                                Index-name-2 }  
          BY { Literal-2 } { Identifier-4 }  
UNTIL Condition-1
```

CAPÍTULO VI – O COMANDO PERFORM

Formato 4 – **PERFORM** Com Opção **VARYING IN-LINE**

```
PERFORM PROCEDURE-NAME-1 { THROUGH  
                           THRU } PROCEDURE-NAME-2 }  
[ WITH TEST { BEFORE }  
             AFTER } ]  
VARYING { Identifier-2 } FROM { Literal-1  
                                Identifier-3  
                                Index-name-2 }  
          BY { Literal-2  
                  Identifier-4 }
```

CAPÍTULO VI – O COMANDO PERFORM

Formato 4 – **PERFORM** Com Opção **VARYING**

IN-LINE **PERFORM** [**WITH TEST** { **BEFORE** }
 { **AFTER** }]

UNTIL Condition-1

VARYING { **Identifier-2** } **FROM** { **Literal-1**
 { **Identifier-3**
 { **Index-name-2** } } }

BY { **Literal-2**
 { **Identifier-4** } } }

MOVE

ADD

PERFORM

IF

END-IF

END-PERFORM

CAPÍTULO VII – PRÁTICAS

Prática II- Aula 9 (Módulos 1 a 4; 6 e 7)

Aula 10 (Módulos 1 a 4; 6 e 7)

Aula 11 (Módulos 1 a 4; 6 e 7)

Aula 12 (Módulos 1 a 4; 6 e 7)

Conhecer o Funcionamento dos Comandos
Aritméticos do **COBOL**.

CAPÍTULO VIII – MATEMÁTICA BÁSICA

Comando ADD	- Soma
Comando SUBTRACT	- Subtrair
Comando MULTIPLY	- Multiplicação
Comando DIVIDE	- Dividir
Comando COMPUTE	- Expressão aritmética

CAPÍTULO VIII – MATEMÁTICA BÁSICA

ADD - (ADIÇÃO)

FORMATO-1

ADD { Identifier-1 } [Identifier-2] TO
 |
 { Literal-1 } [Literal-2]

[Identifier-M ROUNDED] [Identifier-Z ROUNDED]

[ON SIZE ERROR Sentença-imperativa]

ADD - (ADIÇÃO)

FORMATO-2

ADD { Identifier-1 } Identifier-2 Identifier-3 . . .
 | Literal-1 | Literal-2 | Literal-3 . . .

GIVING Identifier-Z ROUNDED

[ON SIZE ERROR Sentença-imperativa]

ADD - (ADIÇÃO)

FORMATO-3

ADD { CORRESPONDIG } Identifier-1 TO Identifier-2
 CORR

CAPÍTULO VIII – MATEMÁTICA BÁSICA

ADD - (CORR)

01 IDENTIFIER-1.

 03 CAMPO-1 PIC (903).

 03 CAMPO-2 PIC 9(05).

 03 CAMPO-3 PIC X(10).

01 IDENTIFIER-2.

 03 CAMPO-1 PIC (903).

 03 CAMPO-2 PIC 9(05).

 03 CAMPO-3 PIC X(10).

ADD CORR IDENTIFIER-1 TO IDENTIFIER-2.

O ADD CORR equivale aos comandos abaixo:

ADD CAMPO-1 OF IDENTIFIER-1 TO CAMPO-1 OF IDENTIFIER-2

ADD CAMPO-2 OF IDENTIFIER-1 TO CAMPO-2 OF IDENTIFIER-2

ADD CAMPO-3 OF IDENTIFIER-1 TO CAMPO-3 OF IDENTIFIER-2

CAPÍTULO VIII – MATEMÁTICA BÁSICA

SUBTRACT - (SUBTRAÇÃO)

FORMATO-1

```
SUBTRACT {Identifier-1}
          {LITERAL-1} Identifier-2 . . .
                         LITERAL-2
FROM Identifier-M [ROUNDED] Identifier-N [ROUNDED]
[ON SIZE ERROR  Sentença-imperativa]
```

CAPÍTULO VIII – MATEMÁTICA BÁSICA

SUBTRACT - (SUBTRAÇÃO)

FORMATO-2

```
SUBTRACT { Identifier-1 } [ Identifier-2 ..... ]  
          { LITERAL-1 }           |  
          { LITERAL-2 }           |  
  
FROM { Identifier-M }  
      { LITERAL-M }  
  
GIVING Identifier-N [ ROUNDED ] [ Identifier-O [ ROUNDED ] ]  
[ ON SIZE ERROR Sentença-imperativa ]
```

SUBTRACT - (SUBTRAÇÃO)

FORMATO-3

SUBTRACT Identifier-1

{ CORREPODING
CORR }

FROM Identifier-2

ON SIZE ERROR Sentença-imperativa

MULTIPLY - (MULTIPLICAÇÃO)

FORMATO-1

MULTIPLY { Identifier-1 } BY Identifier-2

ROUNDED

[ON SIZE ERROR Sentença-imperativa]

MULTIPLY - (MULTIPLICAÇÃO)

FORMATO-2

```
MULTIPLY      { Identifier-1 } BY { Identifier-2 } [ ROUNDED ]  
              { LITERAL-1 }           { LITERAL-2 }
```

```
          GIVING Identifier-3 ROUNDED
```

```
[ ON SIZE ERROR  Sentença-imperativa ]
```

DIVIDE - (DIVISÃO)

FORMATO-1

```
DIVIDE      { Identifier-1 } INTO { Identifier-2 } [ ROUNDED ]  
           { LITERAL-1 }           { LITERAL-2 }
```

```
[ ON SIZE ERROR  Sentença-imperativa ]
```

DIVIDE - (DIVISÃO)

FORMATO-2

```
DIVIDE      { Identifier-1 } { BY      } { Identifier-2 } [ ROUNDED ]
              { LITERAL-1  } { INTO    } { LITERAL-2  }

          GIVING      Identifier-3 [ ROUNDED ]

[ REMAINDER Identifier-4 ]

[ ON SIZE ERROR  Sentença-imperativa ]
```

CAPÍTULO IX – EQUAÇÕES

Comando COMPUTE - Cálculo

Utiliza os Símbolos Aritméticos para Fazer as Representações de Fórmulas Matemáticas.

SIMBOLOGIA:	SOMA	+
	DIFERENÇA	-
	DIVISÃO	/
	MULTIPLICAÇÃO	*
	EXPONENCIAÇÃO	* *

CAPÍTULO IX – EQUAÇÕES

COMPUTE

FORMATO:

COMPUTE Identifier-2 [ROUNDED] = {Identifier-1
LITERAL-1
EXPRESAO ARITMETICA}

[ON SIZE ERROR Sentença-imperativa]

O comando COMPUTE permite montar qualquer equação !

Apresentar Elementos de Decisão

Instrução **IF THEN ELSE** – (Se / Então / Senão)

É Qualquer Sentença que Executa um ou mais Procedimento Dependendo da Ocorrência de uma ou mais Condição.

CAPÍTULO X – ELEMENTOS DE DECISÃO

FORMATO :

```
IF condition      THEN { statement-1
                           NEXT SENTENCE }

ELSE
  { Statement-2
    NEXT SENTENCE }

END-IF.
```

CAPÍTULO X – ELEMENTOS DE DECISÃO

TESTES DE CONDIÇÃO ($>$, $<$ e $=$).

Estes Sinais Equivalem às Seguintes Palavras Reservadas:

>	GREATER THAN
<	LESS THAN
=	EQUAL TO
NOT >	LESS THAN OR EQUAL
NOT <	GREATER THAN OR EQUAL
NOT =	NOT EQUAL TO

Teste de CONDIÇÃO de CLASSE

```
IF Identifier { IS (NUMERIC)  
                { IS NOT (ALPHABETIC) }
```

Exemplo:

```
IF CAMPO IS NUMERIC  
    GO TO REGISTRO-OK  
ELSE  
    MOVE "CAMPO NAO NUMERICO" TO MENSAGEM  
    GO TO ROTINA-ERRO.
```

Teste de NOME-DE-CONDIÇÃO

```
IF STATUS = 1  
      GO TO PROC-HOMEM.  
  
IF STATUS = 2  
      GO TO PROC-MULHER.  
  
IF STATUS = 3  
      GO TO PROC-OUTRO.  
  
IF STATUS GREATER 3  
      GO TO ROT-ERRO.
```

Teste de CONDIÇÃO DE RELAÇÃO

Efetua comparação entre dois operandos.

Formato:

IF {Identifier
LITERAL
EXPRESSÃO ARITMETICA} OPERADOR DE RELAÇÃO {Identifier-1
LITERAL-1
EXPRESSÃO ARITMETICA-1}

Exemplo:

```
IF AC-LIN GREATER 30
    GO TO ROT-CABEC.
IF CODIGO = 2
    MOVE A TO B.
IF CAMPO NOT LESS DADO
    PERFORM ROT-DEFEITO.
```

CAPÍTULO X – ELEMENTOS DE DECISÃO

Teste de CONDIÇÃO DE SINAL

Determina o valor algébrico de um operando aritmético.

Formato:

IF {IDENTIFICADOR
EXPRESSAO ARITMETICA} {IS
IS NOT} {POSITIVE
NEGATIVE
ZERO}

Exemplo-1:

```
IF VALOR IS POSITIVE
    MOVE VALOR TO SAI-VAL
    ADD VALOR TO AC-POSITIVE.
```

```
IF QUALQUER GREATER THAN ( B ** 2 - 4 * A )
    NEXT SENTENCE
ELSE
    GO TO ROTINA-UM.
```

Instrução **EVALUATE**

A instrução **EVALUATE** permite codificar a estrutura **CASE** da Programação Estruturada.

CAPÍTULO X – ELEMENTOS DE DECISÃO

Instrução EVALUATE

```
EVALUATE { Identifier-1  
          Literal-1  
          (Expression-1 / ALSO )  
          TRUE  
          FALSE } Identifier-2  
          Literal-2  
          Expression-2  
          TRUE  
          FALSE  
  
WHEN { ANY  
        Condition-1  
        TRUE  
        FALSE } { Identifier-3  
          Literal-3  
          Arithmetic-expression }  
          Imperative-statement-2  
.  
.  
.  
WHEN OTHER Imperative-statement-3  
END-EVALUATE.
```

CAPÍTULO X – ELEMENTOS DE DECISÃO

```
IF CONTADOR GREATER THAN 100 THEN
  EVALUATE MES
    WHEN 01
      PERFORM TRATA-MES1
    WHEN 02
      PERFORM TRATA-MES2
    WHEN 03
      PERFORM TRATA-MES3
    WHEN OTHER
      PERFORM TRATA-INVALIDO
  END-EVALUATE
ELSE
  EVALUATE OPCAO
    WHEN 'S'
      PERFORM TRATAR-ANO
    WHEN 'N'
      PERFORM TRATA-MESES
    WHEN OTHER
      PERFORM TRATA-OPCAO-INV
  END-EVALUATE
END-IF
```

```
IF CONTADOR GREATER THAN 100 THEN
  IF MES EQUAL 01
    PERFORM TRATA-MES1
  ELSE
    IF MES EQUAL 02
      PERFORM TRATA-MES2
    ELSE
      IF MES EQUAL 03
        PERFORM TRATA-MES3
      ELSE
        PERFORM TRATA-INVALIDO
      END-IF
    END-IF
  END-IF
ELSE
  IF OPCAO EQUAL 'S'
    PERFORM TRATAR-ANO
  ELSE
    IF OPCAO EQUAL 'N'
      PERFORM TRATA-MESES
    ELSE
      PERFORM TRATA-OPCAO-INV
    END-IF
  END-IF
END-IF
```

COMUNICAÇÃO ENTRE PROGRAMAS

O Comando Utilizado Para Chamar Subprogramas é o *CALL*. Este Comando Transfere o Fluxo de Execução Para Outro Programa e Aguarda o Seu Retorno Para Dar Continuidade nos Comandos Após a Sua Utilização.

Instrução CALL **CALL ESTÁTICO**

O Código do Programa Chamado é Incluído no Programa Chamador na Altura da LINKEDICAÇÃO. Desse Modo, Sempre que o Programa Chamado for Alterado, é Necessário Recompilar Todos os Programas Que o Chamam.

Exemplo:

```
CALL 'PTNEM013' USING CAREA-PTNEM013.
```

Instrução CALL **CALL DINÁMICO**

Código do Programa Chamado Apenas é Obtido Durante a Execução do Programa Chamador, pelo que, Sempre que o Programa Chamado for Alterado, o Programa Chamador “Apanha” a Nova Versão.

Exemplo:

CALL PTNEM013 USING CAREA-PTNEM013.

CAPÍTULO XI – PRÁTICAS

Prática III - Aula 15 (Módulos 9 e 10)

Aula 16 (Módulos 9 e 10)

CAPÍTULO XII – ARQUIVOS (PARTE I)

**APRESENTAR OS COMANDOS DE ENTRADA E SAÍDA
UTILIZADOS PARA MANIPULAÇÃO DE ARQUIVOS**

Comandos Apresentados:

OPEN

CLOSE

READ

CAPÍTULO XII – ARQUIVOS (PARTE I)

OPEN

Abrindo os Arquivos

Irá Disponibilizar um Canal com o Arquivo para que o Programa faça as Operações Necessárias.

LER	<input type="checkbox"/>	OPEN	INPUT	NOME-DO-ARQUIVO-DE-ENTRADA
CRIAR	<input type="checkbox"/>	OPEN	OUTPUT	NOME-DO-ARQUIVO-DE-SAIDA
ALTERAR	<input type="checkbox"/>	OPEN	IO	NOME-DO-ARQUIVO-DE-ENTRADA E SAIDA

CLOSE

Fechando os Arquivos

O Comando **Close** Termina o Processar dos Carretéis (ou das Unidades) e dos Arquivos. pode Também Executar a Rebobinação, o Fechamento, e as Operações de Remoção.

CAPÍTULO XII – ARQUIVOS (PARTE I)

READ

Lendo os Arquivos

Para Termos Acesso aos Registros Contidos nos Arquivos Devemos **Ler** Estes Registros, para isso Utilizamos este Comando.

READ (Lendo os Arquivos)

Formato 1 – Acesso Seqüencial

```
READ File-name [ { NEXT } | { PREVIOUS } ] RECORD INTO Identifier  
[ AT END Instrução-imperativa ]  
[ END-READ . ]
```

Apresentar

**COMANDOS DE GRAVAÇÃO
ENTRADA/SAÍDA.**

CAPÍTULO XIII – ARQUIVOS (PARTE II)

Comandos Apresentados:

WRITE

REWRITE

DELETE

WRITE

Gravando Novas Informações e
Imprimindo Relatórios

O Comando **WRITE** é Utilizado para Gravar Novos Registros em um Arquivo ou Imprimir Dados Num Relatório Baseado em Informações Contidas no Registro Informado.

CAPÍTULO XIII – ARQUIVOS (PARTE II)

WRITE

Formato 1 – Geral

WRITE

RECORD-NAME

FROM

Identifier

BEFORE
AFTER

IDENTIFICADOR-2
INTEIRO
MNEMONIC-NAME
PAGE

LINE
LINES

ADVANCING

INVALID KEY Instrução-imperativo
AT {**END-OF-PAGE**
EOP} Instrução-imperativo

END-WRITE

REWRITE

Regravando Novas Informações.

Antes de se Utilizar Estes Comandos, Deveremos *Carregar* este Registro para Nossa *área de FD*, ou seja, no caso de Arquivos *Indexados* Devemos *ler* o Registro que Deverá ser Atualizado, fazer as Modificações Necessárias e Então Regravá-los.

REWRITE

Formato:

REWRITE RECORD-NAME [**FROM** Identifier]

[**INVALID KEY** Instrução-imperativa]

[**END-REWRITE**]

CAPÍTULO XIII – PRÁTICAS

Prática IV - Aula 19 (Módulos 12 e 13)

Aula 20 (Módulos 12 e 13)

Aula 21 (Módulos 12 e 13)

COBOL BÁSICO – PROJETO FINAL

Projeto Final- Aula 22 (Módulos 1 a 14)

Aula 23 (Módulos 1 a 14)

Aula 24 (Módulos 1 a 14)

Aula 25 (Módulos 1 a 14)

Aula 26 (Módulos 1 a 14)

Aula 27 (Módulos 1 a 14)