MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME
MAINFRAME

Informática e Computação

Cursos
Treinamentos
Consultoria
Sistemas

Desde 2000

Ulisses & Moraes – TI

C O B O L
Linguagem de Programação

http://sites.google.com/site/ulissesemoraes/
e-mail: ulissesemoraes@yahoo.com.br

Rua Francisco Perez, 200 – Jd. Monte Alegre
Taboão da Serra – SP
Cel. 9.9879-0971 / 4771-1496

ÍNDICE

Ulisses & Moraes Informática - TI

Linguagem de Programação - COBOL

CAPACITAÇÃO NA LINGUAGEM DE PROGRAMAÇÃO COBOL MVS

Ulisses Moraes - TI

Técnologia da Informação

Treinamento Profissional / Desenvolvimento de Sistemas

* Lógica de Programação - Cobol (Mainframe)
* Fluxograma - Diagrama de Blocos
* MFE - Mainframe Express
* Linguagem de Programação Estruturada - Cobol II MVS (Mainframe)
* JCL - Job Control Language
* Modelagem de Dados (MER) Modelo Conceitual / Lógico / Físico
* Banco de Dados Relacional DB2 / SQL - Query / DML
* CICS - Comand Level
* Vsam KSDS - Processamento Batch / On-Line

Equipamento de Grande Porte

OS/390 MVS Z/OS

"Porque Cobol? Mais de 95% das aplicações financeiras são processadas em Cobol, 75% das transações em Mainframe são feitas usando Cobol. As empresas que possuem mainframes são empresas grandes que possibilitam melhores salários e estabilidade".

Ulisses & Moraes Informática - TI	Conheça o Mercado de Trabalho acesse www.apinfo.com
Prestação de Serviços em Análise de Sistemas, Programação, Treinamentos / Cursos, Manutenção e Desenvolvimento de Projetos, Banco de Dados e Suporte Técnico a Sistemas.	O Site dos profissionais da área de tecnologia da informação
15 Anos na Área de TI Atuando em Desenvolvimento de Sistemas 13 Anos na Área de TI Ministrando Cursos/Treinamentos e Palestras	ulissesemoraes@yahoo.com.br
* Treinamento para Empresas * Treinamento Particular * Aulas em Faculdades	ulisses.souza@spread.com.br
	http://sites.google.com/site/ulissesemoraes/

Ulisses & Moraes Informática – TI

Prestação de Serviços em Análise de Sistemas, Programação,Treinamentos / Cursos, Manutenção e Desenvolvimento de Projetos, Banco de Dados e Suporte Técnico a Sistemas.

15 Anos na Área de TI Atuando em Desenvolvimento de Sistemas
13 Anos na Área de TI Ministrando Cursos/Treinamentos e Palestras

* Treinamento para Empresas
* Treinamento Particular
* Aulas em Faculdades

http://sites.google.com/site/ulissesemoraes/

1. Objetivo

Capacitar o Aluno a Analisar/Desenvolver e a Realizar Manutenções em Programas na Linguagem de Programação Cobol Voltado para a Plataforma Alta Mainframe.

2. Introdução

Este Manual Oferece uma Visão Geral dos Comandos sua Sintaxe, Conceitos e as Melhores Práticas de Programação Estruturada.

O foco do Curso é Totalmente Voltado à Programação COBOL, processamento BATCH.

3. 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).

4. 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 a seguinte estrutura:

5. Coluna de 1 a 6

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

A numeração é uma ordem crescente. (COMPILAÇÃO)

Opcionalmente podem deixar de serem preenchidas ou incluir outros caracteres.

```
      *-----------------------------------------------------------------*
       PROCEDURE    DIVISION    USING    LKG-PARM.
      *-----------------------------------------------------------------*
      *
           PERFORM 0100-00-PROCED-INICIAIS.

LOOP       PERFORM 2000-00-PROCED-PRINCIPAIS
LOOP         UNTIL WS-FS-TCSAER27 EQUAL 10.

           PERFORM 9000-00-PROCED-FINAIS.
      *
      *-----------------------------------------------------------------*
       0100-00-PROCED-INICIAIS     SECTION.
      *-----------------------------------------------------------------*
      *
```

6. 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.

7. 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.

8. 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

9. Palavras Chaves

São essenciais às especificações de um programa. A omissão dessas palavras acarretará resultados errôneos na compilação.

10. Palavras Opcionais

São palavras não obrigatórias, servindo apenas para um melhor entendimento do programa. Dependendo da necessidade da aplicação.

11. Cláusulas Especiais – EJECT

Especifica que a próxima instrução no programa fonte deverá ser impressa no início da próxima página da listagem do programa fonte.

12. Cláusulas Especiais – SKIP1 / SKIP2 / SKIP3

Especifica que a próxima instrução no programa fonte deverá ser impressa precedida de linhas em branco.

13. Cláusulas Especiais – TITLE

Especifica um título a ser impresso no início de cada página da listagem do programa fonte durante a compilação.

14. 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).

15. 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).

16. Seções

Podemos identificar dois tipos de seções:

Definidos na ENVIRONMENT DIVISION e DATA DIVISION conforme requeridos.

Exemplo:

CONFIGURATION SECTION.

WORKING-STORAGE SECTION.

Na PROCEDURE DIVISION para especificar a segmentação do programa.

17. Parágrafos

Na PROCEDURE DIVISION é utilizada para agrupar sentenças, permitindo a alteração do fluxo lógico. (SECTION'S) "Blocos Funcionais / Processos Pré-Definidos)

- 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

18. Sentenças

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

19. IDENTIFICATION DIVISION

Identifica e documenta o programa fonte (Aplicação - informação obrigatória), e outras informações como autor, local de criação, data de criação, data de compilação e descrição do objetivo do programa.

É a primeira das 4 divisões.

Esta divisão não é formada por nenhuma seção.

É utilizada para identificar o programa. (Como se fosse um Cabeçalho).

IDENTIFICATION DIVISION.
PROGRAM-ID. (nome do programa)
AUTHOR. (nome do programador) (**Opcional**)
INSTALLATION. (local de uso ou geração do programa) (**Opcional**)
DATE-WRITTEN. (data em que foi escrito o programa) (**Opcional**)

Obs: a opção DATE-COMPILED não deve ser preenchida, pois o sistema operacional que fará isso.

DATE-COMPILED. (data em que foi compilado o programa) (**Opcional**)
REMARKS / SECURITY. (comentários sobre o programa. Utilizar o REMARKS em programas desenvolvidos em Cobol ANS, isto é, Cobol I). (**Opcional**)

Exemplo:

```
************************************************************************
 IDENTIFICATION              DIVISION.
************************************************************************
*
 PROGRAM-ID.                 PROG001.
 AUTHOR.                     ANTONIO CARLOS.
 DATE-WRITTEN.               19/09/2003.
 SECURITY.
*
************************************************************************
*  SISTEMA......: SIMAN - SISTEMA DE APRENDIZADO NA LINGUAGEM    *
************************************************************************
*  ANALISTA.....: JOSE SILVA                                          *
*  LINGUAGEM....: COBOL/BATCH                                         *
*  PROGRAMADOR..: ANTONIO CARLOS                                      *
*  DATA.........: 19/09/2003                                          *
************************************************************************
*  OBJETIVO.....: A  PARTIR  DO  CADASTRO  DE  PECAS,  GERAR     *
*                 CADASTRO DE PECAS ATUALIZADO.                      *
************************************************************************
```

20. 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.

É a segunda divisão do Cobol.

Identifica a máquina que está sendo usada.

Contém a descrição do computador e a designação dos arquivos para as respectivas unidades de configuração do computador.

A ENVIRONMENT DIVISION está subdividida em duas seções opcionais, isto é, que deverão ser escritas ou não, dependendo da necessidade da aplicação.

CONFIGURATION SECTION. (Seção de Configuração)
INPUT-OUTPUT SECTION. (Seção de Entrada/Saída)

21. CONFIGURATION SECTION

Esta seção é utilizada para fornecer informações sobre o computador.

É dividida em 3 parágrafos:

SOURCE-COMPUTER. (nome-do-computador). (**Opcional**)
OBJECT-COMPUTER. (nome-do-computador). (**Opcional**)
SPECIAL-NAMES. (nome-da-função IS nome-simbólico).

22. 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.

Se a aplicação Manipular informações Financeiras fazendo o uso do Dólar, poderemos omitir esta cláusula.

23. INPUT-OUTPUT SECTION.

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

24. FILE-CONTROL

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

```
*****************************************************************
 FILE-CONTROL.
*****************************************************************
*
       SELECT  CADPECA   ASSIGN  TO  UT-S-CADPECA
               FILE      STATUS  IS  WS-FS-CADPECA.
```

25. SELECT

O Select tem a função de designar um arquivo para um dispositivo de entrada/saída, é necessário um "Select" para cada arquivo.

(DDNAME) ***Nome-do-arquivo*** *-> é dado pelo analista. É o nome pelo qual o arquivo será reconhecido na "DATA DIVISION" e "PROCEDURE DIVISION".*

O mesmo será reconhecido pelo cartão "DD" do "JCL".

Classe *-> especifica o tipo de dispositivo:*

DA = acesso direto (discos magnéticos)

UT = utility (fitas e discos magnéticos)

Organização -> indica a organização do arquivo:

S = arquivos seqüenciais

D = arquivos de acesso direto

I = arquivos de organização indexada

```
*****************************************************************
 FILE-CONTROL.
*****************************************************************
*
       SELECT  CADPECA    ASSIGN  TO  UT-S-CADPECA
               FILE       STATUS  IS  WS-FS-CADPECA.
```

O LABEL **UT-S-** não é obrigatório, porém devemos observar o erro file-status = 35,

- ✓ Este erro poderá ocorrer devido a arquivo vazio criado manualmente.
- ✓ DDNAME não bate entre JCL e seu programa.
- ✓ Ambiente obriga criar o LABEL com o mesmo nome do arquivo DDNAME, ou seja, não utilizar "UT-S-".

Exemplo:

```
*****************************************************************
 ENVIRONMENT                  DIVISION.
*****************************************************************
 CONFIGURATION                SECTION.
*****************************************************************
 SPECIAL-NAMES.               DECIMAL-POINT   IS    COMMA.
*****************************************************************
 INPUT-OUTPUT                 SECTION.
*****************************************************************
 FILE-CONTROL.
*****************************************************************
*
       SELECT  CADPECA   ASSIGN  TO  UT-S-CADPECA
               FILE      STATUS  IS  WS-FS-CADPECA.
*
       SELECT  CADPATU   ASSIGN  TO  UT-S-CADPATU
               FILE      STATUS  IS  WS-FS-CADPATU.

       SELECT  CADVSAM   ASSIGN  TO  DA-I-CADVSAM
               ORGANIZATION      IS  INDEXED
               ACCESS  MODE      IS  DYNAMIC
               RECORD KEY        IS  VSAM-COD-CHAVE
               FILE STATUS       IS  WS-FS-CADVSAM.
*
*****************************************************************
 DATA                         DIVISION.
*****************************************************************
```

26. 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.

27. FILE SECTION.

Descreve o conteúdo e a organização dos arquivos.

O primeiro nível na "FILE SECTION" é por intermédio de uma entrada (FD – FILE DESCRIPTION).

Cada "FD" descreve o arquivo do Select.

O segundo nível é descrito por uma entrada "01".

Indicador	***Uso***
FD	*descrição de arquivos*
SD	*descrição de "sort-files"*

O formato dos números e níveis serve para estruturar logicamente o registro.

As subdivisões de um registro são "itens elementares" (não possuem subdivisões) e "itens de grupo".

Exemplo:

```
*
*****************************************************************
 DATA                          DIVISION.
*****************************************************************
 FILE                          SECTION.
*****************************************************************
* INPUT..: CADPECA - CAD DE PECAS            - LRECL.: 100 BYTES *
*****************************************************************
*
 FD  CADPECA
     RECORDING  MODE       IS  F
     LABEL      RECORD     IS  STANDARD
     BLOCK      CONTAINS   0   RECORDS.

 01          REG-CADPECA    PIC     X(100).
*
*****************************************************************
* OUTPUT.: CADPATU - CAD DE PECAS ATUALIZADO - LRECL.: 100 BYTES *
*****************************************************************
*
 FD  CADPATU
     RECORDING  MODE       IS  F
     LABEL      RECORD     IS  STANDARD
     BLOCK      CONTAINS   0   RECORDS.

 01          REG-CADPATU    PIC     X(100).
*
*****************************************************************
 WORKING-STORAGE               SECTION.
*****************************************************************
*
```

28. FILE DESCRIPTION (FD)

É a descrição do arquivo.

```
                                (V-Formato do Arquivo "Variável")
                                (F-Formato do Arquivo "Fixo")
FD   CADPATU
     RECORDING  MODE      IS  F
     LABEL      RECORD    IS  STANDARD   (Formato do Label)
     BLOCK      CONTAINS  0   RECORDS.   (Quantidade de Blocos)
```

29. RECORDING MODE

(RECORDING MODE IS X) - Designa o formato do registro:

FIXO	(F)
FIXO BLOCADO	(FB)
VARIÁVEL	(V)
VARIÁVEL BLOCADO	(VB)
VARIÁVEL SPANNED	(VS)
VARIÁVEL BLOCADO SPANNED	(VBS)
UNDEFINED	(U)

Se não for colocada a cláusula "RECORDING MODE", o compilador determinará pelo cartão "DD" ou catálogo.

30. LABEL RECORD

(LABEL RECORD IS XXXXXXXX) - Especifica o formato do label.

Quando omitido assume "LABEL STANDARD".

STANDARD -> padrão
OMITTED -> omitido

Para impressora, leitora de cartões, perfuradoras, usar "OMITTED", pois não possuem "LABELS". Os demais casos usar "STANDARD".

31. BLOCK CONTAINS

(BLOCK CONTAINS 9999 RECORDS) - Especifica o tamanho do registro físico.

Se for colocado zero (0), assume informações do cartão "DD".

Se não for colocado "RECORDS", assume "CHARACTERS".

32. RECORD CONTAINS

(RECORD CONTAINS 9999 CHARACTERS) - Especifica o tamanho do registro lógico.

Se esta cláusula for colocada, é feita uma conferência pelo compilador, somando a quantidade de bytes da definição do registro.

33. DATA RECORD

(DATA RECORD IS NOME-DO-DADO-1)
(DATA RECORD ARE NOME-DADO-1, NOME-DADO-2, ...)

Serve apenas como documentação, identificando os registros do arquivo pelo nome.

Exemplo:

```
<...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
       DATA                          DIVISION.
       FILE                          SECTION.
       FD   FITA
            RECORDING  MODE      IS  F
            LABEL      RECORD    IS  STANDARD
            RECORD     CONTAINS  80  CHARACTERS
            BLOCK      CONTAINS  20  RECORDS
            DATA       RECORD    IS  RECIBO.

       01    RECIBO.
         03  NOME                  PIC       X(030).
         03  VALOR                 PIC       9(003)V9(002).
         03  FILLER                PIC       X(005).
```

34. 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
- ✓ Copy books de Subrotinas
- ✓ Áreas do DB2

35. 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:

- ✓ WS-IMPOSTO-RENDA
- ✓ WS-FIM
- ✓ WS-LID-CADPECA
- ✓ WS-GRV-CADPATU
- ✓ WS-DES-CADLIDO
- ✓ WS-GRAVADOS
- ✓ WRK-FLAG
- ✓ W-VALOR
- ✓ AC-ACUMULADOR
- ✓ LT-LITERAL
- ✓ SR-SUBROTINA

```
*
*****************************************************************
 WORKING-STORAGE              SECTION.
*****************************************************************
*
 01          WS-FS-TCSAEX16  PIC     9(002) VALUE ZEROS.
 01          WS-FS-TCSASX16  PIC     9(002) VALUE ZEROS.
*
 01          WS-QTD-GUIAS    PIC     9(018)    VALUE ZEROS.
 01          WS-VLR-GUIAS    PIC     9(016)V99 VALUE ZEROS.
*
```

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

Criar nomes de variáveis com significativos lógicos.

36. Constantes figurativas

É uma palavra associada a um valor particular.

Exemplos:

ZEROS, ZERO, ZEROES → valor zero, ou o próprio numero 0

SPACE, SPACES → valor brancos ou pode ser representado ' '

- ALL '-' → um ou mais ocorrências de literal

Representa uma ou mais ocorrências de caractere que compõe o literal.

- LOW-VALUE, LOW-VALUES → menor valor (hexa 00) – Zeros Binários

Representa uma ou mais ocorrências do caractere X'00' (Representação em Hexadecimal do menor valor na seqüência de caracteres na representação EBCDIC).

A constante figurativa LOW-VALUES é tratado como um literal não numérico.

- HIGH-VALUE, HIGH-VALUES → maior valor (hexa FF)

Representa uma ou mais ocorrências do caractere X'FF' (Representação em Hexadecimal do maior valor na seqüência de caracteres na representação EBCDIC).

A constante figurativa HIGH-VALUES é tratado como um literal não numérico.

37. CLÁUSULA PICTURE (PIC)

É usada para descrição, definição de informações sobre itens, tais como: tamanho, sinal, tipo numérico (Zonado, Compactado, Binário), 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

Picture possíveis:

ALFABÉTICO -> é representado por letras mais o espaço, e o caractere usado é a letra "A"

Exemplos:

```
01  WS-DADO1       PIC          IS   AAA  VALUE  'ABC'.
01  WS-DADO2       PIC          IS   AAA  VALUE  'ABC'.
01  WS-DADO3       PIC  A(3)             VALUE  'BCD'.
```

38. VARIÁVEIS ALFANUMÉRICAS

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

Exemplos:

```
01  WS-DADO1       PIC XXX    VALUE  'ANO'.
01  WS-DADO2       PIC X(005) VALUE  'KKKKK'.
01  WS-NOME        PIC X(030) VALUE  'ANTONIO CARLOS'.
```

39. 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.

40. VARIÁVEIS NUMÉRICAS ZONADAS

Campo zonado é aquele onde um algarismo é representado em um byte no formato zona e dígito.

O tamanho máximo de dígitos deste tipo de campo é 18 que será representado em 18 bytes.

Para informar na definição que o campo é zonado, basta não codificar a cláusula USAGE, ou seja, após o "TAMANHO" colocar o ponto.

Os campos abaixo são chamados de **ZONADOS**

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

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

01  WS-VALOR        PIC S9(013)V99     VALUE  ZEROS.
01  WS-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

41. 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.
```

42. 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.
```

43. 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.
```

Definição	***Picture***	***Valor Real***	***Na memória***
9(04)	*9999*	*502*	*502*
9V9(2)	*9V99*	*1,25*	*125*
9(03)	*999PP*	*43700*	*437*
S9(02)	*S99*	*-21*	*21-*
9(05)	*99.999*	*10.987*	*10987*
9(04)V99	*Z.ZZZ,99*	*25,50*	*002550*
9(03)	*ZZZ*		*000*
S9(03)V99	*999,99CR*	*800,00CR*	*80000(-)*
9(04)	*990099*	*110025*	*1125*
9(06)	*99B99B99*	*12 13 15*	*121315*
9(03)	*$999*	*$371*	*371*
S9(02)	*-99*	*-15*	*15(-)*
S9(02)	*-99*	*16*	*16*
S9(02)	*+99*	*15*	*15(-)*

44. FORMATAÇÃO DE VARIÁVEIS

Cuidados que devemos ter no tratamento de algumas variáveis devido ao seu formato, ou seja, sua **PIC**.

Comando	Origem	Destino	Observações	
Mover	Alfa	Numérico(Zonado)	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
Mover	Alfa pic x(06) 'ABCDE1'	Alfa pic x(10) 'ABCDE1 '	O campo alfa é alinhado da esquerda para a direita.	Joga espaços à direita
Mover	Numérico(Zonado) Pic 9(006) 000123	Alfa Pic x(007) '000123 '	Tem que ter o mesmo tamanho. Tomar cuidado neste caso, devemos ter certeza de que a regra de negócio permite.	Truncar
Mover	Numérico(Zonado) Pic 9(006) 000123	Alfa Pic x(004) '0001'	Tem que ter o mesmo tamanho. Tomar cuidado neste caso, devemos ter certeza de que a regra de negócio permite.	Truncar
Mover	Numérico(comp)	Alf	**Nunca**	Abendar
Mover	Numérico(comp-3)	Alf	**Nunca**	Abendar
Mover	Numérico(comp)	Edição (zz9)	**Depende do Ambiente**	Abendar / Truncar
Mover	Numérico(comp-3)	Edição (zz9)	**Depende do Ambiente**	Abendar / Truncar
Mover	Alf	Edição (zz9)	**Nunca**	Abendar / Truncar
ADD	Edição (zz9)		**Nunca**	Nem Compila
ADD	Alf		**Nunca**	Nem Compila
Display	Numérico(comp)		**Depende do Ambiente**	Apresenta Low-values
Display	Numérico(comp-3)		**Depende do Ambiente**	Apresenta Low-values

```
  APPLID(A01ROSDR)    USER(DWT,X932838)                    J PENDING
> APPLID(A01ROSDR)    USER(DWT,X932838)                    J PENDING
> DSN()               SCRL CSR    COLS 00001 00073                 LINE 000001
> D.SS.TU3013.DT1207.SORT
>       <...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
        ******************************** T O P ***********************************
000001                          E  çoÈð          Éi                    DLR
000002            <             E  çoÈð           Ê%                   DLR
000003            *             E  çoÈð                                DLR
000004            %             E  çoÈð          áa                    DLR
000005            @             E  çoÈð          a ð                   DLR
000006            ð             E  çoÈð          pÂ                    DLR
000007                          E  çoÈð            *                   DLR
000008                          E  çoÈð           i                    DLR
000009                          E  çoÈð           î%                   DLR
000010                          E  çoÈð           ñ                    DLR
000011                          E  çoÈð          èj                    DLR
000012                          E  ç î                                 DLR
000013                          E  ç î             %                   DLR
```

```
        ************************** B O T T O M ***************************

 SET HEX ON
 SET HEX OFF


> APPLID(A01ROSDR)   USER(DWT,X932838)                    J PENDING
> DSN()              SCRL CSR   COLS 00001 00073                  LINE 000001
> D.SS.TU3013.DT1207.SORT
>      <...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
       ******************************** T O P ********************************
000001                          E  çoÈð          Éi                        DLR
       043010000001000000000000C00497800000000078100000000000000000CDD00000000
       01006C00000C00010000000C502864C000000000219C00000000C00000000C4390000C000
       ________________________________________________________________________
000002            <             E  çoÈð          Ê%                        DLR
       043010000004000000000000C00497800000000027600000000000000000CDD00000000
       01006C00000C00010000000C502864C000000000282C00000000C00000000C4390000C000
       ________________________________________________________________________
000003            *             E  çoÈð                                    DLR
       043010000005000000000000C00497800000000021300000000000000000CDD00000000
       01006C00000C00010000000C502864C000000000570C00000000C00000000C4390000C000
       ________________________________________________________________________
000004            %             E  çoÈð          áa                        DLR
       043010000006000000000000C00497800000000048200000000000000000CDD00000000
       01006C00000C00010000000C502864C000000000951C00000000C00000000C4390000C000
       ________________________________________________________________________


> APPLID(A01ROSDR)   USER(DWT,X932838)                    L PENDING
> AWS()              SCRL CSR   COLS 00001 00072                  A<TMP1>2
>       <...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
 ......  ******************************** T O P ********************************
 000001        01  DCLTU3017COMSCTRT.
 000003 00001      10 U00-NUCTRT          PIC S9(10)V USAGE COMP-3.
 000004 00007      10 U00-NURENOCTRT      PIC S9(4) USAGE COMP.
 000005 00009      10 U00-NUADIICTRT      PIC S9(7)V USAGE COMP-3.
 000006 00013      10 U00-NUSEQUCTRTINDL PIC S9(9) USAGE COMP.
 000007 00017      10 U00-NUITEMCTRT      PIC S9(9) USAGE COMP.
 000008 00021      10 U00-CDGARA          PIC S9(6)V USAGE COMP-3.
 000009 00025      10 U00-CDTIPOINEN      PIC  X(1).
 000010 00026      10 U00-CDPESS          PIC S9(10)V USAGE COMP-3.
 000011 00032      10 U00-CDTIPOREMU      PIC  X(2).
 000012 00034      10 U00-CDMOED          PIC  X(3).
 000013 00037      10 U00-PCCOMICOMSPRPT PIC S9(4)V9(5) USAGE COMP-3.
 000014 00042      10 U00-VLCOMICOMS      PIC S9(15)V9(2) USAGE COMP-3.
 000015 00051      10 U00-TSALTE          PIC  X(26).
 000016 00077      10 U00-CDUSUAALTE      PIC  X(10).
 ......  ***************************** B O T T O M ****************************
```

Obs: Em comandos como: (move's, compute e outros...)

A possibilidade de abend depende de vários fatores (Picture, Ambiente, desenvolvimento/homologação/produção, versão do COBOL e outros...)

45. 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).
```

46. 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:

```
01      CAB-01.
  03    FILLER          PIC X(10)   VALUE  SPACES.
  03    FILLER          PIC X(06)   VALUE  'FOLHAS'.
  03    DATA            PIC X(10)   VALUE  SPACES.


01      FILLER          PIC X(01)   VALUE X'7D'.
```

47. CONSTANTES FIGURATIVAS

São constantes definidas pelo compilador

Constantes figurativas	Picture aplicáveis
ZERO ZEROS ZEROES	ALFANUMÉRICAS OU NUMÉRICAS
SPACE SPACES	ALFABÉTICAS OU ALFANUMÉRICAS
HIGH-VALUE (MAIOR VALOR em Hexa) HIGH-VALUES	ALFANUMÉRICAS
ALL '&'	ALFANUMÉRICAS
LOW-VALUE (MENOR VALOR em Hexa) LOW-VALUES	ALFANUMÉRICAS

48. 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.

O mercado trabalha com números impares conforme exemplo abaixo:

```
01          WS-ITEM.
  03        WS-CODIGO        PIC   9(003).
  03        WS-INDICATIVO    PIC   X(001).
  03        WS-DATA.
    05      WS-DIA           PIC   9(002).
    05      WS-MES           PIC   9(002).
    05      WS-ANO.
      07     WS-SC PIC 9(002).
      07     WS-AA PIC 9(002).
  03        WS-CNPJ          PIC   9(014).
```

49. 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".

O item (77) serve para definir acumuladores e áreas auxiliares.

Exemplo:

```
WORKING-STORAGE SECTION.

77     ACU-LIDOS       PIC 9(005) VALUE 0.
77     AUX-NOME        PIC X(020).
```

50. 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          WS-ENTIDADE     PIC     9(005) VALUE ZEROS.
  88        WS-COD-ENTID    VALUE   100,
                                    200,
                                    300.
PROCEDURE DIVISION.

MOVE    PECA-COD-PEC    TO      WS-ENTIDADE.

IF      WS-COD-ENTID
        DISPLAY 'OK'
END-IF.

IF      NOT             WS-COD-ENTID
        DISPLAY 'NAO OK'
END-IF.

IF      WS-COD-ENTID
        NEXT SENTENCE
ELSE
        DISPLAY 'NAO OK'
END-IF.


01          WS-BYTE-LETRA   PIC     X(001) VALUE SPACES.
  88        WS-BYTE-CARAC   VALUE   ' ',
                                    'A',
                                    'B',
                                    'C',
                                    'D',
                                    'E',
                                    'F',
                                    'G',
                                    'H',
                                    'I'.
```

51. 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          WS-DATE         PIC     9(006) VALUE ZEROS.
 01          FILLER          REDEFINES      WS-DATE.
   03        WS-ANO-DATE     PIC     9(002).
   03        WS-MES-DATE     PIC     9(002).
   03        WS-DIA-DATE     PIC     9(002).
```

⇒ Edição de Data (DD/MM/SSAA)

```
*
 01          WS-DAT-EDI      PIC     X(010) VALUE '99/99/2099'.
 01          FILLER          REDEFINES      WS-DAT-EDI.
   03        WS-DIA-EDI      PIC     9(002).
   03        FILLER          PIC     X(001).
   03        WS-MES-EDI      PIC     9(002).
   03        FILLER          PIC     X(001).
   03        WS-SEC-EDI      PIC     9(002).
   03        WS-ANO-EDI      PIC     9(002).
```

⇒ Formatação de Hora Oriunda do Sistema Operacional (HHMMSSNN)

```
*
 01          WS-TIME         PIC     9(008) VALUE ZEROS.
 01          FILLER          REDEFINES       WS-TIME.
   03        WS-HORA-TIME    PIC     9(002).
   03        WS-MIN-TIME     PIC     9(002).
   03        WS-SEG-TIME     PIC     9(002).
   03        WS-MSEG-TIME    PIC     9(002).
```

⇒ Edição de Hora (HH:MM:SS)

```
*
 01          WS-HHH-EDI      PIC     X(008) VALUE '99:99:99'.
 01          FILLER          REDEFINES      WS-HHH-EDI.
   03        WS-HOR-EDI      PIC     9(002).
   03        FILLER          PIC     X(001).
   03        WS-MIN-EDI      PIC     9(002).
   03        FILLER          PIC     X(001).
   03        WS-SEG-EDI      PIC     9(002).
```

⇒ Transformação de Campo Alfanumérico Para Numérico ou Vice-Versa.

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

⇒ Transformação de Campo de Valor Com Vírgula Para Sem Vírgula ou Vice-Versa.

```
*
   01          WS-VLR-CV       PIC     9(016)V99.
   01          WS-VLR-SV       REDEFINES      WS-VLR-CV
                               PIC     9(018).
```

52. LINKAGE SECTION - PARM

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).

É utilizada para receber dados passado pelo cartão "PARM" do JCL.

```
//GPFPBCHM JOB 'GPFPBCHM',CLASS=A,MSGCLASS=X
//*
//STEP1   EXEC PGM=GPFPBCHM,PARM='20070605'
//SYSOUT  DD SYSOUT=*
//
```

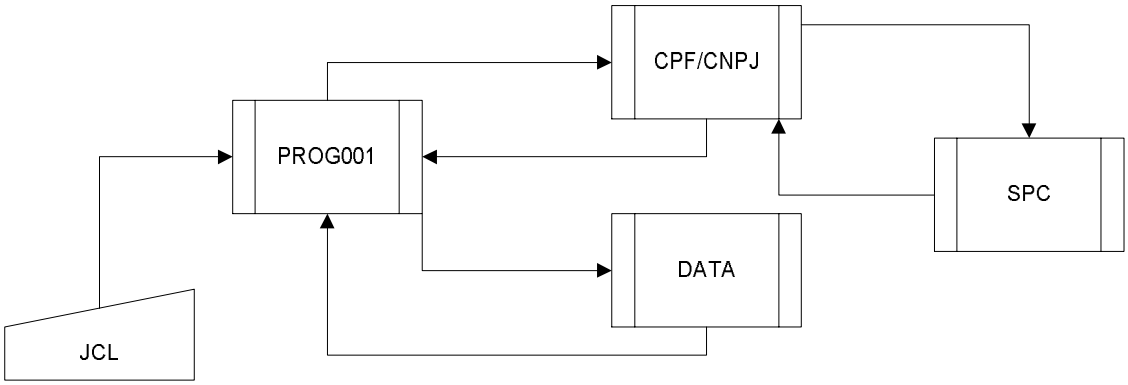
53. DADOS PASSADOS VIA PARM NO CARTÃO JCL

```
*-----------------------------------------------------------------*
 WORKING-STORAGE                SECTION.
*-----------------------------------------------------------------*
*            ACUMULADORES                                         *
*-----------------------------------------------------------------*
*
 01          AC-LID-DEMAB543 PIC     9(004) COMP VALUE ZEROS.
 01          AC-GRV-DEMAB554 PIC     9(004) COMP VALUE ZEROS.
*
*-----------------------------------------------------------------*
 LINKAGE                       SECTION.
*-----------------------------------------------------------------*
*                                (Tamanho da área passada do JCL)
  01          LKG-PARM.
    03        LKG-TAM           PIC    S9(004) COMP.
    03        LKG-DATA          PIC     9(008).
    03        FILLER            REDEFINES      LKG-DATA.
      05      LKG-ANO           PIC     9(004).
      05      LKG-MES           PIC     9(002).
      05      LKG-DIA           PIC     9(002).
*
*-----------------------------------------------------------------*
 PROCEDURE    DIVISION    USING    LKG-PARM.
*-----------------------------------------------------------------*
```

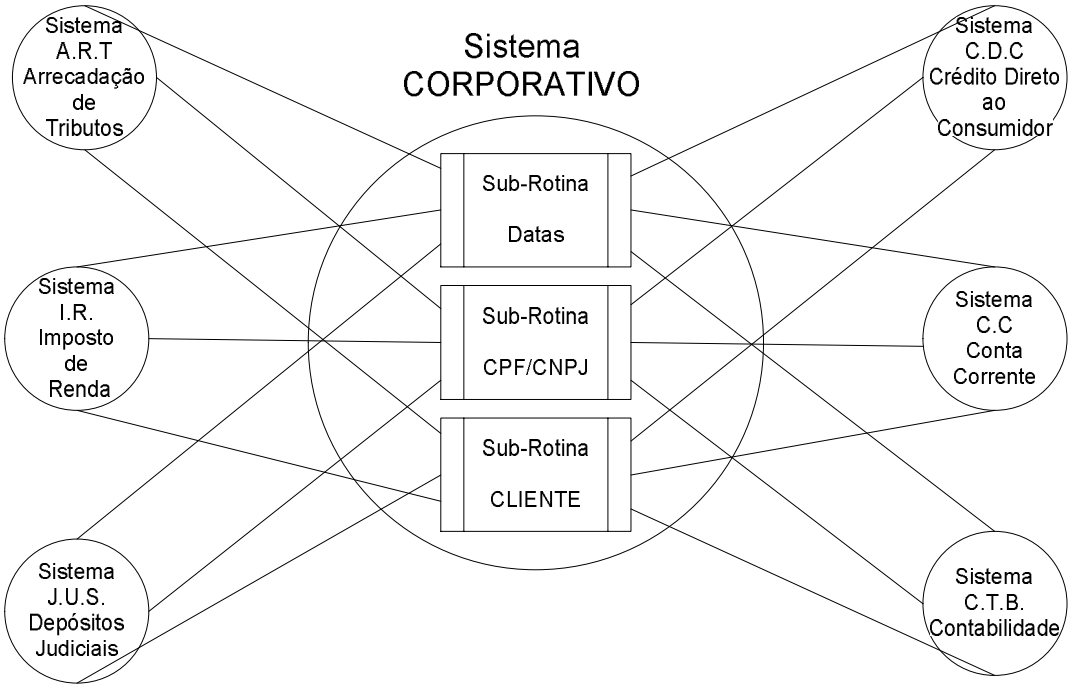
(Sem estes parâmetros a informação não chega até o programa)

54. LINKAGE SECTION – SUB-ROTINA (API)

Área de comunicação em sub-programa (Módulos Batch).



É utilizado para ligar o programa principal em COBOL a outros programas, muito utilizado em programação modular, para comunicação entre o programa principal e as sub-rotinas passando parâmetros



```
*-----------------------------------------------------------------*
 WORKING-STORAGE                SECTION.
*-----------------------------------------------------------------*
*             ACUMULADORES                                        *
*-----------------------------------------------------------------*
*
 01          AC-LID-DEMAB543 PIC     9(004) COMP VALUE ZEROS.
 01          AC-GRV-DEMAB554 PIC     9(004) COMP VALUE ZEROS.
*
*-----------------------------------------------------------------*
 LINKAGE                      SECTION.
*-----------------------------------------------------------------*
*
01           REG-FOCO.
     03      FOCO-CODIGO     PIC     9(005).
     03      FOCO-COD-ERRO-1 PIC     9(003).
     03      FOCO-COD-ERRO-2 PIC     9(003).
     03      FOCO-COD-ERRO-3 PIC     9(003).
     03      FOCO-LIVRE      PIC     X(036).
*
*-----------------------------------------------------------------*
 PROCEDURE    DIVISION   USING    REG-FOCO.
*-----------------------------------------------------------------*
```

Ou se utilize dentro um COPY DEMAB543.

exemplo

COPY DEMAB543. (Melhor Prática)

(Para programas chamadores seja batch ou on-line esta área deve ser definida na Work).

55. Programa Chamador

```
*****************************************************************
 WORKING-STORAGE              SECTION.
*****************************************************************
*
 01          WS-GPFBB008      PIC      X(008) VALUE 'GPFBB008'.
*
*****************************************************************
* SUB
*****************************************************************
*
     COPY    COBI901.
*
*****************************************************************
 LINKAGE                      SECTION.
*****************************************************************
 PROCEDURE                    DIVISION.
*****************************************************************
*
     ANTES DA CHAMADA PASSAR OS PARAMETROS PARA A SUB-ROTINA.

     CALL    WS-GPFBB008      USING       W901-REG.

     APOS A CHAMDADA SEMPRE TESTAR O CODIGO DE RETORNO DA
     SUB-ROTINA.

     IF      W901-COD-RET NOT EQUAL  ZEROS
             ABENDAR
     END-IF.
```

56. Exemplo de um Cabeçalho COBOL MVS (BATCH)

```
*****************************************************************
 IDENTIFICATION              DIVISION.
*****************************************************************
*
 PROGRAM-ID.                 PROG001.
 AUTHOR.                     ANTONIO CARLOS.
 DATE-WRITTEN.               19/09/2003.
 SECURITY.
*
*****************************************************************
*  SISTEMA.......: SIMAN - SISTEMA DE APRENDIZADO NA LINGUAGEM    *
*****************************************************************
*  ANALISTA.....: JOSE SILVA                                     *
*  LINGUAGEM....: COBOL/BATCH                                    *
*  PROGRAMADOR..: ANTONIO CARLOS                                 *
*  DATA.........: 19/09/2003                                     *
*****************************************************************
*  OBJETIVO.....: A  PARTIR  DO  CADASTRO  DE  PECAS,  GERAR     *
*                 CADASTRO DE PECAS ATUALIZADO.                 *
*****************************************************************

*****************************************************************
 ENVIRONMENT                 DIVISION.
*****************************************************************
 CONFIGURATION               SECTION.
*****************************************************************
 SPECIAL-NAMES.              DECIMAL-POINT   IS    COMMA.
*****************************************************************
 INPUT-OUTPUT                SECTION.
*****************************************************************
 FILE-CONTROL.
*****************************************************************
*
       SELECT  CADPECA   ASSIGN  TO  UT-S-CADPECA
               FILE      STATUS  IS  WS-FS-CADPECA.
*
       SELECT  CADPATU   ASSIGN  TO  UT-S-CADPATU
               FILE      STATUS  IS  WS-FS-CADPATU.
*
*****************************************************************
 DATA                        DIVISION.
*****************************************************************
 FILE                        SECTION.
*****************************************************************
* INPUT..: CADPECA - CAD DE PECAS            - LRECL.: 100 BYTES *
*****************************************************************
*
 FD  CADPECA
     RECORDING  MODE      IS  F
     LABEL      RECORD    IS  STANDARD
     BLOCK      CONTAINS  0   RECORDS.

 01          REG-CADPECA    PIC     X(100).
*
```

```
*****************************************************************
* OUTPUT.: CADPATU - CAD DE PECAS ATUALIZADO - LRECL.: 100 BYTES *
*****************************************************************
*
 FD  CADPATU
     RECORDING  MODE      IS  F
     LABEL      RECORD    IS  STANDARD
     BLOCK      CONTAINS  0   RECORDS.

 01          REG-CADPATU    PIC     X(100).
*
*****************************************************************
 WORKING-STORAGE               SECTION.
*****************************************************************
*
 01          WS-LIDOS       PIC     9(007) VALUE ZEROS.
 01          WS-GRAVADOS    PIC     9(007) VALUE ZEROS.
*
*****************************************************************
* INPUT..: CADPECA - CAD DE PECAS            - LRECL.: 100 BYTES *
*****************************************************************
*
 COPY COBI1001.
*
*****************************************************************
* OUTPUT.: CADPATU - CAD DE PECAS ATUALIZADO - LRECL.: 100 BYTES *
*****************************************************************
*
 COPY       COBO1001.
*
*****************************************************************
 LINKAGE                       SECTION.
*****************************************************************
 PROCEDURE                      DIVISION.
*****************************************************************
*
     PERFORM 0100-00-PROCED-INICIAIS.

     PERFORM 0200-00-PROCED-PRINCIPAIS
       UNTIL WS-FS-CADPECA EQUAL 10.

     PERFORM 0300-00-PROCED-FINAIS.
*
*****************************************************************
 0100-00-PROCED-INICIAIS       SECTION.
*****************************************************************
*
```

57. PROCEDURE DIVISION

É a quarta e última divisão de um programa COBOL, descreve todos os procedimentos a serem executados pelo programa, tais como operações e manipulação de dados.

Contém comandos executáveis do programa, isto é, os procedimentos a serem executados.

Toda lógica de Programação é escrita nesta divisão.

58. CLÁUSULA OPEN

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

Sintaxe:

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

INPUT -> arquivos de entrada apenas para leitura

OUTPUT -> arquivos de saída apenas para gravação e impressão

I-O -> arquivos de acesso-direto (VSAM) - (leitura e gravação)

59. START NO VSAM

Esta clausula posiciona um ponteiro no arquivo VSAM, permitindo a leitura a partir de um determinado registro no meio do arquivo.

Clausula utilizada para leitura seqüencial de um arquivo VSAM.

START KRV0999 KEY IS NOT LESS THAN WS-CHV-VSAM.

READ KRV0999 NEXT.

Criar FD para arquivos VSAM conforme modelo abaixo:

```
SELECT  CADVSAM   ASSIGN  TO  DA-I-CADVSAM
        ORGANIZATION       IS  INDEXED
        ACCESS  MODE       IS  DYNAMIC
        RECORD KEY         IS  VSAM-COD-CHAVE
        FILE STATUS        IS  WS-FS-CADVSAM.
```

60. CLÁUSULA READ

Ler um registro do arquivo de entrada.

Sintaxe :

READ ARQUIVO1. ou READ ARQUIVO1 INTO AREA-ARQ1.

ou

```
READ ARQUIVO1  AT END
            MOVE ´SIM´ TO WS-FIM.
```

NOME-DO-ARQUIVO -> definido por uma descrição na "FD"

INTO -> faz com que o registro seja lido e movido para área definida dentro da "WORKING-STORAGE" ou "LINKAGE SECTION".

AT END -> é uma das opções para o controle de fim de arquivo.

61. 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".

WRITE REG-SAIDA FROM AREA-1.

62. 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.
```

63. 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.

64. 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.

65. 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.

66. 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.
```

67. 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.
```

68. CLÁUSULA MOVE

Este comando faz a movimentação de dados dentro do programa.

Sintaxe:

```
MOVE CAMPO1     TO   CAMPO2
                     CAMPO3.

MOVE DADO-A     TO   DADO-2.

MOVE DADO-A     TO   DADO-3
                     DADO-4.
```

Exemplo com literais figurativas:

```
MOVE   SPACES     TO  WS-AREA-LIVRE

MOVE   ZEROS      TO  DATA-8

MOVE   120        TO  NUMERO-FIXO
```

```
MOVE   'CREDITO' TO  WS-CONTABIL

MOVE   'MENSAL'  TO  WS-CABEC1  CABEC2.
```

Todo item de grupo é um item alfanumérico.

Se o campo receptor for alfanumérico.

- *As posições não preenchidas pelo conteúdo do campo emissor, são preenchidas automaticamente com espaços alinhados á esquerda.*

- *Se o tamanho do campo emissor for maior que o campo receptor, os caracteres em excesso serão truncados.*

Se o item receptor for numérico:

- *As posições não preenchidas pelo conteúdo do campo emissor, são preenchidas automaticamente com zeros alinhados á direita.*

69. 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.
```

Nesta forma, soma-se o conteúdo do WS-CAMPO1 ao conteúdo do WS-CAMPO2.

O resultado da soma ficará no WS-CAMPO2.

70. MOVE POSICIONAL

Observe o MOVE abaixo:

```
MOVE   WS-DDMMAAAA(1:2)   TO   WS-DIA.
MOVE   WS-DDMMAAAA(3:2)   TO   WS-MES.
MOVE   WS-DDMMAAAA(5:4)   TO   WS-ANO.
```

Com o Move posicional podemos fazer a manipulação parcial de informações de um campo.

Conforme exemplo acima trabalhamos com as informações de DATA separadamente. Realizamos os moves por parte, 1º. o DIA, 2º. o Mês e por ultimo o ANO.

- Este tipo de move só pode ser realizado em campos Zonados ou Alfanuméricos.
- Em campos Binários, Compactados ou de Edição jamais devemos aplicar esta técnica.

71. CLÁUSULA SUBTRACT

Esta cláusula é utilizada para subtrair itens numéricos

Sintaxe:

SUBTRACT 001 FROM WS-PAGINA.

72. CLÁUSULA MULTIPLY (multiplicação)

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

Sintaxe:

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

A multiplicação é feita "WS-A" por "WS-B" e o resultado é colocado em "WS-C".

O campo "WS-C" pode ser uma picture numérica de edição.

73. CLÁUSULA DIVIDE (divisão)

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

- ✓ Identificação do ano bi-sexto

Work

```
01          WS-BI-SEXTO       PIC          9(004) VALUE ZEROS.
01          WS-RESULTADO      PIC          9(004) VALUE ZEROS.
01          WS-RESTO          PIC          9(002) VALUE ZEROS.

MOVE        WS-ANO            TO           WS-BI-SEXTO.

DIVIDE      WS-BI-SEXTO       BY    4      GIVING      WS-RESULTADO
                                           REMAINDER   WS-RESTO.

IF          WS-RESTO          EQUAL        ZEROS
            MOVE 29           TO           WS-DIA
ELSE
            MOVE 28           TO           WS-DIA
END-IF.
```

74. CLÁUSULA COMPUTE

Operandos que a cláusula compute pode executar:

- *Adição (+);*
- *Subtração (-);*
- *Multiplicação (*);*
- *Divisão (/);*
- *Exponenciação **(**)***

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.

```
COMPUTE   WS-RATEIO   =   (   WS-VLR-BASE   *   WS-PCC-BASE   )   /   100.
```

75. CLÁUSULA ACCEPT

Esta cláusula executa uma operação de entrada.

ACCEPT WS-DATE FROM DATE. FORMATO (**AAMMDD**)
ACCEPT WS-TIME FROM TIME. FORMATO (**HHMMSSNN**)

Sintaxe:

```
WORKING-STORAGE SECTION.  (REDEFINES)


01          WS-DAT-COR      PIC     9(006) VALUE ZEROS.
01          FILLER          REDEFINES      WS-DAT-COR.
  03        WS-ANO-COR      PIC     9(002).
  03        WS-MES-COR      PIC     9(002).
  03        WS-DIA-COR      PIC     9(002).


01          WS-DAT-EDI      PIC     X(010) VALUE '99/99/2099'.
01          FILLER          REDEFINES      WS-DAT-EDI.
  03        WS-DIA-EDI      PIC     9(002).
  03        FILLER          PIC     X(001).
  03        WS-MES-EDI      PIC     9(002).
  03        FILLER          PIC     X(001).
  03        WS-SEC-EDI      PIC     9(002).
  03        WS-ANO-EDI      PIC     9(002).


01          WS-TIM-COR      PIC     9(008) VALUE ZEROS.
01          FILLER          REDEFINES       WS-TIM-COR.
  03        WS-HOR-COR      PIC     9(002).
  03        WS-MIN-COR      PIC     9(002).
  03        WS-SEG-COR      PIC     9(002).
  03        WS-MSG-COR      PIC     9(002).


01          WS-HHH-EDI      PIC     X(008) VALUE '99:99:99'.
01          FILLER          REDEFINES      WS-HHH-EDI.
  03        WS-HOR-EDI      PIC     9(002).
  03        FILLER          PIC     X(001).
  03        WS-MIN-EDI      PIC     9(002).
  03        FILLER          PIC     X(001).
  03        WS-SEG-EDI      PIC     9(002).
```

76. CLÁUSULA SYSIN

Podemos receber Informações passadas pelo SYSIN do JCL.

```
ACCEPT    WS-SYSIN       FROM      SYSIN.
```

```
//GPFPBCHM JOB 'GPFPBCHM',CLASS=A,MSGCLASS=X
//*
//STEP1   EXEC PGM=GPFPBCHM,PARM='20070605'
//SYSOUT  DD SYSOUT=*
//SYSIN   DD *
CARTAO 010027
//*
```

77. CLÁUSULA DISPLAY

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

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

78. CLÁUSULA IF

Esta cláusula indica uma decisão Lógica em um programa.

Exemplo:

```
IF        WS-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.
```

Neste exemplo, o programa indica a existência de uma decisão escrevendo a palavra "IF", seguida de palavras que contenham um teste e o que fazer conforme o resultado do teste significando uma frase condicional.

Outros exemplos:

```
        IF  NOME NOT EQUAL 'PEDRO'
           NEXT SENTENCE
        ELSE
           ADD 1   TO  CONT-NOME-IGUAL.

        MOVE CONT-NOME-IGUAL  TO  RELATORIO-NOME-IGUAL.
```

Quando testar se um campo é numérico, e este for compactado, tomar cuidado com o sinal (C). Nestes casos, colocar o indicador de sinal (S) na frente dos 9's.

Exemplo:

```
        01     CAMPO     9(08).
        01     CAMPO     S9(07) COMP-3.

        IF     IDENT-1  EQUAL   125 OR
               IDENT-2  EQUAL   250
           MOVE 'SIM'  TO WS-ACHOU.
```

79. Operação de Relação

OPERAÇÃO DE RELAÇÃO	SIGNIFICADO
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 THRU 999-99-EXIT.

IF   WS-CODIGO EQUAL 2
     MOVE WS-CODIGO TO   REG-CODIGO.

IF   WS-VALOR  NOT  LESS WS-SALDO
     PERFORM   ROT-GRAVA.
```

80. OPERADORES LÓGICOS

Existem no Cobol 3 (três) operadores lógicos:

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

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

```
IF        WS-VALOR EQUAL 100,00 OR
          WS-VALOR EQUAL 200,00 OR
          WS-VALOR EQUAL 400,00
          PERFORM   880-00-CALCULA-DESCONTO.


IF        WS-VALOR EQUAL 100,00 OR 200,00 OR 400,00
          PERFORM   880-00-CALCULA-DESCONTO.
```

81. TESTES COMPOSTOS

Ocorre o teste composto, quando aparecem os conectores lógicos no "IF", como: "AND", "OR" ou "NOT".

Exemplo:

```
IF         WS-NOME          EQUAL SPACES OR
           WS-ENDERECO      EQUAL ' '
           PERFORM          666-00-TRATA-CADASTRO-PENDENTE
ELSE
 IF        WS-NOME          NOT EQUAL SPACES AND
           WS-ENDERECO      NOT EQUAL SPACES
           PERFORM          667-00-TRATA-CADASTRO-CORRETO
 END-IF
END-IF.
```

82. 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   THRU 0100-99-EXIT.
PERFORM   0200-00-PROCED-PRINCIPAIS THRU 0200-99-EXIT.
PERFORM   0300-00-PROCED-FINAIS     THRU 0300-99-EXIT.

OU

PERFORM   0100-00-PROCED-INICIAIS.
PERFORM   0200-00-PROCED-PRINCIPAIS.
PERFORM   0300-00-PROCED-FINAIS.
```

(Obs : Nunca utilize em um mesmo programa perform's amarrados com **THRU** e **Section**)

Quando dividimos os parágrafos em "SECTION", o CONDITION-CODE de retorno estará na próxima SECTION ou no final da PROCEDURE.

Neste caso, você só poderá usar o GO TO para desvios dentro da mesma SECTION, pois assim, não haverá o risco de destruir o CONDITION-CODE de retorno.

(Obs: Evite o uso de **GO TO**, dificulta a manutenção em programas).

83. 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:

```
MOVE    001        TO        WS-IND.

PERFORM 550-00-ROTINA1 THRU 550-99-EXIT
  UNTIL WS-IND GREATER 050.

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

PERFORM 200-00-PROCED-PRINCIPAIS
  UNTIL WS-FS-CADPECA EQUAL 10.
```

84. PERFORM VARYING

Sintaxe:

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

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


        PERFORM VARYING WS-IND FROM 01 BY 01
                UNTIL   WS-IND GREATER 012
                OR      WS-FLAG-FIM = 'SIM'

           IF   WS-BYTE(WS-IND) IS NUMERIC
                PERFORM 125-00-TRATAR-BYTE
           END-IF

        END-PERFORM.
```

85. CLÁUSULA EXIT

É um ponto comum de finalização para uma série de procedimento(s).

```
        0100-99-EXIT.      (LABEL)
            EXIT.           (COMANDO EXIT)
```

A cláusula "EXIT" deve ser precedida por um nome de parágrafo e deve ser única cláusula do parágrafo.

O programa poderá ter vários EXIT's associados com PERFORM's.

Exemplo:

```
000-00-MODULO-MESTRE.

     PERFORM   100-00-PROCED-INICIAIS   THRU   100-99-EXIT.

     PERFORM   200-00-PROCED-PRINCIPAIS THRU   200-99-EXIT.

     PERFORM   300-00-PROCED-FINAIS     THRU   300-99-EXIT.


000-99-EXIT.
    EXIT.
```

(Obs. Uma section não termina ao encontrar um exit, como muitos profissionais pensam, a section termina ao encontrar um nova section, por isso, quando se utiliza perform SECTION, o exit não é obrigatório).

(Obs. Quando se utiliza (perform THRU, o exit se torna obrigatório).

86. FUNCTION CURRENT-DATE

Formato: **AAAAMMDDHHMMSS**

```
MOVE      FUNCTION CURRENT-DATE (1:4)   TO   WS-ANO.
MOVE      FUNCTION CURRENT-DATE (5:2)   TO   WS-MES.
MOVE      FUNCTION CURRENT-DATE (7:2)   TO   WS-DIA.
```

Podemos trabalhar com Move Posicional, ou itens de Grupo.

87. TABELA INTERNA – SUBSCRITA

As tabelas devem ser construídas na '"Working".

As tabelas podem ser:

- TABELA SUBSCRITA;
- DIRECIONAL;
- BIDIMENSIONAL;
- TRIDIMENSIONAL.

- *O tamanho de uma tabela não pode exceder a 131.071 bytes (128K);*

- *O tamanho de uma tabela que tiver DEPENDING ON não pode exceder a 32.767 bytes (32K);*

- *Cada 1K tem em média 1.024 bytes;*

O indexador de uma tabela indexada pode ser somado ou subtraído. Ex.: TAB (INDEX + 1) TO X.

Pode ser considerada, tabela subscrita, a tabela que você utiliza um índice fora dela. Exemplo de definição da tabela na WORKING-STORAGE SECTION:

```
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).


MOVE  MESES(1)          TO    CAB-MESES.
MOVE  MESES(WS-IND)           TO    CAB-MESES.
```

88. TABELA INTERNA – DIRECIONAL

```
      01            TABELA-ESTADO.
          03        TAB-ESTADOS     OCCURS  50   TIMES.
            05      TAB-CD-SIG      PIC     X(002).
            05      TAB-DS-SIG      PIC     X(050).


MOVE  TAB-CD-SIG(3)           TO    CAB-SIGLA.
MOVE  TAB-CD-SIG(3)           TO    CAB-SIGLA.

MOVE  TAB-DS-SIG(5)           TO    CAB-ESTADO.
MOVE  TAB-DS-SIG(5)           TO    CAB-ESTADO.



      ****************************************************************
      ****************************************************************
      *

       01          WS-VALOR-NUM    PIC     9(015)V99 VALUE ZEROS.
      *
       01          WS-TAB-VLR-NUM  REDEFINES     WS-VALOR-NUM.
         03        WS-TB-VLR-NUM   OCCURS  17    TIMES.
           05      WS-VLR-NUM      PIC     9(001).
        *
      ****************************************************************
      ****************************************************************
      *
       01          WS-VALOR-ALF    PIC     X(017) VALUE SPACES.
      *
       01          WS-TAB-VLR-ALF  REDEFINES     WS-VALOR-ALF.
         03        WS-TB-VLR-ALF   OCCURS  17    TIMES.
           05      WS-VLR-ALF      PIC     X(001).
      *
            MOVE    100,10          TO      WS-VLR-CV.

            DISPLAY 'VALOR - CV  ' WS-VLR-CV.
            DISPLAY 'VALOR - SV  ' WS-VLR-SV.

            MOVE   ' X 1 0 0 , 0 1 X '
                                    TO      WS-VALOR-ALF.

            MOVE    017             TO      WS-IND2.

            PERFORM VARYING WS-IND1 FROM  17 BY    -1
                    UNTIL   WS-IND1 EQUAL    ZEROS

             IF     WS-VLR-ALF(WS-IND1)
                                    IS      NUMERIC
                    MOVE WS-VLR-ALF(WS-IND1)
                                    TO      WS-VLR-NUM(WS-IND2)
                    SUBTRACT 001    FROM    WS-IND2
             END-IF

            END-PERFORM.

            DISPLAY 'VALOR - ALF = ' WS-VALOR-ALF
            DISPLAY 'VALOR - NUM = ' WS-VALOR-NUM
      *
      ****************************************************************
      ****************************************************************
      *
```

89. TABELA INTERNA – BI-DIMENSIONAL

- ✓ **Bi-Dimensional**

Meses (12) Ocorrências
Dias (31) Dias

```
MOVE TAB-ANUAL(2,25)      TO   DADOS-ANO.

MOVE TAB-ANUAL(WS-IND1, WS-IND2)
                          TO   DADOS-ANO.
```

90. TABELA INTERNA – TRI-DIMENSIONAL

- ✓ **Tri-Dimensional**

Meses (12) Ocorrências
Dias (31) Dias
Horas (24) Horas

```
MOVE TAB-ANUAL(3,10,24)  TO   DADOS-ANO.

MOVE TAB-ANUAL(WS-IND1, WS-IND2, WS-IND3)
                          TO   DADOS-ANO.
```

91. INSPECT - REPLACING

A função desta cláusula é substituir um determinado caractere num item, por outro determinado caractere.

Sintaxe:

```
  INSPECT WS-CAMPO REPLACING ALL SPACES BY '_'.

  INSPECT WS-NOME-PEC REPLACING ALL 'E' BY 'U'.

  INSPECT WS-NOME-PEC REPLACING ALL LOW-VALUES  BY SPACES.

COPY      CAVDS020  REPLACING  ==:DS020:==  BY  ==DS020==.
```

REPLACING -> substitui um determinado caractere num item por outro.

Os campos para o INSPECT só podem ser zonados ou alfanuméricos.

92. RETURN-CODE

É uma área que contém um código de retorno qualquer para que este seja checado pelo STEP posterior ao executado, através do cond do JCL.

Exemplo:

```
IDENTIFICATION DIVISION.
..........
ENVIRONMENT DIVISION.
.........
DATA DIVISION.
.........
PROCEDURE DIVISION.

   OPEN INPUT ARQTESTE.

     READ ARQTESTE       AT END
          MOVE 020 TO RETURN-CODE.

   CLOSE ARQTESTE.

   STOP RUN.
```

Deve-se procurar enviar o RETURN-CODE valorizado sempre ao final do processamento, isto é, antes do STOP RUN ou GOBACK.

93. 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.
```

94. ROUNDED

Realizar o arredondamento de dizimas de valores

```
COMPUTE WS-QBR-VLR-UNI ROUNDED = ( PECA-VLR-UNIT / 3 ).
```

95. STRING/UNSTRING

Concatenam variáveis e/ou valores conteúdos fixos.

```
STRING  WS-DIA-DATE '/'
        WS-MES-DATE '/'
       '20'
        WS-ANO-DATE
DELIMITED BY SIZE INTO CAB-DATA.
```

96. COPY

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

```
COPY   COBWS001.

COPY   CAVDS020     REPLACING  ==:DS020:==  BY  ==DS020==.

++INCLUDE COBWS001.

INC COBWS101.

COPY          I#UC2225.
```

97. SORT - SD

Usado para criar arquivo ordenado a partir de um arquivo qualquer.

```
SORT            SORTWS01
                ASCENDING    KEY             NR-BC-SD
                                             NR-AG-SD
                                             NR-CT-SD
                INPUT        PROCEDURE       200-00-CLASSIFICAR
                OUTPUT       PROCEDURE       400-00-GRAVAR-ARQUIVO.
```

*Na **INPUT PROCEDURE** é gerado o arquivo **SORT** SORTWS01*

*Na **OUTPUT PROCEDURE** buscamos os registros no arquivo **SORT** SORTWS01*

98. RETURN

*Usado para transferir dados do arquivo **SORT** para a memória do computador e coloca os mesmos à disposição do programa na **OUTPUT PROCEDURE**.*

```
RETURN    SORTWS01      AT      END
           MOVE 'S'     TO      WS-FIM-SORT.
```

99. RELEASE

*Usado para transferir os dados lidos no arquivo de entrada na **INPUT PROCEDURE** para o arquivo de **SORT**.*

```
RELEASE REG-SORT.
```

Nível 01 do lay-out do arquivo sort definido na SD. (Descrição de Sort)

100. INITIALIZE

Usado para inicializar áreas de trabalho do programa.

```
INITIALIZE          REG-CADPATU.

INITIALIZE          REG-CADFIL1
                    REG-CADFIL2.
```

*Cuidado na utilização deste comando, ele não inicializa **FILLER**, e nem sub-itens.*

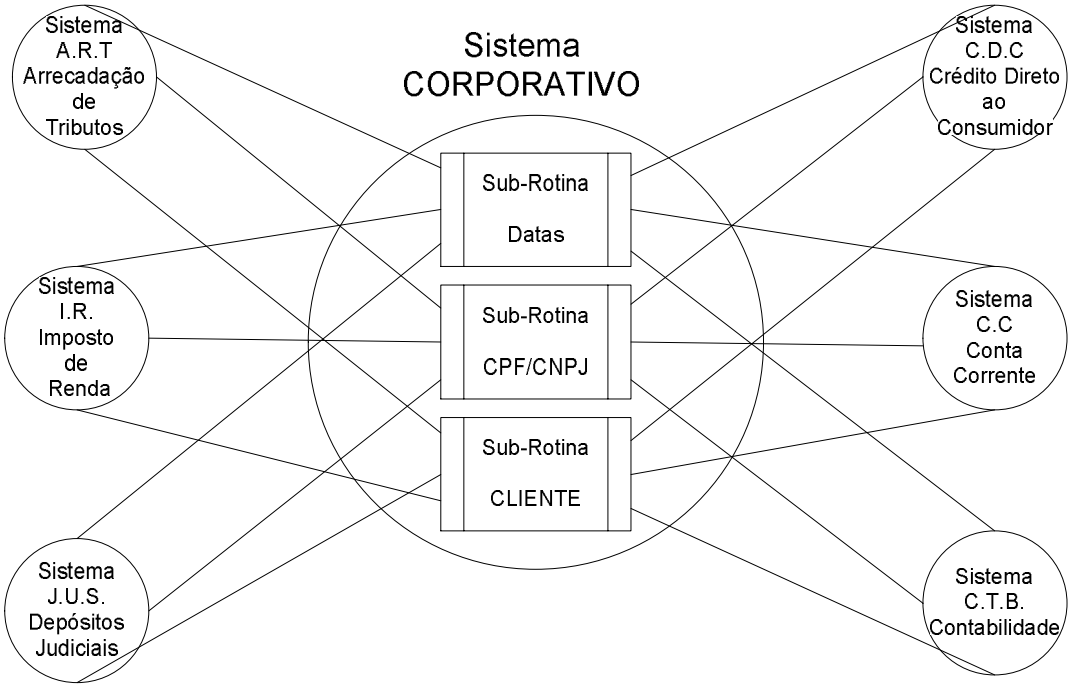
```
   01       REG-CADPECA.
     03     PECA-CODIGO          PIC  9(005).
     03     PECA-NOME            PIC  X(030).
     03     PECA-DATA.
       05   PECA-DIA-ENTREGA     PIC  9(002).
       05   PECA-MES-ENTREGA     PIC  9(002).
       05   PECA-ANO-ENTREGA     PIC  9(002).
     03     FILLER               PIC  X(009).

MOVE      SPACES          TO        REG-CADPECA.

INITIALIZE                REG-CADPECA
                          PECA-DATA.
```

101. SUB-ROTINAS – MODULOS – APIs

Segue abaixo conceito de utilização de sub-rotinas, módulos ou api's.



102. CALL ESTÁTICO

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

```
CALL       'SUBPGM1'      USING          WS-AREA-PGM1
END-CALL.
```

Ou

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

103. CALL DINÂMICO

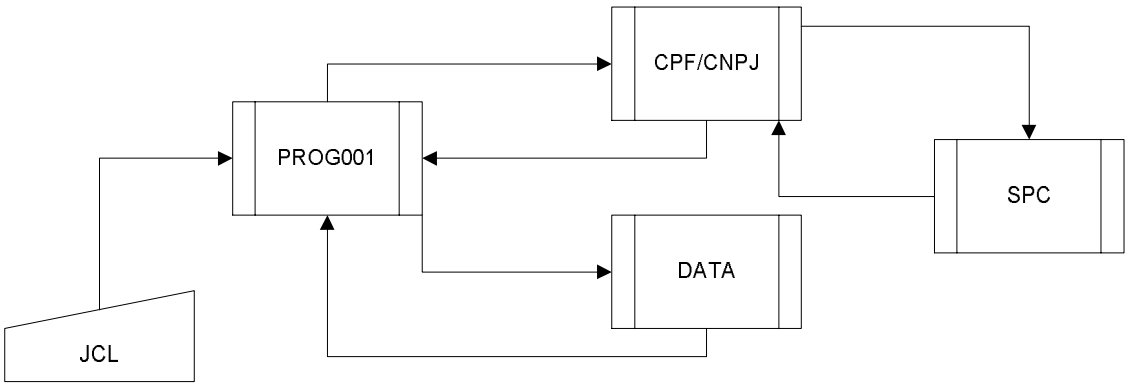
Utilizado para realizar a chamada de um subprograma ou sub-rotina.

```
CALL       WS-SUBPGM1          USING          WS-AREA-PGM1
END-CALL.
```

Ou

```
CALL       WS-SUBPGM1          USING          WS-AREA-PGM1.
```

- ✓ Esquema de Chamadas de Sub-rotinas.



⇒ Inicializar área de trabalho

⇒ Formatar corretamente os campos chaves (INPUT)

⇒ Tratar devidamente os retornos

⇒ Em caso de abend apresentar uma mensagem (técnica) clara e Objetiva

⇒ Sub-Rotina não ABENDA.

104. CARACTER CARRO IMPRESSOR

Tabela de Caracteres do Carro Impressor

Branco	Passa para a linha seguinte
-	Pula 2 linhas
+	Não pula uma linha
1	**Muda de Página**
2	Salta para o canal 2
0	Pula 1 linha

105. Clausula OF

A clausula OF deve ser empregada quando existe uma ambigüidade de campos, ou seja, dois campos com o mesmo nome.

Exemplo:

Arquivo AE

```
01    AE-AREA.
  03  DATA      PIC 9(008).
```

Arquivo AS

```
01    AS-AREA.
  03  DATA      PIC 9(008).
```

No exemplo acima temos o campo DATA nos dois arquivos, neste caso devemos utilizar o **OF**, no comando MOVE do campo do arquivo de entrada AE para o campo do arquivo de Saída AS.

```
MOVE     DATA     OF     AE-AREA
  TO     DATA     OF     AS-AREA.
```

106. 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.

Alguns exemplos de operações de I/O que podem ser testadas o FILE STATUS:

- *OPEN*
- *START*
- *WRITE*
- *READ*
- *REWRITE*
- *CLOSE*

É aconselhável que se teste a STATUS KEY após cada operação de I/O.

Se um valor diferente de zero for encontrado, o correto será terminar o programa e corrigir o erro.

Se outra medida for tomada, é provável que ocorra um abend em uma outra instrução.

File Status	Descrição
00	Execução com sucesso, processamento OK
10	Condição AT END um READ seqüencial foi tentado e não existia um próximo registro lógico no arquivo, porque o fim do arquivo foi atingido, ou o primeiro READ foi executado em um arquivo que não estava presente.Eqüivale ao AT END
22	Foi feita uma tentativa de gravar um registro que criaria uma chave duplicada num arq. relativo; ou foi feita uma tentativa de gravar ou regravar um reg. que criaria uma chave primária duplicada ou uma chave alternada duplicada num arquivo indexado sem a frase DUPLICATES. O valor da chave é aplicado para arq. KSDS em que a chave alternada foi definida com UNIQUE
23	Nenhum registro encontrado. Foi feita uma tentativa de acesso randômico em um reg. que não existe no arq.ou um START, ou READ randômico foi tentado em um arq. que não estava presente.- EOF após START- Registro não encontrado-Opção GREATER THAN, usada e a chave HIGH-VALUES. READ seqüencial após última gravação.
35	Um OPEN INPUT , I-O ou EXTENDED foi tentado em um arq. que não estava presente.FALTOU CARTÃO DD. Arquivo Vazio. confira ddname com select
41	condição de erro de lógica Foi tentado um OPEN para um arquivo já aberto
42	Foi tentado um CLOSE para um arquivo que não estava aberto
46	Um READ seqüencial foi tentado em um arquivo aberto como INPUT ou I/O e não foi estabelecido um próximo reg. porque o READ anterior não foi bem sucedido ou causou um AT END
47	Foi tentado um READ num arquivo que não foi aberto como INPUT ou I/O.
48	Foi tentado um WRITE num arq. que não foi aberto como OUTPUT, I/O ou EXTENDED.

107. ABENDAR PROGRAMA

Forçar uma parada na execução do programa, com stop run ou goback.

- Mensagem para tratamento de erros arquivos

```
  NA ABERTURA,  NA LEITURA,  NA GRAVACAO,  NO FECHAMENTO
****************** XXXXXXXX ******************
*                                            *
*      TERMINO ANORMAL DE PROCESSAMENTO      *
*                                            *
****************** XXXXXXXX ******************
*                                            *
* PROBLEMAS XXXXXXXXXXXXX DO ARQUIVO XXXXXXXX *
*                                            *
*             FILE STATUS....: 99            *
*                                            *
*           PONTO COM ERRO...: 999           *
*                                            *
****************** XXXXXXXX ******************
*     P R O G R A M A  C A N C E L A D O     *
****************** XXXXXXXX ******************
```

- Mensagem para tratamento de erros no Db2

```
  SELECT, INSERT, UPDATE, DELETE, OPEN, FETCH, CLOSE, COUNT…
****************** XXXXXXXX ******************
*                                            *
*      TERMINO ANORMAL DE PROCESSAMENTO      *
*                                            *
****************** XXXXXXXX ******************
*                                            *
*   PROBLEMAS NO XXXXXX DA TABELA XXXXXXXX   *
*                                            *
*              SQLCODE....: +++9             *
*                                            *
*           PONTO COM ERRO...: 999           *
*                                            *
****************** XXXXXXXX ******************
*     P R O G R A M A  C A N C E L A D O     *
****************** XXXXXXXX ******************
```

- Mensagem para tratamento de erros no acesso à sub-rotinas

```
****************** XXXXXXXX ******************
*                                            *
*      TERMINO ANORMAL DE PROCESSAMENTO      *
*                                            *
****************** XXXXXXXX ******************
*                                            *
*   PROBLEMAS NO ACESSO A SUBROTINA XXXXXXXX  *
*                                            *
*             COD.RETORNO...: Z99            *
*                                            *
*           PONTO COM ERRO...: 999           *
*                                            *
****************** XXXXXXXX ******************
*     P R O G R A M A  C A N C E L A D O     *
****************** XXXXXXXX ******************
```

(Obs. Em muitos clientes o tratamento de abend é realizado chamando uma sub-rotina).

A melhor prática é abendar a aplicação utilizando sub-rotina padrão da empresa, apenas mover 12 para o return-code não abenda, ou seja, não para a execução.