

CURSO COBOL

Edit Edit Settings Menu Utilities Compilers Test Help

CURSO COBOL

```
WORKING-STORAGE
*-----*
01 WRK-NUMERO.
03 WRK-NUM1
PIC 9(02) VALUE 1.
*=====*
PROCEDURE
*=====*
0000-INICIO
PERFORM
    UNTIL WRK-NUM1 = 0
ACCEPT WRK-NUM1
DISPLAY '*'
DISPLAY '*' WRK-NUM1 = ' WRK-NUM1
DISPLAY '*'
END-PERFORM.
STOP RUN.
```

F2=Split F3=Exit F4=Rfind F5=Change F6=Up
F9=Swap F10=Left F11=Right F12=Cancel F13=Down

1:3777 04,15 00:00.406 00:06 05/22/12

Objetivo:

Capacitar o aluno a analisar/desenvolver/realizar manutenções em programas na linguagem de programação **Cobol**

O que é COBOL:

- COBOL significa **Common Business Oriented Language**, isto é, Linguagem Comum Orientada para o Comércio.
- O Cobol é um subconjunto de palavras da língua inglesa, ou seja, um número limitado de palavras inglesas sujeita a uma sintaxe própria.
- É uma linguagem que lida com problemas comerciais, envolvendo arquivos de dados de apreciáveis proporções (Seqüências/VSAM/Banco de dados DB2).

Como editar um programa:

É necessário usar uma estrutura definida da maneira de escrever.

O compilador Cobol possui características posicionais, isto é, necessitamos da ordenação de palavras, divisões e seções, usando uma estrutura definida.

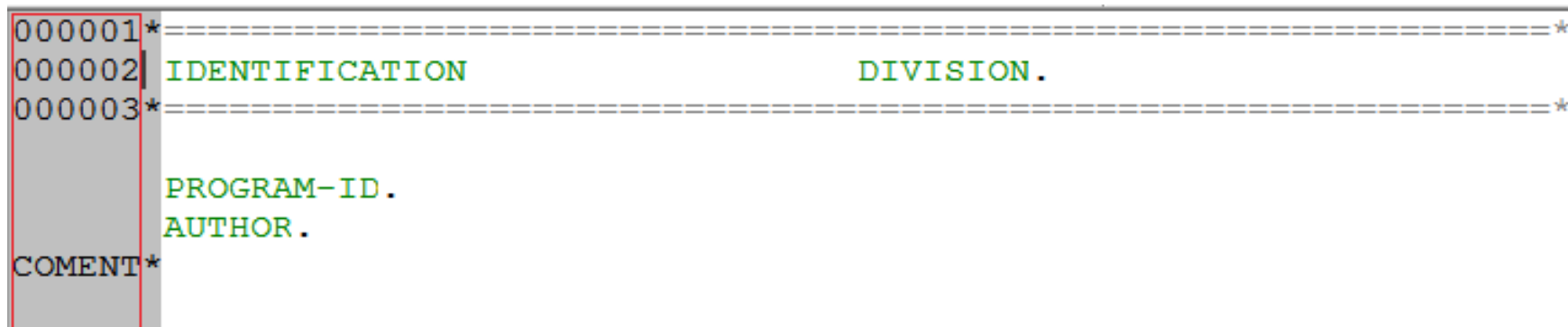
CURSO COBOL

Coluna de 1 a 6:

Essas colunas são usadas para numerar as linhas de um programa.

A numeração é uma ordem crescente.

Opcionalmente podem deixar de serem preenchidas ou incluir outros caracteres.



The diagram illustrates the structure of a COBOL program line. It shows a grid with columns numbered 1 through 6 on the left. The first three lines are separated by dashed lines. The first line contains '000001' in column 1 and '*' in column 6. The second line contains '000002' in column 1, 'IDENTIFICATION' in column 2, and 'DIVISION.' in column 6. The third line contains '000003' in column 1 and '*' in column 6. Below these, there are two more lines: 'PROGRAM-ID.' and 'AUTHOR.'. At the bottom, there is a line labeled 'COMENT' in column 1 and '*' in column 6. A red box highlights the first three lines, indicating the numbering columns.

```
000001*=====*
```

```
000002 IDENTIFICATION DIVISION.
```

```
000003*=====*
```

```
PROGRAM-ID.
```

```
AUTHOR.
```

```
COMENT*
```

Coluna 7:

- Utilizamos o asterisco (*) para inclusão de comentários.
- Utilizamos o hífen (-) para a continuação de não numéricos.
- Utilizamos a barra (/) para o salto de página.

```
000001*=====*
```

000002	IDENTIFICATION	DIVISION.
--------	----------------	-----------

```
000003*=====*
```


	PROGRAM-ID.	.
	AUTHOR.	

```
COMENT*
```

Coluna de 8 a 72:

São usadas para as entradas (palavras ou literais) do programa. Estas colunas estão agrupadas em duas margens 'A' (coluna 8 a 11) e margem 'B' (coluna 12 a 72).

As entradas da margem 'A' são:

- Títulos das divisões, seções e dos parágrafos;
- Descrição dos arquivos;
- Títulos especiais na Procedure Division;
- Os números de nível, como o '77', 01.

As entradas da margem 'B' são:

- Espaços entre as margens (com o objetivo de comunicação visual);
- Continuação das entradas.
- Na procedure os Comandos em Geral, ou seja, Toda Lógica de Programação.

CURSO COBOL

```
*****  
PROCEDURE DIVISION.  
*****
```

```
*****  
      ROTINA PRINCIPAL  
*****
```

```
*****  
0000-INICIAR                                SECTION.  
*****
```

```
      PERFORM 1000-INICIALIZAR.
```

```
      PERFORM 2000-VERIFICAR-VAZIO.
```

```
      PERFORM 3000-PROCESSAR  
              UNTIL WRK-CHV-EARQORIG EQUAL HIGH-VALUES.
```

```
      PERFORM 4000-FINALIZAR.
```

```
*****  
0000-99-FIM.                                EXIT.  
*****
```


CURSO COBOL

```
*-----*
77  FILLER                                PIC  X(050)          VALUE
    '***  ACUMULADORES  ***'.
*-----*
77  ACU-LIDOS-EARQORIG                    PIC  9(009) COMP-3  VALUE ZEROS.
```

```
*****
*  TESTAR FILE-STATUS DO ARQUIVO DE ENTRADA - EARQGRUP
*****
*-----*
1110-TESTAR-FS-EARQORIG                    SECTION.
*-----*

IF  WRK-FS-EARQORIG                        NOT EQUAL ZEROS
    MOVE 'EARQORIG'                        TO WRK-ARQUIVO
```

PALAVRAS RESERVADAS:

Há palavras que são reservadas do Cobol, com propósitos próprios.

São aquelas que têm um significado específico para o compilador COBOL (sintaxe), e não pode ser utilizada fora de sua finalidade dentro de um programa COBOL.

Não podemos criar variáveis, com o mesmo nome de palavras reservadas do cobol.

Exemplo: **FILLER**

Estrutura da linguagem:

A linguagem Cobol é estruturada em:

- Divisões
- Seções
- Parágrafos
- Sentenças
- Cláusulas (nas três primeiras divisões)
- Comandos (Lógica de Programação na Procedure Division)

Cada “DIVISION” do COBOL pode estar dividida em uma ou mais “SECTION”, que por sua vez, cada “SECTION” pode estar dividida em um ou mais “PARÁGRAFOS” e cada “PARÁGRAFO” pode ter um ou uma série de “STATEMENT” (comandos).

Divisões:

As divisões do Cobol para a estruturação do programa e suas funções são quatro:

- Divisão de identificação (IDENTIFICATION DIVISION);
- Divisão de equipamento (ENVIRONMENT DIVISION);
- Divisão de dados (DATA DIVISION);
- Divisão de procedimentos (PROCEDURE DIVISION).

Seções:

Existem 02 tipos de Seções:

- Definidas pelo Cobol (já existentes)
- Definidas pelo programador

Exemplo:

CONFIGURATION SECTION.

WORKING-STORAGE SECTION.

0100-UNICA SECTION.

Parágrafos:

São trechos de código utilizados para separar o código principal dos auxiliares, manter uma organização no programa, reaproveitar codificação.

- O tamanho máximo é de 30 caracteres
- NÃO pode conter espaços e nem caracteres especiais
- Pode conter letras, números ou hífens
- NÃO podemos iniciar ou terminar com o hífen
- NÃO pode ser uma palavra reservada do COBOL

Sentenças:

As sentenças são formadas por uma ou mais cláusulas ou comandos, e terminado por um ponto.

Obs.: O ponto é necessário somente na última sentença do parágrafo.

IDENTIFICATION DIVISION:

Identifica e documenta o programa fonte e outras informações como autor, local de criação, data de criação, data de compilação e descrição do objetivo do programa.

CURSO COBOL

```

*=====*
*IDENTIFICATION                                DIVISION.*
*=====*
*
*PROGRAM-ID.
*AUTHOR.
*
*=====*
*
*PROGRAMA.....:
*PROGRAMADOR.:
*ANALISTA.....:
*DATA.....:
*
*-----*
*OBJETIVO.....:  OBJETIVO DO PROGRAMA
*-----*

```

ENVIRONMENT DIVISION:

É a definição do ambiente físico em que será processado o programa.

Especifica o equipamento usado para compilação e execução do programa, além de associar os arquivos do programa aos diversos periféricos, técnicos especiais de entrada/saída.

CONFIGURATION SECTION:

Identifica características do programa como ex:

- Uso de virgula ao invés de ponto nas decimais

SPECIAL-NAMES DECIMAL-POINT IS COMMA:

Em função de nossa representação do ponto decimal ser diferente da utilizada no país de origem da linguagem Cobol, ou seja, enquanto nos utilizamos Vírgula (,) para representar o ponto decimal, eles utilizam (.), o objetivo desta clausula, quando empregada, é mudar da representação deles para a nossa.

INPUT-OUTPUT SECTION:

Define arquivos utilizados pelo programa efetuando ligações com o equipamento da máquina.

FILE-CONTROL:

É o controle de arquivos, cada arquivo descrito na “DATA DIVISION” deverá ter seu nome simbólico de arquivo descrito após o “select”.

CURSO COBOL

ENVIRONMENT

DIVISION.

CONFIGURATION

SECTION.

SPECIAL-NAMES.

DECIMAL-POINT

IS COMMA.

INPUT-OUTPUT

SECTION.

FILE-CONTROL.

SELECT EARQORIG ASSIGN
FILE STATUS

TO EARQORIG
IS WRK-FS-EARQORIG.

DATA DIVISION:

Define a estrutura lógica dos arquivos e das áreas de trabalho. Descreve os dados que o programa aceitará como entrada e os que serão produzidos como saída.

A DATA DIVISION tem a função de descrever os arquivos e seus registros.

Assim como qualquer área de trabalho necessária ao programa. Essa divisão possui 3 seções que devem aparecer na ordem especificada.

Caso alguma não seja necessária, deve ser omitida:

- 1 – FILE SECTION.
- 2 – WORKING-STORAGE SECTION.
- 3 – LINKAGE SECTION.

FILE SECTION:

Descreve o conteúdo e a organização dos arquivos.

```

=====
DATA DIVISION.
=====
FILE SECTION.
=====
      INPUT - DADOS DO ARQUIVO DE ENTRADA (EARQORIG)
              - LRECL = 290
=====

FD  EARQORIG
   RECORDING MODE IS F
   LABEL RECORD IS STANDARD
   BLOCK CONTAINS 0 RECORDS.

01  FD-EARQORIG                PIC  X(290)                VALUE SPACES.
```

WORKING-STORAGE SECTION:

Esta seção descreve informações sobre as áreas de trabalho:

- Variáveis criadas pelo programador
- Tabelas Internas
- Lay-out de relatórios
- Mensagens utilizadas no Programa
- Copy books de arquivos
- Áreas do DB2

1º Programa

CLÁUSULA DISPLAY:

Esta cláusula serve para escrever dados em um dispositivo de saída. (SYSOUT)

Sintaxe :

DISPLAY 'TOTAL DE REGISTROS LIDOS = ' TOTAL-LIDOS.

CLÁUSULA STOP RUN:

Termina o processamento de um programa.

STOP RUN.

Este comando é obrigatório, podendo existir mais de um comando dentro do mesmo programa.

VARIÁVEIS

Nome de Campos/Variáveis:

Devemos Criar as Variáveis necessárias na WORKING-STORAGE SECTION.

- ✓ Todas as Variáveis de uso do Programador devem ser criadas nesta Área.
- ✓ Em comprimento, um nome de campo não deve exceder a 30 caracteres.
- ✓ O espaço em branco, underline, ou caracteres especiais não são permitidos para a formação de palavras.
- ✓ Uma palavra não pode começar nem terminar com hífen (-).

Exemplo:

- WRK-IMPOSTO-RENDA
- ACU-LIDOS-EARQORIG

```
*-----*  
WORKING-STORAGE SECTION.  
*-----*  
  
01 WRK-CHV-EARQORIG.  
   03 WRK-ORIG-CARTEIRA      PIC X(003)      VALUE SPACES.  
   03 WRK-ORIG-CARTEIRA-2    PIC X(002)      VALUE SPACES.
```

CURSO COBOL

```

*-----*
WORKING-STORAGE                               SECTION.
*-----*

01  WRK-CHV-EARQORIG.
    03  WRK-ORIG-CARTEIRA          PIC X(003)          VALUE SPACES.
    03  WRK-ORIG-CARTEIRA-2       PIC X(002)          VALUE SPACES.

```

Os números de níveis podem começar em 01 até 49, ou utilizar o nível 77.

01.. 49 – Chamadas de variáveis de grupo, item, estruturais..

77 – Variável de valor único

88 – Variáveis do tipo lógica

**** Criar nomes de variáveis com significativos lógicos.

CURSO COBOL

VARIÁVEIS ALFANUMÉRICAS:

ALFANUMÉRICO -> é representado por letras, números e caracteres do Cobol

Exemplos:

01 WRK-DADO1	PIC XXX	VALUE 'ANO'.
01 WRK-ENDER	PIC X(004)	VALUE 'RUA LAPA, 1345'.
01 WRK-NOME	PIC X(030)	VALUE 'SILVIO SANTOS'.

VARIÁVEIS NUMÉRICAS:

NUMÉRICO -> usa-se para representação exclusiva de itens numéricos.

Os caracteres usados são: “9”, “V” e “S”

O tamanho máximo permitido é de 18 bytes, o uso do “V” representa a vírgula e do “S” representa a possibilidade de armazenar o sinal (negativo).

A omissão do sinal significará que o número é positivo.

O uso do “V” define a quantidade de casas decimais, a omissão da vírgula declara um número inteiro.

CLÁUSULA PICTURE (PIC):

É usada para descrição, definição de informações sobre itens, tais como: tamanho, sinal, tipo numérico alfanumérico ou alfabético.

X – Indica campo Alfanumérico

A – Indica campo Alfabético

9 – Indica campo Numérico

V – Indica Vírgula Decimal Implícita

S – Indica Sinal Algébrico

Z – Indica Edição de Campos Numéricos

CURSO COBOL

```
01 WRK-NUM-004    PIC 9(004)          VALUE ZEROS.  
01 WRK-NUM        PIC 9999           VALUE ZEROS.
```

```
01 WRK-VALOR      PIC 9(013)V99       VALUE ZEROS.  
01 WRK-VLR        PIC 9(013)V9(002)  VALUE ZEROS.
```

```
01 WRK-VALOR      PIC S9(013)V99     VALUE ZEROS.  
01 WRK-VLR        PIC S9(013)V9(002) VALUE ZEROS.
```

“S” = é utilizado para apresentação de sinal de negativo (-).

“9” = é utilizado para indicar a posição do campo que contém um dígito de “0” a “9”.

“V” = é usado para mostrar a posição da vírgula decimal. O ponto decimal, se colocado, não faz parte do item

Constantes Figurativas:

É uma palavra associada a um valor particular.

ZEROS, ZERO, ZEROES

→ valor zero, ou o próprio numero 0

SPACE, SPACES

→ valor brancos ou pode ser representado ' '

LOW-VALUE, LOW-VALUES

→ menor valor (hexa 00) – Zeros Binários

HIGH-VALUE, HIGH-VALUES

→ maior valor (hexa FF)

CLÁUSULA VALUE:

É usada para definir um valor inicial para um item da “WORKING-STORAGE SECTION”.

A cláusula VALUE não deve ser especificada para descrições de dados que tenham a cláusula OCCURS. (ex. definição de tabela interna)

Não pode ser usada na “FILE SECTION”, e nem em itens de Grupo (Redefines), neste caso apenas no nível 01.

Exemplo:

```
03  WRK-REG      PIC 9(05)  VALUE ZEROS.  
03  FILLER       PIC X(06)  VALUE 'FOLHAS'.  
03  DATA        PIC X(10)  VALUE SPACES.
```

NÍVEL 77:

Níveis: (77) → designa itens da “WORKING-STORAGE SECTION” que não são subdivisores de outros e por sua vez não são subdivididos.

Quando utilizados devem ser descritos obrigatoriamente dentro da “WORKING-STORAGE SECTION”.

Iniciaremos a Working-Storage com o nível 77, e a partir do momento que declararmos um nível 01, não poderemos mais utilizar o nível 77.

Exemplo:

WORKING-STORAGE SECTION.

```
77  ACU-LIDOS      PIC 9(005)    VALUE ZEROS.  
77  WRK-NOME       PIC X(020)    VALUE SPACES.
```

CURSO COBOL

Itens de grupo – Nível 01:

Níveis: (01) → Podemos criar ITENS DE GRUPO, como uma espécie de hierarquia, sendo que o item principal sempre será o nível 01, os sub-itens serão definidos de 02 à 49.

Iremos utilizar múltiplos de 5 para os níveis:

01	WS-ITEM.		
05	WS-CODIGO	PIC	9 (003).
05	WS-INDICATIVO	PIC	X (001).
05	WS-DATA.		
10	WS-DIA	PIC	9 (002).
10	WS-MES	PIC	9 (002).
10	WS-ANO.		
15	WS-SC	PIC	9 (002).
15	WS-AA	PIC	9 (002).

MOVE

COMANDO MOVE:

Movimenta valores entre variáveis, strings, números ou constantes.

Exemplo:

```
MOVE 'BRASIL' TO WRK-PAIS  
MOVE WRK-SALARIO TO WRK-SAL-FINAL
```

FORMATAÇÃO DE VARIÁVEIS:

Devemos ter certos cuidados no tratamento de algumas variáveis devido ao seu formato, ou seja, sua **PIC**.

Comando	Origem	Destino	Observações	
Mover	Alfa	Numérico	Tem que ter o mesmo tamanho	Abendar / Truncar
Mover	Alfa	Numérico(comp)	Nunca	Abendar
Mover	Alfa	Numérico(comp-3)	Nunca	Abendar
Mover	Alfa pic x(10) 'ABCDE12345'	Alfa pic x(05) 'ABCDE'	O campo alfa é alinhado da esquerda para a direita.	Truncar

VARIÁVEIS EDIÇÃO

VARIÁVEIS DE EDIÇÃO:

Os campos abaixo são chamados de **EDIÇÃO**

01 WS-QTDE	PIC ZZ9.
01 WS-PAGINA	PIC Z.ZZ9.
01 WS-VALOR	PIC Z.ZZZ.ZZ9,99.
01 WS-VALOR	PIC -Z.ZZZ.ZZ9,99.
01 WS-VALOR	PIC Z.ZZZ.ZZ9,99-.
01 WS-VALOR	PIC -.---.--9,99.
01 WS-SQLCODE	PIC +++9.
01 WS-DATA	PIC 99/99/9999.

FILLER

CLÁUSULA FILLER:

É uma palavra reservada do Cobol e é usada para um item elementar ou um item de grupo, e nunca será referenciado, ou seja, não é possível fazer move de FILLER, ou até mesmo IF's.

Pode ser usada na “DATA DIVISION” e suas “SECTIONS”. (Working)

Exemplo:

```
01  REGISTRO.  
   02  FILLER      PIC X(100).
```

Operadores Aritméticos

CLÁUSULA ADD:

Esta cláusula é válida somente para itens elementares numéricos.

Por ela, são somados dois ou mais operando e o resultado guardado numa variável definida pelo programa.

Sintaxe:

ADD WS-CAMPO1 TO WS-CAMPO2.

CLÁUSULA SUBTRACT:

Esta cláusula é utilizada para subtrair itens numéricos

Sintaxe:

SUBTRACT 001 FROM WS-PAGINA.

CLÁUSULA MULTIPLY:

Esta cláusula é usada para multiplicar um item numérico por outro numérico.

Sintaxe:

MULTIPLY WS-A BY WS-B GIVING WS-C.

CLÁUSULA DIVIDE:

Esta cláusula é utilizada para efetuar o comando de divisão entre campos de itens numéricos.

Tomar cuidado em divisões por Zeros (abenda)

Exemplo:

DIVIDE A BY B GIVING C.

Operações que serão efetuadas acima = $(A / B) = C$

CLÁUSULA COMPUTE:

Operandos que a cláusula compute pode executar:

Adição (+);

Subtração (-);

Multiplicação ();*

Divisão (/);

*Exponenciação (**)*

CURSO COBOL

Suponhamos que desejamos calcular uma taxa cujo valor é de 5 percentuais do capital:

```
COMPUTE WS-TAXA = ( WS-CAPITAL * 0,05 ).
```

Outros exemplos:

```
COMPUTE WS-VALOR-A = (WS-TAXA * 0.15 + WS-NUM / WS-DIV).
```

```
COMPUTE RESULTADO = (CAMPO-B * CAMPO-B * CAMPO-A).
```

```
COMPUTE CAMPO-Z = (CAMPO-A / CAMPO-B) * CAMPO-C.
```

```
COMPUTE WS-RESULT1
```

```
    WS-RESULT2
```

```
    WS-RESULT3 = ( WS-BRUTO * 3 / (15 – CALC) ).
```

No compute, as operações obedecem a hierarquia das operações. Caso se queira efetuar uma operação de nível mais inferior antes de uma superior, deve-se colocar a de nível de interesse primeiro entre parênteses.

As operações aritméticas seguem a sua hierarquia, exceto quando estiverem entre parênteses.

Comando de decisão

CLÁUSULA IF:

Esta cláusula indica uma decisão Lógica em um programa.

Exemplo:

```
IF   WRK-FLAG                                EQUAL 'S'  
    MOVE WS-CAMPO1                          TO   WS-CAMPO2  
END-IF.
```

Um programa Cobol poderia testar o rendimento mensal da seguinte forma:

```
IF   WS-RENDIMENTO                          NOT EQUAL 1000,00  
    PERFORM                                  DESCONTO-MAX  
ELSE  
    PERFORM                                  DESCONTO-MIN  
END-IF.
```


Operação de Relação:

<u>OPERAÇÃO DE RELAÇÃO</u>	<u>SIGNIFICADO</u>
GREATER / NOT GREATER	MAIOR QUE / NÃO MAIOR QUE
LESS / NOT LESS	MENOR QUE / NÃO MENOR QUE
EQUAL / NOT EQUAL	IGUAL / NÃO IGUAL

Exemplos:

```
IF AC-LINHA                GREATER      50
   PERFORM ROTINA-CABECALHO
END-IF.
```

CONTINUE:

O comando “CONTINUE” é um comando não operacional. É indicado quando nenhuma instrução executável será utilizada.

```
IF    WS-FS-CADPECA           EQUAL    10
      CONTINUE
ELSE
      ADD 001                  TO    WS-LIDOS
END-IF.
```

EVALUATE:

Realiza o mesmo trabalho de If's encadeados

```
EVALUATE      PECA-COD-PEC

      WHEN      100
        MOVE    'FEDERACAO'  TO      WS-NOME-PEC

      WHEN      200
        MOVE    'SINDICATO'  TO      WS-NOME-PEC

      WHEN      300
        MOVE    'MINISTERIO' TO      WS-NOME-PEC

      WHEN      OTHER
        MOVE    'INVALIDO '  TO      WS-NOME-PEC

END-EVALUATE.
```

Operadores Lógicos

OPERADORES LÓGICOS:

Existem no Cobol 3 (três) operadores lógicos:

<i>OPERADOR LÓGICO</i>	<i>SIGNIFICADO</i>
<i>OR</i>	<i>Se ao menos um for verdadeiro, o resultado será verdadeiro.</i>
<i>AND</i>	<i>Se todos forem verdadeiros, o resultado será verdadeiro</i>
<i>NOT</i>	<i>Negação lógica</i>

Pode-se utilizar parênteses tanto para esclarecer o sentido das comparações, quanto para obter outros efeitos.

NIVEL 88

Variável

tipo Lógica

CURSO COBOL

NÍVEL 88:

Níveis: (88) -> especifica condições que devem ser associadas a valores particulares.

São variáveis do tipo BOOLEANO, que trata de dados lógicos:

TRUE - Verdadeiro

FALSE - Falso

Exemplos:

WORKING-STORAGE SECTION.

01	WRK-ENTIDADE	PIC	9(005)	VALUE ZEROS.
88	WRK-COD-ENTID			VALUE 100,
				200,
				300.

Comando de Repetição

BOA PRÁTICA:

Programas bem projetados e estruturados são aqueles que possuem uma série de construções lógicas, em que a ordem na qual as instruções são executadas é padronizada.

Em programas estruturados, cada conjunto de instruções que realiza uma função específica é definido em uma rotina ou, em COBOL, um parágrafo.

Ele consiste em uma série de instruções relacionadas entre si, em programação estruturada, os parágrafos são executados por meio da instrução **PERFORM.**

CLÁUSULA PERFORM:

Esta cláusula ocasiona a execução de um ou mais procedimentos.

Após a execução dos procedimentos (parágrafos), o controle volta para a instrução seguinte a do “PERFORM”.

PERFORM 0100-00-PROCED-INICIAIS.

PERFORM 0200-00-PROCED-PRINCIPAIS.

PERFORM 0300-00-PROCED-FINAIS.

PERFORM UNTIL:

Com a opção “UNTIL”, os procedimentos serão executados até que a condição após o “UNTIL” seja verdadeira.

No programa, ao encontrar a cláusula “UNTIL”, primeiro é verificado se a condição do “UNTIL” já está satisfeita e depois executa o “PERFORM”.

Sintaxe:

```
PERFORM 200-00-PROCED-PRINCIPAIS  
      UNTIL WS-FIM                EQUAL 'SIM'.
```

CURSO COBOL

PERFORM VARYING:

```
PERFORM 660-00-PROCEDIMENTO1  
  VARYING WS-IND          FROM -01 BY 01  
    UNTIL WS-IND          GREATER 050  
      OR WS-FLAG-FIM EQUAL 'SIM'
```

```
PERFORM 660-00-PROCEDIMENTO1  
  VARYING WS-IND          FROM 500 BY -01  
    UNTIL WS-IND          EQUAL ZEROS
```

CLÁUSULA GO TO:

Permite a transferência da parte do programa que está sendo executada para uma outra.

Sintaxe :

```
125-00-LEITURA  SECTION.
```

```
    READ CADPECA AT END  
      GO TO 125-99-LEITURA.
```

```
    ADD 001 TO WS-LIDOS.
```

```
125-99-LEITURA.
```

Indica-se para uma melhor estruturação da lógica e do programa, não executar o comando GO TO para desviar para fora da rotina em que foi colocado.

Vetores

TABELAS INTERNAS:

Alguns algoritmos mais avançados exigem a definição da mesma variável várias vezes.
Exemplo: Acumular as vendas do ano separadas por mês.

Formato: PIC XXX OCCURS n TIMES

```
01  TABELA-DE-MESES.  
    03  TAB-MESES.  
        05  FILLER      PIC  X(009) VALUE 'JANEIRO ' .  
        05  FILLER      PIC  X(009) VALUE 'FEVEREIRO'.  
        05  FILLER      PIC  X(009) VALUE 'MARCO   ' .  
        05  FILLER      PIC  X(009) VALUE 'ABRIL   ' .  
        05  FILLER      PIC  X(009) VALUE 'MAIO    ' .  
        05  FILLER      PIC  X(009) VALUE 'JUNHO   ' .  
        05  FILLER      PIC  X(009) VALUE 'JULHO   ' .  
        05  FILLER      PIC  X(009) VALUE 'AGOSTO  ' .  
        05  FILLER      PIC  X(009) VALUE 'SETEMBRO'.  
        05  FILLER      PIC  X(009) VALUE 'OUTUBRO ' .  
        05  FILLER      PIC  X(009) VALUE 'NOVEMBRO'.  
        05  FILLER      PIC  X(009) VALUE 'DEZEMBRO'.  
01  TAB-MESES-R REDEFINES TAB-MESES OCCURS 12 TIMES.  
    03  MESES          PIC  X(009).
```

BOOKS

Reaproveitamento de código

COPY:

Usado para incorporar logicamente o conteúdo de arquivos tipo texto ao arquivo do programa fonte.

```
COPY  COBWS001.
```

Trocado dados entre programas

CALL :

Utilizado para realizar a chamada de um subprograma ou sub-rotina.

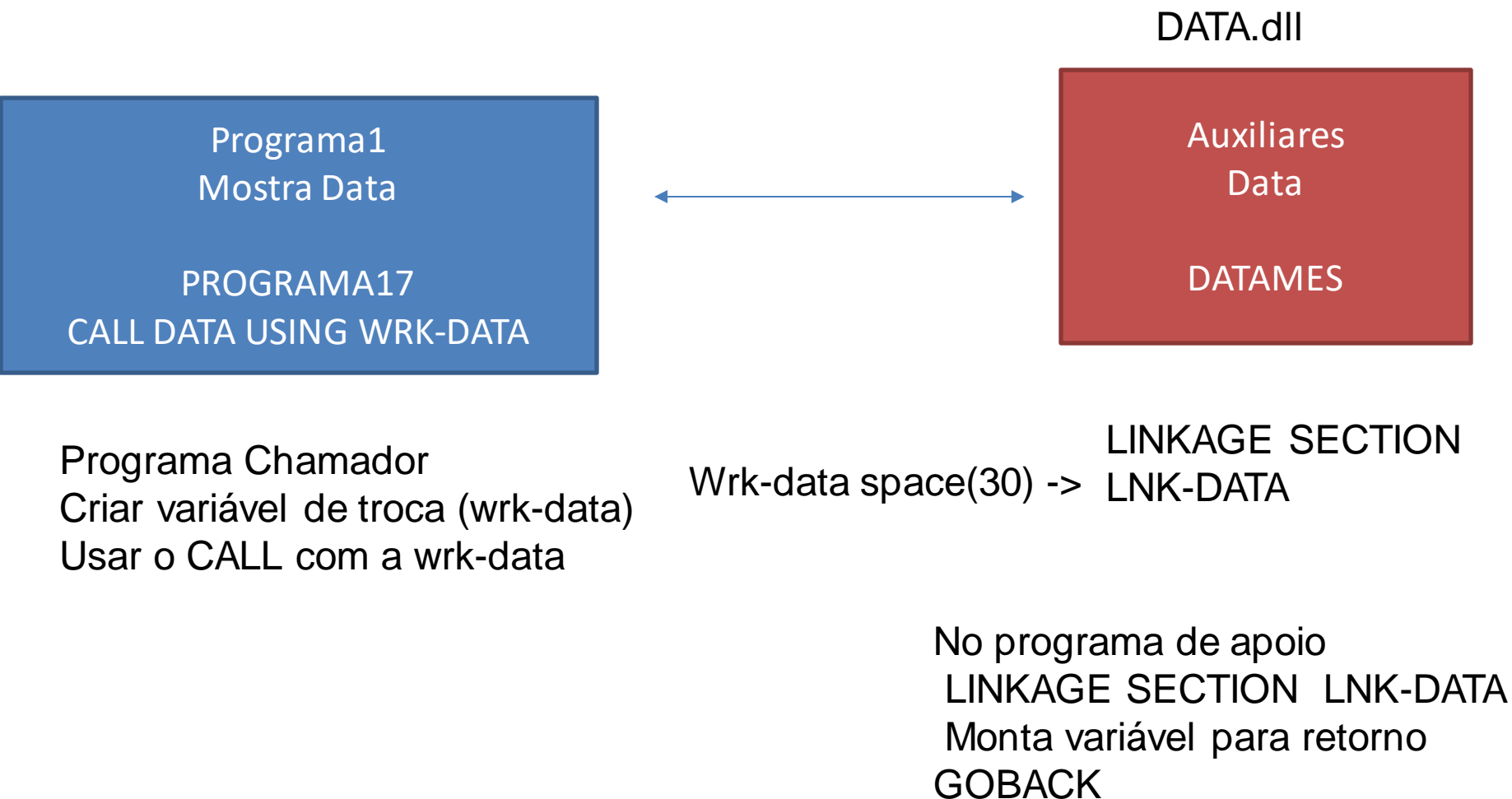
```
CALL  'DATA'           USING                WRK-DATA
```

```
CALL  'WRK-SUBPGM1'    USING                WS-AREA-PGM1
```

LINKAGE SECTION:

O seu funcionamento é parecido com o da Working, com a diferença que os dados aqui declarados serão compartilhados com outro programa, podendo tanto enviar como receber dados nessa área. (É uma área de uso comum para comunicação entre os programas).

CURSO COBOL



CLÁUSULA GOBACK:

Termina o processamento de uma ligação entre programas, ou pode ser utilizado como o “STOP RUN”.

Muito utilizado em programas Batch (módulos Batch, Subrotinas ou API)

Sintaxe:

GOBACK.

Arquivo de Dados

ARQUIVOS DE DADOS

SEQUENCIAIS (QSAM)

VSAM

DB2

ARQUIVOS DE DADOS - SEQUENCIAIS

A organização de arquivo trata do arranjo ou a forma de distribuição dos registros dentro do arquivo, objetivando agilizar o processo de armazenamento e recuperação de dados.

Um arquivo é organizado, logicamente, como uma seqüência de registros que são mapeados nos dispositivos de memória.

ARQUIVOS DE DADOS - SEQUENCIAIS

Os registros podem assumir tamanhos fixos e variáveis:

- *Tamanho fixo - todos os registros do arquivo possuem o mesmo comprimento em bytes;*
- *Tamanho variável - os registros podem assumir um comprimento entre dois limites especificados, máximo e mínimo.*

ARQUIVOS DE DADOS - SEQUENCIAIS

Acesso a um registro

Chaves



00001 ALVARO DIAS
00002 BRUNA
LOMBARDI
00003 SILVIO SANTOS
00004 CARLOS
IMPERIAL

Variavel



QualCodigo: _00003

Processo:

```
Enquanto não for final de arquivo (EOF)
{
  Ler registro
  Comparar Chave com Variavel
  Se Igual mostra NOME
  Senao Ler o próximo registro
}
```

ARQUIVOS DE DADOS - SEQUENCIAIS

Acesso a um registro

O argumento de pesquisa é comparado com cada registro lido, de forma sequencial.

Inclusao de um registro

Deve ser gerado um novo arquivo a partir do atual, intercalando o registro novo com base na chave primaria.

ARQUIVOS DE DADOS - INDEXADOS

Possui índice (CHAVE) para pesquisa

Localização de registro é feito de forma direta, através do índice. Melhor desempenho para pesquisa.

No Mainframe existe diferentes formatos: ESDS, RRDS, KSDS.

ARQUIVOS DE DADOS - INDEXADO

Acesso a um registro

Chave	INDIC
00001 ALVARO DIAS	0000
00002 BRUNA LOMBARDI	1
00004 CARLOS IMPERIAL	0000
00003 SILVIO SANTOS	2
	0000
	3
	4

Variavel-
Chave
Qual Codigo: _00003

Processo:

Ler registro através da Chave (Variavel)
Se encontrada mostra NOME
Senao indica final de arquivo

TABELAS – DB2/SQL

Código funcional	Nome do gerente	Data contratação	Telefone
100	Jorge Luis Almeida	10/12/1999	4521-7767
101	Maria Rita Santos	05/03/2005	4521-9089
102	Claudio Correa	01/06/1971	4521-7766

GERENT

- **E** As tabelas podem se relacionar entre si.

CLIENT

E

Código do cliente	Nome do cliente	Código do gerente	Segmento
900.876	Célia da Silva	102	Prime
899.567	Mário Costa	100	Private
123.321	Elisabeth Alves	101	Varejo

CLÁUSULA OPEN:

Abre arquivo de entrada e saída, seqüenciais e Vsam's

Sintaxe:

OPEN	INPUT	ARQUIV1
		ARQUIV2
	OUTPUT	ARQUIV3
		ARQUIV4
	I-O	ARQUIV5.

CLÁUSULA READ:

Ler um registro do arquivo de entrada.

Sintaxe :

```
READ ARQUIVO1.      ou  
READ  ARQUIVO1 INTO AREA-ARQ1.
```

CLÁUSULA WRITE:

Transfere um registro do programa para um arquivo de saída ou impressora de relatórios.

(nível 01 definido na FD (book))

Sintaxe:

```
WRITE AREA-SAIDA.
```

(nível 01 definido na FD + book (definido na Work)).

Sintaxe:

```
WRITE AREA-SAIDA FROM WS-AREA-1.
```

FROM -> faz com que uma área seja movida da “WORKING-STORAGE
O “WRITE” só deve ser dado em cima do nível “01”.

CLÁUSULA CLOSE:

O CLOSE é utilizado para fechar os arquivos que foram abertos.

Quando este comando não for utilizado, o próprio sistema se encarregará de fechá-los.

Sintaxe:

CLOSE normal p/ disco e fita:

CLOSE CADPECA.

CLOSE CADFIL1
 CADFIL2.

INITIALIZE:

Usado para inicializar áreas de trabalho do programa.

```
INITIALIZE      REG-CADPATU.
```

FILE STATUS:

O FILE STATUS permite ao usuário monitorar a execução de operações de entrada e saída (I/O) requisitadas para os arquivos de um programa.

Após cada operação de I/O, o sistema move um valor para a STATUS KEY

(campo alfa/numérico, com 2 caracteres definidos na WORKING-STORAGE SECTION e especificado na ENVIRONMENT DIVISION, através do SELECT) que acusa o sucesso ou o insucesso da operação.

Qualquer valor movido para a STATUS KEY diferente de zeros, revela que a execução não foi bem sucedida.

Diversos

VARIÁVEIS NUMÉRICAS BINÁRIAS:

O campo binário oferece uma maior capacidade de representação dentro de um byte. Em um campo de 4 casas o valor 6859 é armazenado em dois bytes.

Os campos abaixo são chamados de **BINÁRIOS**

01	WS-COUNT	PIC S9(004)	COMP VALUE +0.
01	WS-VALOR	PIC S9(013)V9(002)	COMP VALUE +0.
01	WS-CODIGO	PIC 9(004)	BINARY.

VARIÁVEIS NUMÉRICAS COMPACTADAS:

Neste campo cada algarismo é representado em meio byte e o meio byte mais à direita contém o sinal do campo.

Os campos abaixo são chamados de **COMPACTADOS**

01	WS-QTDE	PIC S9(004)	COMP-3	VALUE	+0.
01	WS-VALOR	PIC S9(013)V9(002)	COMP-3	VALUE	+0.

CLÁUSULA REDEFINES:

É usada para re-escrever uma área, a redefinição deverá conter a mesma quantidade de bytes do campo ou área anterior e estar no mesmo nível.

Formatação de Data Oriunda do Sistema Operacional (AAMMDD)

01	WRK-DATE	PIC	9(006) VALUE ZEROS.
01	FILLER	REDEFINES	WS-DATE.
05	WRK-ANO-DATE	PIC	9(002).
05	WRK-MES-DATE	PIC	9(002).
05	WRK-DIA-DATE	PIC	9(002).

Transformação de Campo Alfanumérico Para Numérico ou Vice-Versa.

01	WRK-NUM-017	PIC	9(015)V99.
01	WRK-ALF-017	REDEFINES	WS-NUM-017
		PIC	X(017).

CLÁUSULA REDEFINES:

A cláusula REDEFINES é usada quando alguma informação da DATA DIVISION precisa ser definida em dois ou mais formatos:

```
01 WRK-DATA                PIC 9(006).
```

```
01 WRK-DATA-RED            REDEFINES WRK-DATA.  
    03 WRK-DIA              PIC 9(002).  
    03 WRK-MES              PIC 9(002).  
    03 WRK-ANO              PIC 9(002).
```

NEXT SENTENCE:

O comando “NEXT SENTENCE” determina que nada será feito e deve-se continuar o processo após o primeiro ponto final ou “END-IF” encontrado.

```
IF      WS-FS-CADPECA                EQUAL    10
      NEXT SENTENCE
ELSE
      ADD 001                TO    WS-LIDOS
END-IF.
```