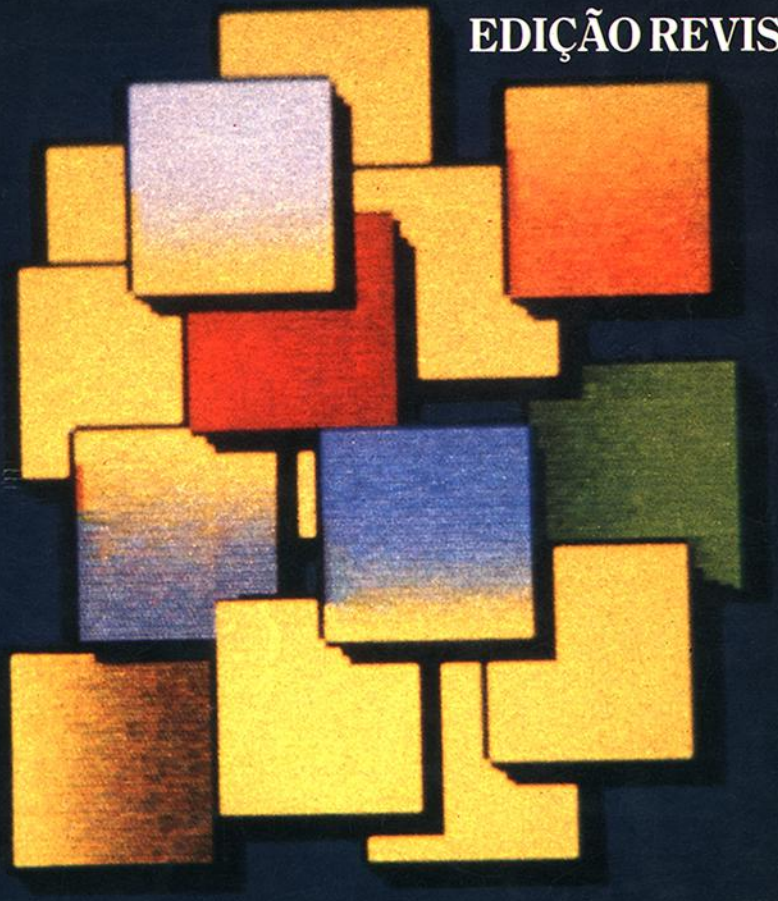
# COBOL
## PARA
## MICROCOMPUTADORES

EDIÇÃO REVISADA



McGraw-Hill

Mutsuo Ono

# COBOL
# para Microcomputadores


Valorize sua formação profissional,
seu futuro, sua consciência

# COBOL

## para Microcomputadores

**Mutsuo Ono**

*Professor graduado em Matemática e Física*
*Faculdade de Filosofia, Ciências e Letras Integrada de Guarulhos*

McGraw-Hill
São Paulo
Rua Tabapuã, 1105, Itaim-Bibi
CEP 04533
(011) 829-8604 e (011) 820-8528

*Rio de Janeiro* • *Lisboa* • *Porto* • *Bogotá* • *Buenos Aires* • *Guatemala* • *Madrid* • *México* • *New York* • *Panamá* • *San Juan* • *Santiago*

Auckland • Hamburg • Kuala Lumpur • London • Milan • Montreal • New Delhi • Paris • Singapore • Sydney • Tokyo • Toronto

*EDITOR*: MILTON MIRA DE ASSUMPÇÃO FILHO

*Consultor Técnico*: JOSÉ PITOL DE LARA
Coordenador do Departamento de Processamento de Dados
F.I.T.O. -- Fundação Instituto Tecnológico de Osasco
Professor da Faculdade Oswaldo Cruz

*Supervisora de Produção Editorial*: Daisy Pereira Daniel
*Supervisor de Produção Gráfica*: José Rodrigues
*Capa*: Hugo Sergio Faleiros
Roberto Bertini Renzetti

**Dados de Catalogação na Publicação (CIP) Internacional**
**(Câmara Brasileira do Livro,SP, Brasil)**

068c Ono, Mutsuo.

COBOL para microcomputadores / Mutsuo Ono. -- São Paulo : McGraw-Hill, 1989, 1990.

1. COBOL (Linguagem de programação para computadores)
2. Microcomputadores – Programação I. Titulo.

89-0078 CDD-001.6424
-001.642

**Índices para catálogo sistemático:**

1. COBOL : Linguagem de programação : Computadores : Processamento de dados 001.6424
2. Microcomputadores : Programação : Processamento de dados 001.642

*A minha esposa Natalina e*
*filhos Alessandro e Alessandra,*
*pela paciência e compreensão.*

# AGRADECIMENTOS

GLEICE C. M. GUERIOS
Mantenedora do Colégio Comercial Brasil

HENRIQUE MORI
Diretor da Unidade I do Colégio Comercial Brasil

ANAI RITSCHEL
Coordenadora da Área Técnica do Colégio Comercial Brasil

VALDEMAR MAZZI
Analista de Sistemas do Centro de Informática nº 2

ANDRÉ LUIZ NAVARRO VALVERDE
Programador do Centro de Informática nº 2

os quais tornaram este livro possível.

# APRESENTAÇÃO

## COBOL 80 / COBOL MS / COBOL MB

O presente trabalho, elaborado com a finalidade de possibilitar a todos os interessados na área de Informática a aprendizagem de Programação COBOL para Micro, apresenta as seguintes características:

*a.* teorias desenvolvidas com intercalação de programas resolvidos e programas propostos;

*b.* explanação dos comandos e instruções utilizadas em cada um dos programas resolvidos;

*c.* instruções e comandos apresentados na seqüência da necessidade e do grau de dificuldade;

*d.* programas rodados em Cobol MS, compatível com Cobol MB e Cobol 80.

Na esperança e expectativa de que esta forma de apresentação torne a Linguagem COBOL mais acessível, produtiva e interessante àqueles que iniciam a teoria e as técnicas desta mesma linguagem, agradeço antecipadamente as sugestões e críticas.

Mutsuo Ono

# SUMÁRIO

**Capítulo 1**

**Capítulo 2**



**Capítulo 3**

**Capítulo 4**



**Capítulo 5**

**Capítulo 6**

## Capítulo 7



## Capítulo 8



## Capítulo 9

**Capítulo 10**



**Capítulo 11**



**Apêndice A**

## Apêndice B



## Apêndice C



## Apêndice D



## Apêndice E

# CAPÍTULO 1

## NOTAÇÃO DE FORMATOS

Ao longo deste trabalho são prescritos "formatos gerais", para várias cláusulas e comandos, a fim de guiar os leitores na escrita de seus próprios comandos. São apresentados num sistema uniforme de notação explicado a seguir:

1. todas as palavras escritas inteiramente em letras maiúsculas são palavras reservadas. Estas palavras têm significados preestabelecidos;

2. as palavras maiúsculas sublinhadas são básicas e devem ser escritas no formato especificado, não devem ser abreviadas e nem ter erros;

3. as palavras maiúsculas não-sublinhadas são opcionais e, portanto, o seu uso é opcional. Porém, quando utilizadas, não devem conter erros de ortografia;

4. onde estiver escrito em minúsculo, tais como nomes de itens ou de procedimentos, são as partes que ficam a cargo do programador, atribuindo as palavras apropriadas durante a programação;

5. as partes entre { } indicam que obrigatoriamente devemos escolher uma das opções adequadas ao se programar;

6. as partes entre [ ] determinam que são opcionais e utilizadas conforme sua necessidade;

7. opções alternativas são representadas através de uma (/), separando as escolhas mutuamente exclusivas.

## DEFINIÇÃO

COBOL é a abreviação de *Common Business Oriented Language*. Essa é uma linguagem de computador orientada para negócios. As regras que comandam o uso da linguagem a fazem aplicável a problemas comerciais. Assim aplicações de natureza científica não podem ser adequadamente manipuladas pelo COBOL.

## CONCEITOS BÁSICOS DE COBOL

O conjunto de caracteres utilizáveis na linguagem COBOL é:

letras de A até Z
branco ou espaço
dígitos de 0 a 9
\+ sinal positivo
\- sinal negativo ou hífen
* asterisco
= sinal de igualdade

> sinal de relação (maior que)
< sinal de relação (menor que)
$ cifrão (dólar)
, vírgula
; ponto-e-vírgula
. ponto
" aspas
( abre parênteses
) fecha parênteses
/ barra

## PONTUAÇÃO

As seguintes regras gerais de pontuação se aplicam à codificação de programas fontes:

1. Como pontuação não convém que ponto-e-vírgula, ponto ou vírgula sejam precedidos por espaço, mas *devem* ser seguidos por espaço;

2. Pelo menos um espaço *deve* aparecer entre duas palavras e/ou literais sucessivos. Dois ou mais espaços sucessivos são tratados como um espaço único, exceto em literais não-numéricos;

3. Caracteres de relações e operadores aritméticos podem sempre ser precedidos por um espaço e seguido por outro;

4. Uma vírgula pode ser usada como separador entre operandos de um comando, ou entre dois subscritos;

5. Um ponto-e-vírgula ou uma vírgula podem ser usados para separar uma série de comandos ou cláusulas.

# FORMATO DA FOLHA DE CODIFICAÇÃO COBOL

```
      0         1    2    3    4    5    6    7
.....1.....678...0.2....0..... 0 ..... 0..... 0..... 0.....2
```

1. colunas 1 a 6: São utilizadas para escrever o número de seqüência de cada programa. Não há necessidade de se digitar esta seqüência no COBOL-80;

2. coluna 7:

   *a.* se um literal alfanumérico se estende por duas ou mais linhas, a primeira linha termina sem aspas, colocado-se um hífen (-) na coluna 7 da linha de continuação e, a partir da coluna 12, uma aspa.

   *b.* linha de comentário: Se houver um asterisco (*) na coluna 7, toda linha é considerada linha de comentário ou linha vazia. Dessa maneira, pode-se inserir um comentário em qualquer parte do programa;

   *c.* se, em vez do asterisco (*) for usada uma barra (/), haverá um avanço para a linha inicial de uma nova folha de formulário da impressora em tempo de compilação;

3. colunas 8 - 72: São colunas onde se escreve o programa propriamente dito;

4. colunas 8 - 11: Representam a margem A; nesta área começam todos os parágrafos (cabeçalhos) e seções.

5. colunas 12 - 72: nesta área estão todos os comandos. Estas colunas representam a margem B.

6. colunas 73 - 80: podem ser escritos os códigos ou o nome de identificação do programa fonte.

## APRESENTAÇÃO ESQUEMÁTICA DE UM PROGRAMA COBOL

IDENTIFICATION DIVISION

PROGRAM-ID. nome-do-programa.
[AUTHOR. comentários.]
[INSTALLATION. comentários.]

[DATE-WRITTEN. comentários.]

[DATE-COMPILED. comentários.]
[SECURITY. comentários.]

ENVIRONMENT DIVISION

[CONFIGURATION SECTION.]

[SOURCE-COMPUTER. entrada.]
[OBJECT-COMPUTER. entrada.]
[SPECIAL-NAMES. entrada.]
[INPUT-OUTPUT SECTION.]

[FILE-CONTROL. entrada.]

DATA DIVISION

[FILE SECTION
  entrada-de-descrição-de-arquivo
  entrada-de-descrição-de-registro.....]

```
[WORKING-STORAGE SECTION
   entrada-de-descrição-de-itens-de-dados.....]

[SCREEN SECTION
   entrada-de-descrição-de-tela
   entrada-de-descrição-de-item-de-dado.....]

[LINKAGE SECTION
   entrada-de-descrição-de-item-de-dado.....]

PROCEDURE DIVISION [USING identificador-1.....]
[nome-de-seção SECTION.]
[nome-de-parágrafo.
   [sentenças.....]
```

## PROGRAMA DE TELA

Este programa utiliza data do sistema, nome-de-condição (nível 88) e comando de decisão.

### 1. Objetivo do Programa

Calcular a área de um triângulo.

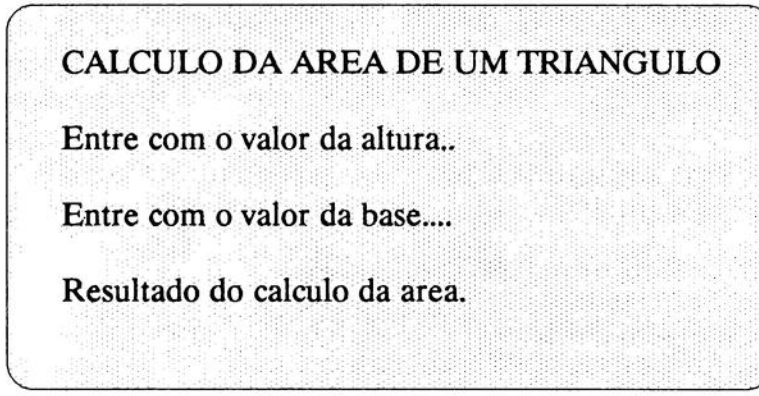
### 2. Procedimentos

1. exibir a tela formatada;

2. aceitar os valores da base e da altura do triângulo;

3. exibir a área calculada.

## 3. Formato da Tela (Video Layout)




## Codificação do Programa

```
1     IDENTIFICATION          DIVISION.
2     PROGRAM-ID.             TRIANG.
3     AUTOR.                  COLEGIO BRASIL.
4     INSTALLATION.           COLEGIO BRASIL.
5     DATE-WRITTEN.           01/02/88.
6     DATE-COMPILED. Tue Jan 01 09:30:35 1988.
7     SECURITY.       LIVRE PARA TODOS.
8     ENVIRONMENT             DIVISION.
9     CONFIGURATION SECTION.
10    SOURCE-COMPUTER. S-600.
11    OBJECT-COMPUTER. S-600.
12    SPECIAL-NAMES.
13          DECIMAL-POINT IS COMMA.
14    DATA DIVISION.
15    WORKING-STORAGE SECTION.
16    77    BASE      PIC 9(04)   VALUE ZEROS.
17    77    ALTURA    PIC 9(04)   VALUE ZEROS.
18    77    HAREA     PIC 9(06)   VALUE ZEROS.
19
20    SCREEN SECTION.
21    01    TELA.
22    02 BLANK SCREEN.
23    02 LINE 04 COLUMN 21 VALUE
24    "CALCULO DA AREA DE UM TRIANGULO".
25    02 LINE 06 COLUMN 15 VALUE
26    "AUTOR: Colegio Brasil - NR 000 - Serie: 2 - TU: X".
```

```
27      2 LINE 10 COLUMN 21 VALUE "Entre com o valor da altura..".
28      02 LINE 12 COLUMN 21 VALUE "Entre com o valor da base....".
29      02 LINE 14 COLUMN 21 VALUE "Resultado do calculo da area.".
30      PROCEDURE DIVISION.
31      INICIO.
32              DISPLAY TELA.
33      DADOS.
34              ACCEPT (10, 51) ALTURA.
35              ACCEPT (12, 51) BASE.
36      CALCULO.
37              MULTIPLY BASE BY ALTURA GIVING HAREA.
38              DIVIDE 2 INTO HAREA.
39      FIM.    DISPLAY (14 51) HAREA.
40              DISPLAY (24, 31) "FIM DE PROGRAMA".
41              STOP RUN.
```

## Explanação e Instruções Introduzidas no Programa

Todo programa COBOL consta de quatro divisões denominadas: IDENTIFICATION DIVISION, ENVIRONMENT DIVISION, DATA DIVISION e PROCEDURE DIVISION.

IDENTIFICATION DIVISION. Serve para identificar o programa e é sempre escrita em primeiro lugar.

ENVIRONMENT DIVISION. Nesta divisão são definidos o equipamento e os arquivos de dados a serem utilizados.

DATA DIVISION. São descritos os arquivos, os registros de dados, definição de áreas de memória e formatação de telas.

PROCEDURE DIVISION. É a última divisão do programa e nela descrevemos a lógica dos procedimentos a seguir.

. O ponto é colocado ao fim de uma sentença que pode constar de quantos comandos se queira. O ponto deve ser seguido de pelo menos um espaço.

PROGRAM-ID. TRIANG. Designação do nome do programa. Em todo programa deve ser escrito este nome-de-parágrafo. Todos os demais parágrafos: AUTHOR, INSTALLATION, DATE-WRITTEN, DATE-COMPILED e SECURITY são opcionais, servindo apenas para informação e documentação.

Nome de programa é qualquer seqüência de caracteres alfanuméricos, sendo que o primeiro caractere deve ser alfabético. Somente os 6 caracteres iniciais do nome são retidos pelo computador.

CONFIGURATION SECTION. Seção de Equipamento. É descrita a configuração do equipamento a ser utilizado para o programa em questão. Para o COBOL-MS, esta Seção é tratada como comentário, portanto poderão existir ou não os parágrafos: SOURCE-COMPUTER, OBJECT-COMPUTER e SPECIAL-NAMES.

SPECIAL-NAMES. DECIMAL-POINT IS COMMA. O uso da vírgula para separar a parte inteira da parte fracionária é especificado por esta entrada.

Se não se deseja que o símbolo monetário seja o sinal de dólar, o programador pode especificar um literal não-numérico de um caractere apenas na cláusula CURRENCY SIGN.

WORKING-STORAGE SECTION. É a Seção da DATA DIVISION que trabalha com a memória, armazenando e reservando resultados.

| 77 | BASE | PIC | 9(4) | VALUE | ZEROS. |
|---|---|---|---|---|---|
| número de nível | nome-de-dado | máscara | tipo e tamanho do campo | valor inicial do campo | constante figurativa |

NÚMEROS DE NÍVEIS (01 a 49, 77 e 88):

Os números de níveis são utilizados para representar a construção de níveis de itens elementares e itens de grupo.

Itens elementares são aqueles que estão na sua forma mais resumida, e itens de grupo são aqueles constituídos por itens elementares.

Um registro é sempre iniciado pelo número de nível 01 que indica ser o maior agrupamento, e à medida que se detalha os dados, os números de níveis crescem.

Pode-se utilizar para um número de nível um valor até 49. O número de nível não necessita crescer de uma forma contínua.

Interpretam-se como sendo de um mesmo item de grupo os campos de número de nível maior que o deste item de grupo, até ser encontrado um número de nível menor ou igual.

O nível 88 é utilizado para itens de condição (nome-de-condição).

Os números de nível menores que 10 podem ser escritos com um dígito apenas.

Na Working-Storage Section pode-se utilizar o número de nível 77 para itens independentes, que não têm relacionamento de níveis com os nomes de dados.

Exemplo:

```
01 REGISTRO.
    02 CODIGO.
        03 NUMERO
        03 DC ....
    02 NOME...
    02 ENDERECO...
```

Um nome de dado ou a palavra-chave FILLER deve ser a primeira palavra seguinte ao número de nível em cada especificação da descrição de registro ou de campos, conforme o formato geral:

**Número de nível {nome-de-dado/FILLER}**

BASE, HAREA e ALTURA. São nomes-de-dados e são atribuídos pelo programador.

**Regras para Formação de Nomes-de-Dados**

*a.* ter de 1 a 30 caracteres;

*b.* pode conter letra, número e hífen (-);

*c.* não deve começar e nem terminar com hífen (-);

*d.* não pode ser uma palavra reservada do COBOL.

PIC. Especifica os tamanhos, os detalhes de edição e as características gerais dos itens.

| | | | |
|---|---|---|---|
| Pode ser: | ALFANUMÉRICO | - | representado por X |
| | ALFABÉTICO | - | representado por A |
| | NUMÉRICO | - | representado por 9 |

Esta cláusula só pode ser utilizada para itens elementares.

Um número inteiro entre parênteses seguido de X, 9 ou o número de repetições deste caractere de PICTURE.

9(4). Especifica que o campo é numérico e ocupa quatro bytes de memória equivalendo a 9999.

VALUE. Indica o valor inicial do item de trabalho.

O formato da cláusula é: **VALUE IS** literal.

ZEROS. Constante figurativa. É uma palavra reservada do COBOL e significa que o nome-de-dado reservado para trabalho tem o seu valor inicial igual a 0000.

As constantes figurativas podem ser:

| | |
|---|---|
| ZEROS ou ZERO: | igual a zeros; |
| SPACES ou SPACE: | igual a brancos; |
| LOW-VALUE: | igual ao menor valor; |
| HIGH-VALUE: | igual ao maior valor; |
| ALL literal: | indica que tal constante não-numérica de uma posição ocupará várias posições em seqüência. |

SCREEN SECTION. Esta Seção define o formato da tela no vídeo.

Da mesma forma que na Working-Storage Section, as descrições podem ser agrupadas através de atribuições de números de níveis apropriados.

BLANK SCREEN. Indica a colocação de espaços em toda a tela (apaga a tela).

LINE e COLUMN. Indica uma posição definida da tela. LINE corresponde às linhas de 1 a 24 e COLUMN corresponde às colunas de 1 a 80.

VALUE. Especifica o 'string' de caracteres que aparece na tela do vídeo quando for executada a instrução DISPLAY.

Exemplo:

```
02 T1 LINE 10 COLUMN 20 VALUE "COLEGIO BRASIL".
```

"CÁLCULO DA ÁREA DE UM TRIÂNGULO". Representa um literal não-numérico. As características do literal não-numérico são seqüências de caracteres entre aspas.

**Regras para Formação de Literal Não-Numérico**

*a.* máximo de 120 caracteres são permitidos;

*b.* exceção de aspas.

Na PROCEDURE DIVISION. Os nomes-de-dados INICIO, DADOS, CALCULO e FIM são denominados parágrafos ou rotinas e são nomes descritos (criados) pelo programador. Todos os parágrafos ou rotinas devem necessariamente iniciar na Margem A, isto é, coluna 8 a 11.

Dentro dos parágrafos colocam-se os comandos que podem formar sentenças.

DISPLAY TELA. DISPLAY é uma instrução reservada do COBOL e tem a função de exibir ou imprimir pequena quantidade de dados no vídeo ou em outro periférico, durante o processamento do programa. No nosso exemplo de programa será exibida a tela formatada na SCREEN SECTION.

ACCEPT (10, 31) BASE. ACCEPT (12, 31) ALTURA. A instrução ACCEPT é utilizada para receber um valor no vídeo durante o processamento do programa.

Neste caso o micro aguarda que sejam digitados os valores numéricos para os campos BASE e ALTURA a fim de que o processamento continue. Após digitar os valores de BASE e ALTURA pressionar a tecla <RETURN> ou <ENTER>.

MULTIPLY BASE BY ALTURA GIVING HAREA. O presente comando especifica que estamos multiplicando o conteúdo do campo BASE pelo conteúdo do campo ALTURA, armazenando o resultado no campo denominado HAREA.

DIVIDE 2 INTO HAREA. Este comando indica que estamos dividindo o conteúdo do campo HAREA por 2 e o resultado será armazenado no próprio campo HAREA.

STOP RUN. encerra o programa, fecha todos os arquivos porventura existentes e passa o controle de processamento para o Sistema Operacional.

## SCREEN SECTION

Numero-de-nível [nome-da-tela]

[BLANK SCREEN]

[LINE number [PLUS] inteiro-1 COLUMN number [PLUS]

inteiro-2] [BLANK LINE/BELL/BLINK/HIGHLIGHT/REVERSE-

VIDEO/UNDERLINE]

[VALUE literal]

[PIC estrutura]

[FROM/TO/USING identificador]

AUTO

SECURE

FULL

REQUIRED

Na PROCEDURE DIVISION a SCREEN SECTION pode ser acessada pelas instruções ACCEPT e DISPLAY. Utilizam-se todos os procedimentos da WORKING-STORAGE SECTION para a sua descrição.

### Line e Column

É utilizada para indicar uma posição definida da tela. LINE representa o número de linha (1 a 24) e COLUMN representa o número de coluna (1 a 80) na tela.

Se LINE e COLUMN não forem especificadas a linha corrente da tela não será ajustada.

A cláusula LINE PLUS inteiro-1 ou COLUMN PLUS inteiro-2 faz com que o inteiro especificado seja adicionado à linha ou coluna corrente, e o resultado seja usado como a linha ou coluna no qual o item de tela corrente deve começar. Se LINE (COLUMN) for especificado sem inteiro, é assumido LINE PLUS 1 (COLUMN PLUS 1).

BLANK LINE. Faz com que seja apagada a tela desde a posição corrente do cursor até o fim da linha corrente, sem alterar a posição do cursor.

BELL. Acionará o alarme sonoro do terminal quando o sistema estiver preparado para ACCEPT (aceitar) um campo. BELL não tem efeito em campos de saída.

BLINK. O efeito da opção BLINK é de apresentar o rótulo da tela piscante.

HIGHLIGHT. O efeito da opção HIGHLIGHT é apresentar o rótulo da tela de modo mais intenso.

REVERSE-VIDEO. Uma declaração REVERSE-VIDEO apresenta o rótulo de modo invertido na tela, isto é, letras em preto sobre fundo verde (no caso dos vídeos monocromáticos).

UNDERLINE. O efeito da opção UNDERLINE é apresentar o rótulo da tela sublinhado.

PIC. Especifica o formato do item elementar de acordo com as regras estabelecidas para a cláusula PIC da WORKING-STORAGE SECTION.

USING/FROM/TO. Quando um item de dado é exibido na tela, FROM ou USING transfere o conteúdo de um item de dado ou de um literal da área para um item temporário que está definido na cláusula PICTURE. Este valor é então exibido no vídeo. Quando um item é aceito

via teclado, TO ou USING, implicitamente, move o conteúdo do item para um item de dado declarado em TO ou USING.

Exemplo:

```
WORKING-STORAGE SECTION.

77 DATA-1     PIC 9(06) VALUE ZEROS.
SCREEN SECTION.
01 TELA1.
     02 TL1-A LINE 10 COLUMN 10 PIC 9(06) USING DATA-1.
:
:
PROCEDURE DIVISION.
:
     ACCEPT TELA1.
```

Ação: os dados digitados serão transferidos da TELA1 para DATA1-A.

FULL. Quando um item de dado é aceito na tela, a cláusula FULL fica aguardando até que todos os caracteres do campo sejam digitados.

Exemplo:

```
05 LINE 10 COLUMN 10 PIC X(05) TO AUX FULL.
```

REQUIRED. Quando um item de dado é aceito na tela, a cláusula REQUIRED fica aguardando até que pelo menos um caractere seja digitado.

Exemplo:

```
05 LINE 05 PIC X(05) TO AUX REQUIRED.
```

AUTO. Especifica que ao preencher todas as posições do campo o cursor salta automaticamente para o próximo campo.

Exemplo:

```
WORKING-STORAGE SECTION.
77 CODIGO   PIC 9(05)      VALUE ZEROS.
```

```
77 CHEQ-DIG PIC 9     VALUE ZERO.
:
SCREEN SECTION.
01 TELA1.
     02 BLANK SCREEN.
     02 LINE 10 COLUMN 10 VALUE "DIGITE O CODIGO E O- DV".

01 TELA2.
     02 T2-A LINE 10 COLUMN 50 PIC 9(06) USING CODIGOAUTO.
     02 T2-B LINE 10 COLUMN 57 PIC 9 USING CHEQ-DIG AUTO.
:
PROCEDURE DIVISION.
     DISPLAY TELA1.
     ACCEPT TELA2.
```

Ação: ao digitar o campo CODIGO, o cursor salta automaticamente para o campo seguinte.

SECURE. Permite inibir o ECO dos caracteres digitados. Preenche os dados digitados com asterisco (*) na execução da instrução ACCEPT.

Exemplo:

```
WORKING-STORAGE SECTION.
77 CHAVE PIC X(09) VALUE SPACES.
:
SCREEN SECTION.
01 TELA1.
     02 BLANK SCREEN.
     02 T1-A LINE 05 COLUMN 10 VALUE "ENTRE COM A CHAVE- =".
01 TELA2.
     02 T2-A LINE 05 COLUMN 30 PIC X(09) USING CHAVE SECURE.
:
PROCEDURE DIVISION.
:
     DISPLAY TELA1.
     ACCEPT TELA2.
     IF CHAVE = "SEGURANCA" GO TO ....
```

Ação: no vídeo aparecerá *********

# INSTRUÇÃO DISPLAY

```
DISPLAY [especificação-da-posição]

    {identificador/constante/ERASE} [UPON nome-mnemônico] [nome-de-tela].
```

O identificador deve referenciar um item de dado com tamanho menor ou igual a 1920 caracteres.

Constante específica um literal não-numérico.

O nome-mnemônico deve ser definido no parágrafo SPECIAL-NAMES da CONFIGURATION SECTION.

Exemplo:

```
SPECIAL-NAMES. PRINTER IS IMPRESSAO.
```

Ação: IMPRESSAO é o nome mnemônico da PRINTER.

O nome-de-tela deve ser definido na SCREEN SECTION da DATA DIVISION.

O comando DISPLAY nome-de-tela provoca a transferência da informação do nome-de-tela (ou cada item de tela elementar subordinado ao nome-de-tela) para a tela.

Para cada item de tela tendo uma especificação VALUE ou USING, o literal especificado ou campo é a fonte do dado apresentado.

A especificação da posição permite definir as coordenadas de saída no vídeo dos dados a serem exibidos. Nessa especificação podemos utilizar um número inteiro para linha e coluna ou LIN e COL que são registradores especiais e representam linha e coluna respectivamente.

LIN e COL podem ser incrementados ou decrementados através de comandos aritméticos e podem variar de 1 a 24 e de 1 a 80 respectivamente.

Exemplos:

```
DISPLAY (01, 01) ERASE.
```

Ação: Limpa a tela a partir da linha e coluna especificada até o fim.

```
DISPLAY (01, 05) "EXEMPLO".
```

Ação: Exibe o literal EXEMPLO a partir da primeira linha e quinta coluna.

```
DISPLAY (01, COL) "EXEMPLO".
```

Ação: Exibe o literal EXEMPLO a partir da primeira linha e o valor atual do COL.

```
DISPLAY (LIN, 20) "EXEMPLO".
```

Ação: Exibe o literal EXEMPLO a partir do valor do LIN e da vigésima coluna.

```
DISPLAY (LIN, COL) "MANUAL".
```

Ação: Exibe o literal MANUAL a partir dos valores especificados em LIN e COL.

```
WORKING-STORAGE SECTION.
77 CT-AUXILIAR PIC 9(05) VALUE 99999.
:
PROCEDURE DIVISION.
:
        DISPLAY (10, 15) CT-AUXILIAR.
```

Ação: Exibe na posição especificada o número 99999.

```
DISPLAY (LIN, 15) "O VALOR E = " CT-AUXILIAR.
```

Ação: Exibe na posição especificada o seguinte:

O VALOR E = 99999.

```
SCREEN SECTION.
01 TELA.
    02 BLANK SCREEN.
    02 LINE 1 COLUMN 1 VALUE "FOLHA DE PAGAMENTO.
    02 LINE 1 COLUMN 33 VALUE "MANUTENCAO DE-ARQUIVOS".
    02 LINE 1 COLUMN 68 VALUE "S E L E T O R".
    02 LINE 8 COLUMN 30 VALUE "S E L E T O R".
    02 LINE 10 COLUMN 30 VALUE "1 - MANUT FUNCIONARIOS.
    02 LINE 12 COLUMN 30 VALUE "2 - MANUT BANCOS".
    02 LINE 14 COLUMN 30 VALUE "3 - MANUT EMPRESA".
    02 LINE 16 COLUMN 30 VALUE "SELECAO OU RETORNO".
:
PROCEDURE DIVISION.
:
    DISPLAY TELA.
```

Ação: Exibe no vídeo os literais definidos na SCREEN SECTION, nas suas respectivas linhas/colunas.

## INSTRUÇÃO ACCEPT

Os formatos da instrução ACCEPT são:

**Formato 1**

```
ACCEPT nome-identificador [FROM DATE/DAY/TIME/SCAPE KEY].
```

Neste formato a instrução ACCEPT permite obter informações dos itens de dados identificados pelo sistema durante a execução do programa.

DATE. Consiste num valor de 6 dígitos com o seguinte formato: AAMMDD, onde AA é o ano, MM é o mês e DD é o dia.

Exemplo:

```
WORKING-STORAGE SECTION.
01 CAMPO-PARA-DATA.
    02 ANO PIC 9(02) VALUE ZEROS.
```

```
           02 MES PIC 9(02) VALUE ZEROS.
           02 DIA PIC 9(02) VALUE ZEROS.
:
PROCEDURE DIVISION.
:
                ACCEPT CAMPO-PARA-DATA FROM DATE.
```

Ação: O programa obtém o conteúdo data através do sistema.

DAY. Consiste num valor de 5 dígitos, com o seguinte formato: AANNN, onde AA é o ano e NNN é o dia do ano em ordem seqüencial de 1 a 366.

TIME. Consiste num valor de 8 dígitos, com o seguinte formato: HHMMSSFF onde HH é hora, MM é minuto, SS é segundo e FF é a décima fração de segundo.

ESCAPE KEY. Um código de dois dígitos gerado pela tecla que terminou a execução mais recente do formato-3 ou formato-4 da instrução ACCEPT. O nome-identificador pode ser interrogado para determinar exatamente qual tecla foi digitada.

O nome-identificador deve conter os seguintes valores referentes ao código de tecla:

00 - tecla ENTER

01 - tecla ESC

02 ... 11 - teclas de função (F1 ... F10)

Com a opção AUTO-SKIP na última instrução ACCEPT o valor do ESCAPE KEY é igual a 00.

O nome-identificador deve especificar um item numérico inteiro sem sinal e o tamanho deve ser compatível com o item de dado definido pelo sistema. Todas as regras da instrução MOVE são válidas para este formato.

Exemplo:

```
WORKING-STORAGE SECTION.
77 DADOS-1 PIC X(10).
77 DADOS-2 PIC 9(05).
77 DADOS-3 PIC X(10).
77 CONTADOR PIC 9(02).

PROCEDURE DIVISION.
⋮
    ACCEPT (10 30) DADOS-1.
    ACCEPT (12 30) DADOS-2.
    ACCEPT (14 30) DADOS-3.
    ACCEPT CONTADOR FROM ESCAPE KEY.
    IF CONTADOR = 01
        GO TO ENTER-DADOS.
⋮
```

**Formato 2**

```
ACCEPT nome-identificador.
```

Este formato permite receber um conjunto de caracteres de entrada (dados digitados via teclado).

Se o campo receptor é alfanumérico ocorre a transferência dos dados de entrada para o campo receptor com o número de caracteres igual ao definido no item receptor, isto é, alinhamento à esquerda. Truncamento à esquerda ocorrerá se o campo receptor for descrito como JUSTIFIED RIGHT. Para um campo receptor numérico ou numérico editado é verificada a validade dos dados de entrada dependendo da cláusula PIC do campo receptor.

Os dígitos contidos neste caso são:

1. dígito de 0 a 9;

2. ponto decimal pode ser especificado apenas uma vez dentro dos dados de entrada e o campo receptor deve conter caracteres que

representam a parte fracionária (9, Z, * ou um caractere de inserção flutuante). Esses caracteres aparecem à direita do ponto decimal (.);

3. o sinal operacional + e - é considerado somente como o primeiro ou último caractere da cadeia de dados de entrada e somente se a PIC do campo receptor contiver um dos indicadores de sinal S, +, -, CR ou DB.

   Se o dado é inválido, o sistema operacional mostra a seguinte mensagem no vídeo:

   "INVALID NUMERIC INPUT PLEASE RETYPE"

A transferência dos dados para o campo receptor será executada como na instrução MOVE, para o campo receptor a partir de um campo fonte hipotético com as seguintes características:

1. PIC com a forma S9...9V9...9;

2. USAGE DISPLAY;

3. comprimento igual ao número de dígitos na cadeia de dados de entrada;

4. tantas posições à direita do ponto decimal fictício quantas forem os dígitos à direita do ponto decimal explícito na cadeia de dados (zero se não houver ponto decimal na cadeia de dados);

5. conteúdo corrente igual à cadeia de dígitos incluído na cadeia de dados de entrada;

6. se o conjunto de dados contém o caractere -, o sinal operacional é indicado separadamente com condições negativas, caso contrário, condições positivas.

**Formato 3**

```
ACCEPT (especificação-da-posição) nome-identificador [WITH PROMPT/
UPDATE/ AUTO-SKIP/ SPACE-FILL/ ZERO-FILL/ TRAILING- SIGN/
LENGTH-CHECK/ BEEP/ LEFT-JUSTIFY/ RIGHT-JUSTIFY/
EMPTY-CHECK/NO-ECHO].

Especificação-da-posição = ([LIN [{ + OU - } inteiro- 1]/inteiro-2],
[COL[{ + ou - } inteiro-3]/inteiro-4])
```

No formato 3 a instrução ACCEPT permite definir a localização do dado de entrada no vídeo e as características do campo de entrada de dados.

Para definir a localização do campo de entrada de dados é necessário especificar a posição inicial do campo (linha e coluna) dentro de parênteses. O LIN e o COL são registradores especiais do COBOL.

Se LIN for especificada, o campo de entrada de dados começará na linha da tela cujo número seja igual ao valor do registrador especial LIN, incrementado (ou decrementado) por inteiro-1. Se inteiro-2 for especificado, o campo de entrada de dados começará na linha da tela inteiro-2. Se LIN, ou inteiro-2 não for especificado, o campo de entrada de dados começará na linha da tela que contiver a posição corrente do cursor.

Se COL for especificado, o campo de entrada de dados começará na coluna da tela cujo número for igual ao valor do registrador especial COL, incrementado (ou decrementado) por inteiro-3. Se inteiro-4 for especificado, o campo de entrada de dados começará na coluna da tela inteiro-4. Se COL, ou inteiro-4 não for especificado, o campo de entrada de dados começará na coluna da tela que contiver a posição corrente do cursor.

Exemplos:

```
ACCEPT    (LIN, COL)
ACCEPT    (10, 30)
ACCEPT    (LIN, 10)
ACCEPT    (20, COL)
ACCEPT    (LIN + 5, COL + 5)
```

Se o campo receptor for definido como item alfanumérico o dado digitado aparece no vídeo na posição especificada. O comprimento do campo de entrada de dado é igual ao do campo receptor. Se a digitação ultrapassa o comprimento do campo de entrada de dado, ocorre a sinalização audível e em conseqüência impedimento de qualquer digitação posterior. Pressionando a tecla ENTER haverá a transferência dos dados digitados para o campo receptor.

Para o campo receptor definido como numérico, o dado digitado deve ser numérico ou caractere de edição. Um campo receptor numérico editado não permite a especificação UPDATE. A seqüência de digitação deve ser: posição do sinal operacional, posições dos dígitos inteiros, posição do ponto decimal (ou vírgula decimal) e posição dos dígitos fracionários.

SPACE-FILL ou ZERO-FILL. Estas duas opções não devem ser especificadas numa mesma instrução ACCEPT.

Após o término da instrução ACCEPT, se o campo receptor é alfanumérico, com a opção SPACE-FILL o campo de entrada de dados e o campo receptor são preenchidos com espaços nas posições onde não forem digitados caracteres. Com a opção ZERO-FILL as posições não-digitadas são preenchidas com ZEROS.

Exemplo:

campo-receptor.....: 02 ALPHA PIC X(10).

conteúdo inicial.. : ABCDEFGHIJ

instrução ACCEPT...: ACCEPT (10, 10) ALPHA WITH SPACE-FILL.

digitação do campo.: COLEGIO

após ENTER.....: COLEGIO

campo receptor.....: COLEGIO

campo-receptor.....: 02 NUMERO PIC 9(5)V99.

conteúdo inicial...: 12345^67

instrução ACCEPT...: ACCEPT (10, 10) NUMERO WITH ZERO-FILL.

digitação do campo..: 987.6

após ENTER.....: 00987^60

campo receptor.....: 00987^60

TRAILING-SIGN. Para o campo receptor numérico ou numérico editado, o TRAILING-SIGN faz com que o sinal operacional apareça na posição mais à direita do campo de entrada de dados. Normalmente o sinal vem à esquerda do campo.

Exemplo:

campo-receptor.....: 03 VALOR PIC S9(04)V99.

conteúdo inicial...: 1234^56

instrução ACCEPT...: ACCEPT (15, 10) VALOR WITH PROMPT
TRAILING-SIGN.

digitação do campo..: 9876.00

digitação do sinal.: 9876.00-

digitação do ponto.: 9876.00-

digitação da fração: 9876.50-

após ENTER.....: 9876.50-

campo receptor.....: 9876^50-

BEEP. Faz com que o alarme seja acionado quando o ACCEPT tiver sido inicializado e o sistema estiver pronto para aceitar a entrada de dados.

LENGTH-CHECK. *a.* faz com que um caractere terminador de campo seja ignorado, a menos que todas as posições dos dados de entrada sejam preenchidas, quando se tratar de campo receptor alfanumérico;

*b.* faz com que o caractere ponto decimal recebido seja ignorado, a menos que todos os dígitos da parte inteira do campo tenham sido digitados; e um caractere terminador de campo seja ignorado, a menos que todas as posições do campo tenham sido preenchidas.

No formato 3 a instrução ACCEPT permite definir a localização do dado de entrada no vídeo e as características do campo de entrada de dados.

Para definir a localização do campo de entrada de dados é necessário especificar a posição inicial do campo (linha e coluna) dentro de parênteses. LIN e COL são registradores especiais do COBOL e têm as mesmas funções descritas na instrução DISPLAY.

PROMPT. Provoca o preenchimento com pontos (.), no campo de entrada de dados, quando o campo receptor é alfanumérico e o preenchimento com (___), no campo de entrada de dados, quando o campo receptor é numérico ou numérico-editado, antes da digitação dos caracteres. No caso do campo receptor ser numérico ou numérico-editado a posição no vídeo do ponto decimal e do sinal operacional é preenchida com ponto decimal e espaço, respectivamente.

Exemplo:

Campo receptor: `03 PRODUTO PIC X(08).`

Instrução ACCEPT: `ACCEPT (10, 10) PRODUTO WITH PROMPT.`

Ação: antes da digitação dos dados aparecerá no vídeo:

Campo receptor: 03 VALOR PIC 9(05)V99.

Instrução ACCEPT: ACCEPT (10, 10) VALOR WITH PROMPT.

Ação: antes da digitação dos dados aparecerá no vídeo:

_ _ _ _ _ . _ _

UPDATE. A opção UPDATE provoca o preenchimento do campo de entrada de dados do vídeo com caracteres contidos no campo receptor, antes da digitação dos caracteres.

Se o operador acionar somente as teclas RETURN ou ENTER o conteúdo do campo receptor permanecerá inalterado. Todas as posições do campo de entrada de dados que forem digitadas pelo operador serão substituídas pelos dados correspondentes do campo receptor.

Exemplo:

Campo receptor: 05 S-N PIC X(08) VALUE "CONFIRMA".

Instrução ACCEPT: ACCEPT (10, 10) S-N WITH UPDATE.

Ação: no vídeo aparecerá CONFIRMA acionando a tecla RETURN o resultado no campo receptor será CONFIRMA; se for acionado CTL X o conteúdo apagará do vídeo.

AUTO-SKIP. Provoca o término de digitação quando forem preenchidas todas as posições do campo. O acionamento da tecla RETURN torna-se desnecessário.

LEFT-JUSTIFY. É tratado como comentário.

RIGHT-JUSTIFY. Os caracteres ocupam as posições mais à direita, isto é, alinham pela direita, após o término da instrução ACCEPT. A transferência dos dados para o campo receptor é controlada pela declaração JUSTIFIED, se especificada. Desta forma esta opção executa o deslocamento somente no campo de entrada de dados (vídeo).

EMPTY-CHECK. Todos os campos sem conteúdo são ignorados até que pelos menos um caractere seja digitado.

NO-ECHO. Inibe o eco dos caracteres digitados, preenchendo com asteriscos o campo de entrada durante a execução da instrução ACCEPT.

**Formato 4**

ACCEPT nome-da-tela [ON **ESCAPE** instrução-incondicional]

Provoca a transferência da informação digitada pelo teclado para a área especificada na SCREEN SECTION. O nome da tela especificada na instrução ACCEPT deve ser um item elementar ou grupo de item da SCREEN SECTION. Se o item do grupo for especificado somente com a cláusula VALUE ou FROM, a execução da instrução não terá nenhum efeito.

Se a tecla ESCAPE é pressionada durante a entrada de dados, todo ACCEPT é terminado sem mover o campo correspondente para o item associado do item TO ou USING, o valor ESCAPE KEY é ajustado em 01, e o comando ON ESCAPE é executado. Se F1 a F10 forem pressionadas, o valor ESCAPE KEY apropriado é ajustado e todo ACCEPT é terminado. Se a tecla ENTER for pressionada, o valor ESCAPE KEY é ajustado em 00 e o cursor é movido para o próximo campo de entrada definido sob o nome-de-tela, se existir. Se o campo corrente é o último campo, todo ACCEPT é terminado.

Quando o campo é terminado por uma tecla de função (F1 a F10) ou ENTER, os conteúdos do campo corrente são movidos para o item associado TO ou USING, exceto no caso quando não forem digitados nenhum caractere de dados nem caracteres de edição naquele campo. Isto permite ao operador tabular para frente ou para trás, através dos campos de entrada, sem afetar o conteúdo dos itens receptores.

Se um campo de entrada for especificado com a cláusula USING, FROM ou TO (na SCREEN SECTION), fará com que a instrução ACCEPT seja executada como na opção UPDATE descrita no formato 3.

# CODIFICAÇÃO DE PROGRAMA DEMONSTRATIVO DAS INSTRUÇÕES SCREEN, SECTION, ACCEPT E DISPLAY

```
1    IDENTIFICATION  DIVISION.
2    PROGRAM-ID.          DEMO1.
3    ENVIRONMENT     DIVISION.
4    CONFIGURATION   SECTION.
5    SPECIAL-NAMES.  DECIMAL-POINT IS COMMA.
6    DATA DIVISION. \I
7    WORKING-STORAGE SECTION.
8    77 ESC-SEL PIC 99 VALUE ZEROS.
9    77 QQ-TECLA PIC X VALUE SPACE.
10   01   DADOS-DE-ENTRADA.
11        02 CPO-01 PIC X(20) VALUE SPACES.
12        02 CPO-02 PIC S9(05) VALUE ZEROS.
13        02 CPO-03 PIC S9(05)V99 VALUE ZEROS.
14   SCREEN SECTION.
15   01   LIMPA-TELA.
16        02 BLANK SCREEN.
17   01   TELA-DEMO.
18        02 LINE 01 COLUMN 25 VALUE
19        "DEMONSTRACAO DO COBOL INTERATIVO".
20        02 LINE 03 COLUMN 16 VALUE "TESTE DE ATRIBUTOS DA SCREEN, DIS
21 -      "PLAY E ACCEPT".
22        02 LINE 05 COLUMN 08 VALUE "1. Digite qualquer tipo de caract
23 -      "eres".
24        02 LINE 09 COLUMN 08 VALUE "2. Digite somente numericos::::::
25 -      "::::".
26        02 LINE 13 COLUMN 08 VALUE "3. Digite numerico decimal::::::::
27 -      "::::".
28        02 LINE 20 COLUMN 08 VALUE "1. Mensagem".
29        02 LINE 22 COLUMN 08 VALUE "2. Mensagem:".
30        02 LINE 24 COLUMN 08 VALUE "3. Mensagem:".
31   01   TELA-DEMO1.
32        02 LINE 01 COLUMN 25 VALUE
33        " DEMONSTRACAO DO COBOL INTERATIVO " REVERSE-VIDEO.
34        02 LINE 03 COLUMN 16 VALUE "TESTE DE ATRIBUTOS DA SCREEN, DIS
35 -      "PLAY E ACCEPT" HIGHLIGHT.
36        02 LINE 05 COLUMN 08 VALUE "1. Digite qualquer tipo de caract
37 -      "eres".
38        02 SS-CPO-01 LINE 05 COLUMN 47 PIC X(20) TO CPO-01 BELL.
39        02 LINE 09 COLUMN 08 VALUE "2. Digite somente numericos::::::
40 -      "::::".
41        02 SS-CPO-02 LINE 09 COLUMN 47 PIC S9(05) TO CPO-02
```

```
42          UNDERLINE.
43          02 LINE 13 COLUMN 08 VALUE "3. Digite numerico decimal:::::::
44 -        ":::::".
45          02 SS-CPO-03 LINE 13 COLUMN 47 PIC S9(05)V99 TO CPO-03
46          SECURE.
47          02 LINE 20 COLUMN 08 VALUE "1. Mensagem:" BLINK.
48          02 LINE 22 COLUMN 08 VALUE "2. Mensagem:" UNDERLINE.
49          02 LINE 24 COLUMN 08 VALUE "3. Mensagem:" HIGHLIGHT.
50    01    TELA-DEMO2.
51          02 LINE 01 COLUMN 25 VALUE
52          " DEMONSTRACAO DO COBOL INTERATIVO " REVERSE-VIDEO.
53          02 LINE 03 COLUMN 16 VALUE "TESTE DE ATRIBUTOS DA SCREEN, DIS
54 -        "PLAY E ACCEPT" REVERSE-VIDEO.
55          02 LINE 05 COLUMN 08 VALUE "1. Digite qualquer tipo de caract
56 -        "eres".
57          02 FF-CPO-01 LINE 05 COLUMN 47 PIC X(20) TO CPO-01 FULL.
58          02 LINE09 COLUMN 08 VALUE"2. Digite somente numericos::::::
59 -        ":::::".
60          02 FF-CPO-02 LINE 09 COLUMN 47 PIC S9(05) TO CPO-02
61          REQUIRED.
62          02 LINE 13 COLUMN 08 VALUE "3. Digite numerico decimal:::::::
63 -        ":::::".
64          02 FF-CPO-03 LINE 13 COLUMN 47 PIC S9(05)V99 TO CPO-03 REVERS
65- E-VIDEO.
66          02 LINE 22 COLUMN 11 VALUE "Mensagem:" BLINK.
67    01    LIMPA-DADOS.
68          02 LD1 LINE 05 COLUMN 47 BLANK LINE.
69          02 LD2 LINE 09 COLUMN 47 BLANK LINE.
70          02 LD3 LINE 13 COLUMN 47 BLANK LINE.
71          02 LIMPA-MENSA.
72          03 LD4 LINE 20 COLUMN 21 BLANK LINE.
73          03 LD5 LINE 22 COLUMN 21 BLANK LINE.
74          03 LD6 LINE 24 COLUMN 21 BLANK LINE.
75    01    MENSAGENS.
76          02 MSG1 LINE 20 COLUMN 21 VALUE "VIDEO NORMAL".
77          02 MSG2 LINE 20 COLUMN 21 VALUE "Opcao PROMPT do ACCEPT".
78          02 MSG3 LINE 20 COLUMN 21 VALUE "Opcao UPDATE do ACCEPT".
79          02 MSG4 LINE 20 COLUMN 21 VALUE "Opcao AUTO-SKIP do ACCEPT".
80          02 MSG5 LINE 20 COLUMN 21 VALUE "Opcao LENGTH-CHECK do ACCEPT
81 -        " ".
82          02 MSG6 LINE 22 COLUMN 21 VALUE "Opcao TRAILING-SIGN do ACCEP
83 -        "T".
84          02 MSG7 LINE 20 COLUMN 21 VALUE "Opcao RIGHT-JUSTIFY do ACCEP
85 -        "T".
86          02 MSG8 LINE 22 COLUMN 21 VALUE "Opcao LEFT-JUSTIFY do ACCEPT
```

```
87 -        " ".
88          02 MSG9 LINE 24 COLUMN 21 VALUE "Opcao EMPTY-CHECK do ACCEPT
89 -        " ".
90          02 MSGA LINE 20 COLUMN 21 VALUE "opcoes BELL, UNDERLINE e SEC
91 -        "URE da SCREEN SECTION".
92          02 MSGB LINE 20 COLUMN 21 VALUE "Opcoes FULL, REQUIRED e REVE
93 -        "RSE-VIDEO da SCREEN SECTION".
94          02 MSGC LINE 22 COLUMN 21 VALUE "Teste de ESCAPE KEY do ACCEP
95 -        "T".
96          02 MSGD LINE 24 COLUMN 21 VALUE "Tente apertar as teclas ESC,
97 -        " F1, F2, F3 ou ENTER".
98          02 MSGE LINE 22 COLUMN 21 VALUE "Teste de ON ESCAPE do ACCEPT
99 -        " ".
100         02 MSGF LINE 24 COLUMN 21 VALUE "Digite sinal negativo"
101         BLINK.
102
103 01      TELA-CONT.
104         02 LINE 24 COLUMN 21 VALUE "Aperte qualquer tecla para c
105 -           "ontinuar ".
106
107         02 COLUMN PLUS 3 PIC X TO QQ-TECLA AUTO.
108
109 01      FIM-PGM.
110         02 LINE 23 COLUMN 30 VALUE "Termino da Demonstracao" BLINK.
111
112 PROCEDURE DIVISION.
113 INICIO.
114         DISPLAY LIMPA-TELA.
115         DISPLAY TELA-DEMO.
116         DISPLAY MSG1.
117         ACCEPT (05 47) CPO-01.
118         ACCEPT (09 47) CPO-02.
119         ACCEPT (13 47) CPO-03.
120         DISPLAY LD6.
121         DISPLAY TELA-CONT.
122         ACCEPT TELA-CONT.
123 OPCAO-01.
124         DISPLAY LIMPA-DADOS.
125         DISPLAY MSG2.
126         ACCEPT (05 47) CPO-01 WITH PROMPT.
127         ACCEPT (09 47) CPO-02 WITH PROMPT.
128         ACCEPT (13 47) CPO-03 WITH PROMPT.
129         DISPLAY LD6.
130         DISPLAY TELA-CONT.
131         ACCEPT TELA-CONT.
```

```
132 OPCAO-02.
133         DISPLAY LIMPA-DADOS.
134         DISPLAY MSG3.
135         ACCEPT (05 47) CPO-01 WITH UPDATE.
136         ACCEPT (09 47) CPO-02 WITH UPDATE.
137         ACCEPT (13 47) CPO-03 WITH UPDATE.
138         DISPLAY LD6.
139         DISPLAY TELA-CONT.
140         ACCEPT TELA-CONT.
141 OPCAO-03.
142         DISPLAY LIMPA-DADOS.
143         DISPLAY MSG4.
144         ACCEPT (05 47) CPO-01 WITH AUTO-SKIP.
145         ACCEPT (09 47) CPO-02 WITH AUTO-SKIP.
146         ACCEPT (13 47) CPO-03 WITH AUTO-SKIP.
147         DISPLAY LD6.
148         DISPLAY TELA-CONT.
149         ACCEPT TELA-CONT.
150 OPCAO-04.
151         DISPLAY LIMPA-DADOS.
152         DISPLAY MSG5.
153         DISPLAY MSG6.
154         ACCEPT (05 47) CPO-01 WITH LENGTH-CHECK.
155         DISPLAY MSGF.
156         ACCEPT (09 47) CPO-02 WITH TRAILING-SIGN.
157         ACCEPT (13 47) CPO-03.
158         DISPLAY LD6.
159         DISPLAY TELA-CONT.
160         ACCEPT TELA-CONT.
161 OPCAO-05.
162         DISPLAY LIMPA-DADOS.
163         DISPLAY MSG7.
164         DISPLAY MSG8.
165         DISPLAY MSG9.
166         ACCEPT (05 47) CPO-01 WITH RIGHT-JUSTIFY.
167         ACCEPT (09 47) CPO-02 WITH LEFT-JUSTIFY.
168         ACCEPT (13 47) CPO-03 WITH EMPTY-CHECK.
169         DISPLAY LD6.
170         DISPLAY TELA-CONT.
171         ACCEPT TELA-CONT.
172 OPCAO-06.
173         DISPLAY LIMPA-TELA.
174         DISPLAY TELA-DEMO1.
175         DISPLAY MSGA. DISPLAY MSGC. DISPLAY MSGD.
176 OPCAO-06-A.
```

```
177         ACCEPT SS-CPO-01.
178  OPCAO-06-B.
179         ACCEPT SS-CPO-02.
180  OPCAO-06-C.
181         ACCEPT SS-CPO-03.
182         ACCEPT (24 59) QQ-TECLA WITH NO-ECHO.
183         ACCEPT ESC-SEL FROM ESCAPE KEY.
184         IF ESC-SEL = 01 GO TO OPCAO-06.
185         IF ESC-SEL = 02 GO TO OPCAO-06-A.
186         IF ESC-SEL = 03 GO TO OPCAO-06-B.
187         IF ESC-SEL = 04 GO TO OPCAO-06-C.
188         DISPLAY LD6.
189         DISPLAY TELA-CONT.
190         ACCEPT TELA-CONT.
191  OPCAO-07.
192         DISPLAY LIMPA-TELA.
193         DISPLAY TELA-DEMO2.
194         DISPLAY MSGB. DISPLAY MSGE. DISPLAY MSGD.
195         ACCEPT TELA-DEMO2 ON ESCAPE GO TO OPCAO-07.
196         DISPLAY LD6.
197         DISPLAY TELA-CONT.
198         ACCEPT TELA-CONT.
199  ROT-FIM.
200         DISPLAY LIMPA-TELA.
201         DISPLAY FIM-PGM.
202         STOP RUN.
```

## Explanação e Instruções Introduzidas no Programa

```
02 LINE 01 COLUMN 25 VALUE
"DEMONSTRACAO DO COBOL INTERATIVO" REVERSE-VIDEO.
```

Ação: a opção REVERSE-VIDEO apresenta o literal de modo invertido na linha e na coluna especificadas.

```
02 LINE 03 COLUMN 16 VALUE "TESTE DE ATRIBUTOS DA
- SCREEN, DISPLAY E ACCEPT" HIGHLIGHT.
```

Ação: a opção HIGHLIGHT apresenta o literal de modo mais intenso na linha e na coluna especificadas.

```
02 SS-CPO-01 LINE 05 COLUMN 47 PIC X(20) TO CPO-01 BELL
```

Ação: a opção BELL aciona o alarme sonoro antes do início da digitação do campo.

```
02 SS-CPO-02 LINE 09 COLUMN 47 PIC S9(05) TO CPO-02 UNDERLINE.
```

Ação: a opção UNDERLINE apresenta o campo de entrada de modo sublinhado.

```
02 SS-CPO-03 LINE 13 COLUMN 47 PIC S9(05)V99 TO CPO-03 SECURE.
```

Ação: a opção apresenta o campo de entrada com asteriscos mesmo durante a execução da instrução ACCEPT.

```
02 FF-CPO-01 LINE 05 COLUMN 47 PIC X(20) TO CPO-01 FULL.
```

Ação: a opção FULL fica aguardando até que todos os caracteres do campo de entrada sejam digitados.

```
02 FF-CPO-02 LINE 09 COLUMN 47 PIC S9(05) TO CPO-02 REQUIRED.
```

Ação: a opção REQUIRED fica aguardando até que pelo menos um caractere do campo de entrada seja digitado.

```
02 LINE 22 COLUMN 11 VALUE "Mensagem:" BLINK.
```

Ação: a opção BLINK apresenta o rótulo da tela de modo piscante.

```
02 LD1 LINE 05 COLUMN 47 BLANK LINE.
02 LD2 LINE 09 COLUMN 47 BLANK LINE.
02 LD3 LINE 13 COLUMN 47 BLANK LINE.
```

Ação: a opção BLANK LINE apaga a tela desde a linha e a coluna especificada até o final da linha.

```
02 COLUMN PLUS 3 PIC X TO QQ-TECLA AUTO.
```

Ação: a opção COLUMN PLUS 3 faz com que seja adicionado mais 3 na coluna anteriormente especificada.

a opção AUTO indica que ao prencher o campo QQ-TECLA o cursor salta automaticamente para o próximo campo.

a opção TO recebe o item via teclado e o envia para o campo QQ-TECLA.

```
ACCEPT (05 47) CPO-01 WITH PROMPT.
```

Ação: a opção WITH PROMPT provoca o preenchimento do campo de entrada com pontos (...), por se tratar de campo alfanumérico.

```
ACCEPT (09 47) CPO-02 WITH PROMPT.
```

Ação: a opção WITH PROMPT provoca o preenchimento do campo de entrada com (_ _ _), por se tratar de campo numérico.

```
ACCEPT (05 47) CPO-01 WITH UPDATE.
```

Ação: a opção WITH UPDATE provoca o preenchimento do campo de entrada de dados com caracteres contidos no campo CPO-01.

```
ACCEPT (05 47) CPO-01 WITH AUTO-SKIP.
```

Ação: a opção WITH AUTO-SKIP torna desnecessário o acionamento da tecla <ENTER>, se todos os caracteres do campo de entrada forem preenchidos.

```
ACCEPT (05 47) CPO-01 WITH LENGTH-CHECK.
```

Ação: a opção WITH LENGTH-CHECK fica aguardando até que todos os caracteres do campo de entrada sejam preenchidos.

```
ACCEPT (09 47) CPO-02 WITH TRAILING-SIGN.
```

Ação: a opção WITH TRAILING-SIGN faz com que o sinal operacional apareça na posição mais à direita do campo de entrada.

```
ACCEPT (05 47) CPO-01 WITH RIGHT-JUSTIFY.
```

Ação: a opção WITH RIGHT-JUSTIFY realiza o alinhamento dos dados do campo de entrada pela direita.

```
ACCEPT (13 47) CPO-03 WITH EMPTY-CHECK.
```

Ação: a opção WITH EMPTY-CHECK fica aguardando até que pelo menos um caractere seja digitado no campo de entrada.

```
ACCEPT (24 59) QQ-TECLA WITH NO-ECHO.
```

Ação: a opção WITH NO-ECHO preenche o campo de entrada com asterisco, servindo para mascarar o caractere digitado.

```
ACCEPT ESC-SEL FROM ESCAPE KEY.
```

Ação: um código de dois dígitos é gerado pela tecla que terminou a execução (ENTER = 00, ESC = 01, F1 = 02, F2 = 03 e F3 = 04). Logo abaixo desta instrução existem testes que determinam qual tecla foi acionada.

```
ACCEPT TELA-DEMO2 ON ESCAPE GO TO OPCAO-07.
```

Ação: se a tecla ESC for pressionada durante a inserção de dados, todo o ACCEPT será terminado sem mover os campos correspondentes aos itens associados especificados pela opção TO na SCREEN SECTION, o valor do ESCAPE KEY é ajustado em 01 e o comando GO TO OPCAO-7 é executado. Se a tecla de função (F1 a F10) for acionada, o valor ESCAPE KEY apropriado é ajustado e o conteúdo dos campos são movidos para o item associado à opção TO. Se a tecla <ENTER> for pressionada, o valor ESCAPE KEY é ajustado em 00 e o cursor passa para o próximo campo de entrada, se existir.

# COMANDOS ARITMÉTICOS

A linguagem COBOL trata cinco comandos aritméticos:

ADD

SUBTRACT

MULTIPLY

DIVIDE

COMPUTE

Todo comando é imperativo ou condicional. Quando um comando aritmético inclui a cláusula ON SIZE ERROR ele é considerado condicional, já que o resultado do ON SIZE ERROR depende de dados.

## Regras Básicas Acerca dos Comandos Aritméticos

1. Todos os nomes-de-dados envolvidos em comandos aritméticos devem ser de itens elementares numéricos definidos na DATA DIVISION.

2. Uma exceção é aberta para os operandos do GIVING que podem ser itens-numéricos-editados.

3. O alinhamento do ponto decimal é feito automaticamente durante as operações.

4. Os campos usados para resultados intermediários garantem a precisão do resultado final, a não ser que haja necessidade de truncamento dos dígitos de mais alta ordem.

**Diretivas de Complementação de Comandos Aritméticos**

Três opções adicionais podem aparecer nos comandos aritméticos: SIZE ERROR, ROUNDED e GIVING.

SIZE ERROR. Se após o alinhamento do ponto decimal e arredondamento do algarismo menos significativo, o valor absoluto do resultado da operação ultrapassar o valor máximo que pode ser colocado na área reservada para o resultado, surgirá a condição de transbordamento. Esta condição poderá surgir no resultado final de uma operação aritmética e não se aplica a resultados intermediários.

Seu formato é: ON **SIZE ERROR** comando-imperativo...

Se a opção estiver presente, e uma condição de erro de tamanho ocorrer, o resultado não será alterado e a seqüência de comandos imperativos especificada para a condição será executada.

Se a opção não estiver presente e a condição de erro de tamanho ocorrer o resultado final será imprevisível.

ROUNDED. Se após o alinhamento do ponto-decimal o resultado contiver mais dígitos que o campo que irá recebê-lo haverá truncamento, a menos que a opção **ROUNDED** tenha sido especificada. Quando a opção **ROUNDED** estiver presente, o dígito menos significativo do nome-de-dado resultante terá seu valor incrementado de 1 sempre que o dígito mais significativo do excesso for maior ou igual a 5.

O arredondamento do resultado negativo é feito primeiro sobre o valor absoluto e posteriormente o resultado é tornado negativo.

**TABELA ILUSTRATIVA**

| | ITEM QUE RECEBE O VALOR CALCULADO | | |
|---|---|---|---|
| VALOR CALCULADO | PIC | VALOR APOS ROUNDED | VALOR SEM ROUNDED |
| -12.36 | S99V9 | -12.4 | -12.3 |
| 8.432 | 9V9 | 8.4 | 8.4 |
| 35.6 | 99V9 | 35.6 | 35.6 |
| 65.6 | S99V | 66 | 65 |
| .0055 | SV9(3) | .006 | .005 |

GIVING. Se num comando aritmético esta opção estiver presente, então o nome-de-dado que segue GIVING receberá o resultado da operação aritmética.

O seu formato é: **GIVING** nome-de-dado-1

Como nome-de-dado-1 não é envolvido nos cálculos ele pode ser um item-de-edição, exceto no comando MULTIPLY.

## Instrução ADD

Função: o comando ADD adiciona dois ou mais valores numéricos e armazena a soma resultante.

O seu formato genérico é:

**ADD** {literal-numérico/nome-de-dado-1} ... {**TO/GIVING**}

nome-de-dado-n [**ROUNDED**] [ON **SIZE ERROR** instrução-incondicional ...]

O comando ADD tem a função de somar dois ou mais valores numéricos e armazenar a soma resultante.

Quando a opção TO é usada, os valores de todos os nomes-de-dados, incluindo nome-de-dado-n, são somados e o resultado é armazenado em nome-de-dado-n.

Quando a opção **GIVING** é usada pelo menos dois nomes-de-dados e/ou literais-numéricos devem seguir a palavra ADD.

Exemplos:

```
ADD NOTA-A NOTA-B TO NOTA-FINAL
```

Ação: NOTA-FINAL = NOTA-A + NOTA-B + NOTA-FINAL

```
ADD NOTA-A TO NOTA-FINAL
```

Ação: NOTA-FINAL = NOTA-A + NOTA-FINAL

```
ADD NOTA-A NOTA-B GIVING NOTA-FINAL
```

Ação: NOTA-FINAL = NOTA-A + NOTA-B

Campo receptor: 77 CONTA-REGISTROS PIC 9(02) VALUE ZEROS.

Comando ADD:

```
ADD 1 TO CONTA-REGISTROS ON SIZE ERROR
        DISPLAY (01, 10) "ULTRAPASSOU LIMITE CONTADOR"
        STOP RUN.
```

Ação: No caso de o conteúdo do campo CONTA-REGISTROS ultrapassar o limite máximo de 99 o comando DISPLAY e STOP RUN serão executados.

Para que isso não ocorra e, também para evitar o uso desta opção, descreva os campos receptores (de destino) com extensão suficiente.

## Instrução SUBTRACT

Função: o comando SUBTRACT subtrai um ou mais itens de dados numéricos de um item especificado e armazena a diferença.

O seu formato genérico é:

**SUBTRACT** {nome-de-dado-1/literal-numérico} **FROM**

nome-de-dado-n [**GIVING** nome-de-dado-n]

literal-numérico-m **GIVING** nome-de-dado-n

[**ROUNDED**] [ON **SIZE ERROR** instrução-incondicional].

No comando SUBTRACT os itens-de-dados que precedem FROM são somados e esta soma é subtraída do item-de-dado que sucede FROM.

O resultado será armazenado em nome-de-dado-n se houver GIVING e, em nome-de-dado-m, se não houver.

Todos os operandos têm de ser itens numéricos exceto nome-de-dado-n que pode ser item-de-edição.

Exemplos:

```
SUBTRACT A B FROM C
```

Ação: C = C - (A + B)

```
SUBTRACT A B 2 FROM C.
```

Ação: C = C - (A + B + 2)

```
SUBTRACT A B 2 FROM 200 GIVING C.
```

Ação: C = 200 - (A + B + 2)

## Instrução MULTIPLY

Função: o Comando MULTIPLY multiplica dois itens de dados numéricos e armazena o produto.

O seu formato genérico é:

**MULTIPLY** {nome-de-dado-1/constante-numérica-1} **BY**

nome-de-dado-2 [**GIVING** nome-de-dado-3]

constante-numérica-2 **GIVING** nome-de-dado-3

[**ROUNDED**] [ON **SIZE ERROR** instrução-incondicional...]

Calcula o produto de dois itens-de-dados numéricos e armazena o resultado.

Quando a opção GIVING é usada, o produto vai para nome-de-dado-3 e quando não, o produto vai para nome-de-dado-2.

Exemplos:

MULTIPLY A BY B

Ação: B = A . B

MULTIPLY A BY 5 GIVING C

Ação: C = A . 5

## Instrução DIVIDE

Função: o comando DIVIDE divide dois valores numéricos e armazena o quociente.

O seu formato genérico é:

**DIVIDE** {nome-de-dado-1/constante-numérica-1} **{BY/INTO}**

{nome-de-dado-2/constante-numérica-2} **GIVING** nome-de-dado-3

**[ROUNDED]** [ON **SIZE ERROR** instrução-incondicional...]

A instrução DIVIDE divide um item numérico por outro armazenando o resultado no item especificado.

A forma BY diz que o primeiro operando (nome-de-dado-1 ou literal-numérico-1) é o dividendo, e que o segundo operando nome-de-dado-2 ou o literal numérico é o divisor.

Para a forma INTO, vale o contrário. Se a opção GIVING não estiver presente, o operando que representar o dividendo deve ser um nome-de-dado que armazenará o quociente. A divisão por zero sempre cria uma condição de erro de tamanho.

Exemplos:

```
DIVIDE A BY B
```

Ação: A = A : B

```
DIVIDE A BY B GIVING C
```

Ação: C = A : B

```
DIVIDE A INTO B
```

Ação: B = B : A

```
DIVIDE A INTO B GIVING C
```

Ação: C = B : A

## Instrução COMPUTE

Função: calcula o valor de uma expressão aritmética e o armazena num item-numérico ou de edição especificado.

O seu formato genérico é:

**COMPUTE** nome-de-dado-1 **[ROUNDED**...] {nome-de-dado-2/ constante-numérica/ expressão-aritmética} [ON **SIZE ERROR** comando-imperativo]

Uma expressão aritmética é uma combinação de literais- numéricos, nomes-de-dados, operadores aritméticos e parênteses de tal forma que o conjunto tenha um sentido matemático. Em geral nomes-de-dados numa expressão aritmética designam itens-numéricos. Nome-de-dados consecutivos (ou literais) devem estar separados por um operador aritmético. De cada lado de um operador aritmético deve haver um ou mais espaços. Os operadores são:

\+ para adição

\- para subtração

* para multiplicação

/ para divisão

** para exponenciação

Quando se usam parênteses valem as seguintes regras de pontuação:

a) um abre parênteses é precedido por um ou mais espaços;

b) um fecha parênteses é seguido por um ou mais espaços.

Exemplos:

```
COMPUTE F = A + B / (C - D * E)
```

| | | |
|---|---|---|
| Ação: | F = A + B / (C - R1) | onde R1 = D * E |
| | F = A + B / R2 | onde R2 = C - R1 |
| | F = A + R3 | onde R3 = B : R2 |
| | F = R4 | onde R4 = A + R3 |

```
COMPUTE X = -Y
```

## INSTRUÇÃO STOP RUN

A instrução STOP RUN encerra o programa, fecha todos os arquivos porventura existentes e passa o controle de processamento para o Sistema Operacional.

O seu formato genérico é:

```
STOP {RUN/constante}
```

Tem a função específica para permitir o término de execução de um programa objeto, total ou temporariamente. Quando for especificada uma constante (literal) será suspensa temporariamente a execução do programa após apresentar o literal no vídeo. A execução será retomada quando o operador pressionar a tecla RETURN ou ENTER.

## EXERCÍCIOS

Exercício 1

### 1. Objetivo Do Programa

Calcular a área de uma circunferência.

### 2. Procedimentos

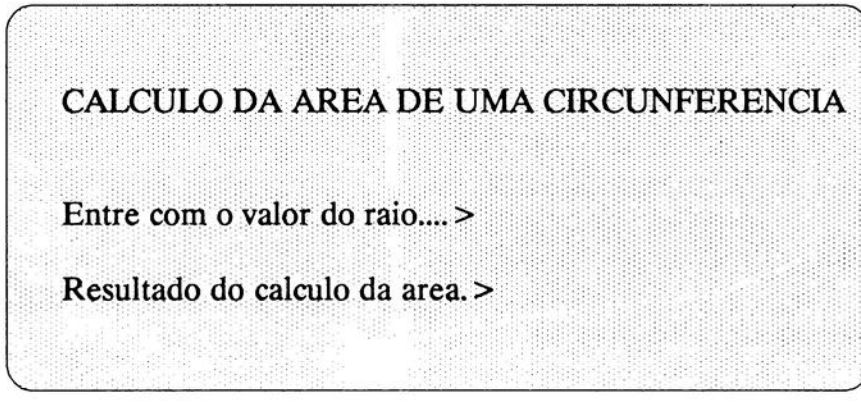
1. exibir a tela formatada;

2. aceitar o valor do raio;

3. exibir a área calculada;

4. exibir mensagem "FIM DE PROGRAMA".

## 3. Formato da Tela (Video Layout)




Exercício 2

## 1. Objetivo do Programa

Calcular a média aritmética das notas de um aluno.

## 2. Procedimentos

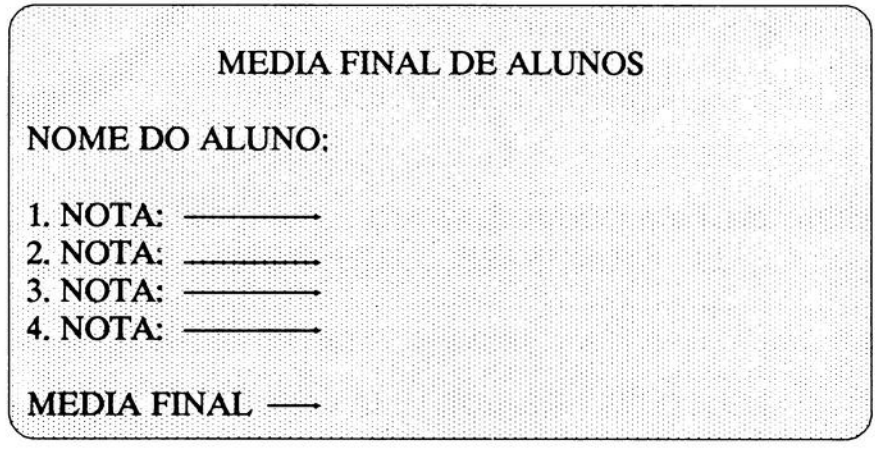
1. exibir a tela formatada conforme layout;

2. aceitar o nome do aluno e as quatro notas;

3. exibir a média calculada;

4. exibir mensagem "FIM DE PROGRAMA".

## 3. Formato da Tela (Video Layout)




Exercício 3

## 1. Objetivo do Programa

Elaborar um programa de tela que faça o controle de custo de uma determinada mercadoria de uma certa distribuidora.

## 2. Procedimentos

1. exibir a tela conforme layout;

2. entrar com o nome do produto;

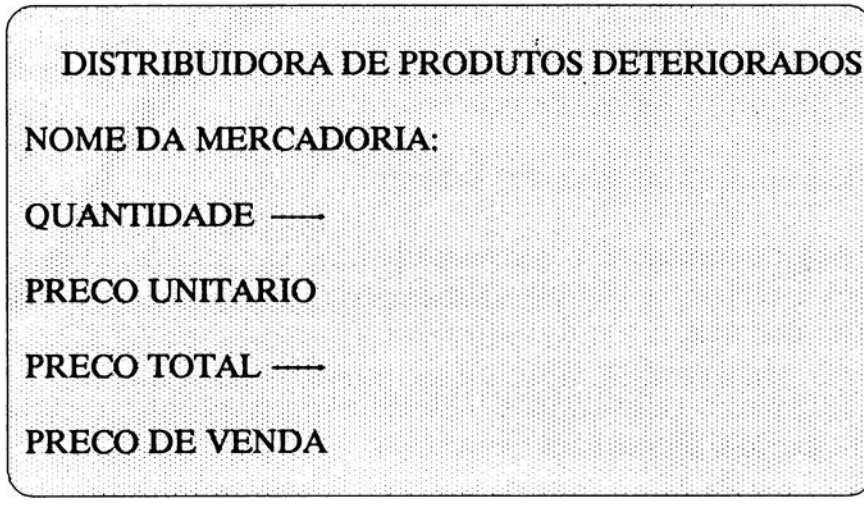
3. entrar com a quantidade e o preço da mercadoria;

4. calcular o valor total da mercadoria;

5. calcular o preço de venda, acrescendo 30% no valor total;

6. condicionar fim de programa.

## 3. Formato da Tela (Video Layout)




## 4. Tamanho dos Campos

| | |
|---|---|
| MERCADORIA | PICX(20) |
| QUANTIDADE | PIC 9(04) |
| PRECO UNITARIO | PIC 9(04)V99 |
| PRECO TOTAL | PIC 9(05)V99 |
| PRECO DE VENDA | PIC 9(06)V99 |

## COMPLEMENTO DA CLÁUSULA PICTURE (PIC)

Vamos tratar somente dos caracteres numéricos, pois os alfanuméricos já foram tratados anteriormente.

O item numérico consiste numa combinação dos seguintes caracteres: 9 V S P.

*a.* O caractere 9 indica que esta posição contém um valor numérico. O número máximo de caracteres permitidos na picture é de 18.

Exemplo válido: 03 CAMPO-A PIC 9(10).
Exemplo inválido: 03 CAMPO-B PIC 9(20).

*b.* O caractere V indica a posição em que é assumido o ponto decimal e não ocupa posição de memória.

Exemplo: 03 CRUZADO PIC 9(05)V99.

Ação: O caractere V não ocupa posição na memória. O campo é de 7 posições e indicamos que ele deve possuir 5 posições para os inteiros e duas partes decimais.

Se for um valor 1234567 na memória, será interpretado como 12345.67 quando o programa for executado.

*c.* O caractere S indica que o campo possui um sinal. Quando o campo for definido sem o caractere S será considerado positivo. Este caractere é importante e essencial quando se deseja que o campo contenha um sinal.

Exemplo:
```
77 AREA-AUX PIC S9(05) VALUE -10000.
77 AUX-AREA PIC S9(05) VALUE ZEROS.
ADD AREA-AUX -10000 GIVING AUX-AREA.
```

Ação: Após a execução do comando o campo AUX-AREA conterá -20000.

*d.* O caractere P indica a posição fictícia do ponto decimal e não ocupa posição de memória, ou seja, é tratado como se contivesse zeros para cada P indicado.

Exemplo: Suponha que queiramos que o computador interprete 25 como 25000; a cláusula PICTURE deverá ser a seguinte:

```
77 WS-AUX1 PIC 9(02)PPP.
MOVE 25 TO WS-AUX1.
```

Ação: Resultará no campo WS-AUX1 o valor 25000 após a execução da instrução MOVE.

# FUNÇÃO DOS CARACTERES DE EDIÇÃO

O item numérico pode, ainda, ser utilizado para representar os caracteres de edição. Esses tipos de caracteres de edição são utilizados para imprimir relatórios e exibir pequena quantidade de dados no vídeo.

São funções de edição:

*a.* eliminação dos zeros à esquerda;

*b.* impressão do ponto decimal onde o alinhamento decimal está implícito;

*c.* impressão do sinal de $ e a vírgula;

*d.* impressão de asteriscos para proteger a verificação;

*e.* impressão de sinais de + e de - para indicar o valor do campo;

*f.* impressão de símbolos de crédito e de débito para aplicações de contas;

*g.* impressão de espaços, zeros e barras como separadores de campos.

São caracteres de edição: 9 V . CR DB , $ + * B 0 - P /.

Significado de cada um desses caracteres:

indica que um ponto decimal deve ser de fato inserido com conseqüente alinhamento dos campos. Todos os caracteres PICTURE que seguirem o (.) devem ser do mesmo tipo.

Z e * são caracteres de substituição. Cada um representa a posição de um dígito. Os zeros à esquerda, cujas posições correspondem a Z ou *, são substituídos por branco ou *. A supressão de zeros termina quando for encontrado o primeiro caractere diferente de zero ou o ponto decimal. Z ou * só podem aparecer à direita do ponto decimal real se todos os caracteres da PICTURE forem Z ou * (todos iguais).

CR e DB são chamados símbolos de crédito e débito e só podem aparecer à direita numa PICTURE. Estes caracteres ocupam 2 posições e informam que o símbolo especificado deve aparecer nas posições indicadas se o valor do item a editar for negativo. Se o valor for positivo ou zero, tais posições ficarão em branco. CR e DB e + e - são mutuamente exclusivos.

indica que uma vírgula deve ser inserida entre dígitos. Toda vírgula da PICTURE é contada no tamanho do campo impresso, mas não representa nenhum dígito do campo fonte (campo interno).

A vírgula pode também fazer parte de uma cadeia flutuante conforme a discussão que se segue. Uma cadeia flutuante é uma seqüência contínua de caracteres que possua apenas $ ou + ou - ou uma cadeia que contenha várias ocorrências de um só destes caracteres além das vírgulás e/ou ponto decimal.

Exemplos de cadeias flutuantes:

```
$$,$$$,$$
+ + + + +
--,---,---
```

| | |
|---|---|
| + - | os caracteres + e - podem aparecer sozinhos numa PICTURE ou numa cadeia flutuante. Como caractere de controle de sinal fixo + ou - tem que aparecer como último caractere da PICTURE. O caractere + indica que se o campo a editar for positivo ou zero aparecerá o + no relatório e se for negativo aparecerá o -. |
| | O - na PICTURE indica que branco ou - será editado, dependendo do campo fonte ser positivo ou negativo respectivamente. |
| B | cada ocorrência do B numa PICTURE representa um branco no valor final editado. |
| / | cada barra na PICTURE indica a presença de uma barra no item final editado. |
| 0 | cada ocorrência de zero numa PICTURE acarreta um zero no valor final editado. |

Os caracteres S V CR e DB devem ser definidos somente uma vez dentro de uma PICTURE.

Seguem alguns exemplos válidos de uso de PICTURE para editar dados numéricos.

## Tabela de Exemplos Válidos

| Campo Transmissor | | Campo Receptor | |
|---|---|---|---|
| PICTURE | VALOR DO DADO | PICTURE | VALOR EDITADO |
| 999V99 | 12345 | 999.99 | 123.45 |
| 99V99 | 0012 | 99.99 | 00.12 |
| 9(5) | 12345 | 9(5).9 | 12345.0 |
| 9(6)V99 | 00123456 | 999.999,99 | 001.234,56 |
| 999V99 | 01234 | ZZ9.99 | 12.34 |
| S999V99 | 02345 | ZZZ.ZZ | 23.45 |
| S9(3)V99 | 00345 | ***.99 | **3.45 |
| S9(5) | 00345 | *****.99 | **345.00 |
| S9(5) | 02345 | ZZZ9.99CR | 2345.00CR |
| S9(5) | -02345 | ZZZ9.99CR | 2345.00 |
| S9(5)V99 | -02345 | ZZ9.99DB | 23,45DB |
| S9(5)V99 | 02345 | ZZ9.99DB | 23,45 |
| 9(4) | 1234 | $$$$$99 | $1234 |
| 9(4) | -1234 | -999 | -1234 |
| 99V99 | 1234 | + +9.99 | + 12,34 |

# CAPÍTULO 2

## PROGRAMA DE TELA

Este programa utiliza a data do sistema, nome-de-condição (nível 88) e comando de decisão.

## 1. Objetivo do Programa

Elaborar um programa utilizando os recursos da SCREEN SECTION para a tela e consistir os dados digitados.

## 2. Procedimentos

1. exibir a tela formatada;

2. aceitar a data do sistema;

3. exibir a data no canto superior esquerdo do vídeo;

4. aceitar o nome que deve ser diferente de brancos;

5. aceitar a idade que deve ser > 14 e < 30;

6. aceitar o sexo que deve ser "F" ou "M";

7. aceitar o salário que deve ser > 4999 e < 50001;

8. calcular o salário novo aplicando mais 25% no salário digitado; exibir no formato numérico editado;

9. condicionar fim de programa;

10. em toda consistência não aceita deve ser exibida mensagem antes de retornar ao item.

## 3. Formato da Tela (Video Layout)

```
EM 99/99/99          CONSISTENCIA DE DADOS

AUTOR. _____ NR __ SERIE __ TURMA __

NOME:

IDADE:

SEXO:

SALARIO:

SALARIO ATUAL:

          CONTINUA (s/n).....( )

MENSAGEM:
```

## Codificação do Programa

*Observação*: Todos os parágrafos considerados opcionais não estão descritos nesta codificação.

```
1 *----------> Inicio fisico do programa
2    IDENTIFICATION DIVISION.
3    PROGRAM-ID. DADOS.
4    AUTHOR. COLEGIO BRASIL.
5    ENVIRONMENT DIVISION.
6    CONFIGURATION SECTION.
7    SPECIAL-NAMES. DECIMAL-POINT IS COMMA.
8    DATA DIVISION.
9    WORKING-STORAGE SECTION.
10   01 AREAS-DE-TRABALHO.
11        02 WS-NOME PIC X(30) VALUE SPACES.
12        02 WS-IDADE PIC 9(02) VALUE ZEROS.
13             88 IDADE VALUE 15 THRU 29.
14        02 WS-SEXO PIC X VALUE SPACE.
15             88 FM VALUE "F" "M" "f" "m".
16        02 WS-SALARIO PIC 9(05)V99 VALUE ZEROS.
17        02 WS-SAL-ATUAL PIC ZZ.ZZ9,99 VALUE ZEROS.
18        02 WS-CONT PIC X VALUE SPACE.
19   01 MENSAGENS-DE-CRITICA.
20        02 MENSA1PIC X(30) VALUE
21             "NOME INVALIDO <REDIGITE>".
22        02 MENSA2 PIC X(30) VALUE
23             "IDADE INVALIDA <REDIGITE>".
24        02 MENSA3 PIC X(30) VALUE
25             "SEXO INVALIDO <REDIGITE>".
26
27        02 MENSA4 PIC X(30) VALUE
28             "SALARIO INVALIDO <REDIGITE>".
29        02 MENSA5 PIC X(30) VALUE SPACES.
30        02 MENSA6 PIC X(30) VALUE
31             "FIM DE PROGRAMA".
32        02 MENSA7 PIC X(30) VALUE
33             "OPCAO INVALIDA <REDIGITE>".
34   01 DATA-DO-SISTEMA.
35        02 ANO PIC 9(02) VALUE ZEROS.
36        02 MES PIC 9(02) VALUE ZEROS.
37        02 DIA PIC 9(02) VALUE ZEROS.
38   SCREEN SECTION.
39   01 TELA.
```

```
40          02 BLANK SCREEN.
41          02 LINE 02 COLUMN 02 VALUE "EM".
42          02 LINE 02 COLUMN 28 VALUE
43              "C O N S I S T E N C I A D E D A D O S".
44          02 LINE 04 COLUMN 25 VALUE
45          "AUTOR: COLEGIO BRASIL - N. 000 - SERIE: 2. - TU: X".
46          02 LINE 08 COLUMN 21 VALUE "NOME    ".
47          02 LINE 10 COLUMN 21 VALUE "IDADE--  ".
48          02 LINE 12 COLUMN 21 VALUE "SEXO     ".
49          02 LINE 14 COLUMN 21 VALUE "SALARIO ".
50          02 LINE 16 COLUMN 21 VALUE "SALARIO ATUAL   ".
51          02 LINE 19 COLUMN 21 VALUE "CONTINUA (S/N) < >".
52          02 LINE 23 COLUMN 21 VALUE "MENSAGEM:".
53      PROCEDURE DIVISION.
54      ROT-INICIO.
55          DISPLAY TELA.
56          ACCEPT DATA-DO-SISTEMA FROM DATE.
57          DISPLAY (02 05) DIA "/" MES "/" ANO.
58      ROT-NOME.
59          ACCEPT (08 39) WS-NOME WITH PROMPT.
60          DISPLAY (23 31) MENSA5.
61          IF WS-NOME = SPACES
62              DISPLAY (23 31) MENSA1 GO TO ROT-NOME.
63      ROT-IDADE.
64          ACCEPT (10 39) WS-IDADE WITH PROMPT.
65          DISPLAY (23 31) MENSA5.
66          IF IDADE NEXT SENTENCE
67          ELSE DISPLAY (23 31) MENSA2 GO TO ROT-IDADE.
68      ROT-SEXO.
69          ACCEPT (12 39) WS-SEXO WITH PROMPT.
70          DISPLAY (23 31) MENSA5.
71          IF FM NEXT SENTENCE
72          ELSE DISPLAY (23 31) MENSA3 GO TO ROT-SEXO.
73      ROT-SALARIO.
74          ACCEPT (14 39) WS-SALARIO WITH PROMPT.
75          DISPLAY (23 31) MENSA5.
76          IF WS-SALARIO > 04999,00 OR < 50001,00 NEXT SENTENCE
77          ELSE DISPLAY (23 31) MENSA4 GO TO ROT-SALARIO.
78      ROT-CALCULO.
79          COMPUTE WS-SAL-ATUAL = WS-SALARIO * 25 / 100 + WS-SALARIO
80          DISPLAY (16 39) WS-SAL-ATUAL.
81      ROT-CONTINUA.
82          ACCEPT (19 37) WS-CONT WITH PROMPT.
83          DISPLAY (23 31) MENSA5.
84          IF WS-CONT = "S" OR "s" GO TO ROT-INICIO.
```

```
85  IF WS-CONT = "N" OR "n" DISPLAY (23 31) MENSA6 STOP RUN
86  ELSE DISPLAY (23 31) MENSA7 GO TO ROT-CONTINUA.
87
```

## Explanação e Instruções Introduzidas no Programa

Na WORKING-STORAGE SECTION temos 3 níveis, 01 que são denominados itens-de-grupo. Cada um desses itens-de-grupo são constituídos por um ou mais itens-elementares. Convém observar que os itens-de-grupo não devem ter as cláusulas PIC e VALUE.

```
88 IDADE VALUE 15 THRU 29.
88 FM VALUE "F" "M" "f" "m".
```

Os níveis 88 são utilizados para representar um valor específico de um item-elementar que são denominados nomes-de- condição.

```
02 WS-SAL-ATUAL PIC ZZ.ZZ9,99 VALUE ZEROS.
```

Este formato de Picture representa o numérico editado onde os "Z" significam a supressão de ZEROS à esquerda do último significativo e o . e a , significam inserção desses caracteres na representação do campo WS-SAL-ATUAL.

```
01 MENSAGENS-DE-CRITICA.
```

Este item-de-grupo contém todas as mensagens que serão utilizadas na confecção do procedimento e são utilizados com o comando DISPLAY.

O conteúdo dessas mensagens poderia ser descrito diretamente na Procedure Division.

```
01 DATA-DO-SISTEMA.
```

Este item-de-grupo foi descrito para receber a data do sistema operacional. Observar que esta data está no formato ano, mês, dia.

```
Na PROCEDURE DIVISION.

ACCEPT DATA-DO-SISTEMA FROM DATE.
```

O nome-de-dado DATA-DO-SISTEMA irá receber a data atual do sistema.

```
IF WS-NOME = SPACES (condição simples)
```

A instrução IF avalia se a condição especificada é verdadeira ou não. Se verdadeira, serão executadas as duas instruções seguintes. Caso contrário, passa para o parágrafo/instrução seguinte. Observar que só devemos colocar um ponto (.) após a última instrução.

```
GO TO ROT-NOME.
```

A instrução GO TO provoca um desvio para o parágrafo ou rotina especificada que é ROT-NOME.

```
88 IDADE VALUE 15 THRU 29.

IF IDADE NEXT SENTENCE

ELSE DISPLAY (23 31) MENSA2

GO TO ROT-IDADE.
```

Neste conjunto de instruções o nome-de-dado IDADE é o nome-de-condição especificado no nível 88 do nome-de-campo WS-IDADE. Significa que se for digitado um valor entre 15 e 29 a condição será verdadeira, caso contrário será falsa. Esta condição é equivalente a:

```
IF WS-IDADE > 14 OR < 30 .....

IF WS-SALARIO > 04999,00 OR < 50001,00 .....
```

Nestas decisões temos uma condição composta representada pela palavra reservada OR.

## INSTRUÇÃO NOME-DE-CONDIÇÃO - NÍVEL 88

A entrada que descreve um nome-de-condição começa com o número de nível 88 e atribui um valor, uma lista de valores ou uma faixa de valores a esse nome. Se no momento da referência, na PROCEDURE DIVISION, ao nome-de-condição, o mesmo igualar o valor, ou um dos valores da lista ou pertencer à faixa especificada, então a sentença condicional será verdadeira, caso contrário será falsa.

O formato geral é:

```
88 VALUE IS {literal-1 [literal-2...]/literal-1 THRU literal-2}
```

Um nome-de-condição substitui, na PROCEDURE DIVISION, uma condição de relação simples.

Um nível 88 deve ser precedido por outro nível 88 (caso de vários níveis 88 pertencentes a um item elementar) ou por um item elementar. Itens de dados indexados não podem ser seguidos por itens de nível 88.

Cada nome de condição se refere a um item elementar de tal forma que pode ser qualificado pelo nome do item elementar.

Pode pertencer a um item elementar (variável condicional) que requeira subscritos. Neste caso qualquer referência ao nome-de-condição na PROCEDURE DIVISION deverá ter os subscritos pertinentes ao item elementar ao qual ao mesmo está subordinado. O tipo de literal usado deve ser consistente com o definido para a variável condicional.

Exemplo 1:

```
03 FREQUENCIA PIC 9 VALUE ZERO.
88 SEMANALVALUE 1.
88 QUINZENAL VALUE 2.
88 MENSAL VALUE 3.
```

Na PROCEDURE DIVISION, teremos:

```
IF MENSAL GO TO ROT-MES.
```

com o mesmo efeito de:

```
IF FREQUENCIA = 3 GO TO ROT-MES.
```

Exemplo 2:

```
03  TURMA PIC X VALUE SPACE.
    88 TURMA-OK VALUE "A" THRU "D".
```

Na PROCEDURE DIVISION teremos:

```
IF TURMA-OK GO TO ROT-PROCES.
```

com o mesmo efeito de:

```
IF TURMA = "A" OR "B" OR "C" OR "D" GO TO ROT- PROCES.
```

## INSTRUÇÃO IF

O comando IF permite que o programador especifique uma série de comandos para o caso de uma condição ser verdadeira. Opcionalmente, uma série de comandos pode ser especificada no caso de a condição ser falsa.

Seu formato geral é:

```
IF condição {NEXT SENTENCE/comando...} { ELSE [ NEXT
SENTENCE/comando... ] }
```

A primeira série de comandos será executada se a condição for verdadeira e a segunda se a condição for falsa. Um ponto-período terminando a sentença segue a segunda série caso ela exista. Caso não exista, segue a primeira série.

Seja a condição verdadeira ou falsa, a próxima sentença somente será processada após a execução da série de comandos apropriada. Exceções a esta regra são:

. ocorrência de um GO TO como um dos comandos;

. o fluxo normal do programa ser interrompido por causa de um comando PERFORM ativo.

A cláusula NEXT SENTENCE especifica que determinada condição será executada somente no caso contrário à afirmação estabelecida pelo IF e que o mesmo nada faça se a condição for obedecida.

Uma condição pode ser simples ou composta. Uma condição simples, por sua vez, pode ser:

- uma relação condicional;
- um teste de classe;
- um teste de sinal;
- um teste de nome-de-condição

## Relação Condicional

A relação condicional compara o valor de dois operandos.

| | |
|---|---|
| = | igual a |
| NOT = | diferente de |
| < | menor que |
| NOT < | maior ou igual |
| > | maior que |
| NOT > | menor ou igual |

As palavras reservadas **EQUAL TO**, **LESS THAN** e **GREATER THAN** substituindo = < >, nesta ordem, são válidas para o COBOL.

Os tipos de comparações permitidas são:

*a.* **comparações numéricas**

Os operandos são comparados após alinhamento de ponto decimal. Os resultados são conforme definido matematicamente. Qualquer valor negativo é menor do que ZERO, que por sua vez é maior do que qualquer valor positivo. Um nome-de-índice ou item-de-dado de índice (subscritor/indexador) pode aparecer em uma comparação. A comparação de quaisquer dois operandos é permitida, independentemente dos formatos especificados em suas respectivas cláusulas USAGE, e independentemente do comprimento.

*b.* **comparações de caracteres**

Comparações entre itens de comprimentos diferentes são permitidas, sendo assumidos espaços para prolongar o comprimento do item mais curto, caso seja necessário. As relações são as implícitas no código ASCII. As letras A-Z são em seqüência ascendente, e os dígitos são menores que as letras. Itens de grupo são tratados simplesmente como seqüência de caracteres quando comparados.

Exemplos:

```
IF A = B GO TO AAA.

IF A NOT = B GO TO BBB.

IF A > B GO TO CCC.

IF A NOT < B GO TO DDD.

IF A < B GO TO EEE ELSE GO TO FFF.
```

## Teste de Classe

Seu formato é:

nome-de-dado IS **[NOT]** {**NUMERIC/ALPHABETIC**}

A condição examina o conteúdo do nome-de-dado. Se o teste é numérico (NUMERIC) a instrução verifica se cada caractere é um número e para teste alfabético (ALPHABETIC), a instrução verifica se cada caractere é um alfabeto ou espaço.

A opção numérica é válida para itens-de-grupo ou alfanumérico. A opção alfabética é válida apenas para itens-de-grupo ou alfanuméricos.

Exemplos:

```
02 MATRICULA PIC X(10).

IF MATRICULA NOT NUMERIC GO TO ERRO-NUMERO.

02 NOME PIC X(30).

IF NOME IS ALPHABETIC PERFORM ROT-GRAVA.
```

## Teste de Sinal

Formato: {expressão-aritmética/nome-de-dado} IS **[NOT]**

**NEGATIVE/ZERO/POSITIVE**

A condição testa se o dado é maior, igual ou menor que zero.

O teste só é válido para item-numérico.

A opção POSITIVE verifica se o resultado da expressão aritmética ou o conteúdo do nome-de-dado tem valor maior que zero. A opção

NEGATIVE terá valor verdadeiro conforme a expressão aritmética, ou nome-de-dado, seja menor que zero e a opção ZERO será satisfeita quando o valor algébrico, do lado esquerdo do operador, for nulo.

Exemplo:

```
02 A PIC S9(03).

IF A IS POSITIVE

    DISPLAY (10 10) "POSITIVO"

ELSE

    DISPLAY (10 10) "NEGATIVO".

02 NUMERO PIC S9(05).

IF NUMERO +1 IS POSITIVE

    DISPLAY (10 10) "Positivo"

ELSE

    DISPLAY (10 10) "Negativo".
```

## Teste de Nome-de-Condição

Seu formato é: **[NOT]** nome-de-condição

O nome-de-condição é definido no nível 88, na DATA DIVISION.

Exemplo:

```
DATA DIVISION.
:
    03 SEXO PIC .
        88 HOMEM VALUE "1".
        88 MULHER VALUE "3".
```

```
PROCEDURE DIVISION.
:
    IF HOMEM GO TO ROT-MACHO.
    GO TO ROT-FRAGIL.
```

## Condição Composta

As palavras reservadas AND e OR permitem testar as condições simples da seguinte maneira:

1. AND. Se a primeira condição simples é verdadeira e a segunda condição simples também, então a condição composta é verdadeira. Se uma das condições simples é falsa, então a condição composta também é falsa.

2. OR. Se a primeira ou a segunda condição é verdadeira então a condição composta é verdadeira. Se as duas condições são falsas, então a condição composta é falsa.

O operador NOT também pode aparecer em condições compostas:

Formato:

[NOT] condição-1 [{**AND/OR**} [NOT] condição-2],

onde condição-1 e condição-2 podem ser:

1. uma condição simples, exemplo: **A > B**;

2. negação de uma condição simples, exemplo: **NOT (A > B)**;

3. condição composta, exemplo: **A > B AND C IS POSITIVE**;

4. a negação de uma condição composta, exemplo:

**NOT (A > B AND C IS POSITIVE)**;

5. qualquer combinação das condições acima.

As seguintes regras definem o cálculo das condições compostas:

1. o par de parênteses mais interno é tratado em primeiro lugar;

2. as expressões aritméticas são reduzidas a um único valor numérico;

3. as relações condicionais, testes de classe, testes de sinal e testes de nomes de condição são calculados;

4. todos os **NOT** são efetuados da esquerda para a direita;

5. todos os **AND** são efetuados da esquerda para a direita;

6. todos os **OR** são efetuados da esquerda para a direita;

7. o próximo par de parênteses mais interno é tratado de acordo com as regras 2 e 6.

Exemplos:

```
IF A < B OR C = D OR E NOT > F GO TO AAA.

IF A = 1 AND B = 2
    GO TO ROT-CERTO
ELSE
    GO TO ROT-ERRADO.

IF NOT (A AND B > C OR D IS POSITIVE OR F < 100 GO TO ROT-QQ.

IF CP1 = CP2 AND BAND = "I" OR XAVE = 0 GO TO ROT-PROC.
```

Possibilidades na execução da instrução do último exemplo:

| Valor do Dado | | | | ROT-PROC sera executado? |
|---|---|---|---|---|
| CP1 | CP2 | BAND | XAVE | |
| 10 | 10 | "I" | 1 | Sim |
| 10 | 11 | "I" | 1 | Não |
| 10 | 11 | "I" | 0 | Sim |
| 10 | 10 | "M" | 1 | Não |
| 6 | 3 | "M" | 0 | Sim |
| 6 | 6 | "M" | 1 | Não |

## INSTRUÇÃO GO TO

Este comando quebra a seqüência natural de execução do programa.

Seu formato é:

**GO** TO nome-de-procedimento....[DEPENDING ON nome-de-dado]

A forma GO TO nome-de-procedimento transfere o fluxo do programa para o parágrafo ou para a seção designada.

O nome-de-dado deve conter valor entre 1 e n e deve ter n nomes-de-procedimentos. O nome-de-dado deve ser um item numérico elementar com dado numérico interno. Qualquer erro de especificação de controle segue normal sem executar a instrução GO TO.

Exemplo:

```
    GO TO CASO-1 CASO-2 CASO-3 CASO-4 DEPENDING ON CASO.

CASO-1.
:
    GO TO SEGUE.
CASO-2.
:
    GO TO SEGUE.
CASO-3.
```

```
        GO TO SEGUE.
CASO-4.
    :
    :
SEGUE.
    :
    :
```

No exemplo, se CASO = 3 então o fluxo é desviado para o parágrafo CASO-3. O comando GO TO SEGUE é necessário para que apenas um parágrafo seja executado, aquele selecionado pelo valor presente de CASO. Se CASO valer ZERO ou mais do que quatro também se cairá em CASO-1.

## INSTRUÇÃO ALTER

O comando ALTER é usado para modificar um comando simples GO TO em qualquer lugar da PROCEDURE DIVISION, mudando assim a seqüência da execução dos comandos do programa.

Formato:

**ALTER** parágrafo **TO** [PROCEED TO] nome-de-procedimento.

No parágrafo especificado deve ter apenas um comando GO TO simples.

Exemplo:

```
ROT-A.
    GO TO ABRIR.
ABRIR.
    OPEN INPUT ARQUIVO.
    ALTER ROT-A TO PROCEED TO LER.
LER.
    READ ARQUIVO AT END GO TO FIM-PGM.
    :
    :
```

Ação: na primeira vez em que o programa passa pelo parágrafo ROT-A, abre o arquivo e o lê e nas outras vezes vai direto para o parágrafo LER.

## INSTRUÇÃO EXIT

Função: É usado quando se precisa criar um ponto de fim em um parágrafo ou seção.

Formato:

```
nome-de-parágrafo. EXIT.
```

O EXIT deve aparecer no parágrafo isoladamente. Tal recurso é usado quando se deseja um ponto final, para o qual os comandos precedentes podem transferir o controle do programa, se for o caso de ignorar alguma parte da seção ou parágrafo.

Exemplo:

```
LEITURA.

    IF CHAVE = 9
        GO TO SAIDA
    ELSE
        MOVE REGISTRO TO AREA-TRABALHO
        GO TO LEITURA.
    SAIDA.
        EXIT.
```

# EXERCÍCIOS

Exercício 1

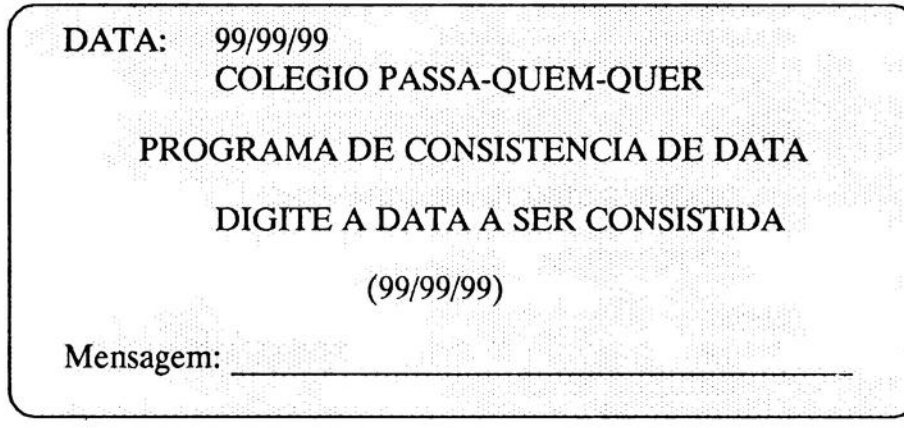
## 1. Objetivo do Programa

Elaborar um programa para realizar a consistência de uma data qualquer.

## 2. Procedimentos

1. exibir a tela formatada;
2. aceitar a data do sistema e exibi-la no canto superior esquerdo;
3. receber a data a ser consistida:
   *a.* o dia deve ser > 0 ou < ou = a 31;
   *b.* o mês deve ser > 0 ou < ou = a 12;
   *c.* se o dia for 31, o mês deverá ser 01, 03, 05, 07, 08, 10 ou 12;
   *d.* se o mês for 02 o dia deverá ser até 29. Caso seja 29, verificar se o ano é bissexto;
3. emitir mensagem de consistência;
4. condicionar final de processamento do programa.

## 3. Formato da Tela (Video Layout)



Exercício 2

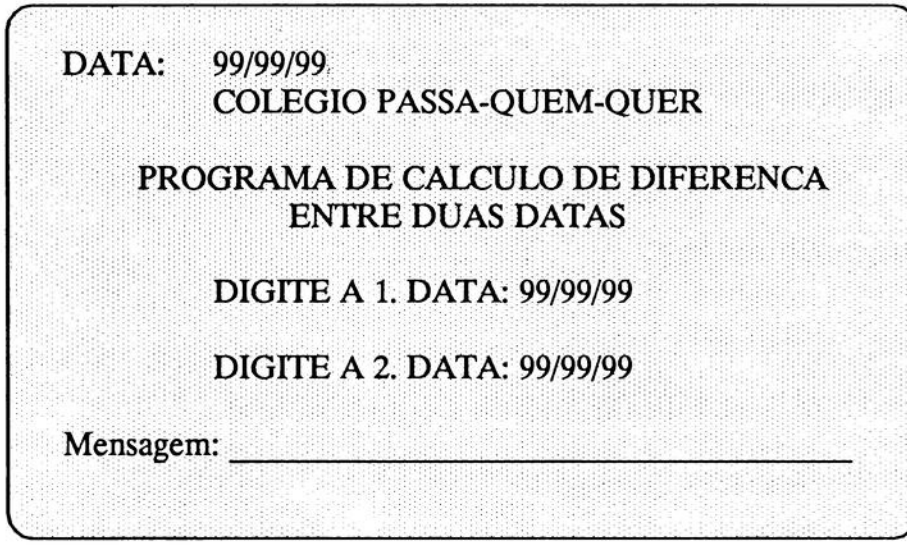
## 1. Objetivo do Programa

Elaborar um programa para calcular a diferença, em dias, de duas datas quaisquer.

## 2. Procedimentos

1. exibir a tela formatada;
2. aceitar a data do sistema e exibi-la no canto superior esquerdo;
3. entrar com as duas datas a serem consistidas;
4. emitir mensagem de consistência;
5. condicionar final de processamento do programa.

## 3. Formato da Tela (Video Layout)



# CAPÍTULO 3

## PROGRAMA PARA CRIAÇÃO DE ARQUIVO SEQÜENCIAL

### 1. Objetivo do Programa

O propósito deste programa é criar um arquivo seqüencial.

### 2. Procedimentos

1. exibir tela;

2. receber a data do sistema e exibi-la na tela;

3. receber os campos;

4. consistência:

*a*. código do funcionário:

- diferente de zeros;
- 9999 encerrar programa;

*b*. nome do funcionário diferente de brancos;

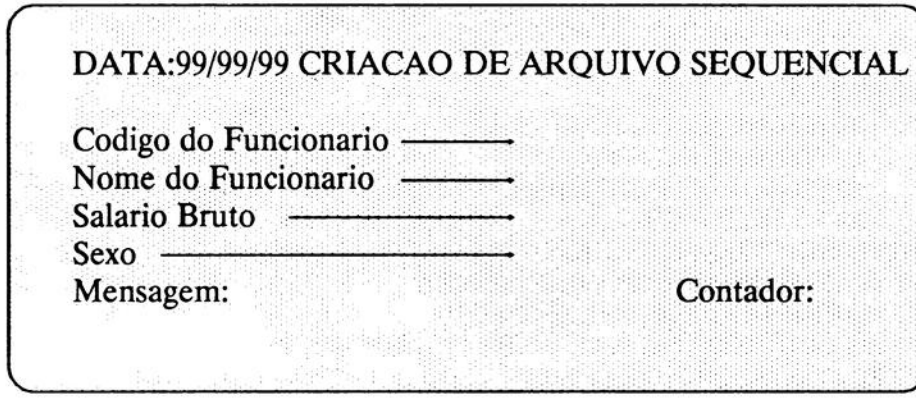
*c*. salário bruto diferente de zeros;

*d*. sexo = "F" ou "M" ou "1" ou "2";

5. gravar o registro;

6. exibir, para cada registro gravado, a seqüência, através de um contador.

## 3. Formato da Tela (Video Layout)




## 4. Record Layout

| codigo do funcionario | nome do funcionario | salario bruto | sexo |
|---|---|---|---|
| 9(4) | x(30) | 9(5)v99 | x |

## Codificação do Programa

*Observação:* As instruções e comandos já utilizados anteriormente não serão comentados.

```
1   IDENTIFICATION DIVISION.
2   PROGRAM-ID. CRIA.
3   AUTHOR. COLEGIO BRASIL.
4   ENVIRONMENT DIVISION.
5   CONFIGURATION SECTION.
6   SPECIAL-NAMES. DECIMAL-POINT IS COMMA.
7   INPUT-OUTPUT SECTION.
8   FILE-CONTROL.
9       SELECT  CRIA01 ASSIGN TO DISK
10                  ORGANIZATION IS SEQUENTIAL
11                  ACCESS MODE IS SEQUENTIAL
12                  FILE STATUS IS WS-STATUS.
13  DATA DIVISION.
14  FILE SECTION.
15  FD CRIA01
16      LABEL RECORD IS STANDARD
17      RECORD CONTAINS 42 CHARACTERS
18      DATA RECORD IS REG-CRIA01
19      VALUE OF FILE-ID IS "CRIA01".
20  01 REG-CRIA01.
21      02 FD-CODIGO PIC 9(04).
22      02 FD-NOME PIC X(30).
23      02 FD-SALARIO PIC 9(05)V99.
24      02 FD-SEXO PIC X.
25  WORKING-STORAGE SECTION.
26  77 WS-STATUS PIC X(02) VALUE SPACES.
27  77 WS-LIMPA PIC X(30) VALUE SPACES.
28  77 WS-CONTADOR PIC 9(03) VALUE ZEROS.
29  77 ED-CONTADOR PIC ZZ9 VALUE ZEROS.
30  01 WS-DATA.
31      02 WS-ANO PIC 9(02) VALUE ZEROS.
32      02 WS-MES PIC 9(02) VALUE ZEROS.
33      02 WS-DIA PIC 9(02) VALUE ZEROS.
34  SCREEN SECTION.
35  01 TELA.
36      02 BLANK SCREEN.
37      02 LINE 01 COLUMN 01 VALUE "DATA".
38      02 LINE 01 COLUMN 24 VALUE "CRIACAO DE ARQUIVO SEQUENCIAL".
39      02 LINE 03 COLUMN 19 VALUE
```

```
40          "Autor: Colegio Brasil - Nr: 000 - Serie: 2. Turma: Z".
41          02 LINE 06 COLUMN 19 VALUE "Codigo do Funcionario -".
42          02 LINE 08 COLUMN 19 VALUE "Nome do Funcionario -".
43          02 LINE 10 COLUMN 19 VALUE "Salario Bruto -".
44          02 LINE 12 COLUMN 19 VALUE "Sexo -".
45          02 LINE 20 COLUMN 19 VALUE "Mensagem:".
46          02 LINE 20 COLUMN 65 VALUE "Contador".
47      PROCEDURE DIVISION.
48      INICIO.
49          DISPLAY (01 01) ERASE.
50          OPEN OUTPUT CRIA01.
51          IF WS-STATUS = "00"
52              NEXT SENTENCE
53          ELSE
54              DISPLAY (20 29) "Erro de Abertura"
55              STOP RUN.
56          ACCEPT WS-DATA FROM DATE.
57      VIDEO.
58          DISPLAY TELA.
59          DISPLAY (01 06) WS-DIA "/" WS-MES "/" WS-ANO.
60      LIMPA-DADOS.
61          DISPLAY (06 43) WS-LIMPA.
62          DISPLAY (08 43) WS-LIMPA.
63          DISPLAY (10 43) WS-LIMPA.
64          DISPLAY (12 43) WS-LIMPA.
65      A-CODIGO.
66          ACCEPT (06 43) FD-CODIGO WITH PROMPT.
67          IF FD-CODIGO = ZEROS
68              DISPLAY (20 29) "Codigo Invalido - Redigite"
69              GO TO A-CODIGO.
70          IF FD-CODIGO = 9999
71              GO TO FIM.
72      A-NOME.
73          ACCEPT (08 43) FD-NOME WITH PROMPT.
74          IF FD-NOME = SPACES
75              DISPLAY (20 29) "Nome em Branco - Redigite"
76              GO TO A-NOME.
77      A-SALARIO.
78          ACCEPT (10 43) FD-SALARIO WITH PROMPT.
79          IF FD-SALARIO = ZEROS
80              DISPLAY (20 29) "Salario Invalido - Redigite"
81              GO TO A-SALARIO.
82      A-SEXO.
83          ACCEPT (12 43) FD-SEXO WITH PROMPT.
84          IF FD-SEXO = "F" OR "M" OR 1 OR 2
```

```
85              NEXT SENTENCE
86          ELSE
87              DISPLAY (20 29) "Sexo Invalido - Redigite"
88              GO TO A-SEXO.
89  GRAVAR.
90          WRITE REG-CRIA01.
91          IF WS-STATUS = "00"
92              NEXT SENTENCE
93          ELSE
94              DISPLAY (20 29) "Erro de Gravacao = " WS-STATUS
95              STOP RUN.
96          ADD 1 TO WS-CONTADOR.
97          MOVE WS-CONTADOR TO ED-CONTADOR.
98          DISPLAY (20 75) ED-CONTADOR.
99          GO TO LIMPA-DADOS.
100 FIM.
101         DISPLAY (01 01) ERASE.
102         DISPLAY (10 40) "Fim de Programa".
103         CLOSE CRIA01.
104         STOP RUN.
105
```

## Explanação e Instruções Introduzidas no Programa

INPUT-OUTPUT SECTION. Seção de entrada e saída. Toda vez que o programa manipular arquivo esta seção se torna obrigatória, como neste exemplo.

FILE-CONTROL. Neste parágrafo é feita a atribuição de nome de arquivos, descrição das características físicas e atribuição do número de dispositivo de entrada/saída para tais arquivos.

SELECT CRIA01 ASSIGN TO DISK. Está atribuindo o arquivo CRIA01 para um dispositivo de entrada/saída que neste caso é o disco flexível (DISK).

ORGANIZATION SEQUENTIAL. Declaração de que o arquivo tem os registros em posição fisicamente seqüencial. Esta cláusula é opcional, quando se tratar de arquivo seqüencial.

ACCESS MODE SEQUENTIAL. Declaração de que o acesso aos registros do arquivo é de modo seqüencial. Esta cláusula, também, é opcional, quando se quer este tipo de acesso.

FILE STATUS WS-STATUS. Esta cláusula é utilizada para verificar se há ocorrência de erro após a execução de uma operação de entrada e saída. O campo WS-STATUS está definido na WORKING-STORAGE SECTION, com PICTURE X(02).

Para arquivos seqüenciais temos os seguintes códigos:

00 - operação completa sem erros;

10 - encontrou EOF (fim de arquivo);

30 - erro permanente;

24 - não há área disponível no disco;

91 - arquivo com estrutura destruída.

FILE SECTION. Seção de arquivos.

```
FD  CRIA01
    LABEL RECORD STANDARD
    RECORD CONTAINS 42 CHARACTERS
    DATA RECORD REG-CRIA01
    VALUE OF FILE-ID "CRIA01".
```

Neste conjunto de descrições devemos considerar como obrigatório a FD nome-de-arquivo, LABEL RECORD STANDARD e VALUE OF FILE-ID IS literal.

LABEL. Só existe nos rolos de fita e nos discos, não existindo nos cartões perfurados ou impressora. Portanto, quando não há label escrevemos LABEL RECORD OMITTED.

RECORD CONTAINS 42 CHARACTERS. Especifica que cada registro do arquivo denominado CRIA01 contém 42 caracteres de comprimento (tamanho).

DATA RECORD REG-CRIA01. Especifica que o nome do registro será REG-CRIA01 (nível 01 da descrição do registro).

VALUE OF FILE-ID "CRIA01". Indica que o nome do arquivo especificado no diretório do disquete possui o nome de CRIA01.

```
01  REG-CRIA01.
    02 FD-CODIGO        PIC 9(04).
    02 FD-NOME          PIC X(30).
    02 FD-SALARIO       PIC 9(05)V99.
    02 FD-SEXO          PIC X.
```

O conjunto acima especifica o nome do registro e a descrição dos nomes-de-campos com seus respectivos tamanhos para referências posteriores na PROCEDURE DIVISON.

OPEN OUTPUT CRIA01. Especifica que o arquivo CRIA01 será aberto no modo OUTPUT, isto é, para saída (gravação).

```
IF WS-STATUS = "00" NEXT SENTENCE
ELSE DISPLAY (20, 29) "ERRO DE ABERTURA" STOP RUN.
```

Neste conjunto de instruções é verificado se o arquivo especificado foi aberto sem erros.

"00". Todos os campos descritos como alfanuméricos (PIC X), quando forem referenciados os seus conteúdos, deverão estar entre aspas se não contiverem valores numéricos.

```
ADD 1 TO WS-CONTADOR.
MOVE WS-CONTADOR TO ED-CONTADOR.
DISPLAY (15, 65) ED-CONTADOR.
```

Especifica que será somado 1 inteiro no campo numérico WS-CONTADOR, movimentado para o campo numérico editado

ED-CONTADOR para ser exibido no vídeo. Campos definidos como numéricos de edição não podem estar comprometidos dentro de uma operação aritmética.

WRITE REG-CRIA01. Grava o registro REG-CRIA01 no arquivo CRIA01.

## INSTRUÇÃO MOVE

Este comando tem por fim transferir dados de um campo (fonte) para outro (receptor).

O seu formato é:

**MOVE** {nome-de-dado-1/literal} **TO** nome-de-dado-2 [nome-de-dado-3].....

O dado representado por nome-de-dado-1 ou literal é movido para nome-de-dado-2. Pode haver mais do que um campo receptor.

Exemplo: MOVE ZERO TO A B C.

Quando o campo-receptor for um item de grupo, a transferência será feita sem considerar a estrutura do nível do grupo envolvido e sem edição.

Na execução de um comando MOVE são válidas as seguintes afirmações:

1. numérico para numérico ou de edição:

   *a.* os itens são alinhados pelo ponto decimal com geração de zeros ou truncamento em ambas as extremidades dependendo do número de significativos no campo-fonte;

   *b.* quando os tipos dos campos fonte e receptor diferem, a conversão para o tipo do receptor é feita;

*c.* os itens podem receber tratamento de edição como supressão de zeros não-significativos, inclusão do cifrão ou de um ponto-decimal explícito de acordo com a PICTURE do campo-receptor;

2. fonte e receptor não-numéricos:

*a.* os caracteres são gravados no receptor da esquerda para a direita;

*b.* se o campo-receptor for mais longo que o campo-fonte então será completado com brancos;

3. se **receptor** for o menor o MOVE termina quando ele estiver pre**enchido;**

*a.* se o campo-fonte e o campo-receptor forem de algum modo superpostos (uso de REDEFINES) o resultado do MOVE será imprevisível.

Exemplos de uso do comando MOVE:

| **Campo-Fonte** | | **Campo-Receptor** | | |
|---|---|---|---|---|
| PICTURE | VALOR | PICTURE | VALOR ANTES DO MOVE | VALOR APOS O MOVE |
| 99V99 | 1234 | S99V99 | 9876- | 1234+ |
| 99V99 | 1234 | 99V9 | 987 | 123 |
| S9V9 | 12- | 99V999 | 98765 | 01200 |
| XXX | A2B | XXXXX | Y9X8W | A2Bbb |
| 9V99 | 123 | 99.99 | 87.65 | 01.23 |

## ARQUIVO

Denomina-se arquivo ao conjunto de registros que possuem uma organização lógica.

Classificação dos arquivos:

- seqüencial;
- indexado;
- relativo;

## ARQUIVO SEQÜENCIAL

Um arquivo seqüencial em disco flexível possui registros armazenados conforme a ordem em que os mesmos foram gravados.

## INSTRUÇÃO OPEN

Formato: **OPEN {INPUT/I-O/OUTPUT/EXTEND}** nome-de-arquivo.....

Função: preparação necessária para conectar o arquivo com o programa.

- opção INPUT. O OPEN INPUT abre o arquivo e também lê o primeiro registro para a memória o que permite que o primeiro READ não tenha atraso.

- opção OUTPUT. O OPEN OUTPUT torna disponível a área para a preparação de um registro o qual será transmitido para o dispositivo de saída após a execução de um WRITE; abre o arquivo e posiciona no primeiro registro do arquivo. Se o arquivo já contiver dados os mesmos serão perdidos.

- opção I-O. Somente é permitida para arquivos em disco. Permite o uso do comando REWRITE para alterar registros.

- opção EXTEND. Posiciona o arquivo para escrita no primeiro registro livre. Assim, se no momento do OPEN EXTEND o arquivo tiver dez registros o primeiro WRITE gravará o décimo primeiro registro.

## INSTRUÇÃO CLOSE

Formato: **CLOSE** {nome-de-arquivo [WITH LOCK].....}

Função: responsável pelos procedimentos finais em relação aos arquivos, isto é, após completar o processamento de um arquivo o mesmo deve ser fechado (colocação de uma marca de fim).

- opção LOCK. Não permite nova abertura dentro do mesmo programa.

# INSTRUÇÃO READ

Formato: **READ** nome-de-arquivo RECORD [**INTO** nome-de-dado]
[AT **END** instrução-incondicional].

Função: lê o registro do arquivo no qual foi especificado INPUT ou I-O na instrução OPEN, isto é, torna disponível o próximo registro lógico do arquivo designado, atualizando também o valor do FILE STATUS, se for especificado.

Uma vez que em algum instante será atingido o fim do arquivo o programador pode usar a cláusula AT END. A palavra END é seguida por um número qualquer de comandos imperativos, cada um dos quais será executado somente na situação de fim de arquivo. O último comando imperativo será seguido por um ponto-de-período que indicará o fim de sentença. Se a situação de fim de arquivo ocorrer, e não houver a cláusula AT END, então o procedimento para esta situação será executado. Se o AT END não for incluído e nem houver a declaração de FILE STATUS para este arquivo, haverá então um contexto de erro de I-O em tempo de execução.

Quando há registros para serem lidos, a execução bem-sucedida do comando READ é seguida imediatamente pela execução da próxima sentença.

A opção INTO leva imediatamente uma cópia do registro lido para nome-de-dado. Este não deve ser definido como registro do arquivo.

Quando forem definidos 2 ou mais registros subordinados na mesma FD, esses registros ocuparão a mesma área na memória.

Exemplo:

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
        SELECT CADASTRO ASSIGN TO DISK
                FILE STATUS FS.
```

```
DATA DIVISION.
FILE SECTION.
FD CADASTRO LABEL RECORD STANDARD
            VALUE OF FILE-ID "CADASTRO".

01 REGISTRO PIC X(80).
WORKING-STORAGE SECTION.
01 AREA-TRABALHO.
    02 COD PIC 9(05).
    02 NOME PIC X(40).
    02 ENDERECO PIC X(35).
:
PROCEDURE DIVISION.
:

*..... > Exemplo de leitura simples.
    READ CADASTRO AT END GO TO FECHAMENTO.
    MOVE REGISTRO DO AREA-TRABALHO.


*..... > Exemplo de leitura com opção INTO.
    READ CADASTRO INTO AREA-TRABALHO AT END GO TO FECHAMENTO.
```

Ação: a opção INTO transfére o registro para o campo AREA-TRABALHO sem utilização da instrução MOVE.

## INSTRUÇÃO WRITE

A Instrução WRITE procede apenas para arquivo seqüencial em disco e fita.

Formato: **WRITE** nome-de-registro [**FROM** nome-de-dado].

Função: grava o registro no arquivo que foi especificado com OUTPUT ou I-O na instrução OPEN.

É importante lembrar que se executa um READ sobre arquivo e WRITE sobre registro.

O nome-de-registro deve ser definido no nível 01 da FD (DATA DIVISION).

Se os dados para saída foram desenvolvidos na WORKING-STORAGE SECTION a opção **FROM** permite ao usuário estipular que o dado designado (nome-de-dado) deve ser copiado na área "nome-de-registro" e então sair a partir daí. O nome-de-registro e o nome-de-dado devem se referir a áreas de armazenamento separadas.

Exemplo:

```
FD ARQUIVO LABEL RECORD STANDARD
              VALUE OF FILE-ID "CADASTRO".
01   REGISTRO.
     02 CODIGO        PIC 9(06).
     02 NOME          PIC X(40).
     02 ENDERECO      PIC X(34).
:
:
PROCEDURE DIVISION.
INICIO.   OPEN OUTPUT ARQUIVO.
:
:
GRAVACAO. WRITE REGISTRO.
:
```

## INSTRUÇÃO REWRITE

Formato: **REWRITE** nome-de-registro [**FROM** nome-de-dado].

Função: Substitui os dados de um registro qualquer de um arquivo seqüencial em disco.

Antes do REWRITE, deverá ser executado um READ referente ao mesmo registro.

A instrução OPEN referente ao arquivo deve ser especificada com I-O.

Nome-de-registro é o nome de um registro lógico na FILE SECTION da DATA DIVISION e pode ser qualificado. Nome de registro e nome-de-dado devem se referir a áreas de memórias separadas.

Se a opção **FROM** é incluída no comando, o efeito é como se o comando **MOVE** tivesse sido executado antes do **REWRITE.**

Se o registro que está sendo regravado for maior que o registro do arquivo, somente serão reescritos tantos caracteres quantos couberem.

Exemplo:

```
DATA DIVISION.
FILE SECTION.

FD ARQUIVO LABEL RECORD STANDARD
              VALUE OF FILE-ID "ARQ1".
01   REGISTRO.
     02 CODIGO   PIC 9(06).
     02 VALOR    PIC 9(06)V99.
:
:
PROCEDURE DIVISION.
INICIO.  OPEN I-O ARQUIVO.
LER-REGRAVAR.
     READ ARQUIVO AT END GO TO FIM.
     ACCEPT (05 10) VALOR-NOVO.
     MOVE VALOR-NOVO TO VALOR.
     REWRITE REGISTRO.
     GO TO LER-REGRAVAR.
FIM.
:
```

Se ocorrer um erro de E/S, o item FILE STATUS do arquivo, se especificado, será ajustado ao código de dois caracteres apropriados, caso contrário, assumirá o valor "00".

Se ocorrer um erro de E/S e for do tipo que é pertinente a uma cláusula **AT END** ou **INVALID KEY**, então os comandos imperativos em tal cláusula, se estiverem presentes no comando que provocou o erro, serão executados. Porém se houver uma cláusula apropriada, então a lógica do programa seguirá deste modo:

1. se existir um procedimento declarativo de erro associado, ele será executado automaticamente; a lógica escrita pelo usuário deve determinar qual ação a ser tomada devido a existência de erro. Após o retorno do tratamento de erro, o fluxo normal do programa segue para o comando seguinte;

2. se não houver nenhum declarativo de procedimento de erro aplicável, mas existir um item FILE STATUS associado, pressupõe-se que o usuário poderá basear as suas ações no teste do item STATUS, de forma que seja permitido o fluxo normal para o comando seguinte.

Se não existir nenhuma das cláusulas acima (INVALID KEY/AT END, Declarativo de Procedimento de Erro ou FILE STATUS), então o manipulador de erro em tempo de execução receberá o controle; a localização de erro (número da linha do programa fonte) será apresentada na console, e a execução terminada anormalmente.

Estas informações se aplicam ao procedimento de qualquer arquivo cuja organização seja seqüencial, line sequential, indexada ou relativa.

## EXERCÍCIOS

Exercício 1

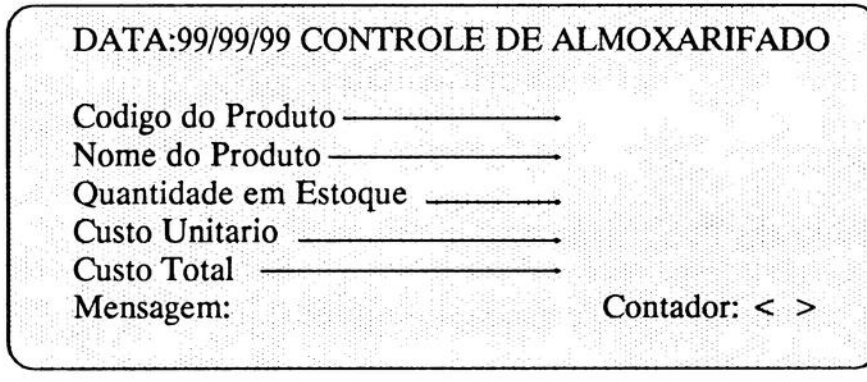
### 1. Objetivo do Programa

Criar um arquivo de organização seqüencial.

## 2. Procedimentos

1. exibir tela;
2. receber a data do sistema e exibi-la no vídeo;
3. receber os campos;
4. consistência dos campos:
   *a*. código diferente de zeros; código = 9999 encerrar programa;
   *b*. nome diferente de brancos;
   *c*. quantidade de estoque maior ou igual a 10;
   *d*. custo unitário diferente de zeros;
   *e*. custo total = quantidade de estoque x custo unitário.
5. gravar registro;
6. exibir a seqüência dos registros que estão sendo gravados através de um contador.

## 3. Formato da Tela (Video Layout)



## 4. Record Layout

| codigo do produto | nome do produto | quantidade estoque | custo unitario | custo total |
|---|---|---|---|---|
| 9(4) | x(30) | 9(04) | 9(05)v99 | 9(06)v99 |

Exercício 2

## 1. Objetivo do Programa

Listar no vídeo os registros do arquivo seqüencial CRIA02.

## 2. Procedimentos

1. data do sistema;

2. exibir tela;

3. ler o arquivo até o fim ou outra condição proposta;

4. calcular 25% do custo total e exibi-lo juntamente com o restante dos dados;

5. exibir um conjunto de 7 registros com espaço duplo;

6. prever opção de continuar ou não após listar o conjunto de 7 registros;

7. no final do processo exibir o maior custo total majorado.

## 3. Formato da Tela (Video Layout)

```
DATA:99/99/99          CONTROLE DE ALMOXARIFADO

CODIGO   N O M E        ESTOQUE  V.UNITARIO    V.TOTAL

9999     X---(30)---X   Z.ZZ9    ZZ.ZZ9,99     ZZZ.ZZ9,99

9999     X---(30)---X   Z.ZZ9    ZZ.ZZ9,99     ZZZ.ZZ9,99

Mensagem:
```

Exercício 3

## 1. Objetivo do Programa

Confeccionar um programa que faça a manutenção de um arquivo seqüencial de clientes de uma loja.

## 2. Procedimentos

1. data do sistema;

2. exibir tela principal (menu);

3. selecionar a opção desejada;

4. INCLUSÃO - consistir todos os campos e gravar;

5. ALTERAÇÃO - localizar o registro a ser alterado, exibi-lo no vídeo, consistir o campo a alterar e regravar o registro;

6. EXCLUSÃO - localizar o registro a ser excluído, exibi-lo no vídeo, confirmar e excluir. Para excluir é necessário utilizar um arquivo auxiliar;

7. CONSULTA - localizar o registro a ser consultado e exibi-lo no vídeo;

8. FIM - encerrar o processamento.

## 3. Formato da Tela (Video Layout)

**Formato da Tela 1 (Video Layout)**

```
DATA: 99/99/99 MANUTENCAO DO ARQUIVO DE CLIENTES

Autor: ..... Nr: ...      Serie: ...      Turma: ...

        TELA PRINCIPAL
        (1) INCLUSAO
        (2) ALTERACAO
        (3) EXCLUSAO
        (4) CONSULTA
        (5) FIM

Mensagem:
```

**Formato da Tela 2 (Video Layout)**

```
DATA: 99/99/99 MANUTENCAO DO ARQUIVO DE CLIENTES

Autor: ..... Nr: ...      Serie: ...      Turma: ...

                (*)_____________
                (0) CODIGO:
                (1) NOME:
                (3) ENDERECO:
                (4) BAIRRO:
                (5) CEP:
                (6) CIDADE:
Mensagem:
```

(*) INCLUSÃO, ALTERAÇÃO, EXCLUSÃO, CONSULTA

## 4. Formato de Registro (Record Layout)

| CODIGO | NOME | ENDERECO | BAIRRO | CEP | CIDADE |
|---|---|---|---|---|---|
| 9(04) | X(30) | X(30) | X(20) | 9(05) | X(20) |

# CAPÍTULO 4

## PROGRAMA DE EMISSÃO DE LISTAGEM

### 1. Objetivo do Programa

Emitir um relatório a partir do arquivo seqüencial denominado CADASTRO.

### 2. Procedimentos

1. ler o arquivo;

2. acrescer 35% no salário bruto, se o sexo for "M" e 45% se o sexo for "F";

3. do novo salário bruto calcular 9% referente ao valor do INPS e o salário líquido;

4. imprimir 25 linhas com espaço duplo em cada folha, colocando os cabeçalhos em todas as páginas;

5. no final, exibir o total de registros impressos.

## 3. Record Layout

| CODIGO | NOME DO FUNCIONARIO | SALARIO | SEXO |
|---|---|---|---|
| 9(04) | X(30) | 9(05)V99 | X |

## 4. Gabarito de Impressão (Printer Layout)

```
DATA 99/99/99  RELACAO DE PAGAMENTOS DA EMPRESA XYZ  PAG. Z9

CODIGO  NOME DO FUNCIONARIO  SAL. ANT.  SAL. ATUAL  INPS  LIQUIDO

9999    X---(30)---X         ZZ.ZZ9,99  ZZ.ZZ9,99  ZZ.ZZ9,99  ZZ.ZZ9,99

9999    X---(30)---X         ZZ.ZZ9,99  ZZ.ZZ9,99  ZZ.ZZ9,99  ZZ.ZZ9,99

        TOTAL DE REGISTROS IMPRESSOS = ZZ9
```

# Codificação do Programa

```
1   IDENTIFICATION  DIVISION.
2   PROGRAM-ID.     CADAS.
3   AUTHOR.         COLEGIO BRASIL.
4   ENVIRONMENT     DIVISION.
5   CONFIGURATION   SECTION.
6   SPECIAL-NAMES.  DECIMAL-POINT IS COMMA.
7   INPUT-OUTPUT    SECTION.
8   FILE-CONTROL.
9       SELECT CADASTRO ASSIGN TO DISK
10          ORGANIZATION IS LINE SEQUENTIAL
11          FILE STATUS IS FS.
12      SELECT RELATO ASSIGN TO PRINTER.
13  DATA DIVISION.
14  FILE SECTION.
15  FD  CADASTRO LABEL RECORD STANDARD
16              VALUE OF FILE-ID "CLIENTES".
17  01  REG-CADASTRO.
18      02 FD-CODIGO    PIC 9(04).
19      02 FD-NOME      PIC X(30).
20      02 FD-SALARIO   PIC 9(05)V99.
21      02 FD-SEXO      PIC X.
22      FD  RELATO  LABEL RECORD OMITTED
23                  LINAGE 55 TOP 6 BOTTOM 5.
24  01  REG-RELATO PIC X(132).
25  WORKING-STORAGE SECTION.
26  77  FS             PIC X(02)     VALUE  SPACES.
27  77  WS-SALATU      PIC 9(05)V99  VALUE  ZEROS.
28  77  WS-INPS        PIC 9(05)V99  VALUE  ZEROS.
29  77  WS-LIQUIDO     PIC 9(05)V99  VALUE  ZEROS.
30  77  CT-PAG         PIC 9(02)     VALUE  ZEROS.
31  77  CT-IMP         PIC 9(03)     VALUE  ZEROS.
32  01  DATA-DO-SISTEMA.
33      02 ANO-SIS      PIC 9(02)     VALUE  ZEROS.
34      02 MES-SIS      PIC 9(02)     VALUE  ZEROS.
35      02 DIA-SIS      PIC 9(02)     VALUE  ZEROS.
36  01  CAB01.
37      02 FILLER       PIC X(04)     VALUE  SPACES.
38      02 FILLER       PIC X(06)     VALUE  "DATA:".
39      02 DATA-CAB01.
40          03 DIA-CAB01    PIC 99/   VALUE  ZEROS.
41          03 MES-CAB01    PIC 99/   VALUE  ZEROS.
42          03 ANO-CAB01    PIC 99    VALUE  ZEROS.
```

```
43          02 FILLER           PIC X(32)       VALUE   SPACES.
44          02 FILLER           PIC X(36)       VALUE   "RELACAO DE PAGAMENTO
45      -   "S DA EMPRESA XYZ".
46          02 FILLER           PIC X(29)       VALUE   SPACES.
47          02 FILLER           PIC X(08)       VALUE   "PAG:".
48          02 PAG-CAB01        PIC Z9          VALUE   ZEROS.
49          02 FILLER           PIC X(07)       VALUE   SPACES.
50      01  CAB02.
51          02 FILLER           PIC X(10)       VALUE   SPACES.
52          02 FILLER           PIC X(20)       VALUE   "CODIGO".
53          02 FILLER           PIC X(33)       VALUE
54          "NOME DO FUNCIONARIO".
55          02 FILLER           PIC X(15)       VALUE   "SALARIO ANT.".
56          02 FILLER           PIC X(17)       VALUE   "SALARIO ATUAL".
57          02 FILLER           PIC X(11)       VALUE   "INPS".
58          02 FILLER           PIC X(24)       VALUE   "VALOR LIQUIDO".
59      01  DETALHE.
60          02 FILLER           PIC X(11)       VALUE   SPACES.
61          02 CODIGO-DET       PIC 9(04)       VALUE   ZEROS.
62          02 FILLER           PIC X(10)       VALUE   SPACES.
63          02 NOME-DET         PIC X(30)       VALUE   SPACES.
64          02 FILLER           PIC X(10)       VALUE   SPACES.
65          02 SALANT-DET       PIC ZZ.ZZ9,99   VALUE   ZEROS.
66          02 FILLER           PIC X(06)       VALUE   SPACES.
67          02 SALATU-DET       PIC ZZ.ZZ9,99   VALUE   ZEROS.
68          02 FILLER           PIC X(06)       VALUE   SPACES.
69          02 INPS-DET         PIC ZZ.ZZ9,99   VALUE   ZEROS.
70          02 FILLER           PIC X(06)       VALUE   SPACES.
71          02 LIQUIDO-DET      PIC ZZ.ZZ9,99   VALUE   ZEROS.
72          02 FILLER           PIC X(13)       VALUE   SPACES.
73      01  TOTAL.
74          02 FILLER           PIC X(40)       VALUE   SPACES.
75          02 FILLER           PIC X(32)       VALUE
76          "TOTAL DE REGISTROS IMPRESSOS = ".
77          02 CONT-TOT         PIC ZZ9         VALUE   ZEROS.
78          02 FILLER           PIC X(62)       VALUE   SPACES.
79      SCREEN SECTION.
80      01  TELA.
81          02 BLANK SCREEN.
82          02 LINE 10 COLUMN 20 VALUE " I M P R I M I N D O . . . "
83                REVERSE-VIDEO BLINK.
84      PROCEDURE DIVISION.
85      INICIO.
86          DISPLAY (01 01) ERASE.
87          OPEN INPUT CADASTRO.
```

```
88          IF FS NOT = "00
89              IF FS NOT = "30"
90                  DISPLAY (23 35) "ARQUIVO NAO SE ENCONTRA NO DISCO"
91                  STOP RUN
92              ELSE
93                  DISPLAY (23 35) "ARQUIVO CADASTRO DANIFICADO"
94                  DISPLAY (24 35) "CODIGO DO STATUS = " FS
95                  STOP RUN
96          ELSE
97              NEXT SENTENCE.
98          OPEN OUTPUT RELATO.
99          ACCEPT DATA-DO-SISTEMA FROM DATE.
100         MOVE DIA-SIS TO DIA-CAB01.
101         MOVE MES-SIS TO MES-CAB01.
102         MOVE ANO-SIS TO ANO-CAB01.
103         DISPLAY TELA.
104     CABECALHO.
105         MOVE SPACES TO REG-RELATO.
106         WRITE REG-RELATO BEFORE ADVANCING 1 LINE.
107         ADD 1 TO CT-PAG.
108         MOVE CT-PAG TO PAG-CAB01.
109         WRITE REG-RELATO FROM CAB01 BEFORE ADVANCING 3 LINES.
110         WRITE REG-RELATO FROM CAB02 BEFORE ADVANCING 2 LINES.
111     LER.
112         READ CADASTRO AT END GO TO FIM.
113         IF FD-SEXO = "M"
114             COMPUTE WS-SALATU = FD-SALARIO * 1,35
115         ELSE
116             COMPUTE WS-SALATU = FD-SALARIO * 1,45.
117         COMPUTE WS-INPS = WS-SALATU * 9 / 100.
118         COMPUTE WS-LIQUIDO = WS-SALATU - WS-INPS.
119         MOVE FD-CODIGO TO CODIGO-DET.
120         MOVE FD-NOME TO NOME-DET.
121         MOVE FD-SALARIO TO SALANT-DET.
122         MOVE WS-SALATU TO SALATU-DET.
123         MOVE WS-INPS TO INPS-DET.
124         MOVE WS-LIQUIDO TO LIQUIDO-DET.
125         WRITE REG-RELATO FROM DETALHE BEFORE ADVANCING 2 LINES
126             AT EOP PERFORM CABECALHO.
127         ADD 1 TO CT-IMP.
128         GO TO LER.
129     FIM.
130         MOVE SPACES TO REG-RELATO.
131         WRITE REG-RELATO BEFORE ADVANCING 2 LINES.
132         MOVE CT-IMP TO CONT-TOT.
```

```
133     WRITE REG-RELATO FROM TOTAL BEFORE ADVANCING 1 LINE.
134     CLOSE CADASTRO RELATO.
135     DISPLAY (01 01) ERASE.
136     DISPLAY (10 35) "PROGRAMA ENCERRADO".
137     STOP RUN.
```

## Explanação e Instruções Introduzidas no Programa

Para se definir uma linha a ser impressa no formulário contínuo, devemos descrever todos os espaços dessa linha mesmo os espaços que ficarão em branco.

```
01 TOTAL.
    02 FILLER PIC X(40) VALUE SPACES.
    02 FILLER PIC X(27) VALUE "TOTAL DE REGISTROS-LIDOS = ".
    02 CONT-TOT PIC ZZ9 VALUE ZEROS.
    02 FILLER PIC X(62) VALUE SPACES.
```

Na definição acima teremos o literal TOTAL DE REGISTROS LIDOS = centralizado no formulário contínuo.

A palavra reservada FILLER, ou simplesmente F, é utilizada para denominar campos que não serão referenciados (utilizados) na PROCEDURE DIVISION.

```
FD RELATO   LABEL RECORD OMITTED
            LINAGE 55 TOP 6 BOTTOM 5.
```

Quando se descreve o arquivo para saída na PRINTER (impressora) declaramos sempre que o seu rótulo (LABEL) é OMITTED.

LINAGE 55. É a declaração de que o corpo do relatório terá a quantidade de linhas declarada.

TOP 6. É a declaração de que ficarão 6 linhas em branco antes do corpo do relatório.

BOTTOM 5. É a declaração de que ficarão 5 linhas em branco após o corpo do relatório.

```
02 DATA-CAB01.
    03 DIA-CAB01      PIC 99/      VALUE  ZEROS.
    03 MES-CAB01      PIC 99/      VALUE  ZEROS.
    03 ANO-CAB01      PIC 99       VALUE  ZEROS.
:
:
02 SALANT-DET PIC ZZ.ZZ9,99        VALUE  ZEROS.
:
```

99/. Representa campo numérico editado, isto é, a / estará presente no instante da impressão.

ZZ.ZZ9,99. Representa campo numérico editado, onde serão suprimidos os zeros à esquerda a partir do primeiro significativo.

WRITE REG-RELATO BEFORE ADVANCING 1 LINE. Imprime o conteúdo do registro REG-RELATO.

WRITE REG-RELATO FORM DETALHE BEFORE ADVANCING 2 LINES AT EOP PERFORM CABECALHO. Imprime o conteúdo do campo DETALHE através do registro REG-RELATO. Após a impressão salta uma linha. Testa se há fim de página através da decisão AT EOP. Caso afirmativo é executado a rotina de CABECALHO. Caso negativo o processo continua sem desvio temporário.

PERFORM CABECALHO. Desvia o programa, temporariamente, para a rotina denominada CABECALHO, executa a mesma, e retorna para o passo seguinte ao comando PERFORM.

## CLÁUSULAS LINAGE, TOP, BOTTOM e FOOTING

Formato Genérico:

**LINAGE** IS {nome-de-dado-1/inteiro-1} LINES [WITH **FOOTING** AT {nome-de-dado-2/inteiro-2}] [LINES AT **TOP** {nome-de-dado-3/inteiro-3}] [LINES AT **BOTTOM** {nome-de-dado- 4/inteiro-4}]

Possuem a função de determinar o número de linhas lógicas de uma página impressa.

Regras Sintáticas:

1. todos os nomes-de-dados são inteiros sem sinal;

2. LINAGE define a área a ser impressa denominada "corpo";

3. o tamanho da página corresponde à soma de todos os valores em cada frase, exceto o valor de FOOTING;

4. O TOP define o número de linhas da página antes do corpo e o BOTTOM define o número de linhas da página depois do corpo;

5. se não for especificado o TOP ou o BOTTOM assumirá como ZERO;

6. a área denominada FOOTING corresponde à parte do corpo da página que está entre a linha indicada no FOOTING e a última linha do corpo;

7. a cláusula LINAGE cria um contador de linhas. Este contador informa a posição atual da linha dentro de uma página.

   Na execução da instrução WRITE , o contador é incrementado automaticamente:

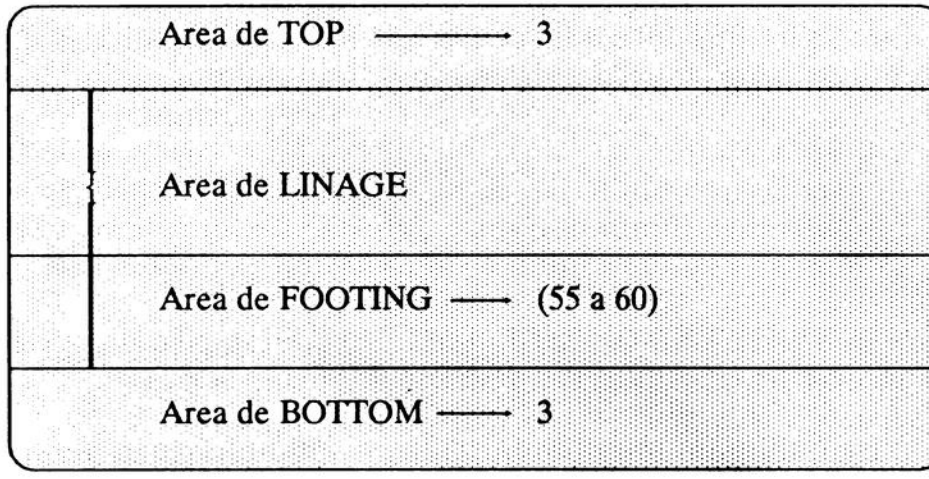
   - ADVANCING PAGE da instrução WRITE e OVERFLOW, o contador de página é colocado para 1;

- ADVANCING operando LINE(S), o contador é incrementado com o valor especificado no ADVANCING;

- quando não for especificado ADVANCING o contador é incrementado de 1.

Exemplo:

```
FD IMPRESSAO    LABEL RECORD OMITTED
                LINAGE 60    FOOTING 55
                TOP 3        BOTTOM 3.
```




## INSTRUÇÃO WRITE ( PARA IMPRESSÃO DE RELATÓRIOS)

Formato:

**WRITE** nome-do-registro **[FROM** nome-de-dado-1]
[{**AFTER/BEFORE**} ADVANCING {operando LINE(S)/PAGE} [AT
{**END-OF-PAGE/EOP**} Instrução-Incondicional].

Função: Grava (imprime) o registro no arquivo (relatório) que foi especificado com OUTPUT na instrução OPEN.

Regras Sintáticas:

1. com a opção FROM os dados do nome-de-dado-1 serão transferidos para a área do registro antes de serem enviados para a impressora;

2. a opção ADVANCING permite controlar o salto de linhas na impressora;

3. se for especificada a palavra AFTER, a impressão dos dados será feita após saltar o número de linhas indicadas no operando.

   Se for BEFORE a impressão será feita antes de saltar o número de linhas indicado no operando;

4. após a execução da instrução WRITE o contador de linhas verifica se ocorreu fim de página, através da comparação com o valor especificado na cláusula LINAGE. Se for positivo, o sistema passa a executar a instrução incondicional da cláusula EOP.

Se a frase **END-OF-PAGE** (ou EOP) for especificada, a cláusula **LINAGE** deve ser especificada na descrição do arquivo associado.

Uma condição **END-OF-PAGE** (ou EOP) é alcançada sempre que um comando WRITE com a frase END-OF-PAGE (ou EOP) provoca uma impressão ou espaçamento dentro da área do rodapé do corpo da página.

Isto ocorre quando tal comando WRITE faz o **LINAGE-COUNTER** (contador de linhas gerado pelo compilador) igual ou maior que o valor especificado pelo valor **FOOTING**, se especificado. Neste caso, depois de o comando WRITE ser executado, o comando imperativo da frase END-OF-PAGE (ou EOP) também será.

Uma condição de "OVERFLOW" é atingida sempre que um comando WRITE não pode ser totalmente acomodado dentro do corpo da página. Isto ocorre quando um comando WRITE faz com que o LINAGE-COUNTER exceda o valor especificado, como tamanho do corpo da página, na cláusula LINAGE. Neste caso, o registro é impresso antes ou depois

(dependendo da frase usada) que a impressora seja reposicionada na primeira linha da página lógica seguinte.

O comando imperativo na cláusula END-OF-PAGE (ou EOP), se especificado, é executado depois de o registro ser escrito e a impressora reposicionada.

Se não for especificado nenhum valor FOOTING, na cláusula LINAGE, ou se as condições de END-OF-PAGE (ou EOP) e "OVERFLOW" ocorrerem simultaneamente, então somente a condição de "OVERFLOW" será efetivada.

Exemplo:

```
FD IMPRESSAO    LABEL RECORD OMITTED
                LINAGE IS 55 LINES
                WITH FOOTING AT 50 LINES
                AT TOP 6 LINES
                AT BOTTOM 5 LINES.
01 LINHA        PIC X(132).
:
:
PROCEDURE DIVISION.
:
:
    WRITE LINHA FROM DETALHE BEFORE ADVANCING 2
    LINES AT EOP GO TO ROT-FOOTING.
:
:
CABECALHO.
    MOVE SPACES TO LINHA.
    WRITE LINHA BEFORE ADVANCING 1 LINE
    GO TO ROT-IMP.
ROT-FOOTING.
    MOVE SPACES TO LINHA.
    WRITE LINHA BEFORE ADVANCING 4 LINES.
    WRITE LINHA FROM RODAPE BEFORE ADVANCING 1 LINE.
    GO TO CABECALHO.
```

Ação: no instante em que o LINAGE encontra o valor 50, o fluxo do programa será desviado para ROT-FOOTING, detectado pela condição EOP.

# INSTRUÇÃO PERFORM

A instrução PERFORM permite desviar o programa para uma posição diferente daquela que normalmente executaria dentro da seqüência normal, executando uma ou mais vezes.

**Formato 1**

**PERFORM** nome-de-parágrafo.

Função:

1. executar todas as instruções no parágrafo;

2. transferir o controle para o próximo passo da seqüência que segue a declaração PERFORM.

Exemplo:

```
    PERFORM TOTAL-RTN.
    GO TO INICIO.
    :
    :
TOTAL-RTN.
    ADD VALOR1 TO TOTAL1.
    ADD VALOR2 TO TOTAL2.
    :
    :
```

**Formato 2**

**PERFORM** nome-de-parágrafo-1 THRU nome-de-parágrafo-n

Função: Todas as declarações que começam no nome-de-parágrafo-1 e terminam no parágrafo-n serão executadas e o controle transferido para a declaração que está depois do PERFORM.

Exemplo:

```
STEP1.      ADD A TO B.
            IF C > ZERO PERFORM STEP2 THRU STEP4
                        GO TO STEPFIM
            ELSE NEXT SENTENCE.
STEP2.      MULTIPLY C BY C.
STEP3.      MOVE B TO CAMPOA.
STEP4.      ADD 1 TO C.
STEP5.      GO TO STEP1.
STEP FIM.
:
:
```

**Formato 3**

**PERFORM** nome-de-parágrafo-1 **[THRU** nome-de-pagrafo-n**]**
{inteiro/nome-de-dado} **TIMES**

Função: Possibilita a execução de um parágrafo ou um conjunto de parágrafos várias vezes.

Regras Sintáticas:

1. o nome-de-dado deve ser numérico, inteiro e ser definido na DATA DIVISION.

2. se o nome-de-dado tiver conteúdo ZERO o(s) parágrafo(s) não será(ão) performado(s).

Exemplo 1:

```
            PERFORM ROT-1 6 TIMES.
            STOP RUN.
            :
            :
ROT-1.
            ADD X TO Y.
            MULTIPLY Y BY Z.
```

Exemplo 2:

```
77 CONTROLE    PIC 9          VALUE   ZERO.
:
:
PROCEDURE DIVISION.
:
:
          ACCEPT (10 10) CONTROLE.
          PERFORM ROT-1 CONTROLE TIMES.
          STOP RUN.
:
ROT-1.
          ADD X TO Y.
          MULTIPLY Y BY Z.
```

**Formato 4**

**PERFORM** nome-de-parágrafo-1 [**THRU** nome-de-parágrafo-n] **UNTIL** condição.

Função: Executa um parágrafo ou um conjunto de parágrafos até que a condição seja verdadeira.

Exemplo:

```
WORKING-STORAGE SECTION.
77  CONTROLE        PIC 9          VALUE   7.
:
:
PROCEDURE DIVISION.
:
:
     MOVE ZEROS TO TOTAL.
     PERFORM ROTINA-1 UNTIL CONTROLE = 0
     GO TO OUTRA-ROTINA.
ROTINA-1.
     ADD 1 TO TOTAL.
     SUBTRACT 1 FROM CONTROLE.
OUTRA-ROTINA.
:
:
```

**Formato 5**

**PERFORM** nome-de-parágrafo-1 **[THRU** nome-de-parágrafo-n]
**VARYING** nome-de-dado-1 **FROM** {nome-de-dado-2/inteiro-1}
**BY** {nome-de-dado-3/inteiro-2} **UNTIL** condição.

Este formato de PERFORM executa as seguintes funções:

1. o nome-de-dado-1 é utilizado como um contador ou indicador de looping;

2. inicializa este contador em nome-de-dado-2 ou inteiro-1;

3. incrementa o contador por inteiro-2 ou pelo conteúdo do nome-de-dado-3;

4. testa o indicador de looping para a condição especificada. Enquanto a condição não é satisfeita a declaração PERFORM é executada.

Exemplo:

Suponhamos que desejamos somar todos os números inteiros ímpares de 1 a 1001.

```
WORKING-STORAGE SECTION.
77  TOT-IMPAR          PIC 9(07)     VALUE  ZEROS.
77  IMPAR              PIC 9(04)     VALUE  ZEROS.
:
PROCEDURE DIVISION.
:
:
    PERFORM     ROT-IMPAR VARYING IMPAR FROM 1 BY 2 UNTIL
                IMPAR > 1001.
    DISPLAY (05 20) "SOMA DOS IMPARES = " TOT-IMPAR.
    STOP RUN.
ROT-IMPAR.
    ADD IMPAR TO TOT-IMPAR.
```

# EXERCÍCIOS

Exercício 1

## 1. Objetivo do Programa

Emitir um relatório a partir do arquivo line sequential de materiais denominado ARQMAT.

## 2. Procedimentos

1. ler o arquivo ARQMAT;
2. imprimir 25 linhas por página em espaço duplo;
3. imprimir cabeçalho em todas as páginas;
4. ao final do processo imprimir uma folha em separado dos totais de produtos, custo total e preço médio.

## 3. Record Layout

| CODIGO | NOME | QTDE ESTOQUE | PRECO UNIT. | PRECO TOT. |
|---|---|---|---|---|
| 9(05) | X(15) | 9(07) | 9(05)V99 | 9(06)V99 |

## 4. Gabarito de Impressão (Printer Layout) - 80 colunas

```
DATA 99/99/99        RELATORIO DE MATERIAL EM ESTOQUE   PAG. ZZ9

CODIGO   NOME      QTDE ESTOQUE    CUSTO UNITARIO    CUSTO TOTAL
9999     X---15---X  Z.ZZZ.ZZ9      ZZ.ZZ9,99        ZZZ.ZZ9,99
9999     X---15---X  Z.ZZZ.ZZ9      ZZ.ZZ9,99        ZZZ.ZZ9,99
```

## 5. Gabarito de Impressão da Última Página

```
DATA 99/99/99        RELATORIO DE MATERIAL EM ESTOQUE   PAG. ZZ9

QUANTIDADE DE PRODUTO          PRECO MEDIO        PRECO TOTAL
Z.ZZZ.ZZ9                      ZZZ.ZZ9,ZZ         Z.ZZZ.ZZ9,99
```

Exercício 2

## 1. Objetivo do Programa

Emitir um relatório a partir do arquivo line sequential da revendedora de automóveis BRASILCAR denominado ARQCAR.

## 2. Procedimentos

1. ler o arquivo ARQCAR;

2. imprimir 24 linhas por página em espaço duplo;

3. imprimir cabeçalho em todas as páginas;

4. ao final do processo, listar em folha separada o total de carros existentes na revendedora, conforme layout de impressão da última folha do relatório.

*Observação:* Os carros de fabricação anteriores a 1982 deverão ser acumulados no ano de 1982.

## 3. Record Layout

| CODIGO | MARCA | C O R | A N O | CHAPA | MODELO |
|---|---|---|---|---|---|
| 9(03) | X(15) | X(10) | 9(04) | X(06) | X(10) |

## 4. Gabarito de Impressão (Printer Layout)

```
DATA 99/99/99        BRASILCAR REVEDENDORA DE VEICULOS   PAG: ZZ9

 CODIGO    MARCA COR        ANO        MODELO

 999       XXXX  XXX        9999       XXXXXX

 999       XXXX  XXX        9999       XXXXXX
```

## 5. Gabarito de Impressão da Última Página

```
DATA 99/99/99        BRASILCAR REVEDENDORA DE VEICULOS   PAG: ZZ9

              RESUMO DE EXISTENCIA POR ANO

 -1982-    -1983-    -1984-    -1985-    -1986-    -1987-    -1988-

 Z.ZZ9     Z.ZZ9     Z.ZZ9     Z.ZZ9     Z.ZZ9     Z.ZZ9     Z.ZZ9
```

# CAPÍTULO 5

## PROGRAMA UTILIZANDO TABELA

### 1. Objetivo do Programa

Exibir no vídeo o mês por extenso.

### 2. Procedimentos

1. aceitar a data do sistema e exibi-la na tela;

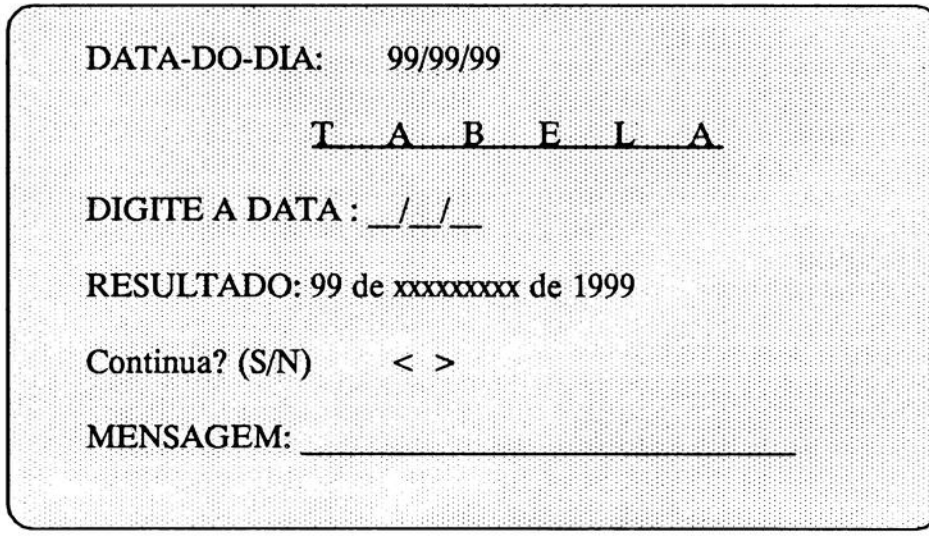
2. aceitar uma data qualquer;

3. exibir o mês digitado por extenso;

4. o mês por extenso será pesquisado em uma tabela;

5. condicionar fim de processamento.

## 3. Formato da Tela (Video Layout)




## Codificação do Programa

```
1   IDENTIFICATION      DIVISION.
2   PROGRAM-ID.         TABELA.
3   AUTHOR.             COLEGIO BRAISL.
4   ENVIRONMENT         DIVISION.
5   DATA                DIVISION.
6   WORKING-STORAGE     SECTION.
7   77 CONT             PIC X          VALUE  SPACE.
8   77 BRANCO           PIC X(30)      VALUE  SPACES.
9   01 MESES-DO-ANO.
10      02 F            PIC X(09)      VALUE  "JANEIRO".
11      02 F            PIC X(09)      VALUE  "FEVEREIRO".
12      02 F            PIC X(09)      VALUE  "MARCO".
13      02 F            PIC X(09)      VALUE  "ABRIL".
14      02 F            PIC X(09)      VALUE  "MAIO".
15      02 F            PIC X(09)      VALUE  "JUNHO".
16      02 F            PIC X(09)      VALUE  "JULHO".
17      02 F            PIC X(09)      VALUE  "AGOSTO".
18      02 F            PIC X(09)      VALUE  "SETEMBRO".
19      02 F            PIC X(09)      VALUE  "OUTUBRO".
20      02 F            PIC X(09)      VALUE  "NOVEMBRO".
```

```
21          02 F                PIC X(09)     VALUE   "DEZEMBRO".
22      01  TABELA-MESES REDEFINES MESES-DO-ANO.
23          02 MES-T            PIC X(09)     OCCURS      12 TIMES.
24      01  DATA-QUALQUER.
25          02 DIA              PIC 99        VALUE   ZEROS.
26          02 MES              PIC 99        VALUE   ZEROS.
27          02 ANO              PIC 99        VALUE   ZEROS.
28      01  DATA-DO-SISTEMA.
29          02 AA               PIC 99        VALUE   ZEROS.
30          02 MM               PIC 99        VALUE   ZEROS.
31          02 DD               PIC 99        VALUE   ZEROS.
32      SCREEN SECTION.
33      01 TELA.
34          02 BLANK SCREEN.
35          02 LINE 01 COLUMN 01 VALUE "DATA-DO-DIA:".
36          02 LINE 03 COLUMN 31     VALUE "T     A    B    E    L    A".
37          02 LINE 04 COLUMN 31     VALUE "——————————————————————".
38          02 LINE 05 COLUMN 21 VALUE
39          "Autor: Colegio Brasil - Nr. 000 - Serie: 2. Tu: Z".
40          02 LINE 09 COLUMN 31 VALUE "DIGITE A DATA:".
41          02 LINE 11 COLUMN 31 VALUE "RESULTADO:".
42          02 LINE 15 COLUMN 31 VALUE "Continua? (S/N) < >".
43          02 LINE 23 COLUMN 21 VALUE "MENSAGEM:".
44      PROCEDURE DIVISION.
45      INICIO.
46          ACCEPT DATA-DO-SISTEMA FROM DATE.
47          DISPLAY TELA.
48          DISPLAY (01 14) DD "/" MM "/" AA.
49      LACO-1.
50          DISPLAY (09 46) " / / ".
51          ACCEPT (09 46) DIA WITH PROMPT AUTO-SKIP.
52          DISPLAY (23 31) BRANCO.
53          IF DIA < 1 OR > 31
54              DISPLAY (23 31) "DIA INVALIDO"
55              GO TO LACO-1.
56      LACO-2.
57          ACCEPT (09 49) MES WITH PROMPT AUTO-SKIP.
58          DISPLAY (23 31) BRANCO.
59          IF MES < 1 OR > 12
60              DISPLAY (23 31) "MES INVALIDO"
61              GO TO LACO-2.
62      LACO-3.
63          ACCEPT (09 52) ANO WITH PROMPT AUTO-SKIP.
64          DISPLAY (23 31) BRANCO.
65          IF ANO = ZEROS
```

```
66              DISPLAY (23 31) "ANO INVALIDO"
67              GO TO LACO-3.
68  MOSTRA.
69          DISPLAY (11 42) DIA.
70          DISPLAY (11 46) "de"
71          DISPLAY (11 49) MES-T (MES).
72          DISPLAY (11 59) "de 19"
73          DISPLAY (11 64) ANO.
74  CONTINUA.
75          ACCEPT (15 48) CONT WITH UPDATE.
76          DISPLAY (21 31) BRANCO.
77          IF CONT = "S"
78              DISPLAY (11 42) BRANCO
79              GO TO LACO-1.
80          IF CONT = "N"
81              DISPLAY (01 01) ERASE
82              DISPLAY (23 31) "FIM DE PROGRAMA"
83              STOP RUN
84          ELSE
85              DISPLAY (23 31) "OPCAO INVALIDA"
86              GO TO CONTINUA.
87
```

## Explanação e Instruções Introduzidas no Programa

01 TABELA-MESES REDEFINES MESES-DO-ANO. a especificação REDEFINES (redefinir) permite dar mais de uma definição à mesma área. No programa, o item-de-grupo MESES-DO-ANO de 108 posições é redefinida com o nome de TABELA-MESES e recebe nova subdivisão. Desta maneira, os dois itens-de-grupo ocupam a mesma área de memória.

02 MES-T PIC X(09) OCCURS 12 TIMES. no COBOL, para indicar a ocorrência repetida de um item que tem o mesmo formato, usamos uma cláusula OCCURS.

No programa temos, antes da redefinição, 12 campos com 09 posições cada. Após a redefinição do item-de-grupo utilizamos a cláusula OCCURS que permite a indicação dos itens com uma simples entrada, no caso de nível 02.

Com uma cláusula OCCURS foi indicada a ocorrência repetida de itens. No caso MES-T define 12 itens, cada um com uma picture de X(09).

DISPLAY (11 45) MES-T(MES). Para se fazer referência a um campo definido na tabela, necessitamos de um indicador de posição e para isso utilizamos um subscritor. No programa, (MES) é o subscritor para referenciar a posição do campo na tabela. Por exemplo: se o campo MES tiver o valor 02 é o mesmo que escrever: MES-T(02), isto é, indicar a segunda ocorrência da tabela que no caso é o mês de FEVEREIRO.

## CLÁUSULA REDEFINES

Formato Genérico:

```
Número-de-nível nome-de-dado-1 REDEFINES nome-de-dado-2.
```

Sua função é indicar a redefinição de uma mesma área de memória por itens de dados distintos.

Regras Sintáticas:

1. a cláusula REDEFINES deve ser escrita logo após o nome-de-dado-1. Na definição do nome-de-dado-2 não é permitida a utilização da cláusula REDEFINES nem OCCURS;

2. os números de níveis do nome-de-dado-1 e nome-de-dado-2 devem ser iguais, porém diferentes de 88;

3. a cláusula REDEFINES não deve constar no nível 01 da FILE SECTION. Isto porque o fato de escrever vários níveis 01 dentro de uma mesma FD da FILE SECTION significa utilizar uma mesma área de memória. Portanto, obtém-se o mesmo efeito desta cláusula;

4. a redefinição prevalece até surgir um número de nível menor ou igual ao nome-de-dado-2;

5. o nome-de-dado-1 e os nomes-de-dados subordinados ao mesmo não devem conter a cláusula VALUE, exceto o nível 88;

6. o tamanho das áreas ocupadas pelo nome-de-dado-1 e nome-de-dado-2 deve ser o mesmo;

7. o item-de-dado (nome-de-dado-2) deve ser escrito logo após o item-de-dado (nome-de-dado-1) que indica a área da memória a ser redefinida;

8. um mesmo item pode ser redefinido quantas vezes for necessário.

Exemplo 1:

```
02 BANCOS.
    03 FILLER              PIC X(11)    VALUE  "AUGUSTA".
    03 FILLER              PIC X(11)    VALUE  "RODOVIARIA".
    03 FILLER              PIC X(11)    VALUE  "LUZ".
    03 FILLER              PIC X(11)    VALUE  "LIBERDADE".
    03 FILLER              PIC X(11)    VALUE  "V.CARRAO".
    03 FILLER              PIC X(11)    VALUE  "IBIRAPUERA".
02 NOMES-DE-AGENCIAS REDEFINES BANCOS.
    03 AGENCIA             PIC X(11)    OCCURS 6 TIMES.


02 TIPO-DE-ESTRUTURA      PIC 9(01).
02 ESTRUTURA-1.
    03 DATA-INICIO         PIC 9(06).
    03 DATA-FIM            PIC 9(06).
02 ESTRUTURA-2 REDEFINES ESTRUTURA-1.
    03 ANOS                PIC 9(02).
    03 MESES               PIC 9(02).
    03 DIAS                PIC 9(02).
    03 HORAS               PIC 9(02).
    03 MINUTOS             PIC 9(02).
    03 SEGUNDOS            PIC 9(02).
```

# CLÁUSULA OCCURS

Permite substancial redução de trabalho quando se trata de especificar itens que se repetem. Especifica o número de vezes que um dado item (de grupo ou elementar) se repete com o mesmo formato. As cláusulas associadas a um item cuja descrição inclua um OCCURS são extensivas a todas as repetições deste mesmo item. O nome-de-dado que contém a cláusula OCCURS deverá estar subscrito (ou indexado) toda vez que for referenciado na PROCEDURE DIVISION. Se este nome-de-dado corresponder a um item-de-grupo, então toda ocorrência de item subordinado a ele (na PROCEDURE DIVISION) deverá estar subscrita (ou indexada).

A cláusula OCCURS é proibida nos níveis 01 e 77.

Seu formato é:

Número-de-nível nome-de-dado **OCCURS** inteiro **TIMES**
[**INDEXED** BY nome-de-índice].

O subscrito permite o acesso a um item de uma tabela (ou matriz).

O uso da cláusula OCCURS na descrição de um item implica o uso do subscrito, ou índice, quando quisermos referenciar tal item.

Acerca do subscrito valem as afirmações:

- é um inteiro > 0 que indica a posição de um elemento dentro de uma tabela;
- o subscrito pode ser um literal ou um nome-de-dado cujo valor seja um inteiro positivo;
- o subscrito deve estar sempre entre parênteses e após o nome-de-dado que o usa.

No máximo três cláusulas OCCURS podem governar um item. Conseqüentemente são necessários um, dois ou três subscritores.

Exemplo:

```
WORKING-STORAGE SECTION.
77 SUB              PIC 9          VALUE   ZERO.
77 MOSTRA           PIC X(07)      VALUE   SPACES.
01 SEMANA.
    02 FILLER       PIC X(07)      VALUE   "SEGUNDA".
    02 FILLER       PIC X(07)      VALUE   "TERCA".
    02 FILLER       PIC X(07)      VALUE   "QUARTA".
    02 FILLER       PIC X(07)      VALUE   "QUINTA".
    02 FILLER       PIC X(07)      VALUE   "SEXTA".
    02 FILLER       PIC X(07)      VALUE   "SABADO".
    02 FILLER       PIC X(07)      VALUE   "DOMINGO".
01 SEMAN REDEFINES SEMANA.
    02 SEMA         PIC X(07)      OCCURS 7 TIMES.
:
PROCEDURE DIVISION.
:
    ADD 2 TO SUB.
    MOVE SEMA(SUB) TO MOSTRA.
    DISPLAY (10 10) MOSTRA.
```

Ação: Sendo 2 o valor do campo SUB, o item da tabela a ser movimentado será TERCA.

Para manipular o indexador dentro de um programa COBOL necessitamos da instrução SET.

## INSTRUÇÃO SET

**Formato 1**

**SET** {nome-de-índice-1/nome-de-dado-1} **TO** {nome-de-índice- 2/inteiro-1}.

**Formato 2**

**SET** nome-de-índice-3 {**UP BY/DOWN BY**}
{nome-de-índice-4/nome-de-dado-2/inteiro-2}.

Permite a manipulação do nome-de-índice para acessar dados da tabela criada pela cláusula OCCURS com a frase INDEXED BY.

Regras Sintáticas:

1. o formato 1 transfere o conteúdo do campo ou inteiro após o TO para o campo antes do TO;

2. o formato 2 é equivalente a redução (DOWN) ou incremento (UP) aplicados a cada um dos valores do nome-de-índice-3 especificado imediatamente após o verbo SET. A quantidade a ser reduzida ou incrementada é especificada na palavra escrita após a palavra BY;

3. todos os nomes-de-dados da instrução SET devem conter valores inteiros.

Exemplo:

```
WORKING-STORAGE SECTION.
77  RECEPTOR PIC 9(07) VALUE ZEROS.
01  TABELA.
    02 FILLER          PIC X(07)     VALUE  "1111111".
    02 FILLER          PIC X(07)     VALUE  "2222222".
    02 FILLER          PIC X(07)     VALUE  "3333333".
    02 FILLER          PIC X(07)     VALUE  "4444444".
01 TABEL REDEFINES TABELA.
    02 TAB             PIC 9(07)     OCCURS 4 TIMES INDEXED BY IND.
:
PROCEDURE DIVISION.
INICIO.
    SET IND TO 1.
    MOVE 5 TO LIN.
    PERFORM CONT 4 TIMES.
    STOP RUN.
```

```
CONT.
    MOVE TAB(IND) TO RECEPTOR.
    DISPLAY (LIN. 10) RECEPTOR.
    SET IND UP BY 1
    ADD 2 TO LIN.
```

Ação: SET IND TO 1: o conteúdo do indexador terá valor igual a 1;

SET IND UP BY 1: o conteúdo do indexado será incrementado de 1.

## INDEXAÇÃO RELATIVA

Podemos ter também uma referência relativa quando queremos comparar elementos de mesmo conjunto. Teremos então o seguinte formato:

```
nome + - constante-inteira
```

Exemplo:

```
MOVE TAB (IND + 1) TO RECEPTOR.
```

## COMO CARREGAR UMA TABELA

**Processo 1**

A tabela é pequena e vai ser descrita com dados no próprio programa:

```
01 SEMANA.
    02 FILLER      PIC X(07)    VALUE   "SEGUNDA".
    02 FILLER      PIC X(07)    VALUE   "TERCA".
    02 FILLER      PIC X(07)    VALUE   "QUARTA".
```

```
    02 FILLER          PIC X(07)     VALUE   "QUINTA".
    02 FILLER          PIC X(07)     VALUE   "SEXTA".
    02 FILLER          PIC X(07)     VALUE   "SABADO".
    02 FILLER          PIC X(07)     VALUE   "DOMINGO".
01  RSEMANA REDEFINES SEMANA.
    02 SEMA            PIC X(07)     OCCURS 7 TIMES.
```

**Processo 2**

A tabela será carregada a partir de um arquivo de entrada:

```
DATA    DIVISION.
FILE    SECTION.
FD      ARQUIVO     LABEL RECORD STANDARD
                    VALUE OF FILE-ID "ARQUIVO".
01      ENT-TABELA      PIC X(80).
WORKING-STORAGE SECTION.
77  SUB                 PIC 9(03)     VALUE   1.
01  TABELA.
    02 ELEM-TABELA      PIC X(80)     OCCURS 100 TIMES.
:
PROCEDURE DIVISION.
INICIO.
    OPEN INPUT ARQUIVO.

LER-CARREGAR.
    READ ARQUIVO AT END GO TO FECHAMENTO.
    MOVE ENT-TABELA TO ELEM-TABELA(SUB).
    ADD 1 TO SUB.
    GO TO LER-CARREGAR.
FECHAMENTO.
    CLOSE ARQUIVO.
```

Exemplo de Definição de Tabela:

```
01  TABELA-MES.
    02 F          PIC 99   VALUE   31.
    02 F          PIC 99   VALUE   28.
    02 F          PIC 99   VALUE   31.
    02 F          PIC 99   VALUE   30.
    02 F          PIC 99   VALUE   31.
    02 F          PIC 99   VALUE   30.
```

```
    02 F          PIC 99   VALUE   31.
    02 F          PIC 99   VALUE   31.
    02 F          PIC 99   VALUE   30.
    02 F          PIC 99   VALUE   31.
    02 F          PIC 99   VALUE   30.
    02 F          PIC 99   VALUE   31.
01 TABELA REDEFINES TABELA-MES.
    02 TAB        PIC 99   OCCURS 12 TIMES.
```

A mesma tabela poderia ser definida da forma abaixo:

```
01  TABELA-MES.
    02 FILLER     PIC X(24) VALUE
    "312831303130313130313031".
01  TABELA REDEFINES TABELA-MES.
    02 TAB        PIC 99   OCCURS 12 TIMES.
```

# EXERCÍCIOS

Exercício 1

## 1. Objetivo do Programa

Elaborar um programa no qual exige uma manipulação simples de tabela.

## 2. Procedimentos

1. exibir a tela formatada com a respectiva data do sistema;

2. entrar com a sigla do Estado ou Território;

3. quando for digitada uma sigla diferente à da tabela, exibir mensagem "ESTADO INEXISTENTE";

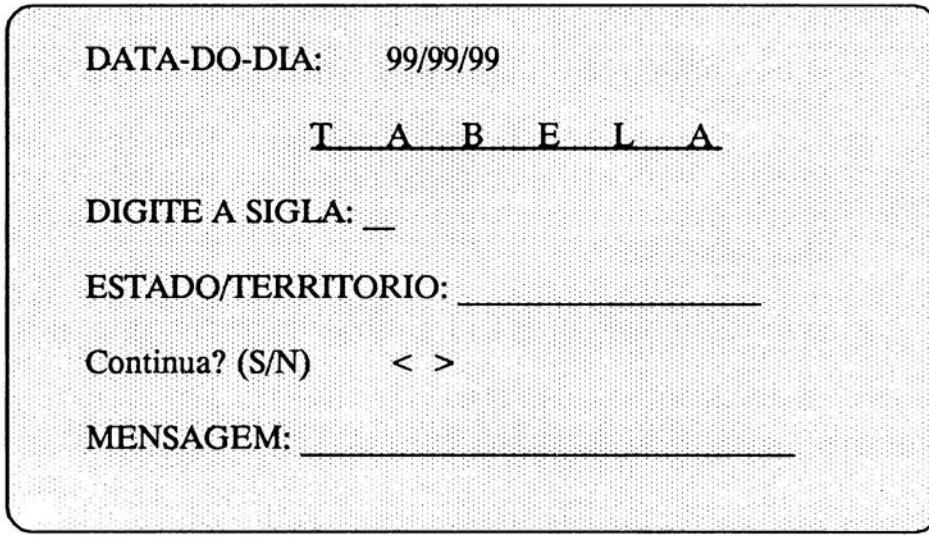
4. mostrar no vídeo o Estado ou Território correspondente;

5. condicionar final de processamento;

6. Estados e Territórios com as respectivas siglas:

| **Estado/Território** | **Sigla** |
|---|---|
| ACRE | AC |
| ALAGOAS | AL |
| AMAZONAS | AM |
| AMAPÁ | AP |
| BAHIA | BA |
| CEARÁ | CE |
| DISTRITO FEDERAL | DF |
| ESPÍRITO SANTO | ES |
| FERNANDO DE NORONHA | FN |
| GOIÁS | GO |
| MARANHÃO | MA |
| MINAS GERAIS | MG |
| MATO GROSSO DO SUL | MS |
| MATO GROSSO | MT |
| PARÁ | PA |
| PARAÍBA | PB |
| PERNAMBUCO | PE |
| PIAUÍ | PI |
| PARANÁ | PR |
| RIO DE JANEIRO | RJ |
| RIO GRANDE DO NORTE | RN |
| RONDÔNIA | RO |
| RORAIMA | RR |
| RIO GRANDE DO SUL | RS |
| SANTA CATARINA | SC |
| SERGIPE | SE |
| SÃO PAULO | SP |

## 3. Formato da Tela (Video Layout)




Exercício 2

## 1. Objetivo do Programa

Elaborar um programa para emitir listagem com pesquisa de tabela.

## 2. Procedimentos

1. aceitar a data do sistema;

2. ler o arquivo line sequential "TABELA", carregando-o na memória, até que seja fim de arquivo, num total de no máximo 20 registros;

3. ler o arquivo line sequential ARQMAR;

4. pesquisar o código do veículo na tabela carregada;

5. quando for encontrado o código correspondente do veículo, imprimir o registro conforme o gabarito de impressão;

6. se não houver código correspondente, o registro lido deverá ser desprezado;

7. antes da impressão do registro deverá ser calculado o valor do veículo que será o produto do campo valor da tabela com 75% a mais desse valor;

8. no final, imprimir totais de registros lidos, impressos e rejeitados.

## 3. Record Layout do Arquivo Tabela

| CODIGO | MODELO | VALOR |
|---|---|---|
| 9(03) | X(15) | 9(7)V99 |

## 4. Record Layout do Arquivo Arqmar

| MATRICULA | N O M E | CIDADE | ESTADO | CODIGO |
|---|---|---|---|---|
| 9(06) | X(30) | X(20) | XX | 99 |

## 5. Gabarito de Impressão (Printer Layout)

```
DATA: 99/99/99    RELACAO DE CLIENTES CONTEMPLADOS PAG: Z9

MATRI CULA  NOME          CIDADE        ESTADO  MODELO        PRECO

999999      X---30---X    X---20---X    XX      X---15---X    Z.ZZZ.ZZ9,99

999999      X---30---X    X---20---X    XX      X---15---X    Z.ZZZ.ZZ9,99


        - TOTAL DE CLIENTES CADASTRADOS.....> ZZ9
        - TOTAL DE REGISTROS IMPRESSOS......> ZZ9
        - TOTAL DE REGISTROS DESPREZADOS.....> ZZ9
```

# CAPÍTULO 6

## PROGRAMA DE MANUTENÇÃO DE ARQUIVO INDEXADO

### 1. Objetivo do Programa

Elaborar um programa que faça a manutenção do cadastro de clientes de uma loja.

### 2. Procedimentos

1. consistir todos os campos;

2. após a execução de qualquer função retornar ao menu.

### 3. Formato da Tela (Video Layout)

Video Layout (1)

```
EM 99/99/99     CADASTRO DE CLIENTES

          FUNÇÃO DESEJADA:    < >

          <1> INCLUSAO
          <2> ALTERACAO
          <3> EXCLUSAO
          <4> CONSULTA
          <5> FIM

   MENSAGEM:
```

Video Layout (2)

```
EM 99/99/99     CADASTRO DE CLIENTES

                FUNCAO: xxxxxxxxx

          0 - Codigo...:
          1 - Nome.....:
          2 - Endereco.:
          3 - Bairro...:
          4 - Cidade...:
          5 - C E P ...:

   MENSAGEM:
```

## 4. Record Layout

| CODIGO | NOME | ENDERECO | BAIRRO | CIDADE | CEP |
|---|---|---|---|---|---|
| 9(04) | X(30) | X(30) | X(20) | X(20) | 9(05) |

## Codificação do Programa

```
1   IDENTIFICATION  DIVISION.
2   PROGRAM-ID.     EXCLI.
3   AUTHOR.         COLEGIO BRASIL.
4   ENVIRONMENT     DIVISION.
5   CONFIGURATION   SECTION.
6   SPECIAL-NAMES.  DECIMAL-POINT IS COMMA.
7   INPUT-OUTPUT    SECTION.
8   FILE-CONTROL.
9       SELECT  ARQCLI ASSIGN TO    DISK
10      ORGANIZATION                INDEXED
11      ACCESS MODE                 DYNAMIC
12      RECORD KEY                  FD-CODIGO
13      FILE STATUS                 FS.
14  DATA DIVISION.
15  FILE SECTION.
16  FD  ARQCL I LABEL RECORD STANDARD
17          VALUE OF FILE-ID "ARQCLI".
18  01  REG-ARQCLI.
19      02 FD-CODIGO.
20          03 CODIGO            PIC 9(04).
21      02 FD-NOME               PIC X(30).
22      02 FD-END                PIC X(30).
23      02 FD-BAIRRO             PIC X(20).
24      02 FD-CIDADE             PIC X(20).
25      02 FD-CEP                PIC 9(05).
26   WORKING-STORAGE SECTION.
27  77  WS-SPACE                 PIC X(40)   VALUE   SPACES.
28  77  FS                       PIC X(02)   VALUE   SPACES.
29  77  WS-FUNC                  PIC 9       VALUE   ZERO.
30  77  MSG                      PIC X(09)   VALUE   SPACES.
31  77  WS-CONF                  PIC X       VALUE   SPACE.
32  01  WS-DATA-SIS.
33      02 AA                     PIC 9(02)   VALUE   ZEROS.
34      02 MM                     PIC 9(02)   VALUE   ZEROS.
35      02 DD                     PIC 9(02)   VALUE   ZEROS.
36  01  WS-MENSAGENS.
37      02 MENSA1                 PIC X(30)   VALUE "Funcao Errada - REDIGITE".
38      02 MENSA2                 PIC X(30)   VALUE "Campo Invalido".
39      02 MENSA3                 PIC X(30)   VALUE "Cliente Ja Cadastrado".
40      02 MENSA4                 PIC X(30)   VALUE "Cliente Nao Cadastrado".
41  SCREEN SECTION.
42  01  TELA1.
```

```
43          02 BLANK SCREEN.
44          02 LINE 01 COLUMN 01 VALUE "EM:".
45          02 LINE 01 COLUMN 26 VALUE " CADASTRO DE CLIENTES "
46              REVERSE-VIDEO.
47          02 LINE 03 COLUMN 19 VALUE
48          "Autor: Colegio Brasil - Nr: 000 - Serie: 2 - Turma: Z".
49          02 LINE 06 COLUMN 29 VALUE "Funcao Desejada: < >".
50          02 LINE 08 COLUMN 29 VALUE " <1> INCLUSAO".
51          02 LINE 10 COLUMN 29 VALUE " <2> ALTERACAO".
52          02 LINE 12 COLUMN 29 VALUE " <3> EXCLUSAO".
53          02 LINE 14 COLUMN 29 VALUE " <4> CONSULTA".
54          02 LINE 16 COLUMN 29 VALUE " <5> FIM".
55          02 LINE 21 COLUMN 29 VALUE "MENSAGEM:".
56      01  TELA2.
57          02 LINE 06 COLUMN 31 VALUE "FUNCAO".
58          02 LINE 08 COLUMN 19 VALUE "0 - Codigo...:".
59          02 LINE 10 COLUMN 19 VALUE "1 - Nome.....:".
60          02 LINE 12 COLUMN 19 VALUE "2 - Endereco.:".
61          02 LINE 14 COLUMN 19 VALUE "3 - Bairro...:".
62          02 LINE 16 COLUMN 19 VALUE "4 - Cidade...:".
63          02 LINE 18 COLUMN 19 VALUE "5 - C E P ...:".
64          02 LINE 21 COLUMN 19 VALUE "MENSAGEM:".
65      01  LIMPATELA.
66          02 LINE 08 COLUMN 34 BLANK LINE.
67          02 LINE 10 COLUMN 34 BLANK LINE.
68          02 LINE 12 COLUMN 34 BLANK LINE.
69          02 LINE 14 COLUMN 34 BLANK LINE.
70          02 LINE 16 COLUMN 34 BLANK LINE.
71          02 LINE 18 COLUMN 34 BLANK LINE.
72      01  TELA-OPCAO.
73          02 LINE 06 COLUMN 39 PIC X(09) USING MSG REVERSE-VIDEO.
74      PROCEDURE DIVISION.
75      *----
76      INICIO.
77          DISPLAY (01 01) ERASE.
78          OPEN I-O ARQCLI.
79          IF FS NOT = "00"
80              IF FS = "30"
81                  CLOSE ARQCLI OPEN OUTPUT ARQCLI CLOSE ARQCLI
82                  GO TO INICIO
83              ELSE
84                  DISPLAY (24 35) "File Status --->" FS
85                  STOP RUN
86              ELSE
87                  NEXT SENTENCE.
```

```
88              ACCEPT WS-DATA-SIS FROM DATE.
89      TELA.
90          DISPLAY TELA1.
91          DISPLAY (01 04) DD "/" MM "/" AA.
92      ESCOLHA.
93          ACCEPT (06 47) WS-FUNC WITH AUTO-SKIP.
94          DISPLAY (21 29) WS-SPACE.
95          IF WS-FUNC = 0 OR > 5
96              DISPLAY (21 24) MENSA1
97              GO TO ESCOLHA.
98          GO TO INCLUSAO ALTERACAO EXCLUSAO CONSULTA FIM
99              DEPENDING ON WS-FUNC.
100     *——     Inicio dos procedimentos de inclusao.
101     INCLUSAO.
102         DISPLAY (05 01) ERASE.
103         DISPLAY TELA2.
104     INCLUIR.
105         MOVE "INCLUSAO" TO MSG.
106         DISPLAY TELA-OPCAO.
107         ACCEPT (08 34) CODIGO WITH PROMPT.
108         DISPLAY (21 29) WS-SPACE.
109         IF CODIGO = ZEROS
110             DISPLAY (21 29) MENSA2
111             GO TO INCLUIR.
112         READ ARQCLI INVALID KEY GO TO ROT-NOME.
113         DISPLAY (21 29) MENSA3 GO TO INCLUIR.
114     ROT-NOME.
115         ACCEPT (10 34) FD-NOME WITH PROMPT.
116         DISPLAY (21 29) WS-SPACE.
117         IF FD-NOME = SPACES
118             DISPLAY (21 29) MENSA2 GO TO ROT-NOME.
119     ROT-END.
120         ACCEPT (12 34) FD-END WITH PROMPT.
121         DISPLAY (21 29) WS-SPACE.
122         IF FD-END = SPACES
123             DISPLAY (21 29) MENSA2 GO TO ROT-END.
124     ROT-BAIRRO.
125         ACCEPT (14 34) FD-BAIRRO WITH PROMPT.
126         DISPLAY (21 29) WS-SPACE.
127         IF FD-BAIRRO = SPACES
128             DISPLAY (21 29) MENSA2 GO TO ROT-BAIRRO.
129     ROT-CIDADE.
130         ACCEPT (16 34) FD-CIDADE WITH PROMPT.
131         DISPLAY (21 29) WS-SPACE.
132         IF FD-CIDADE = SPACES
```

```
133             DISPLAY (21 29) MENSA2 GO TO ROT-CIDADE.
134 ROT-CEP.
135         ACCEPT (18 34) FD-CEP WITH PROMPT.
136         DISPLAY (21 29) WS-SPACE.
137         IF FD-CEP = ZEROS
138             DISPLAY (21 29) MENSA2 GO TO ROT-CEP.
139 ROT-GRAVA.
140         WRITE REG-ARQCLI INVALID KEY
141         DISPLAY (21 29) "Erro de Gravacao - FS = " FS
142         STOP RUN.
143 ROT-RETORNO.
144         DISPLAY (21 29) WS-SPACE.
145         DISPLAY (21 29) "Continua Inclusao? < > ".
146         ACCEPT (21 49) WS-CONF WITH AUTO-SKIP.
147         IF WS-CONF = "S" OR "s"
148             DISPLAY LIMPATELA
149             GO TO INCLUIR.
150         GO TO TELA.
151 *---      Inicio dos procedimentos de alteracao.
152 ALTERACAO.
153         PERFORM INCLUSAO.
154         MOVE "ALTERACAO" TO MSG.
155         DISPLAY TELA-OPCAO.
156 ALTERAR.
157         ACCEPT (08 34) CODIGO WITH PROMPT.
158         DISPLAY (21 29) WS-SPACE.
159         IF CODIGO = ZEROS
160             DISPLAY (21 29) MENSA2 GO TO ALTERAR.
161         READ ARQCLI INVALID KEY
162             DISPLAY (21 29) MENSA4 GO TO ALTERAR.
163         PERFORM MOSTRA.
164 ALTERA.
165         DISPLAY (21 29) WS-SPACE.
166         DISPLAY (21 29) "Digite o n. do campo p/ alterar < >".
167         ACCEPT (21 62) WS-FUNC WITH AUTO-SKIP.
168         IF WS-FUNC = 0 OR > 5
169             GO TO ALTERA.
170         GO TO CPO-NOME CPO-END CPO-BAIRRO CPO-CIDADE CPO-CEP
171             DEPENDING ON WS-FUNC.
172 CPO-NOME.
173         PERFORM ROT-NOME.
174         GO TO ROT-REGRAVA.
175 CPO-END.
176         PERFORM ROT-END.
177         GO TO ROT-REGRAVA.
```

```
178 CPO-BAIRRO.
179     PERFORM ROT-BAIRRO.
180     GO TO ROT-REGRAVA.
181 CPO-CIDADE.
182     PERFORM ROT-CIDADE.
183     GO TO ROT-REGRAVA.
184 CPO-CEP.
185     PERFORM ROT-CEP.
186 ROT-REGRAVA.
187     DISPLAY (21 29) WS-SPACE.
188     DISPLAY (21 29) "Continua alterando mesmo registro? < >".
189     ACCEPT (21 65) WS-CONF WITH AUTO-SKIP.
190     IF WS-CONF = "S" OR "s"
191         GO TO ALTERA.
192     IF WS-CONF = "N" OR "n"
193         NEXT SENTENCE
194     ELSE
195         GO TO ROT-REGRAVA.
196     REWRITE REG-ARQCLI INVALID KEY
197         DISPLAY (21 29) WS-SPACE
198         DISPLAY (21 29) "Erro de Regravacao - FS = " FS
199         STOP RUN.
200     DISPLAY (21 29) WS-SPACE.
201     DISPLAY (21 29) "Continua Alterando? < >".
202     ACCEPT (21 50) WS-CONF WITH AUTO-SKIP.
203     IF WS-CONF = "S" OR "s"
204         DISPLAY LIMPATELA GO TO ALTERAR.
205     GO TO TELA.
206 *---    Inicio dos procedimentos para exclusao de registro.
207 EXCLUSAO.
208     PERFORM INCLUSAO.
209     MOVE "EXCLUSAO" TO MSG.
210     DISPLAY TELA-OPCAO.
211 EXCLUIR.
212     PERFORM ALTERAR.
213     DISPLAY (21 29) WS-SPACE.
214     DISPLAY (21 29) "Confirma Exclusao? < >".
215     ACCEPT (21 49) WS-CONF WITH AUTO-SKIP.
216     IF WS-CONF = "S" OR "s"
217         NEXT SENTENCE
218     ELSE
219         DISPLAY LIMPATELA GO TO EXCLUIR.
220     DELETE ARQCLI INVALID KEY
221         DISPLAY (21 29) WS-SPACE
222         DISPLAY (21 29) "Erro de Exclusao - FS = " FS
```

```
223             STOP RUN.
224         DISPLAY (21 29) WS-SPACE.
225         DISPLAY (21 29) "Continua Exclusao? < >".
226         ACCEPT (21 49) WS-CONF WITH AUTO-SKIP.
227         IF WS-CONF = "S" OR "s"
228             DISPLAY LIMPATELA GO TO EXCLUIR.
229         GO TO TELA.
230 *---->    Inicio dos procedimentos para consulta.
231  CONSULTA.
232         PERFORM INCLUSAO.
233         MOVE "CONSULTA" TO MSG.
234         DISPLAY TELA-OPCAO.
235  CONSULTAR.
236         PERFORM ALTERAR.
237         DISPLAY (21 29) WS-SPACE.
238         DISPLAY (21 29) "Continua Consulta? < >".
239         ACCEPT (21 49) WS-CONF WITH AUTO-SKIP.
240         IF WS-CONF = "S" OR "s"
241             DISPLAY LIMPATELA GO TO CONSULTAR.
242         GO TO TELA.
243 *---->    Rotina de exibicao de registro na tela.
244  MOSTRA.
245         DISPLAY (10 34) FD-NOME.
246         DISPLAY (12 34) FD-END.
247         DISPLAY (14 34) FD-BAIRRO.
248         DISPLAY (16 34) FD-CIDADE.
249         DISPLAY (18 34) FD-CEP.
250 *---->    Rotina de encerramento normal de programa.
251  FIM.
252         DISPLAY (01 01) ERASE.
253         DISPLAY (21 29) "FIM DE PROCESSAMENTO".
254         CLOSE ARQCLI.
255         STOP RUN.
256
257
```

## Explanação e Instruções Introduzidas no Programa

```
SELECT  ARQCLI ASSIGN TO DISK
        ORGANIZATION INDEXED
        ACCESS MODE DYNAMIC
        RECORD KEY FD-CODIGO
        FILE STATUS FS.
```

ORGANIZATION INDEXED. A organização é seqüencial indexada. Antes do acesso a cada registro, o programa consulta uma tabela de índices associada ao arquivo.

ACCESS MODE DYNAMIC. Arquivo de disco pode ser processado tanto seqüencialmente como randomicamente (aleatoriamente).

RECORD KEY FD-CODIGO. FD-CODIGO é um campo chaveado dentro do arquivo.

READ ARQCLI INVALID KEY GO TO ROT-NOME. Este comando torna disponível, para processamento, o registro lógico de acordo com o valor da chave especificada na cláusula RECORD KEY. Se não for encontrado o registro especificado, o comando após a cláusula INVALID KEY será executado.

WRITE REG-ARQCLI INVALID KEY DISPLAY (21 29) "ERRO DE GRAVACAO - FS = "FS" STOP RUN. Este comando grava o registro no arquivo. Caso não seja executada com sucesso esta operação, então as instruções após a cláusula INVALID KEY serão acionadas.

REWRITE REG-ARQCLI INVALID KEY DISPLAY (21 29) WS-SPACE DISPLAY (21 29) "ERRO DE REGRAVACAO - FS = " FS STOP RUN. Altera valores nos campos do registro. Assim como na instrução anterior a cláusula INVALID KEY será acionada se a regravação não for bem-sucedida.

DELETE ARQCLI INVALID KEY DISPLAY (21 29) WS-SPACE DISPLAY (21 29) "ERRO DE EXCLUSAO - FS = " FS STOP RUN. Apaga o registro do arquivo indexado.

OPEN I-O ARQCLI. Abre o arquivo ARQCLI no modo INPUT e OUTPUT, possibilitando dessa forma a leitura e a gravação de registros.

# PROCESSAMENTO DE ARQUIVOS DE ORGANIZAÇÃO INDEXADA

A organização indexada é capaz de recuperar/gravar registros de um arquivo de dados em disco, a partir de um diretório de ponteiros, chamado índice de controle. Tais ponteiros permitem uma localização direta de registros que tenham valores únicos de uma determinada chave.

O acesso pode ser seqüencial, aleatório e dinâmico.

Seqüencial: o acesso é feito em ordem crescente de valores de RECORD KEY;

Aleatório: a ordem de acesso aos registros é controlada pelo programador. A mecânica consiste em colocar o valor de RECORD KEY desejado e depois realizar o acesso;

Dinâmico: a lógica do programa pode alterar o modo de acesso de seqüencial para aleatório e vice-versa, tantas vezes quantas for conveniente.

**Considerações de Sintaxe (ENVIRONMENT)**

A cláusula SELECT deve especificar ORGANIZATION IS INDEXED, e o formato da cláusula ACCESS é:

```
ACCESS MODE IS {SEQUENTIAL/RANDOM/DYNAMIC}.
```

As cláusulas ASSIGN e FILE STATUS são idênticas ao descrito para organização seqüencial.

A cláusula RECORD KEY tem o formato:

```
RECORD KEY IS nome-de-dado,
```

onde nome-de-dado é um item contido na descrição do registro do arquivo em questão e é alfanumérico. O tamanho máximo da chave é de 60 bytes e o seu valor nunca poderá ser zero.

Se o modo de acesso especificado for RANDOM (aleatório) o valor do nome-de-dado indica o registro envolvido no próximo DELETE, READ, REWRITE ou WRITE. Cada registro deve ter um valor de RECORD KEY único (diferente do valor de todos os outros registros).

A cláusula ALTERNATE RECORD KEY tem o formato: ALTERNATE RECORD KEY IS nome-de-dado-1 WITH DUPLICATES, onde nome-de-dado-1 é um item contido na descrição do registro do arquivo em questão.

Esta cláusula indica que registros do arquivo indexado podem ser localizados através de chaves alternativas num total de onze, além, é claro, da chave principal.

A declaração WITH DUPLICATES especifica que mesmo havendo chave alternativa repetida a pesquisa será realizada.

Exemplo de descrição:

```
SELECT  ARQCLI ASSIGN TO DISK
        ORGANIZATION INDEXED
        ACCESS MODE DYNAMIC
        RECORD KEY FD-CODIGO
        ALTERNATE RECORD KEY FD-NOME WITH DUPLICATES
        ALTERNATE RECORD KEY FD-CEP WITH DUPLICATES
        FILE  STATUS FS.
```

## INSTRUÇÃO READ

Dois formatos são possíveis:

**Formato 1**

**READ** nome-de arquivo **NEXT** RECORD [INTO nome-de arquivo-1]
[AT **END** comando-imperativo].

**Formato 2**

**READ** nome-de-arquivo [**KEY IS** nome-de-dado-1] RECORD
[**INTO** nome-de-dado-2] [**INVALID KEY** comando-imperativo].

O formato 1 com a opção NEXT é usado para leitura seqüencial num arquivo com modo de acesso DYNAMIC. A cláusula AT END será executada quando for atingida a condição de fim lógico do arquivo. Se o programa não contiver esta cláusula deverá contar com a declaração de FILE STATUS para este arquivo.

No formato 2 INVALID KEY especifica a ação a ser tomada, no caso de RECORD KEY não se igualar à chave existente no arquivo.

As regras para arquivos seqüenciais com respeito à frase INTO também se aplicam para arquivos indexados.

A opção KEY IS nome-de-dado-1 é utilizada para o caso em que existe chave (ALTERNATE RECORD KEY IS) alternativa, possibilitando, dessa forma, pesquisa de um registro do arquivo por mais de uma chave (RECORD KEY).

## INSTRUÇÃO WRITE

Escreve um registro lógico num arquivo aberto como OUTPUT ou como I-O.

O seu formato é:

**WRITE** nome-de-registro [**FROM** nome-de-dado-1]
[**INVALID KEY** comando-imperativo].

No momento da execução deste comando um valor único deve estar no campo definido como RECORD KEY em nome-de-registro.

No caso de um valor RECORD KEY inválido as operações que seguem o INVALID KEY serão executadas. Se a cláusula INVALID KEY não for especificada o controle será passado à declaração FILE STATUS para este arquivo.

Uma situação de RECORD KEY inválida é atingida se:

1. para acesso seqüencial as chaves não estiverem em ordem ascendente em dois WRITE sucessivos;

2. o valor da chave não for único:

3. o espaço alocado em disco acabar.

## INSTRUÇÃO REWRITE

Substitui um registro lógico existente. O seu formato é:

**REWRITE** nome-de-registro [**FROM** nome-de-dado]
[**INVALID KEY** comando-imperativo...].

Para que um REWRITE seja válido é mister que o último READ no mesmo arquivo tenha sido bem-sucedido, caso o modo de acesso seja seqüencial. Se o valor de RECORD KEY válido para este REWRITE não for o mesmo do último READ ou se o READ anterior terminou em erro, então a condição de chave inválida existe, e os comandos imperativos têm vez. Para arquivo em modo de acesso aleatório ou dinâmico, o registro a ser substituído é especificado pela chave; não é necessário nenhum comando READ anterior.

## INSTRUÇÃO DELETE

Remove um registro lógico do arquivo indexado. Seu formato é:

**DELETE** nome-de-arquivo RECORD [**INVALID KEY** comando-imperativo].

No caso de modo de acesso seqüencial o último READ deve ter sido bem-sucedido. Este será o registro deletado.

Para um arquivo com modo de acesso dinâmico ou aleatório, o registro deletado é aquele cuja chave seja igual a RECORD KEY. Se não existir tal registro, serão executados os comandos da cláusula INVALID KEY (se especificado).

# INSTRUÇÃO START

Este comando permite que se posicione um arquivo indexado para leitura a partir de um valor de chave especificado. Isto é permitido para arquivos nos modos de acesso seqüencial e dinâmico. Seu formato é:

**START** nome-de-arquivo **[KEY IS {GREATER/NOT LESS THAN/EQUAL TO}** nome-de-dado**] [INVALID KEY** comando-imperativo...**]**

Nome-de-dado deve ser a RECORD KEY declarada e o valor a ser procurado no arquivo deve ser armazenado previamente ali. A inclusão deste comando pressupõe que o arquivo tenha sido aberto como INPUT ou I-O.

Se a cláusula KEY não for especificada, um registro com a chave igual a RECORD KEY será procurado. Com a inclusão desta cláusula será procurado um registro tal que sua chave satisfaça a relação especificada. Se a condição estipulada não for satisfeita os comandos que seguem o INVALID KEY serão executados.

**Abertura de Arquivos Indexados**

Com exceção da cláusula EXTEND, a abertura dos arquivos indexados obedece às mesmas regras da abertura de arquivos já apresentados. Deve, no entanto, testar o FILE STATUS para saber se o comando foi bem-sucedido.

FILE STATUS para arquivo indexado:

| Códigos | Significado |
|---|---|
| 00 | operação completada sem erros |
| 10 | encontrou EOF (AT END) |
| 21 | erro na estrutura do arquivo |
| 22 | chave duplicada |
| 23 | o registro não se encontra no arquivo |
| 24 | não há área disponível no disco |
| 30 | erro permanente |
| 91 | erro na estrutura do arquivo |

Comentários:

1. a sinalização do FILE STATUS 21 decorre se num acesso seqüencial a gravação não obedece uma seqüência ascendente de chave, no arquivo indexado. Também ocorre se a chave foi alterada por uma instrução anterior à operação do REWRITE;

2. para a operação de OPEN INPUT e OPEN I-O, o FILE STATUS com código 30 significa que o arquivo não se encontra no disco;

3. a sinalização de FILE STATUS 91 ocorre na execução do OPEN INPUT ou OPEN I-O quando a estrutura do arquivo está destruída. No caso de OPEN INPUT, o arquivo permanece aberto e permite a execução da instrução READ. No caso de OPEN I-O o arquivo não é considerado como aberto e não permite nenhuma operação de entrada e saída;

4. a sinalização de FILE STATUS 21, 22, 23 e 24 é detectada também pela função INVALID KEY.

## EXERCÍCIOS

Exercício 1

### 1. Objetivo do Programa

Elaborar um programa que faça manutenção do arquivo indexado ARQMAT.

### 2. Procedimentos

1. o programa deverá possibilitar inclusão, alteração, exclusão e consulta de registros do arquivo ARQMAT;

2. consistir a validade dos campos quando se tratar de inclusão e alteração;

3. campo chave = CODIGO.

## 3. Formato da Tela (Video Layout)

```
DATA: 99/99/99          CONTROLE DE ESTOQUE

ESCOLHA A FUNCAO: __ (IN - AL - EX - CO - FF)

     CODIGO..................> 99999
     NOME....................> X--(30)--X
     QUANTIDADE ESTOQUE....> 9999
     VALOR UNITARIO.........> 9999999.99
     VALOR TOTAL.............> 99999999.99

MENSAGEM:
```

onde: IN = inclusão

AL = alteração

EX = exclusão

CO = consulta

FF = fim de processamento

## 4. Record Layout

| CODIGO | NOME | QTDE ESTOQUE | VALOR UNIT. | VALOR TOT. |
|---|---|---|---|---|
| 9(5) | X(30) | 9(4) | 9(7)V99 | 9(8)V99 |

Observação: CODIGO = RECORD KEY do registro.

Exercício 2

## 1. Objetivo do Programa

Elaborar um programa que faça a manutenção do arquivo indexado ARQPRO.

## 2. Procedimentos

1. código do produto é o campo chave;

2. calcular o DV (dígito de verificação) do código aplicando o módulo 10;

3. deverá possibilitar inclusão, alteração, exclusão, consulta e emissão de listagem;

4. para emissão de listagem:

   *a.* imprimir cabeçalho em todas as páginas;
   *b.* imprimir 25 linhas por página com espaço duplo;
   *c.* aceitar, via vídeo, o início e o fim dos registros que serão impressos (instrução START);
   *d.* valor-total = valor-unitário x quantidade-de-estoque.

## 3. Formato de Tela (Video Layout)

Formato da Tela (Video Layout) (1)

```
DATA: 99/99/99 CIA PRODUTOS DETERIORADOS S/A

    FUNCAO ESCOLHIDA:        < >

          <1>    INCLUSAO
          <2>    ALTERACAO
          <3>    EXCLUSAO
          <4>    CONSULTA
          <5>    LISTAGEM
          <6>    FIM

    MENSAGEM:
```

Formato da Tela (video Layout) (2)

```
EM 99/99/99          CONTROLE DE ESTOQUE

ESCOLHA A FUNCAO: __ (IN - AL - EX - CO - FF)

        CODIGO.....> 99999
        NOME.....> X--(30)--X
        QUANTIDADE ESTOQUE....> 9999
        VALOR UNITARIO.....> 9999999.99
        VALOR TOTAL.....> 99999999.99

MENSAGEM:
```

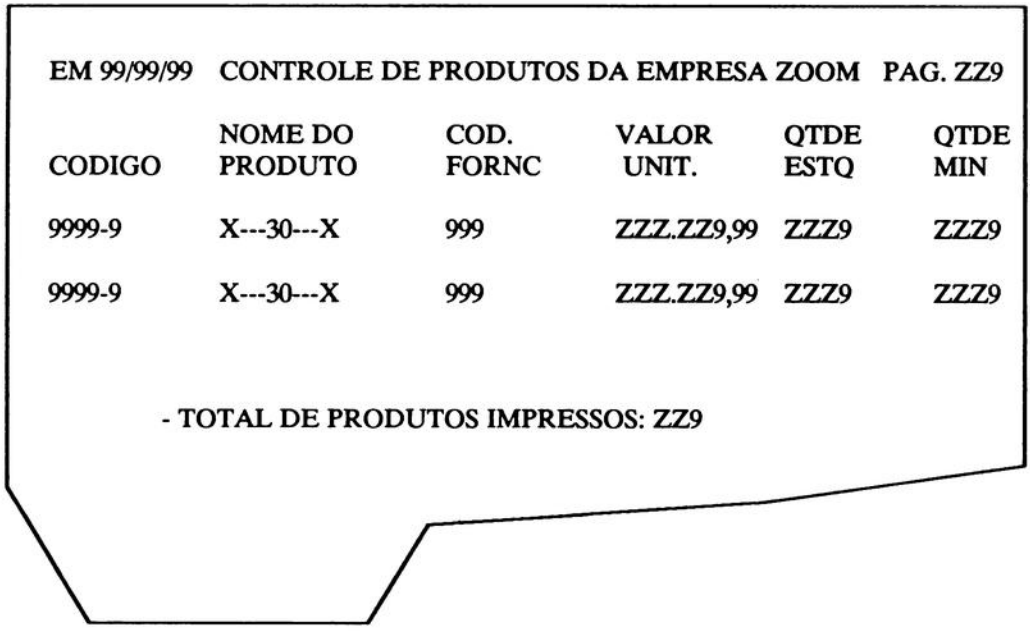
*Observação:* para a função de LISTAGEM a tela terá o seguinte formato:

```
DATA: 99/99/99       CIA PRODUTOS DETERIORADOS S/A

        FUNCAO: LISTAGEM

CODIGO DO REGISTRO INICIAL: _______
CODIGO DO REGISTRO FINAL..: _______

Contador de registros impressos: ___

MENSAGEM:
```

## 4. Record Layout

| CODIGO | | NOME DO PRODUTO | CODIGO FORNEC | VALOR UNIT. | QUANTIDADE ESTOQUE | QUANTIDADE MINIMA |
|---|---|---|---|---|---|---|
| NR | DV | | | | | |
| 9999 | 9 | X(30) | 999 | 9(6)V99 | 9999 | 9999 |

## 5. Gabarito de Impressão (Printer Layout)



EM 99/99/99 CONTROLE DE PRODUTOS DA EMPRESA ZOOM PAG. ZZ9

| CODIGO | NOME DO PRODUTO | COD. FORNC | VALOR UNIT. | QTDE ESTQ | QTDE MIN |
|---|---|---|---|---|---|
| 9999-9 | X---30---X | 999 | ZZZ.ZZ9,99 | ZZZ9 | ZZZ9 |
| 9999-9 | X---30---X | 999 | ZZZ.ZZ9,99 | ZZZ9 | ZZZ9 |

- TOTAL DE PRODUTOS IMPRESSOS: ZZ9

## DEFINIÇÃO DA ORGANIZAÇÃO DE ARQUIVOS RELATIVOS

A organização relativa é restrita a arquivos em disco. Os registros são diferenciados com base em um número de registro relativo que vai de 1 a 32.767. Ao contrário do caso de arquivo indexado, em que o campo identificador de chave ocupa uma parte do registro de dados, os números do registro relativo são conceituais e não são inseridos nos registros de dados.

Um arquivo de organização relativa pode ser acessado de modo seqüencial, dinâmico ou aleatório. No modo de acesso seqüencial, os registros são acessados em ordem ascendente dos números de registros.

No modo de acesso aleatório, a seqüência de acesso aos registros é controlada pelo programa, pela substituição do número em um item de chave relativa. No modo de acesso dinâmico, o programa pode ser de acesso aleatório ou seqüencial.

## Considerações sobre Sintaxe

A cláusula SELECT da ENVIRONMENT DIVISION deve especificar

**ORGANIZATION** IS **RELATIVE,**

e o formato da cláusula ACCESS é:

**ACCESS** MODE IS **SEQUENTIAL / RANDOM / DYNAMIC**

Os formatos das cláusula ASSIGN, RESERVE e FILE STATUS são idênticos àqueles usados para arquivos de organização seqüencial ou indexada. Os valores do Status 2, quando Status 1 é igual a 2 são:

Na especificação FD para um arquivo relativo, devem ser

2 ——→ para tentativa de WRITE de uma chave duplicada

3 ——→ para registro inexistente

4 ——→ para falta de espaço em disquete

necessariamente utilizadas as cláusulas LABEL RECORD STANDARD e VALUE OF FILE-ID.

## Cláusula RELATIVE KEY

Além das cláusulas usuais na entrada SELECT, é necessária uma cláusula da forma:

**RELATIVE** KEY IS nome-de-dado-1

para o modo de acesso aleatório ou dinâmico. É também necessária para o modo de acesso seqüencial, se um comando START existe para tal arquivo.

O nome-de-dado-1 deve ser descrito como um item inteiro binário sem sinal, não contido dentro de qualquer descrição de registro do arquivo em si. Seu valor deve ser positivo e diferente de zero.

## Comandos da Procedure Division para Arquivos Relativos

Na PROCEDURE DIVISION, os comandos OPEN, CLOSE, READ, WRITE, REWRITE, DELETE e START estão disponíveis, assim como para os arquivos cuja organização é indexada. Os formatos do comando OPEN e CLOSE para arquivo seqüencial são aplicáveis aos arquivos relativos, exceto para a frase EXTEND.

## INSTRUÇÃO READ

**Formato 1**

**READ** nome-de-arquivo [**NEXT**] RECORD [**INTO** nome-de-dado]
[AT **END** instrução-incondicional.....]

**Formato 2**

**READ** nome-de-arquivo RECORD [**INTO** nome-de-dado]
[**INVALID** KEY instrução-incondicional.....]

O formato 1 deve ser usado para todos os arquivos em modo de acesso seqüencial. A frase NEXT deve estar presente para se conseguir o acesso seqüencial se o modo de acesso declarado no arquivo é dinâmico. A cláusula AT END, se dada, é executada quando a condição lógica de fim de arquivo ocorre, se não for dada, a seção DECLARATIVES receberá o controle, se estiver presente.

O formato 2 é usado para se conseguir acesso aleatório com modo de acesso declarado aleatório ou dinâmico.

Se for definida uma chave relativa (na cláusula SELECT do arquivo), a execução de um comando READ no formato 1 atualizará o conteúdo do item RELATIVE KEY nome-de-dado-1, de forma que conterá o número do registro recuperado.

Para um formato 2 do READ, o registro recuperado é aquele cujo número de registro relativo pré-armazenado no item RELATIVE KEY. Se não existir tal registro, porém, ocorrerá a condição INVALID KEY, manipulada pela:

*a.* instrução-incondicional mencionada na parte de INVALID KEY do READ; ou

*b.* seção DECLARATIVES associada.

As regras para arquivos seqüenciais referentes à frase INTO também são aplicadas a arquivos relativos.

Exemplos:

**Formato 1**

```
READ ARQUIVO NEXT RECORD INTO WS-REGISTRO
AT END GO TO FIM-PROGRAMA.
```

**Formato 2**

```
READ ARQUIVO RECORD INTO WS-REGISTRO
INVALID KEY DISPLAY
"Registro não encontrado".
```

## INSTRUÇÃO WRITE

O formato do comando WRITE é o mesmo para arquivo relativo e arquivo indexado.

Formato:

```
WRITE nome-de-registro [FROM nome-de-dado]
[INVALID KEY instrução-incondicional.....]
```

Se o modo de acesso é seqüencial, então o término do comando WRITE faz com que o número de registro relativo do registro recém gravado seja colocado no item RELATIVE KEY.

Se o modo de acesso é aleatório ou dinâmico, então o usuário deve pré-atribuir o valor do item RELATIVE KEY a fim de designar para o registro de um número ordinal (relativo). A condição INVALID KEY ocorre se já existe um registro com o número ordinal especificado, ou se o espaço no disco for insuficiente.

Exemplo:

```
WRITE REGISTRO FROM WS-REGISTRO
INVALID KEY DISPLAY "Reg não encontrado".
```

## INSTRUÇÃO REWRITE

O formato do comando REWRITE é o mesmo para arquivo relativo e indexado.

Formato:

```
REWRITE nome-de-registro [FROM nome-de-dado]
[INVALID KEY comando-imperativo.....]
```

Para um arquivo de acesso seqüencial, a ação imediatamente anterior deve ter sido um READ satisfatório; o registro previamente disponível é substituído no arquivo com a execução do comando REWRITE. Se o READ anterior não foi satisfatório, um erro em tempo de execução ocorrerá. A cláusula INVALID KEY não é permitida para acesso seqüencial.

Para um arquivo com modo de acesso dinâmico ou aleatório, o registro substituído com a execução do REWRITE é aquele cujo número ordinal é pré-estabelecido no item RELATIVE KEY. Se não existir tal item, ocorre a condição de INVALID KEY.

Exemplo:

```
REWRITE REGISTRO FROM WS-REGISTRO
INVALID KEY PERFORM ROTINA-DE-ERRO.
```

## INSTRUÇÃO DELETE

O formato do comando DELETE é o mesmo para o arquivo relativo e arquivo indexado.

Formato:

```
DELETE nome-de-arquivo RECORD
[INVALID KEY instrução-incondicional.....]
```

Para um arquivo em modo de acesso seqüencial, a ação imediatamente anterior deve ter sido um comando READ satisfatório; o registro, então tornado disponível previamente, é logicamente removido do arquivo. Se o READ anterior não foi satisfatório, o erro em tempo de execução ocorrerá. A frase INVALID KEY não pode ser especificada para arquivos em modo de acesso seqüencial.

Para um arquivo no modo de acesso dinâmico ou aleatório, a ação de remoção pertence a qualquer registro que for designado pelo valor no item RELATIVE KEY. Se não existir tal registro numerado, ocorre a condição de INVALID KEY.

Exemplo:

```
DELETE ARQUIVO RECORD
INVALID KEY DISPLAY "Registro não encontrado".
```

## INSTRUÇÃO START

O formato do comando START é o mesmo para arquivo relativo e para arquivo indexado.

Formato:

**START** nome-de-arquivo **KEY** IS [{**GREATER** THAN / **NOT_LESS** THAN / **EQUAL TO**}] nome-de-dado-1 [**INVALID** KEY instrução-incondicional.....]

A execução deste comando especifica a posição inicial para a operação de leitura; somente é permitido para um arquivo cujo modo de acesso é definido como seqüencial ou dinâmico.

Nome-de-dado-1 pode ser somente aquele do item RELATIVE KEY previamente declarado, e o número de registro relativo deve ser armazenado nele antes do START ser executado. Quando se executa este comando, o arquivo associado deve estar aberto em modo INPUT ou I-O.

Se a frase KEY não está presente, a igualdade entre o registro no arquivo e a chave é procurada. Se a relação GREATER ou NOT LESS é especificada, o arquivo é posicionado para o próximo acesso no primeiro registro maior ou igual ao valor indicado na chave.

Se não houver nenhum registro relativo, as instruções incondicionais da cláusula INVALID KEY são executadas ou a Seção DECLARATIVES, se existir, serão executadas.

Exemplo:

```
START ARQUIVO KEY IS EQUAL TO CHAVE-RELATIVA
INVALID KEY DISPLAY "Chave não encontrada".
```

# CAPÍTULO 7


## PROGRAMA SORT

### 1. Objetivo do Programa

Emitir um relatório, classificado em ordem crescente de nome de funcionários.

### 2. Procedimentos

1. ler o arquivo FUNARQ;

2. desprezar todos os registros cujo campo número de dependentes seja igual a ZERO e campo salário menor do que Crz 5.000,00;

3. emitir um relatório de todos os demais registros, classificados por ordem crescente do campo nome-do-funcionário;

4. calcular o salário líquido obtido:

SALARIO-LIQUIDO = SALARIO - (INPS + IR);

5. imprimir 25 linhas de detalhe por página com espaço duplo;

6. imprimir cabeçalho em todas as páginas;

7. no final emitir total de registros lidos e impressos.

## 3. Record Layout

| CODIGO DO FUNC. | NOME DO FUNC. | SALARIO | SEXO | INPS | IR | NR DEP |
|---|---|---|---|---|---|---|
| 9(10) | X(30) | 906)V99 | X | 9(05)V99 | 9(5)V99 | 9 |

## 4. Gabarito de Impressão (Printer Layout)

```
EM 99/99/99 RELACAO DE FUNCIONARIOS COM DEPENDENTES  FLS: ZZ9

CODIGO     NOME          SALARIO    INSP       I.R.       DEP. LIQUIDO

9999999999 X---(30)---X ZZZ.ZZ9,99 ZZ.ZZ9,99  ZZ.ZZ9,99    9  ZZZ.ZZ9,99

9999999999 X---(30)---X ZZZ.ZZ9,99 ZZ.ZZ9,99  ZZ.ZZ9,99    9  ZZZ.ZZ9,99


     - TOTAL DE CADASTRADOS.....  : Z.ZZ9

     - TOTAL DE IMPRESSOS.......  : Z.ZZ9
```

# Codificação do Programa

```
1   IDENTIFICATION  DIVISION.
2   PROGRAM-ID.     SORPGM.
3   AUTHOR.         COLEGIO BRASIL.
4   ENVIRONMENT     DIVISION.
5   CONFIGURATION   SECTION.
6   SPECIAL-NAMES.  DECIMAL-POINT IS COMMA.
7   INPUT-OUTPUT    SECTION.
8   FILE-CONTROL.
9       SELECT FUNARQ ASSIGN TO DISK
10      ORGANIZATION LINE SEQUENTIAL
11      FILE STATUS IS STAT.
12      SELECT SORARQ ASSIGN TO DISK
13      SORT STATUS IS ST.
14      SELECT RELATO ASSIGN TO PRINTER.
15  DATA DIVISION.
16  FILE SECTION.
17  FD  FUNARQ LABEL RECORD STANDARD
18      VALUE OF FILE-ID IS "FUNARQ".
19  01  REG-FUNARQ.
20      02 CODIGO              PIC 9(10).
21      02 NOME                PIC X(30).
22      02 SALARIO             PIC 9(06)V99.
23      02 SEXO                PIC X.
24      02 INPS                PIC 9(05)V99.
25      02 IR                  PIC 9(05)V99.
26      02 NR-DEP              PIC 9.
27  SD  SORARQVALUE OF FILE-ID IS "SORARQ".
28  01  REG-SORARQ.
29      02 SOR-COD             PIC 9(10).
30      02 SOR-NOME            PIC X(30).
31      02 SOR-SAL             PIC 9(06)V99.
32      02 SOR-SEX             PIC X.
33      02 SOR-INPS            PIC 9(05)V99.
34      02 SOR-IR              PIC 9(05)V99.
35      02 SOR-DEP             PIC 9.
36  FD  RELATO  LABEL RECORD OMITTED
37              LINAGE 54 TOP 6 BOTTOM 6.
38  01  LINHA                 PIC X(132).
39  WORKING-STORAGE SECTION.
40  77  STAT                  PIC X(02)     VALUE  SPACES.
41  77  WS-LIQ                PIC 9(06)V99  VALUE  ZEROS.
42  77  WS-PAG                PIC 9(03)     VALUE  ZEROS.
```

```
43  77  WS-LIDOS          PIC 9(04)      VALUE  ZEROS.
44  77  WS-IMP            PIC 9(04)      VALUE  ZEROS.
45  77  ST                PIC X(02)      VALUE  SPACES.
46  01  WS-DATA.
47      02   WS-ANO        PIC 9(02)      VALUE  ZEROS.
48      02 WS-MES          PIC 9(02)      VALUE  ZEROS.
49       02 WS-DIA         PIC 9(02)      VALUE  ZEROS.
50  01  CAB1.
51      02 FILLER          PIC X(03)      VALUE  "EM".
52      02 DATA-CAB1.
53           03 DIA-CAB1   PIC 99/        VALUE  ZEROS.
54           03 MES-CAB1   PIC 99/        VALUE  ZEROS.
55           03 ANO-CAB1   PIC 99         VALUE  ZEROS.
56      02 FILLER          PIC X(29)      VALUE  SPACES.
57      02 FILLER          PIC X(75)      VALUE  "RELACAO DE FUNCIO
58  -   "NARIOS COM DEPENDENTES".
59      02 FILLER          PIC X(08)      VALUE  "PAGINA:".
60      02 PAG-CAB1        PIC ZZ9        VALUE  ZEROS.
61      02 FILLER          PIC X(07)      VALUE  SPACES.
62  01  CAB2.
63      02 FILLER          PIC X(05)      VALUE  SPACES.
64      02 FILLER          PIC X(25)      VALUE  "CODIGO".
65      02 FILLER          PIC X(35)      VALUE  "NOME DO FUNCIONARIO".
66      02 FILLER          PIC X(16)      VALUE  "SALARIO".
67      02 FILLER          PIC X(13)      VALUE  "INPS".
68      02 FILLER          PIC X(13)      VALUE  "I R".
69      02 FILLER          PIC X(09)      VALUE  "ND".
70      02 FILLER          PIC X(16)      VALUE  "LIQUIDO".
71  01  DETALHE.
72      02 FILLER          PIC X(03)          VALUE SPACES.
73      02 DET-COD         PIC 9(10)          VALUE ZEROS.
74      02 FILLER          PIC X(12)          VALUE SPACES.
75      02 DET-NOME        PIC X(30)          VALUE SPACES.
76      02 FILLER          PIC X(10)          VALUE SPACES.
77      02 DET-SAL         PIC ZZZ.ZZ9,99     VALUE ZEROS.
78      02 FILLER          PIC X(04)          VALUE SPACES.
79      02 DET-INPS        PIC ZZ.ZZ9,99      VALUE ZEROS.
80      02 FILLER          PIC X(05)          VALUE SPACES.
81      02 DET-IR          PIC ZZ.ZZ9,99      VALUE ZEROS.
82      02 FILLER          PIC X(05)          VALUE SPACES.
83      02 DET-ND          PIC 9(02)          VALUE ZEROS.
84      02 FILLER          PIC X(06)          VALUE SPACES.
85      02 DET-LIQ         PIC ZZZ.ZZ9,99     VALUE ZEROS.
86      02 FILLER          PIC X(07)          VALUE SPACES.
87  01  TOT-1.
```

```
88          02 FILLER          PIC X(40)          VALUE   SPACES.
89          02 FILLER          PIC X(28)          VALUE   "- TOTAL DE CADAST
90    -     "RADOS.....>".
91          02 TOT-LIDOS       PIC Z.ZZ9          VALUE   ZEROS.
92          02 FILLER          PIC X(59)          VALUE   SPACES.
93    01    TOT-2.
94          02 FILLER          PIC X(40)          VALUE   SPACES.
95          02 FILLER          PIC X(28)          VALUE   "- TOTAL DE IMPRES
96    -     "SOS.....>".
97          02 TOT-IMP         PIC Z.ZZ9          VALUE   ZEROS.
98          02 FILLER          PIC X(59)          VALUE SPACES.
99    PROCEDURE DIVISION.
100   INICIO.
101   SORT SORARQ ASCENDING KEY SOR-NOME
102         INPUT PROCEDURE ENTRADA
103         OUTPUT PROCEDURE SAIDA.
104         STOP RUN.
105   ENTRADA SECTION.
106   INI-ENTRADA.
107         OPEN INPUT FUNARQ.
108         DISPLAY (01 01) ERASE.
109         IF STAT = "00" NEXT SENTENCE
110         ELSE
111         DISPLAY (23 10) "ERRO DE ABERTURA - STATUS = " STAT
112         CLOSE FUNARQ STOP RUN.
113   LER-ENTRADA.
114         READ FUNARQ AT END CLOSE FUNARQ
115         GO TO FIM-ENTRADA.
116         ADD 1 TO WS-LIDOS.
117         IF NR-DEP = ZERO AND SALARIO < 5000
118               GO TO LER-ENTRADA.
119               MOVE REG-FUNARQ TO REG-SORARQ.
120         RELEASE REG-SORARQ.
121*        IF ST = "00" NEXT SENTENCE
122*        ELSE DISPLAY (23 10) "ERRO NO RELEASE - STATUS = "
123*              ST CLOSE FUNARQ STOP RUN.
124               GO TO LER-ENTRADA.
125   FIM-ENTRADA.
126         EXIT.
127
128   SAIDA SECTION.
129   INI-SAIDA.
130         ACCEPT      WS-DATA FROM DATE.
131         MOVE        WS-DIA      TO  DIA-CAB1.
132         MOVE        WS-MES      TO  MES-CAB1.
```

```
133       MOVE        WS-ANO      TO  ANO-CAB1.
134       OPEN    OUTPUT RELATO.
135       PERFORM CAB-SAIDA.
136  LER-SAIDA.
137       RETURN SORARQ AT END GO TO FIM-SAIDA.
138       IF ST = "00"
139           NEXT SENTENCE
140       ELSE
141           DISPLAY (23 10) "ERRO NO RETURN - STATUS = " ST
142           STOP RUN.
143           COMPUTE WS-LIQ = SOR-SAL - (SOR-INPS + SOR-IR).
144           MOVE SOR-COD        TO DET-COD.
145           MOVE SOR-NOME       TO DET-NOME.
146           MOVE SOR-SAL        TO DET-SAL.
147           MOVE SOR-INPS       TO DET-INPS.
148           MOVE SOR-IR         TO DET-IR.
149           MOVE SOR-DEP        TO DET-ND.
150           MOVE WS-LIQ         TO DET-LIQ.
151           WRITE LINHA FROM DETALHE BEFORE ADVANCING 2
152               LINES AT EOP PERFORM CAB-SAIDA.
153           ADD1 TO WS-IMP.
154           GO TO LER-SAIDA.
155  CAB-SAIDA.
156       ADD 1 TO WS-PAG.
157       MOVE WS-PAG     TO PAG-CAB1.
158       WRITE LINHA FROM CAB1 BEFORE ADVANCING 2 LINES.
159       WRITE LINHA FROM CAB2 BEFORE ADVANCING 2 LINES.
160  FIM-SAIDA.
161       MOVE WS-LIDOS  TO TOT-LIDOS.
162       MOVE WS-IMP TO TOT-IMP.
163       MOVE SPACES     TO LINHA.
164       WRITE LINHA BEFORE ADVANCING 2 LINES.
165       WRITE LINHA FROM TOT-1 BEFORE ADVANCING 2 LINES.
166       WRITE LINHA FROM TOT-2 BEFORE ADVANCING 2 LINES.
167       CLOSE RELATO.
168       DISPLAY (23 10) "PROGRAMA ENCERRADO".
169  EXIT-SAIDA.
170       EXIT.
```

## Explanação e Instruções Introduzidas no Programa

Chamamos de SORT a operação de classificar (colocar em ordem) os registros de um arquivo segundo um critério: ordem crescente ou ordem decrescente.

SELECT SORARQ ASSIGN TO DISK
SORT STATUS IS ST. Especifica o arquivo de trabalho do SORT. O identificador ST do SORT STATUS está definido na WORKING-STORAGE SECTION.

SD SORARQ VALUE OF FILE-ID "SORARQ". SD são as iniciais de SORT DESCRIPTION = descrição da classificação, isto é, todo arquivo de classificação é definido como um arquivo comum; entretanto, em vez de FD escreve-se SD. Neste caso, não é permitido escrever a especificação LABEL RECORD IS .....

```
SORT SORARQ ASCENDING KEY SOR-NOME
INPUT PROCEDURE ENTRADA
OUTPUT PROCEDURE SAIDA.
```

SORT. É o comando que inicia a rotina de classificação.

ASCENDING KEY SOR-NOME. Indica que o arquivo de entrada será classificado pelo campo SOR-NOME em ordem crescente (ASCENDING).

INPUT PROCEDURE ENTRADA. ENTRADA é o nome da seção de entrada do programa de classificação. Esta rotina deve ser escrita pelo programador, pois é necessário fazer uma verificação antes da classificação, tais como: abertura, leitura, transferência e encerramento dos arquivos.

OUTPUT PROCEDURE SAIDA. SAIDA é o nome da seção do programa de classificação. Também, esta rotina deve ser escrita pelo programador, pois nesta seqüência devem ser dados alguns comandos antes da saída dos registros.

ENTRADA SECTION. Início da seção de entrada do programa de classificação.

RELEASE REG-SORARQ. O comando RELEASE (liberar) coloca o registro de entrada à disposição do arquivo de classificação. O nome escrito depois da palavra RELEASE é o nome-de-registro do arquivo de trabalho. Este comando só pode ser usado dentro de um procedimento INPUT PROCEDURE.

RETURN SORARQ AT END GO TO FIM-SAIDA. Terminado o processo de classificação, a cláusula RETURN (devolver) efetua a transferência de um registro do arquivo de trabalho para a memória principal. O nome escrito depois da palavra RETURN é o nome do arquivo de trabalho. Este comando só pode ser utilizado dentro de um procedimento OUTPUT PROCEDURE. O comando RETURN deve conter uma condição AT END.

## ELEMENTOS DO SORT

Para o uso do SORT, o programa em COBOL tem de prever informações na ENVIRONMENT, DATA e PROCEDURE DIVISION.

SELECT para o SORT: o seguinte formato deve ser usado para se definir o arquivo de SORT (arquivo de trabalho):

```
SELECT nome-de-arquivo ASSIGN TO DISK

SORT STATUS IS nome-de-dado.
```

FILE SECTION para o SORT: devemos descrever uma SD (SORT DESCRIPTION), com o formato abaixo:

```
SD nome-de-arquivo

    VALUE OF FILE-ID IS literal.
```

```
01 nome-do-registro.
    02 nome-de-dado-1 PIC.....
    02 nome-de-dado-2 PIC.....
```

Não é necessário escrever o LABEL do arquivo, pois o compilador proverá esta informação e acusará erro se a ela for escrita.

PROCEDURE DIVISION para o SORT: o comando SORT providencia todas as informações e controles para a classificação. Estas informações e controles agem diretamente nas operações do SORT. Estas operações estão divididas em duas fases:

1. obtêm e passam para o SORT os registros a serem classificados, através da INPUT PROCEDURE ou USING;

2. obtêm e passam para o programa em COBOL os registros da fase anterior, já classificados, através da OUTPUT PROCEDURE ou GIVING.

## INSTRUÇÃO SORT

```
SORT    nome-de-arquivo ON {ASCENDING/DESCENDING} KEY
        nome-de-dado-1..... ON {ASCENDING/DESCENDING}
        KEY nome-de-dado-2..... INPUT PROCEDURE IS
        nome-de-seção-1 THRU nome-de-seção-2 [USING
        nome-de-arquivo-2] OUTPUT PROCEDURE IS nome-
        de-seção-3 THRU nome-de-seção-4 [GIVING nome-de-arquivo-3].
```

nome-de-arquivo-1. É o nome do arquivo na SD e define os registros para o SORT.

ASCENDING e DESCENDING. São opções que especificam em qual ordem os registros serão classificados (ascendente ou descendente) respectivamente, baseado em uma ou mais chaves (KEY). Pelo menos um ASCENDING ou DESCENDING tem de ser especificado no comando SORT.

Os nome-de-dado-1, nome-de-dado-2.... definidos no parâmetro KEY são itens do registro definidos na SD.

A ordem de importância das chaves (KEY) definidas no comando SORT, são da esquerda para a direita dentro da sentença do comando SORT, não importando se ascendente ou descendente.

Nome-de-seção-1. É o nome da primeira seção da INPUT PROCEDURE.

Nome-de-seção-2. É o nome da última seção da INPUT PROCEDURE. É opcional e deve aparecer quando o procedimento terminar em uma seção diferente da que foi iniciada.

INPUT PROCEDURE. Indica ao SORT que serão processados os registros antes de passá-los para a fase de classificação. Estes procedimentos podem estar em uma ou mais seções (SECTION) da INPUT PROCEDURE.

A INPUT PROCEDURE consiste numa ou mais Seções, escritas consecutivamente e estas seções não podem fazer parte de outros procedimentos. Na INPUT PROCEDURE é obrigatório pelo menos um comando RELEASE.

O controle é passado para os comandos que compõem a INPUT PROCEDURE, somente pelo comando SORT no programa.

A saída da INPUT PROCEDURE será provida pelo compilador que irá prever todo o mecanismo de saída na última seção mencionada na INPUT PROCEDURE, e somente será ativada quando os procedimentos forem desviados para a última seção referenciada na INPUT PROCEDURE. Somente então os registros passados para o SORT serão classificados.

USING. Se a opção USING é usada, todos os registros do nome-de-arquivo-2 serão transferidos automaticamente para nome-de-arquivo-2. Em tempo de execução o comando SORT abre (OPEN) automaticamente o nome-de-arquivo-2.

A organização do nome-de-arquivo-2 tem de ser seqüencial. Com a opção USING o compilador irá prever todas as rotinas de OPEN, READ, RELEASE e CLOSE do nome-de-arquivo-2 sem que o programador especifique essas funções.

Nome-de-seção-3. É o nome da primeira seção da OUTPUT PROCEDURE.

Nome-de-seção-4. É o nome da última seção que compõe a OUTPUT PROCEDURE. E é somente necessária quando o procedimento termina em uma seção diferente da que foi iniciada.

OUTPUT PROCEDURE. Indica ao SORT que o programa irá processar os registros depois de classificados. Estes procedimentos podem estar em uma ou mais seções (SECTIONS) da OUTPUT PROCEDURE.

A OUTPUT PROCEDURE consiste numa ou mais seções, escritas consecutivamente e estas seções não podem fazer parte de outros procedimentos diferentes dos definidos na OUTPUT PROCEDURE. Na OUTPUT PROCEDURE é obrigatório pelo menos um comando de RETURN (ordem para retirar o registro já classificado do arquivo do SORT).

O controle é passado para os comandos que compõem a OUTPUT PROCEDURE, somente pelo comando SORT no programa; o comando RETURN da OUTPUT PROCEDURE é controlado por procedimentos do SORT. Na OUTPUT PROCEDURE podem ser incluídos procedimentos para criar, alterar e modificar registros depois de classificados.

Se a OUTPUT PROCEDURE é especificada, o controle somente passa para os procedimentos quando a fase de saída do programa SORT for executada pelo comando RETURN. A saída da OUTPUT PROCEDURE será provida pelo compilador, que irá prover todo o mecanismo de saída da última seção mencionada na OUTPUT PROCEDURE, e somente será ativada quando o procedimento for desviado para a última seção referenciada na OUTPUT PROCEDURE.

GIVING. Se a opção GIVING é usada, todos os registros classificados são automaticamente transferidos para o nome-de-arquivo-3 como se fosse um procedimento de saída implícito para o comando SORT. No instante da execução do comando SORT, nome-de-arquivo-3 não deve estar aberto. A instrução SORT inicia o processamento deste, libera os registros do arquivo para o arquivo-3 e termina o processamento fechando o arquivo-3.

## Comando RELEASE

O comando RELEASE transfere os registros lidos na INPUT PROCEDURE para a fase de entrada nas operações do SORT.

Formato: **RELEASE** nome-de-registro [**FROM** identificador].

O comando RELEASE só pode ser usado na INPUT PROCEDURE associada com o comando SORT, isto é, se a opção INPUT PROCEDURE for especificada, a única maneira de passar os registros lidos pelo COBOL na INPUT PROCEDURE para o SORT é através do RELEASE.

Nome-de-registro. É o nome de registro definido na SD.

Quando o FROM é usado, equivale à transferência dos dados do identificador para o nome-de-registro.

## Comando RETURN

O comando RETURN passa para o COBOL os registros, individualmente já classificados na fase final do SORT.

Formato:

**RETURN** nome-de-arquivo-sort [**INTO** identificador]
**AT END** comando-imperativo.

Nome-de-arquivo-sort é o nome usado na SD.

Todas as referências dos registros lidos pelo comando RETURN têm o mesmo formato definido na SD, isto é, todos os nomes-de-dados estão com os valores do último comando RETURN executados e disponíveis para o usuário.

Se o operando INTO for especificado, após o comando RETURN, todo o registro definido na SD será transferido para o identificador, que é uma área definida pelo usuário.

O comando RETURN somente é usado se associado a uma OUTPUT PROCEDURE.

O comando imperativo na frase AT END somente será executado quando todos os registros classificados forem lidos pelo comando RETURN, ou seja, quando o arquivo classificado chegar ao fim.

Quando da utilização dos comandos USING e GIVING devem ser satisfeitas as seguintes condições:

1. para o comando USING o arquivo de entrada deve ser seqüencial, o arquivo de trabalho deve possuir o mesmo layout do arquivo de entrada e não poderá haver seleção de registros;

2. para o comando GIVING o arquivo de saída deve ser seqüencial e possuir o mesmo layout do arquivo de trabalho e não poderá haver seleção de registros.

Por outro lado, pode-se combinar as opções INPUT PROCEDURE e OUTPUT PROCEDURE com as opções USING e GIVING, respeitando-se as regras para cada comando específico.

Exemplo:

```
SORT arquivo-de-trabalho ASCENDING KEY chave
    USING arquivo-de-entrada
    GIVING arquivo-de-saida.
STOP RUN.
```

# EXERCÍCIOS

Exercício 1

## 1. Objetivo do Programa

Emitir um relatório classificado pelo campo nome do produto.

## 2. Procedimentos

1. ler o arquivo CADPROD;

2. desprezar todos os registros cuja quantidade esteja abaixo do limite inferior;

3. valor em estoque = preço unitário x qtdade estoque;

4. ICM = valor em estoque x 15 : 100;

5. emitir relatório dos registros não desprezados, classificados pelo campo nome do produto;

6. imprimir 55 linhas de detalhe por página;

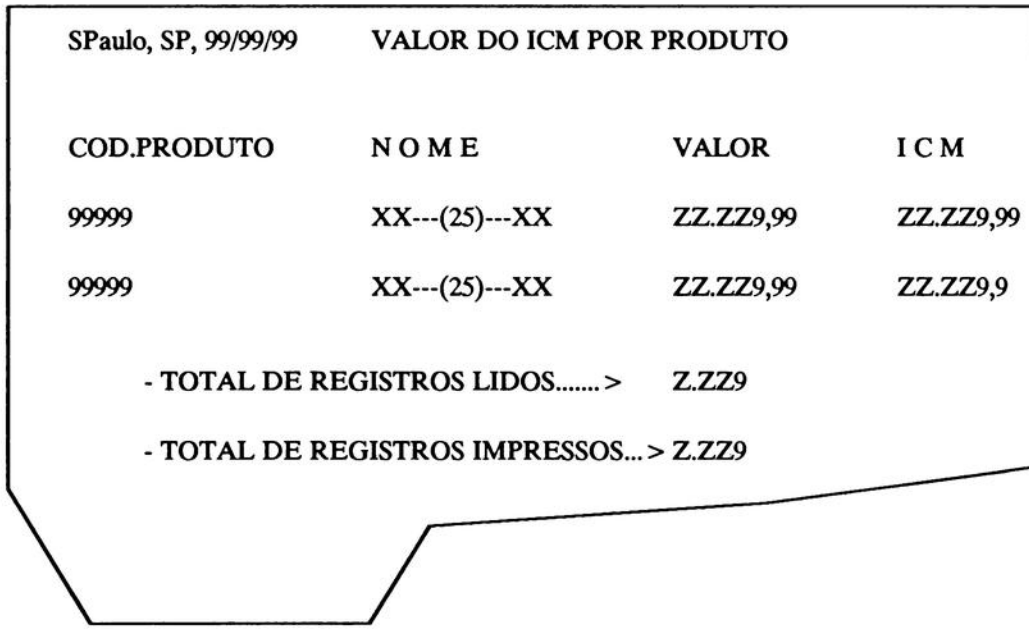
7. imprimir cabeçalho em todas as páginas;

8. no final imprimir o total de registros lidos e impressos.

## 3. Record Layout

| CODIGO DO PRODUTO | NOME DO PRODUTO | LIMITE SUPER. | LIMITE INFER. | PRECO UNITARIO | QTDADE ESTOQUE |
|---|---|---|---|---|---|
| 9(05) | X(25) | 9(05) | 9(05) | 9(05)V99 | 9(05) |

## 4. Gabarito de Impressão (Printer Layout)




Exercício 2

## 1. Objetivo do Programa

Classificar os registros do arquivo CEP01, armazenando-os no arquivo CEP02, para posterior consulta.

## 2. Procedimentos

1. serão aproveitados somente os registros de entrada nos quais o campo CEP terminar em 00;

2. após o processo de classificação, deverão ser transferidos para o arquivo de saída um único registro para cada número de CEP.

## 3. Record Layout de Entrada e de Saída

| cep | endereco | cidade | estado |
|---|---|---|---|
| 9(05) | x(30) | x(20) | x(02) |

## 4. Vídeo Layout Livre

# CAPÍTULO 8

## COMUNICAÇÃO ENTRE PROGRAMAS

### 1. Objetivo do Programa

Exemplificar comunicação entre dois programas.

### 2. Procedimentos

1. o programa principal deverá enviar um determinado número para o subprograma;

2. o subprograma realizará o cálculo desse número com um determinado valor;

3. após o cálculo, o produto da operação será exibido no vídeo através do programa principal.

## Codificação do Programa Principal

```
1  IDENTIFICATION  DIVISION.
2  PROGRAM-ID.     PRINCIP.
3  ENVIRONMENT     DIVISION.
4  CONFIGURATION   SECTION.
5  SPECIAL-NAMES.  DECIMAL-POINT IS COMMA.
6  DATA DIVISION.
7  WORKING-STORAGE SECTION.
8  77  ENVIA      PIC 9(06)    VALUE   ZEROS.
9  77 RECEBE      PIC 9(06)    VALUE   ZEROS.
10 PROCEDURE DIVISION.
11 INICIO.
12      MOVE ZEROS TO ENVIA RECEBE.
13      DISPLAY (01 01) ERASE.
14      DISPLAY (10 15) "TESTE DE INSTRUCAO CALL"
15      DISPLAY (15 15) "DIGITE UM NUMERO - P/ENCERRAR 999999".
16      ACCEPT (17 15) ENVIA WITH PROMPT.
17      IF ENVIA = 999999 STOP RUN.
18      CALL "SUBPROG" USING ENVIA RECEBE.
19 RETORNO.
20      DISPLAY (19 15) "RESPOSTA = " RECEBE.
21      DISPLAY (17 15) " "
22      STOP "<TECLE ENTER> ENTER".
23      GO TO INICIO.
```

## Codificação do Subprograma

```
1  IDENTIFICATION  DIVISION.
2  PROGRAM-ID.     SUBPROG.
3  ENVIRONMENT     DIVISION.
4  CONFIGURATION   SECTION.
5  SPECIAL-NAMES.  DECIMAL-POINT IS COMMA.
6  DATA DIVISION.
7  WORKING-STORAGE SECTION.
8  77  CALCULO        PIC 9(06) VALUE ZEROS.
9  LINKAGE SECTION.
10 77  ENVIADO        PIC 9(06).
11 77  REMETIDO       PIC 9(06).
12 PROCEDURE DIVISION USING ENVIADO REMETIDO.
13 INICIO.
14      DISPLAY (01 01) ERASE.
15      DISPLAY (10 15) "PASSAGEM PELO SUBPROGRAMA".
```

```
16      MOVE ENVIADO TO CALCULO.
17      COMPUTE REMETIDO = CALCULO * 3.
18  SAIDA.
19      EXIT PROGRAM.
```

## Explanação e Instruções Introduzidas no Programa

CALL "SUBPROG" USING ENVIA RECEBE. O comando CALL transfere o controle, durante a execução do programa, "PRINCIP" para o programa "SUBPROG".

O literal SUBPROG corresponde ao nome do programa chamado.

Os nomes-de-dados ENVIA e RECEBE na especificação USING designam campos de dados do programa que chama, aos quais também o programa chamado deve ter acesso.

A comunicação entre os programas é feita através da seqüência em que são apresentados os nomes-de-dados na especificação USING. Assim, os campos ENVIA e RECEBE correspondem aos nomes-de-dados especificados no cabeçalho PROCEDURE DIVISION do programa chamado.

LINKAGE SECTION. Todos os campos de dados apresentados na especificação USING do comando CALL devem ser escritos na LINKAGE SECTION do programa SUBPROG. Trata-se de uma segunda descrição, por isso, não provoca nenhuma reserva de área de memória.

PROCEDURE DIVISION USING ENVIADO REMETIDO. Os campos ENVIADO e REMETIDO, citados após a especificação USING do cabeçalho PROCEDURE DIVISION, devem ser definidos na LINKAGE SECTION do mesmo programa.

EXIT PROGRAM. Indica o término lógico de um programa chamado. Este comando devolve o controle ao programa que chama. O controle é transferido para o comando que vem imediatamente após o comando CALL.

## Comunicação entre Programas com a Linkage Section

Módulos de programas COBOL compilados separadamente podem ser combinados em um único programa executável. A comunicação entre programas é possível através do uso da LINKAGE SECTION da DATA DIVISION. Um módulo lança outro através do comando CALL e da palavra-chave USING, seguida dos parâmetros que devem ser passados. Esta lista de parâmetros é a mesma que segue a palavra USING, no cabeçalho PROCEDURE DIVISION do módulo que está sendo lançado.

A LINKAGE SECTION descreve dados que estarão disponíveis na memória de dados reservada a outro módulo (o que chama). Assim sendo, as descrições de dados da LINKAGE SECTION não acarretam reserva da área de memória.

Todas as regras de descrição de dados aplicáveis na WORKING-STORAGE valem na LINKAGE SECTION com as seguintes ressalvas:

1. a cláusula VALUE só pode ser associada a itens do nível 88;

2. é da responsabilidade do programador garantir o alinhamento entre os argumentos do programa que chama e do programa que é chamado.

Lista USING em subprogramas COBOL:

As diferenças básicas entre um subprograma e um programa são:

- o subprograma tem LINKAGE SECTION;

- o cabeçalho PROCEDURE DIVISION recebe um apêndice;

- os comandos de fim são diferentes.

O cabeçalho PROCEDURE DIVISION num subprograma tem a forma:

**PROCEDURE DIVISION [USING** nome-de-dado.....**]**

O USING precede à lista que contém cada um dos níveis 01 e 77 da LINKAGE SECTION. Quando um programa chama este subprograma, o CALL tem o efeito de passar ordenadamente os endereços dos itens que constam do seu próprio USING. A passagem é posicional e, portanto, não interessam os nomes no instante da passagem.

## INSTRUÇÃO CALL

O seu formato é:

**CALL** { identificador / literal } **[USING** nome-de-dado-1 ... nome-de-dado-n**]**
**[ON OVERFLOW** comando-imperativo**]**

Literal é o nome do subprograma que aparece como PROGRAM-ID de um módulo compilado separadamente.

Os nomes-de-dados que constam do USING tornam-se disponíveis ao subprograma chamado pela passagem de endereços. Estes endereços são então associados aos itens descritos na LINKAGE SECTION e arrolados no USING do cabeçalho da PROCEDURE DIVISION do subprograma chamado. Por isso, o número de itens no USING do CALL deve ser o mesmo do USING no subprograma chamado.

### Frase ON OVERFLOW

Caso a memória seja insuficiente para acomodar o subprograma especificado pela instrução CALL e a frase ON OVERFLOW for mencionada, não ocorrerá a transferência de comando para o subprograma e a instrução imperativa da frase ON OVERFLOW será executada. Se a frase ON OVERFLOW não estiver presente e a memória for insuficiente para carregar o programa, o programa terminará anormalmente.

## INSTRUÇÃO EXIT PROGRAM

O seu formato é: EXIT PROGRAM

O EXIT PROGRAM retorna o controle ao programa que chamou o módulo em que aparece.

O retorno se dá no primeiro comando executável após o CALL.

## INSTRUÇÃO CHAIN

O seu formato é: **CHAIN** {literal/identificador-1} [**USING** identificador-2.....}

Literal e identificador-1 devem ser alfanuméricos, e o identificador-1 deve conter um espaço para terminação. Cada ocorrência do identificador-2 deve estar definida na WORKING- STORAGE ou LINKAGE SECTION ou na área de registro de um arquivo aberto no momento em que o comando CHAIN é executado.

Quando o comando CHAIN é executado, o valor do literal ou identificador-1, até (não incluindo) o primeiro espaço encontrado (ou o fim do literal), é interpretado como o nome de um arquivo de programa executável. O programa nomeado é carregado na memória e executado. Todo programa e estrutura de dados do programa principal são permanentemente destruídos, exceto a cláusula USING que pode ser usada para transferir parâmetros para o programa chamado.

### Formato da PROCEDURE DIVISION usando CHAINING

**PROCEDURE DIVISION [CHAINING** nome-de-dado-1.....]

A cláusula CHAINING deve ser especificada quando for transferido o controle de execução de um outro programa através da instrução CHAIN, contendo a cláusula USING.

Os valores dos itens-de-dados, nomeados no cabeçalho da PROCEDURE DIVISION, são estabelecidos no início do programa usando os conteúdos dos itens-de-dados, posicionalmente correspondentes, nomeados na chamada do comando CHAIN.

## INSTRUÇÃO CANCEL

A instrução CANCEL é usada para remover sub-rotinas da memória, liberando-as para outros subprogramas ou para outra finalidade.

Formato:

```
CANCEL  { identificador-1 }  [ identificador-2 ]
        { literal-1       }  [ literal-2       ]
```

Os identificadores ou literais especificam os subprogramas que serão liberados.

Os identificadores devem ser definidos como dados alfanuméricos. Se o subprograma a ser liberado não for previamente chamado, o comando CANCEL é ignorado.

## LINKAGE SECTION

A estrutura da LINKAGE SECTION corresponde à estrutura da WORKING-STORAGE SECTION.

```
[LINKAGE SECTION]
[77 nome-de-dados.....]
[descrição de registro.....]
```

Quando se deseja ter acesso a um campo de dados descrito na LINKAGE SECTION, o nome de dados no cabeçalho PROCEDURE DIVISION é tornado igual ao nome de dados correspondente na instrução CALL, sendo assim determinado o endereço do campo no programa que chama.

## INSTRUÇÃO SEARCH

**Formato 1**

Uma pesquisa linear de uma tabela pode ser feita usando-se o comando SEARCH.

O formato geral é:

**SEARCH** tabela [**VARYING** identificador/nome-de-índice]
[AT **END** comando-imperativo-1] {**WHEN** condição {NEXT
SENTENCE/comando-imperativo-2}}.....

A tabela é o nome de item que contém a cláusula OCCURS com INDEXED BY. A tabela deve ser escrita sem subscritos ou índices, pois a instrução SEARCH provoca a variação do nome-de- índice associado a esta tabela.

A frase VARYING permite 4 opções:

1. sem a frase VARYING a pesquisa será feita variando o primeiro nome-de-índice da lista na cláusula OCCURS;

2. VARYING nome-de-índice de uma outra tabela a pesquisa será feita variando o primeiro nome-de-índice especificado na cláusula OCCURS e, ao mesmo tempo, haverá variação do nome-de-índice da frase VARYING da mesma maneira;

3. VARYING nome-de-índice definido na tabela - este nome-de-índice específico é o único que é variado;

4. VARYING nome-de-índice - tanto este dado, como o primeiro nome-de-índice relacionado para a tabela, são variados simultaneamente.

O valor inicial do nome-de-índice antes da pesquisa é definido com a instrução SET. Se o nome-de-índice contém um valor que excede o valor máximo declarado na cláusula OCCURS, é executada a instrução incondicional-1 da frase AT END. Se a instrução SEARCH foi escrita sem a frase AT END, não será executada, passando o controle para a instrução seguinte.

Se o valor do índice é válido então, cada condição especificada na frase WHEN é examinada. Se a condição é verdadeira então é executada a instrução incondicional correspondente, terminando a pesquisa. Caso contrário, o índice é incrementado de 1 e passa a repetir a mesma operação.

Se uma tabela possui várias dimensões (várias cláusulas OCCURS num mesmo item-de-grupo), o nome-de-índice deve ser definido a cada frase INDEXED BY das cláusulas OCCURS. Somente um índice é variado (acompanhado pelo outro nome-de-índice ou item-de-dado da frase VARYING. Para pesquisa de todo o conjunto de tabelas bi ou tridimensional, haverá a necessidade de se executar várias instruções SEARCH utilizando vários nomes-de-índice, com valor inicial definido pela instrução SET no momento apropriado a cada uma delas com a instrução PERFORM ou VARYING.

Exemplo:

```
:
WORKING-STORAGE SECTION.
77  CODIGO      PIC 9(03)      VALUE   ZEROS.
01 TABELA-DE-MUNICIPIO.
     02 TABELA OCCURS 1000 TIMES INDEXED BY IND.
          03 NUMERO PIC 9(06).
          03 NOMEPIC X(24).
:
PROCEDURE DIVISION.
:
PESQ-TABELA.
     ACCEPT (05, 10) CODIGO WITH PROMPT.
     IF CODIGO = 999999 GO TO FIM.
     IF CODIGO = ZEROS GO TO PESQ-TABELA.
     SET IND TO 1.
     SEARCH TABELA VARYING IND AT END GO TO ROT-ERRO
          WHEN CODIGO = NUMERO(IND)
          PERFORM ROT-EXIBE

          GO TO PESQ-TABELA.
```

**Formato 2**

A instrução SEARCH (formato 2) lida com tabelas de dados ordenados.

Formato Geral:

**SEARCH ALL** tabela [AT **END** instrução-incondicional-1...]
**WHEN** condição {instrução-incondicional-2.../ NEXT SENTENCE}.

A cláusula WHEN pode ser definida somente uma vez. A condição deverá ser uma relação simples ou nome-de-condição. A relação deve conter o nome da tabela e o índice.

O nome-de-dado é considerado como uma chave para pesquisa na tabela e deve ser definido na cláusula OCCURS com o seguinte formato:

```
{ASCENDING/DESCENDING} KEY IS nome-de-dado....
```

onde nome-de-dado é definido na descrição de dado. A frase KEY nome-de-dado indica que a tabela será pesquisada em ordem ascendente ou descendente a partir de um valor do nome-de-dado. Pode-se especificar mais de uma frase KEY.

Numa condição simples deve ser escrito somente um teste de igualdade (usando a relação = ou IS EQUAL TO), ou pode ser definida uma conexão lógica usando AND (o OR não deve ser usado). Nesta condição, o predicado pode ser um identificador ou literal, porém, o identificador não deve ser referenciado na cláusula KEY ou indexado pelo primeiro nome-de-índice associado à tabela.

Se a condição não for satisfeita para todos os valores de índice, o controle passa para a instrução-incondicional-1 se existir a cláusula AT END. Caso contrário, executa a instrução seguinte. Se a condição for satisfeita o controle passa para a instrução incondicional-2.

A definição do valor inicial do nome-de-índice (pela instrução SET) é ignorada neste formato.

Exemplo:

```
FILE SECTION.
FD  ARQMESTRE       LABEL RECORD STANDARD
                    VALUE OF FILE-ID "MESTRE".
01 REGISTRO-MESTRE.
    02 ESTADO           PIC 9(02).
    02 ZONA             PIC 9(02).
    02 MUNICIPIO        PIC 9(04).
    02 AREA.
```

```
           03 IDTF          PIC 9(06).
           03 NUM           PIC 9(06).
:
WORKING-STORAGE SECTION.
01 TABELA-MUNICIPIO.
     02 TAB-MUN OCCURS 9999 TIMES INDEXED BY IA
          ASCENDING KEY MUN.
          03  MUN           PIC 9(04).
          03  NOME          PIC X(24).
:
PROCEDURE DIVISION.
:
LER.
     READ ARQMESTRE AT END GO TO FIM.


PESQUISA-TABELA.
     SEARCH ALL TAB-MUN AT END PERFORM ROT-ERRO
          WHEN MUN(IA) = MUNICIPIO
               MOVE NOME(IA) TO DET-IMP-MUNICIPIO.
     :
OUTRO-PARAGRAFO.
```

# CAPÍTULO 9


## INSTRUÇÃO INSPECT

### 1. Objetivo do Programa

Permite ao programador o exame de uma determinada seqüência de caracteres. Pode-se combinar várias das seguintes ações:

### 2. Procedimentos

1. contar as ocorrências de um dado caractere;

2. substituir certo caractere por um alternativo;

3. qualificar e limitar as operações acima, condicionando-as à ocorrência de caracteres específicos.

O formato do comando INSPECT é:

**INSPECT** nome-de-dado-1 [cláusula TALLYING]
[cláusula REPLACING]

Sendo que as duas cláusulas têm os formatos:

**Formato 1. cláusula TALLYING**

**TALLYING** nome-de-dado-2 **FOR** {**CHARACTERS** {**ALL/LEADING**}
operando-3} [{**BEFORE/AFTER**} INITIAL operando-4].

**Formato 2. cláusula REPLACING**

**REPLACING** {**CHARACTERS** {**ALL/LEADING/FIRST**} operando-5}
**BY** [{**BEFORE/AFTER**} INITIAL operando-7].

Nos formatos descritos operando-n pode ser:

1. um literal entre apóstrofos de um caractere;

2. uma constante figurativa que signifique um caractere;

3. o nome-de-dado de um item de tamanho unitário.

A cláusula TALLYING, ou a cláusula REPLACING, ou ambas, devem constar sempre de um INSPECT. Quando ambas ocorrerem, TALLYING deve vir primeiro.

O funcionamento de ambas as cláusulas é:

A cláusula TALLYING determina comparação caractere a caractere a partir da esquerda de nome-de-dado-1 com operando-3. Se a cláusula AFTER INITIAL operando-4 estiver presente, então as comparações iniciarão apenas depois do ponto em que ocorrer, pela primeira vez, operando-4.

Se BEFORE INITIAL operando-4 tiver sido especificado, as comparações terminam quando pela primeira vez for encontrado operando-4 (se não ocorrer, a comparação prosseguirá até o último caractere de nome-de-dado-1).

A cláusula REPLACING permite a substituição de caracteres sob condições especificadas.

Se AFTER INITIAL operando-7 estiver presente, então as substituições só serão efetuadas após a primeira ocorrência de operando-7; se, a cláusula BEFORE INITIAL operando-7 estiver presente, as substituições serão efetuadas até a primeira ocorrência de operando-7.

Se um INSPECT contiver TALLYING e REPLACING então tudo se passa como se dois comandos INSPECT, um contendo TALLYING e outro contendo REPLACING tivessem sido escritos. O comando TALLYING será executado primeiro.

Quando TALLYING for usado o resultado da contagem é adicionado ao valor do nome-de-dado-2. O usuário deve portanto mover zero para nome-de-dado se quiser o valor absoluto da contagem.

Abaixo, estão descritas as funções de cada uma das palavras reservadas que aparecem em INSPECT. A palavra TALLYING conta e a palavra REPLACING substitui.

CHARACTERS. Qualquer caractere do código ASCII. Se especificarmos AFTER ou BEFORE INITIAL operando-n, podemos contar quantos caracteres do campo examinado precedem ou sucedem operando-n, para TALLYING.

ALL operando-m. Conta ou substitui todas as ocorrências de operando-m, até operando-p ou após operando-p, se BEFORE INITIAL operando-p estiver presente. Na ausência de BEFORE INITIAL e AFTER INITIAL, examina todo o campo reservado para nome-de-dado- 2.

LEADING operando-m. Conta ou substitui todas as ocorrências de operando-i que aparecerem no começo do campo sob exame, consecutivas,

sem interrupção. O mesmo procedimento se aplica à parte do campo que sucede imediatamente operando-j, se AFTER INITIAL operando-j estiver presente. Não faz sentido usar a opção LEADING combinada com BEFORE INITIAL operando-j.

FIRST operando-5. Especifica que somente o primeiro caractere encontrado igual ao operando-5 participa da substituição pelo operando-6.

Exemplos:

*a.* INSPECT NOME TALLYING CONTADOR FOR CHARACTERS
AFTER INITIAL "P".

| | antes | depois |
|---|---|---|
| NOME | AB5PS134)7(9 | AB5PS134)7(9 |
| CONTADOR | 0 | 8 |

*b.* INSPECT NOME REPLACING LEADING ZEROS BY "9" AFTER INITIAL "P".

| NOME(antes) | NOME(depois) | Comentário |
|---|---|---|
| BA004P0020AB | BA004P9920AB | (trocou 00 por 99) |
| ABCPDP5004P0 | ABCPDP5004P0 | (nada mudou, o primeiro P não é seguido por 0) |

*c.* INSPECT DADOS TALLYING NUMERO FOR LEADING "L"
BEFORE INITIAL "A".

| DADOS(antes) | LAGOA | ANALISTA | LANCA |
|---|---|---|---|
| NUMERO(depos) | 1 | 0 | 1 |

*d.* MOVE ZEROS TO CONTADOR.
INSPECT ITEM TALLYING CONTADOR FOR ALL "L"
REPLACING LEADING"A" BY "E" AFTER INITIAL "L".

| | | | |
|---|---|---|---|
| ITEM(antes) | SALARIO | BALEEIRA | SALADA |
| ITEM(depois) | SALERIO | BALEEIRA | SALADA |
| CONTADOR | 1 | 1 | 1 |

*e.* INSPECT ARTRAB REPLACING ALL SEPARADORES BY TRANSFORMACAO

| | |
|---|---|
| ARTRAB(antes com 16 posições) | RIO - SAO PAULO |
| SEPARADORES | (espaços) |
| TRANSFORMACAO | (.) |
| ARTRAB(resultado) | RIO.-.SAO.PAULO. |

## Codificação do Programa

Abaixo um programa desenvolvido para melhores esclarecimentos.

```
1    IDENTIFICATION  DIVISION.
2    PROGRAM-ID.     INSPC.
3    ENVIRONMENT     DIVISION.
4    CONFIGURATION   SECTION.
5    SPECIAL-NAMES.  DECIMAL-POINT IS COMMA.
6    DATA DIVISION.
7    WORKING-STORAGE SECTION.
8    77  CTRL-ESC       PIC 9(02)    VALUE  ZEROS.
9    77  OPCAO          PIC 9        VALUE  ZEROS.
10   77  NOME           PIC X(20)    VALUE  SPACES.
11   77  CONTADOR       PIC 9(02)    VALUE  ZEROS.
12   77  SEPARADORES    PIC X        VALUE  " ".
13   77 TRANSFORMA       PIC X        VALUE  "*".
14   77 NUMERO           PIC X(08)    VALUE  SPACES.
15   01 DATA-SYS.
16        02 AA          PIC 99 VALUE ZEROS.
17        02 MM          PIC 99 VALUE ZEROS.
18        02 DD          PIC 99 VALUE ZEROS.
19   SCREEN SECTION.
20   01 TELA.
21        02  BLANK SCREEN.
22        02  LINE 02 COLUMN 01 VALUE "DATA".
23        02  LINE 02 COLUMN 28 VALUE "= = = = = = = = = = = = = = = = = = = = =".
24        02  LINE 03 COLUMN 28 VALUE "= COMANDO INSPECT =".
25        02  LINE 04 COLUMN 28 VALUE "= = = = = = = = = = = = = = = = = = = = =".
26        02  LINE 06 COLUMN 03 VALUE "OPCOES:".
27        02  LINE 08 COLUMN 03 VALUE "1 - Examinar quantidade d
28   -    "e caracteres apos a letra 'Z'".
29        02  LINE 10 COLUMN 03 VALUE "2 - Trocar caracteres 'B'
30   -    " apos a letra 'A'".
31        02  LINE 12 COLUMN 03 VALUE "3 - Examinar quantidade d
32   -    "e caracteres antes da letra 'M'".
33        02  LINE 14 COLUMN 03 VALUE "4 - Examinar e trocar car
34   -    "acteres 'A' por 'E' que estiverem apos 'L'".
35        02  LINE 16 COLUMN 03 VALUE "5 - Examinar e trocar bra
36   -    "ncos por '*'".
37        02  LINE 18 COLUMN 03 VALUE "6 - Examinar e trocar bra
38   -    "ncos por ZEROS".
39        02  LINE 20 COLUMN 03 VALUE "7 - Encerra testes".
40        02  LINE 22 COLUMN 03 VALUE "OPCAO (1/7) < > DIGI
```

```
41  -    "TE:".
42       02 NOME-S   LINE 22 COLUMN 32 PIC X(20) USING NOME
43                   REVERSE-VIDEO.
44       02 NUMER-S  LINE 22 COLUMN 32 PIC X(08) USING NUMERO
45                   REVERSE-VIDEO.
46       02   LINE 24 COLUMN 20 VALUE "RESULTADO:".
47       02   LINE 24 COLUMN 53 VALUE "QUANTIDADE:".
48  01   LIMPA-MENSA.
49       02   LINE 24 COLUMN 32 VALUE "".
50       02   LINE 24 COLUMN 65 VALUE "".
51
52  PROCEDURE DIVISION.
53  INICIO.
54       ACCEPT DATA-SYS FROM DATE.
55       DISPLAY TELA.
56       DISPLAY (2 8) DD "/" MM "/" AA.
57  ROT-OPCAO.
58       DISPLAY (22 32) " ".
59       ACCEPT (22 18) OPCAO WITH AUTO-SKIP.
60       DISPLAY LIMPA-MENSA.
61       GO TO ROT1, ROT2, ROT3, ROT4, ROT5, ROT6, ROT7
62            DEPENDING ON OPCAO.
63       GO TO ROT-OPCAO.
64  ROT1.
65       DISPLAY (22 32) " ".
66       ACCEPT NOME-S.
67       IF NOME = SPACES GO TO ROT1.
68       MOVE ZEROS TO CONTADOR.
69       INSPECT NOME TALLYING CONTADOR FOR CHARACTERS AFTER INITIAL
70       "Z".
71       DISPLAY (24 32) NOME.
72       DISPLAY (24 65) CONTADOR.
73       GO TO ROT-OPCAO.
74  ROT2.
75       DISPLAY (22 32) " ".
76       ACCEPT NOME-S.
77       IF NOME = SPACES GO TO ROT2.
78       MOVE ZEROS TO CONTADOR.
79       INSPECT NOME REPLACING ALL "B" BY "X" AFTER INITIAL "A".
80       DISPLAY (24 32) NOME.
81       DISPLAY (24 65) CONTADOR.
82       GO TO ROT-OPCAO.
83  ROT3.
84       DISPLAY (22 32) " ".
85       ACCEPT NOME-S.
```

```
86          IF NOME = SPACES GO TO ROT3.
87          MOVE ZEROS TO CONTADOR.
88          INSPECT NOME TALLYING CONTADOR FOR CHARACTERS BEFORE INITIAL
89          "M".
90          DISPLAY (24 32) NOME.
91          DISPLAY (24 65) CONTADOR.
92          GO TO ROT-OPCAO.
93    ROT4.
94          DISPLAY (22 32) " ".
95          ACCEPT NOME-S.
96          IF NOME = SPACES GO TO ROT4.
97          MOVE ZEROS TO CONTADOR.
98          INSPECT NOME TALLYING CONTADOR FOR ALL "L" REPLACING ALL
99          "A" BY "E" AFTER INITIAL "L".
100         DISPLAY (24 32) NOME.
101         DISPLAY (24 65) CONTADOR.
102         GO TO ROT-OPCAO.
103   ROT5.
104         DISPLAY (22 32) "".
105         ACCEPT NOME-S.
106         IF NOME = SPACES GO TO ROT5.
107         MOVE ZEROS TO CONTADOR.
108         INSPECT NOME REPLACING ALL SEPARADORES BY TRANSFORMA.
109         DISPLAY (24 32) NOME.
110         DISPLAY (24 65) CONTADOR.
111         GO TO ROT-OPCAO.
112   ROT6.
113         DISPLAY (22 32) "".
114         ACCEPT NUMER-S.
115         IF NUMERO = SPACES GO TO ROT6.
116         MOVE ZEROS TO CONTADOR.
117         INSPECT NUMERO REPLACING ALL SPACES BY ZEROS.
118         DISPLAY (24 32) NUMERO.
119         DISPLAY (24 65) CONTADOR.
120         GO TO ROT-OPCAO.
121   ROT7.
122         STOP RUN.
```

## Explanação e Instruções Introduzidas no Programa

INSPECT NOME TALLYING CONTADOR FOR CHARACTERS AFTER INITIAL "Z".

Ação: especifica que serão contados e armazenados no campo CONTADOR, a quantidade de caracteres do campo NOME que estiverem localizados após a primeira letra *Z*.

```
INSPECT NOME REPLACING ALL "B" BY "X" AFTER INITIAL
"A".
```

Ação: especifica que serão substituídos todos os caracteres *B* por *X* que estiverem localizados após a primeira letra *A*.

```
INSPECT NOME TALLYING CONTADOR FOR
CHARACTERS BEFORE INITIAL "M".
```

Ação: especifica que serão contados e armazenados no campo CONTADOR a quantidade de caracteres do campo NOME existentes antes da primeira letra *M*.

```
INSPECT NOME TALLYING CONTADOR FOR ALL "L"
REPLACING ALL "A" BY "E" AFTER INITIAL "L".
```

Ação: especifica que serão contados e armazenados em CONTADOR todos os caracteres *L* e todos os caracteres *A* serão substituídos por *B* que estiverem após a primeira letra *L*.

```
INSPECT NOME REPLACING ALL SEPARADORES BY
TRANSFORMA.
```

Ação: especifica que todos os caracteres idênticos ao conteúdo do campo SEPARADORES serão substituídos pelos caracteres do campo TRANSFORMA.

```
INSPECT NUMERO REPLACING ALL SPACES BY ZEROS.
```

Ação: especifica que todos os espaços do campo NUMERO serão preenchidos por zeros.

# INSTRUÇÃO STRING

O comando STRING permite a concatenação de múltiplos valores de item-de-dado enviados em um só item receptor.

O formato geral do comando STRING é:

**STRING** {operando-1... **DELIMITED** BY {operando-2/**SIZE**}}...
**INTO** identificador-1 [WITH **POINTER** identificador-2]
[ON **OVERFLOW** comando-imperativo]

O termo operando-1 significa uma literal não-numérico, uma constante figurativa de um caractere, ou nome-de-dado. Identificador-1 é o nome do item-de-dado receptor que deve ser alfanumérico sem símbolo de edição ou a cláusula JUSTIFIED. O identificador-2 é um contador e deve ser um item-de-dado inteiro numérico elementar de tamanho suficiente (mais um) para apontar as posições de caractere dentro de identificador-1.

Se não houver nenhuma frase POINTER, o valor padrão do POINTER lógico é um.

O valor lógico do POINTER designa a posição inicial do campo receptor no qual a colocação do dado começa. Durante o movimento para o campo receptor, os critérios para término de uma fonte individual são controlados pela frase "DELIMITED BY".

DELIMITED BY SIZE: o campo fonte inteiro é movido (a menos que o campo receptor venha cheio).

DELIMITED BY operando-2: a seqüência de caracteres especificada pelo operando-2 é um padrão de pesquisa que, quando for igual a uma seqüência contígua dos caracteres enviados termina a função para o operando atual enviado (e provoca a comutação automática para o operando enviado em seguida, se houver). Os caracteres de combinação nos campos de envio não são movidos para o identificador-1.

Se em qualquer ponto o apontador POINTER lógico (que é automaticamente incrementado de um para cada caractere armazenado no

identificador-1) for menor que um ou maior que o tamanho do identificador-1, não ocorrerá nenhum movimento de dado, e o comando imperativo dado na frase OVERFLOW (se especificado) será executado. Se não existir nenhuma frase OVERFLOW, o controle será transferido para o próximo comando executável.

Não há nenhum preenchimento de espaço automático em nenhuma posição do identificador-1. Isto é, as posições não acessadas não sofrem mudanças após o término do comando STRING.

Após o término do comando STRING, se houver uma frase POINTER, o valor resultante do identificador-2 é igual ao seu valor original mais o número de caracteres movidos durante a execução do comando STRING.

## INSTRUÇÃO UNSTRING

O comando UNSTRING faz com que os dados de um único campo sejam separados em subcampos que são colocados em múltiplos campos receptores.

O formato geral do comando é:

**UNSTRING** identificador-1 [**DELIMITED** BY [**ALL**] operando-1 [**OR** [**ALL**] operando-2] ... ] **INTO** {identificador-2 [**DELIMITER** IN identificador-3] [**COUNT** IN identificador-4]}... [WITH **POINTER** identificador-5] [**TALLYING** IN identificador-6] [**ON OVERFLOW** comando imperativo].

Os critérios para separação de subcampos podem ser dados na frase "**DELIMITED BY**". Cada vez que uma sucessão de caracteres combina com um dos literais não-numéricos, constantes figurativas de um caractere, ou valores de item-de-dado nomeados pelo operador-1, o conjunto atual de caracteres enviados é terminado e movido para o campo receptor seguinte, especificado pela cláusula INTO. Quando a frase ALL é especificada, mais de uma ocorrência contígua no operando-1 em identificador-1, é tratado

como uma ocorrência. A cadeia delimitadora não é movida para o campo receptor corrente.

Quando existem dois ou mais delimitadores, existe uma condição "OR". Cada delimitador é comparado com o campo emissor na ordem especificada no comando UNSTRING.

O identificador-1 deve ser um item-de-grupo ou um item-de-seqüência de caracteres (alfanuméricos). Quando um item-de-dado é empregado como qualquer operando-1, esse operando deverá também ser um item-de-grupo, ou um item de uma seqüência de caracteres.

Os campos receptores (identificador-2) podem ser qualquer um dos seguintes tipos de itens:

1. um item alfabético não-editado;

2. um item de cadeia de caracteres (alfanumérico);

3. um item de grupo;

4. um item decimal externo (numérico, usage DISPLAY) cuja PIC não contenha nenhum caractere P.

Quando qualquer exame encontra dois delimitadores contíguos, a área receptora corrente é preenchida com brancos ou zeros dependendo do seu tipo. Se houver uma frase "DELIMITED BY", no comando UNSTRING, então existirão frases "DELIMITER IN" seguindo qualquer item receptor identificador-2 mencionado na cláusula INTO. Neste caso, o(s) caracter(es) que delimitam o dado movido no identificador-2 são armazenados no identificador-3, que deverá ser um item alfanumérico. Além disso, se uma frase "COUNT IN" estiver presente, o número de caracteres que foram movidos para o identificador-2 será somado para o identificador-4, que deve ser um item inteiro numérico elementar.

Se houver uma frase "POINTER", então o identificador-5 deve ser um item numérico inteiro, e seu valor inicial é o valor do pointer lógico inicial (caso contrário, o valor do pointer lógico é considerado um). O exame de caracteres-fonte começa na posição no identificador-1 especificada pelo pointer lógico; após o término do comando UNSTRING, o valor final do pointer lógico será copiado novamente no identificador-5.

Se em qualquer ocasião o valor do "pointer" lógico for menor que um ou exceda o tamanho do identificador-1, ocorre "OVERFLOW" e o controle passa para o comando imperativo dado na cláusula "ON OVERFLOW", se especificado.

"OVERFLOW" também ocorre quando todos os campos receptores foram preenchidos antes de esvaziar o campo-fonte.

Durante o curso da pesquisa do campo-fonte (procurando combinação de seqüências delimitadoras), uma seqüência de caracteres de comprimento variável é desenvolvida e, quando completada pelo reconhecimento de um delimitador ou aquisição de tantos caracteres quanto o tamanho do campo receptor corrente pode reter, é movido para o campo receptor corrente de acordo com o modo padrão do comando MOVE.

Se houver uma frase "TALLYING IN", o identificador-6 deve ser um item-numérico inteiro. O número de campos receptores ativados, mais o valor inicial do identificador-6 será produzido no identificador-6 após o comando UNSTRING.

Qualquer subscrição ou indexação, associada com o identificador-1, 5 ou 6, é avaliada somente uma vez no começo do comando UNSTRING. Qualquer subscrição associada com operando- 1 ou identificador-2, 3, ou 4 é avaliada imediatamente antes do acesso ao item-de-dado.

## Codificação do Programa

Programa exemplo envolvendo os comandos STRING e UNSTRING.

```
1   IDENTIFICATION  DIVISION.
2   PROGRAM-ID.     STRUNSTR.
3   ENVIRONMENT     DIVISION.
4   CONFIGURATION SECTION.
5   SPECIAL-NAMES. DECIMAL-POINT IS COMMA.
6   DATA DIVISION.
7   WORKING-STORAGE SECTION.
8   77  VAGO               PIC X          VALUE   SPACES.
9   01  DATA-SYS.
10      02 AA               PIC 99         VALUE   ZEROS.
11      02 MM               PIC 99         VALUE   ZEROS.
12      02 DD               PIC 99         VALUE   ZEROS.
13  01  CAMPOS-RECEPTORES.
14          03 CPO1         PIC X(10)      VALUE   SPACES.
15          03 CPO2         PIC X(10)      VALUE   SPACES.
16          03 CPO3         PIC X(10)      VALUE   SPACES.
17          03 CPO4         PIC X(10)      VALUE   SPACES.
18  01  CONTADOR           PIC 9(02)      VALUE   ZEROS.
19  01  ESC                PIC 9(02)      VALUE   01.
20  01  RECEPTOR           PIC X(48)      VALUE   SPACES.
21  01  RECEPTOR-1         PIC X(40)      VALUE   SPACES.
22  01  CONT-POINTER       PIC 9(02)      VALUE   01.
23  01  SUB                PIC 9(01)      VALUE   ZEROS.
24  01  CONT-REG           PIC 9(02)      VALUE   01.
25  01  TABELA-UNSTRING.
26      02 TAB-UNSTR OCCURS 4 TIMES.
27          03 RECEPT       PIC X(10).
28          03 DELM         PIC X(03).
29          03 CONT         PIC 9(02).
30  SCREEN SECTION.
31  01 TELA.
32      02          BLANK SCREEN.
33      02          LINE 02 COLUMN 01   VALUE   "DATA".
34      02          LINE 02 COLUMN 28   VALUE   "= = = = = = = = = = = = = = = = =
35  -   "= = = = =" FOREGROUND-COLOR 7 BACKGROUND-COLOR 0.
36      02  LINE 03 COLUMN 28 VALUE "= COMANDOS STRING E UNSTR
37  -   "ING =" FOREGROUND-COLOR 7 BACKGROUND-COLOR 0.
38      02  LINE 04 COLUMN 28 VALUE "= = = = = = = = = = = = = = = = = = = = = =
39  -   "= = = = =" FOREGROUND-COLOR 7 BACKGROUND-COLOR 0.
```

```
40          02  LINE 06 COLUMN 02 VALUE "-O comando STRING permite
41      -   " a concatenacao de varios itens de dados num unico".
42          02  LINE 07 COLUMN 02 VALUE "item.".
43          02  LINE 09 COLUMN 02 VALUE "-O comando UNSTRING permi
44      -   "te transferir um conjunto de caracteres de um item".
45          02  LINE 10 COLUMN 02 VALUE " para um multiplo campo r
46      -   "eceptor (separando em subcampos).".
47          02  LINE 12 COLUMN 21 VALUE "FUNCOES:".
48          02  LINE 14 COLUMN 11 VALUE "Para o 1. exemplo de STRI
49      -   "NG aperte... <F1>".
50          02  LINE 16 COLUMN 11 VALUE "Para o 2. exemplo de STRI
51      -   "NG aperte... <F2>".
52          02  LINE 18 COLUMN 11 VALUE "Para o 1. exemplo de UNST
53      -   "RING aperte <F3>".
54          02  LINE 20 COLUMN 11 VALUE "Para o 2. exemplo de UNST
55      -   "RING aperte <F4>".
56          02  LINE 22 COLUMN 11 VALUE "Para voltar ao inicio ape
57      -   "rte..... <ES>".
58          02  LINE 24 COLUMN 11 VALUE "Para encerrar aperte.....
59      -   "..... <F5>".
60      01  TELA2.
61          02  BLANK SCREEN.
62          02  LINE 02 COLUMN 02 VALUE "No 1. exemplo do comando
63      -   "STRING, digite 4 campos que serao concatenados em um".
64          02  LINE 04 COLUMN 02 VALUE "unico campo. Estes campos
65      -   " fonte serao delimitados por /.".
66      01  TELA3.
67          02  LINE 06 COLUMN 02 VALUE "Entre com o 1. campo com
68      -   "ate 10 CHAR :".
69          02  LINE 08 COLUMN 02 VALUE "Entre com o 2. campo com
70      -   "ate 10 CHAR :".
71          02  LINE 10 COLUMN 02 VALUE "Entre com o 3. campo com
72      -   "ate 10 CHAR :".
73          02  LINE 12 COLUMN 02 VALUE "Entre com o 4. campo com
74      -   "ate 10 CHAR :".
75      01  TELA3-A.
76          02  LINE 24 COLUMN 18 VALUE "Aperte <ESC> para voltar
77      -   "a tela inicial".
78      01  TELA4.
79          02  BLANK SCREEN.
80          02  LINE 02 COLUMN 07 VALUE "No 2. exemplo do comando
81      -   "STRING, digite 3 campos que serao conca-".
82          02  LINE 04 COLUMN 07 VALUE "tenados em um unico campo
83      -   ". Estes campos fonte serao delimitados".
84          02  LINE 06 COLUMN 07 VALUE "por brancos.".
```

```
85  01  TELA5.
86          02  BLANK SCREEN.
87          02  LINE 02 COLUMN 07 VALUE "No 1. exemplo do comando
88  -       "UNSTRING, digite um campo de 40 caracteres".
89          02  LINE 04 COLUMN 07 VALUE "no formato identico a xxx
90  -       "xx/xxxxx/xxxxx/xxxxx.".
91          02  LINE 07 COLUMN 10 VALUE "ENTRE COM O CAMPO:".
92          02  LINE 09 COLUMN 20 VALUE "Resultado apos a execucao
93  -       ":".
94          02  SS-REC LINE 07 COLUMN 29 PIC X(48) USING RECEPTOR
95  REVERSE-VIDEO.
96          02  LINE 11 COLUMN 20 VALUE "1. Campo:".
97          02  LINE 13 COLUMN 20 VALUE "2. Campo:".
98          02  LINE 15 COLUMN 20 VALUE "3. Campo:".
99          02  LINE 17 COLUMN 20 VALUE "4. Campo:".
100 01 TELA6.
101         02  BLANK SCREEN.
102         02  LINE 02 COLUMN 07 VALUE "No 2. exemplo do comando
103 -       "UNSTRING, digite um campo de 23 caracteres".
104         02  LINE 04 COLUMN 07 VALUE "no formato identico a xxx
105 -       " xxx xxx xxx ou xxx/xxx/xxx/xxx ou xxx*xxx*xxx*xxx".
106         02  LINE 07 COLUMN 10 VALUE "ENTRE COM O CAMPO:".
107         02  UU-REC LINE 07 COLUMN 29 PIC X(48) USING RECEPTOR-1
108             REVERSE-VIDEO.
109         02  LINE 09 COLUMN 20 VALUE "Resultado apos a execucao
110 -       ":".
111         02  LINE 11 COLUMN 20 VALUE "1. Campo:".
112         02  LINE 11 COLUMN 43 VALUE "Delimitador:".
113         02  LINE 11 COLUMN 60 VALUE "Contador:".
114         02  LINE 13 COLUMN 20 VALUE "2. Campo:".
115         02  LINE 13 COLUMN 43 VALUE "Delimitador:".
116         02  LINE 13 COLUMN 60 VALUE "Contador:".
117         02  LINE 15 COLUMN 20 VALUE "3. Campo:".
118         02  LINE 15 COLUMN 43 VALUE "Delimitador:".
119         02  LINE 15 COLUMN 60 VALUE "Contador:".
120         02  LINE 17 COLUMN 20 VALUE "4. Campo:".
121         02  LINE 17 COLUMN 43 VALUE "Delimitador:".
122         02  LINE 17 COLUMN 60 VALUE "Contador:".
123         02  LINE 19 COLUMN 20 VALUE "Pointer:".
124         02  LINE 19 COLUMN 43 VALUE "Tallying:".
125
126
127 PROCEDURE DIVISION.
128 INICIO.
129         ACCEPT DATA-SYS FROM DATE.
```

```
130         DISPLAY TELA.
131         DISPLAY (2 8) DD "/" MM "/" AA.
132 ROT-OPCAO.
133         DISPLAY (19 60) "<1>".
134         ACCEPT (19 61) VAGO WITH AUTO-SKIP.
135         ACCEPT ESC FROM ESCAPE KEY.
136         IF ESC = 06 GO TO FIM.
137         IF ESC = 01 GO TO INICIO.
138         IF ESC = 02 GO TO STRING-1.
139         IF ESC = 03 GO TO STRING-2.
140         IF ESC = 04 GO TO UNSTRING-1.
141         IF ESC = 05 GO TO UNSTRING-2.
142         GO TO ROT-OPCAO.
143 STRING-1.
144         MOVE 01 TO CONTADOR.
145         MOVE SPACES TO RECEPTOR, CAMPOS-RECEPTORES.
146         DISPLAY TELA2. DISPLAY TELA3. DISPLAY TELA3-A.
147         ACCEPT (06 41) CPO1 WITH AUTO-SKIP.
148         ACCEPT (08 41) CPO2 WITH AUTO-SKIP.
149         ACCEPT (10 41) CPO3 WITH AUTO-SKIP.
150         ACCEPT (12 41) CPO4 WITH AUTO-SKIP.
151         STRING CPO1 "/" CPO2 "/" CPO3 "/" CPO4 "/" DELIMITED BY
152             SPACES INTO RECEPTOR POINTER CONTADOR.
153         DISPLAY (15 02) "Resultado da concatenacao:".
154         DISPLAY (15 29) RECEPTOR.
155         SUBTRACT 1 FROM CONTADOR.
156         DISPLAY (17 02) "Contador: " CONTADOR.
157 E-OPCAO.
158         DISPLAY (24 69) "<1>".
159         ACCEPT (24 70) VAGO WITH AUTO-SKIP.
160         ACCEPT ESC FROM ESCAPE KEY.
161         IF ESC = 01 GO TO INICIO.
162         GO TO E-OPCAO.
163
164 STRING-2.
165         MOVE 01 TO CONTADOR.
166         MOVE SPACES TO RECEPTOR, CAMPOS-RECEPTORES.
167         MOVE SPACES TO CPO1, CPO2, CPO3.
168         DISPLAY TELA4 DISPLAY TELA3 DISPLAY TELA3-A.
169         DISPLAY (12 02) ".
170         ACCEPT (06 41) CPO1.
171         ACCEPT (08 41) CPO2.
172         ACCEPT (10 41) CPO3.
173         STRING CPO1 DELIMITED SPACE
174             CPO2 DELIMITED SPACE
```

```
175             CPO3 DELIMITED SPACE
176             INTO RECEPTOR POINTER CONTADOR.
177         DISPLAY (15 02) "Resultado da concatenacao:".
178         DISPLAY (15 29) RECEPTOR.
179         SUBTRACT 1 FROM CONTADOR.
180         DISPLAY (17 02) "Contador: " CONTADOR.
181 E-OPCAO.
182         DISPLAY (24 69) "<1>".
183         ACCEPT (24 70) VAGO WITH AUTO-SKIP.
184         ACCEPT ESC FROM ESCAPE KEY.
185         IF ESC = 01 GO TO INICIO.
186         GO TO EE-OPCAO.
187
188 UNSTRING-1.
189         MOVE SPACES TO RECEPTOR.
190         MOVE SPACES TO CPO1, CPO2, CPO3, CPO4.
191         DISPLAY TELA5. DISPLAY TELA3-A.
192         ACCEPT SS-REC.
193         UNSTRING RECEPTOR DELIMITED BY "/"
194         INTO CPO1, CPO2, CPO3, CPO4.
195         DISPLAY (11 30) CPO1.
196         DISPLAY (13 30) CPO2.
197         DISPLAY (15 30) CPO3.
198         DISPLAY (17 30) CPO4.
199 U-OPCAO.
200         DISPLAY (24 69) "<1>".
201         ACCEPT (24 70) VAGO WITH AUTO-SKIP.
202         ACCEPT ESC FROM ESCAPE KEY.
203         IF ESC = 01 GO TO INICIO.
204         GO TO U-OPCAO.
205
206 UNSTRING-2.
207         MOVE ZEROS TO SUB. PERFORM ZERAR 4 TIMES.
208         MOVE 01 TO CONT-POINTER, CONT-REG.
209         MOVE SPACES TO RECEPTOR-1.
210         DISPLAY TELA6. DISPLAY TELA3-A.
211         ACCEPT UU-REC.
212         UNSTRING RECEPTOR-1 DELIMITED BY SPACE OR "/" OR "*" INTO
213             RECEPT(1) DELIMITER DELM(1) COUNT CONT(1)
214             RECEPT(2) DELIMITER DELM(2) COUNT CONT(2)
215             RECEPT(3) DELIMITER DELM(3) COUNT CONT(3)
216             RECEPT(4) DELIMITER DELM(4) COUNT CONT(4)
217             POINTER CONT-POINTER
218             TALLYING IN CONT-REG.
219         SUBTRACT 1 FROM CONT-POINTER.
```

```
220         SUBTRACT 1 FROM CONT-REG.
221         DISPLAY (11 30) RECEPT(1)
222                 (13 30) RECEPT(2)
223                 (15 30) RECEPT(3)
224                 (17 30) RECEPT(4)
225                 (11 56) DELM(1)
226                 (13 56) DELM(2)
227                 (15 56) DELM(3)
228                 (17 56) DELM(4)
229                 (11 70) CONT(1)
230                 (13 70) CONT(2)
231                 (15 70) CONT(3)
232                 (17 70) CONT(4)
233                 (19 30) CONT-POINTER
234                 (19 55) CONT-REG.
235 UU-OPCAO.
236         DISPLAY (24 69) "<1>".
237         ACCEPT (24 70) VAGO WITH AUTO-SKIP.
238         ACCEPT ESC FROM ESCAPE KEY.
239         IF ESC = 01 GO TO INICIO.
240         GO TO UU-OPCAO.
241 FIM.
242         DISPLAY (01 01) ERASE.
243         DISPLAY (10 18) "FIM DE PROCESSAMENTO".
244         STOP RUN.
245 ZERAR.
246         ADD 1 TO SUB.
247         MOVE SPACES TO RECEPT(SUB), DELM(SUB).
248         MOVE ZEROS TO CONT(SUB).
249
```

## Explanação e Instruções Introduzidas no Programa

```
STRING CPO1 "/" CPO2 "/" CPO3 "/" CPO4 "/" DELIMITED
BY SPACES INTO RECEPTOR POINTER CONTADOR.
```

Ação: realiza a concatenação dos campos CPO1, CPO2, CPO3 e CPO4 no campo RECEPTOR separados por /.

```
STRING CPO1 DELIMITED SPACE
CPO2 DELIMITED SPACE
CPO3 DELIMITED SPACE
INTO RECEPTOR POINTER CONTADOR.
```

Ação: realiza a concatenação dos campos CPO1, CPO2 e CPO3 no campo RECEPTOR e a variável CONTADOR armazena a quantidade de caracteres processados.

```
UNSTRING RECEPTOR DELIMITED BY "/"
INTO CPO1, CPO2, CPO3, CPO4.
```

Ação: transfere o conjunto de caracteres do campo RECEPTOR que, está delimitado por /, para os campos CPO1, CPO2, CPO3 e CPO4.

```
UNSTRING RECEPTOR-1 DELIMITED BY SPACE OR "/" OR "*"
        INTO RECEPT(1) DELIMITER DELM(1) COUNT
        CONT(1) RECEPT(2) DELIMITER DELM(2) COUNT
        CONT(2) RECEPT(3) DELIMITER DELM(3) COUNT
        CONT(3) RECEPT(4) DELIMITER DELM(4) COUNT
        CONT(4) POINTER CONT-POINTER
        TALLYING IN CONT-REG.
```

Ação: transfere o conjunto de caracteres do campo RECEPTOR-1, que podem estar delimitados por espaços "/" ou "*", para os campos da tabela RECEPT(subscritor). A opção DELIMITER transfere os caracteres delimitadores para os campos da tabela DELM(subscritor). A opção COUNT armazena a quantidade de caracteres em CONT(subscritor), especificados em RECEPT(subscritor). A opção POINTER indica a posição do caractere em processamento do campo RECEPTOR-1 armazenando na variável CONT-POINTER. A opção TALLYING especifica que será armazenada em CONT-REG a soma do valor inicial com o número de campos receptores ativados.

# CAPÍTULO 10

## COMANDOS DE DEPURAÇÃO DINÂMICA

Inicialmente um programa COBOL pode conter erros de sintaxe e de lógica. Os erros de sintaxe, que descontinuam a compilação, são acusados pelo compilador. Uma vez eliminados tais erros, o programa é compilado e pode agora ter fim anormal, pela presença de erros de lógica, quando lançado em execução.

## INSTRUÇÕES READY TRACE E RESET TRACE

Esta fase, de depuração mais difícil, pode ser encurtada se soubermos quais parágrafos foram executados, em que ordem, e em que parágrafo se estava quando houve o fim anormal. Estas informações são providas pelo comando READY TRACE. Seu formato é:

```
READY TRACE.
```

e pode ser desativado por

**RESET TRACE.**

Quando o modo TRACE está ativado os nomes-de-procedimentos são listados na ordem em que são alcançados durante a execução. O RESET TRACE inibe esta impressão.

READY TRACE e RESET TRACE podem ser escritos em qualquer parte da PROCEDURE DIVISION, a partir da coluna 12.

Com a lista de nomes-de-procedimentos por onde o programa passou, o programador pode agora usar outro comando para saber valores de dados envolvidos em operações no parágrafo em que o programa parou.

Para esse fim existe o comando EXHIBIT.

## INSTRUÇÃO EXHIBIT

Formato:

**EXHIBIT NAMED** {[especificação-da-posição]
{identificador/literal/**ERASE**}} ... [UPON nome-mnemônico]

O comando EXHIBIT produz uma impressão dos valores dos literais indicados, ou itens-de-dados no formato nome-de-dado = valor. A especificação-da-posição e a frase UPON têm o mesmo efeito que o comando DISPLAY.

## DECLARATIVES E A SENTENÇA USE

A parte declarative fornece um método de incluir procedimentos que são executados não como parte do programa, mas sim quando ocorre uma condição que não pode normalmente ser testada pelo programador.

Ainda que o sistema verifique automaticamente tanto a criação de rótulos (labels) padrões e execute rotinas de recuperação de erro no caso de erros de entrada/saída, procedimentos adicionais podem ser especificados pelo programador COBOL.

Como esses procedimentos são executados somente em caso de erro na leitura ou na escrita, não podem aparecer na seqüência normal de declarações de procedimentos. Precisam ser escritos no início da PROCEDURE DIVISION em uma subdivisão chamada DECLARATIVES.

Procedimentos relacionados são precedidos por uma sentença USE que especifica suas funções. Uma seção DECLARA TIVES só termina com a ocorrência de outro nome-de-seção com o comando USE ou com as palavras-chaves END DECLARATIVES.

As palavras-chaves DECLARATIVES e END DECLARATIVES devem começar na Área A e ser seguidas por um ponto.

```
PROCEDURE DIVISION.
DECLARATIVES.
{nome-de-seção SECTION. USE sentença.
{nome-de-parágrafo. {sentença} ... } ... }
END DECLARATIVES.
```

A sentença USE define a aplicabilidade da seção associada ao arquivo.

Uma sentença USE, quando presente, deve seguir imediatamente o cabeçalho da seção na área do DECLARATIVES da PROCEDURE DIVISION e também ser seguido por um ponto e um espaço. O restante da seção deve consistir em zero, um ou mais parágrafos de procedimento que define os métodos a serem usados. A sentença USE em si nunca é executada; em vez disso, ela define as condições para a execução do procedimento USE.

O formato geral da sentença USE é:

```
USE AFTER STANDARD EXCEPTION/ERROR PROCEDURE
ON nome-de-arquivo ... /INPUT/OUTPUT/I-O/EXTEND.
```

As palavras EXCEPTION e ERROR podem ser usadas intercambiavelmente. A seção DECLARATIVES associada é executada (pelo mecanismo PERFORM) depois dos procedimentos de recuperação de E/S padrões para os arquivos designados, ou depois que surge a condição INVALID KEY ou AT END num comando desprovido destas cláusulas. Um nome-de-arquivo dado não pode ser associado com mais de uma seção DECLARATIVES.

Dentro de uma seção DECLARATIVES não deve haver nenhuma referência a nenhum procedimento não-declarative. Igualmente, na parte não-declarative não deve existir nenhuma referência a nomes-de-procedimentos que aparecem na seção DECLARATIVES, exceto os comandos PERFORM que referenciam a um comando USE e seus procedimentos.

Uma saída da seção DECLARATIVES é inserida pelo compilador seguindo o último comando na seção. Toda lógica do programa dentro da seção deve ser levada a este ponto de saída.

## Codificação do Programa

```
1   IDENTIFICATION  DIVISION.
2   PROGRAM-ID.     DEBUG.
3   ENVIRONMENT     DIVISION.
4   CONFIGURATION   SECTION.
5   SPECIAL-NAMES.  DECIMAL-POINT IS COMMA.
6   INPUT-OUTPUT SECTION.
7   FILE-CONTROL.
8       SELECT  ARQMAT      ASSIGN TO DISK
9               ORGANIZATION IS LINE SEQUENTIAL
10              FILE STATUS IS FS.
11      SELECT IMPRES ASSIGN TO PRINTER.
12  DATA DIVISION.
13  FILE SECTION.
14  FD  ARQMAT LABEL RECORD STANDARD
15              VALUE OF FILE-ID "ARQMAT".
16  01  REG-ARQMAT.
17      02  FD-CODIGO      PIC 9(05).
18      02  FD-NOME        PIC X(15).
19      02  FD-ESTOQUE     PIC 9(07).
```

```
20        02  FD-UNITARIO        PIC 9(05)V99.
21        02  FD-TOTAL           PIC 9(06)V99.
22    FD  IMPRES  LABEL RECORD OMITTED
23                LINAGE 55 TOP 6 BOTTOM 5.
24    01  REG-IMPRES             PIC X(132).
25    WORKING-STORAGE SECTION.
26    77  FS                     PIC X(02)      VALUE   SPACES.
27    77  AC-ESTOQUE             PIC 9(08)      VALUE   ZEROS.
28    77  AC-TOTAL               PIC 9(08)V99   VALUE   ZEROS.
29    77  CAL-MAJ                PIC ZZZ.ZZZ.ZZ9,99   VALUE   ZEROS.
30    77  CT-PAG                 PIC 9(02)      VALUE   ZEROS.
31    01  DATA-DO-SISTEMA.
32        02 ANO-SIS              PIC 9(02)      VALUE   ZEROS.
33        02 MES-SIS              PIC 9(02)      VALUE   ZEROS.
34        02 DIA-SIS              PIC 9(02)      VALUE   ZEROS.
35    01  CAB01.
36        02 FILLER               PIC X(10)      VALUE   SPACES.
37        02 FILLER               PIC X(06)      VALUE   "DATA:".
38        02 DATA-CAB01.
39            03 DIA-CAB01        PIC 99/        VALUE   ZEROS.
40            03 MES-CAB01        PIC 99/        VALUE   ZEROS.
41            03 ANO-CAB01        PIC 99         VALUE   ZEROS.
42        02 FILLER               PIC X(26)      VALUE   SPACES.
43        02 FILLER               PIC X(68)      VALUE   "RELATORIO DE MATERIAL E
44    -   "M ESTOQUE".
45        02 FILLER               PIC X(05)      VALUE   "PAG:".
46        02 PAG-CAB01            PIC ZZ9        VALUE   ZEROS.
47        02 FILLER               PIC X(06)      VALUE   SPACES.
48    01  CAB02.
49        02 FILLER               PIC X(14)      VALUE   SPACES.
50        02 FILLER               PIC X(17)      VALUE   "CODIGO".
51        02 FILLER               PIC X(24)      VALUE
52        "NOME DO PRODUTO".
53        02 FILLER               PIC X(27)      VALUE
54        "QUANTIDADE EM ESTOQUE".
55        02 FILLER               PIC X(21)      VALUE
56        "CUSTO UNITARIO".
57        02 FILLER               PIC X(29)      VALUE
58        "CUSTO TOTAL".
59    01  CAB03.
60        02 FILLER               PIC X(22)      VALUE   SPACES.
61        02 FILLER               PIC X(42)      VALUE
62        "QUANTIDADE DE PRODUTO".
63        02 FILLER               PIC X(34)      VALUE
64        "PRECO MEDIO".
```

```
65          02 FILLER                 PIC X(34)      VALUE
66          "PRECO TOTAL".
67      01  DETALHE.
68          02 FILLER                 PIC X(15)          VALUE  SPACES.
69          02 WS-CODIGO              PIC 9(05)          VALUE  ZEROS.
70          02 FILLER                 PIC X(11)          VALUE  SPACES.
71          02 WS-NOME                PIC X(15)          VALUE  SPACES.
72          02 FILLER                 PIC X(14)          VALUE  SPACES.
73          02 WS-ESTOQUE             PIC Z.ZZZ.ZZ9      VALUE  ZEROS.
74          02 FILLER                 PIC X(17)          VALUE  SPACES.
75          02 WS-UNITARIO            PIC ZZ.ZZ9,99      VALUE  ZEROS.
76          02 FILLER                 PIC X(09)          VALUE  SPACES.
77          02 WS-TOTAL               PIC ZZZ.ZZ9,99     VALUE  ZEROS.
78          02 FILLER                 PIC X(19)          VALUE  SPACES.
79      01  TOTAL.
80          02 FILLER                 PIC X(27)          VALUE  SPACES.
81          02 WS-TOTAL-PROD          PIC Z.ZZZ.ZZ9      VALUE  ZEROS.
82          02 FILLER                 PIC X(28)          VALUE  SPACES.
83          02 WS-MEDIO               PIC ZZZ.ZZ9,99     VALUE  ZEROS.
84          02 FILLER                 PIC X(23)          VALUE  SPACES.
85          02 WS-PRECO-TOT           PIC Z.ZZZ.ZZ9,99   VALUE  ZEROS.
86          02 FILLER                 PIC X(23)          VALUE  SPACES.
87      SCREEN SECTION.
88      01  TELA.
89          02 BLANK SCREEN.
90          02 LINE 01 COLUMN 20 VALUE " TESTE DE DEBUG E DECLARATIVES"
91              REVERSE-VIDEO BLINK.
92      PROCEDURE DIVISION.
93
94      DECLARATIVES.
95      VERIFICACAO SECTION.
96          USE AFTER STANDARD ERROR PROCEDURE ON ARQMAT.
97      ACAO.
98          DISPLAY (23 01) ".
99          DISPLAY (23 20) "INTERACAO COM DECLARATIVES".
100         IF FS = 10 PERFORM ROT-FIM STOP RUN.
101     END DECLARATIVES.
102
103     ROT-INICIO.
104         DISPLAY (01 01) ERASE.
105         OPEN INPUT ARQMAT.
106             IF FS NOT = "00
107                 IF FS NOT = "30"
108                     DISPLAY (23 35) "ARQUIVO NAO SE ENCONTRA NO DISCO"
109                     STOP RUN
```

```
110                 ELSE
111                     DISPLAY (23 35) "ARQUIVO ARQMAT DANIFICADO"
112                     DISPLAY (24 35) "CODIGO DO STATUS = " FS
113                     STOP RUN
114                 ELSE
115             NEXT SENTENCE.
116         OPEN OUTPUT IMPRES.
117         ACCEPT DATA-DO-SISTEMA FROM DATE.
118         MOVE DIA-SIS TO DIA-CAB01.
119         MOVE MES-SIS TO MES-CAB01.
120         MOVE ANO-SIS TO ANO-CAB01.
121         DISPLAY TELA. MOVE 1 TO LIN.
122     ROT-CAB.
123         MOVE SPACES TO REG-IMPRES.
124         WRITE REG-IMPRES BEFORE ADVANCING 1 LINE.
125         ADD 1 TO CT-PAG.
126         MOVE CT-PAG TO PAG-CAB01.
127         WRITE REG-IMPRES FROM CAB01 BEFORE ADVANCING 3 LINES.
128         WRITE REG-IMPRES FROM CAB02 BEFORE ADVANCING 2 LINES.
129     ROT-LER.
130         READ ARQMAT.
131         ADD FD-ESTOQUE     TO AC-ESTOQUE.
132         ADD FD-TOTAL       TO AC-TOTAL.
133         MOVE FD-CODIGO     TO WS-CODIGO.
134         MOVE FD-NOME       TO WS-NOME.
135         MOVE FD-ESTOQUE    TO WS-ESTOQUE.
136         MOVE FD-UNITARIO   TO WS-UNITARIO.
137         MOVE FD-TOTAL      TO WS-TOTAL.
138         ADD 1 TO LIN.
139         IF LIN = 24 DISPLAY (02 01) ERASE MOVE 2 TO LIN.
140         COMPUTE CAL-MAJ = FD-UNITARIO * 1,55.
141         EXHIBIT NAMED (LIN, 1) FD-NOME " " CAL-MAJ.
142         WRITE REG-IMPRES FROM DETALHE BEFORE ADVANCING 2 LINES
143             AT EOP PERFORM ROT-CAB.
144         GO TO ROT-LER.
145     ROT-FIM.
146         MOVE SPACES TO REG-IMPRES.
147         WRITE REG-IMPRES BEFORE ADVANCING PAGE.
148         ADD 1 TO CT-PAG.
149         MOVE CT-PAG TO PAG-CAB01.
150         WRITE REG-IMPRES FROM CAB01 BEFORE ADVANCING 3 LINE.
151         WRITE REG-IMPRES FROM CAB03 BEFORE ADVANCING 2 LINE.
152         COMPUTE WS-MEDIO = AC-TOTAL / AC-ESTOQUE.
153         MOVE AC-ESTOQUE TO WS-TOTAL-PROD.
154         MOVE AC-TOTALTO WS-PRECO-TOT.
```

```
155     WRITE REG-IMPRES FROM TOTAL BEFORE ADVANCING 2 LINE.
156     CLOSE ARQMAT IMPRES.
157     DISPLAY (24 35) "PROGRAMA DE PROCESSAMENTO".
158
159
```

## Explanação e Instruções Introduzidas no Programa

```
        DECLARATIVES.
        VERIFICACAO SECTION.
        USE AFTER STANDARD ERROR PROCEDURE ON ARQMAT.
ACAO.
        DISPLAY(23 01)".
        DISPLAY (23 20) "INTERACAO COM DECLARATIVES".
        IF FS = 10 PERFORM ROT-FIM STOP RUN.
END DECLARATIVES.
```

Ação: A instrução USE AFTER STANDARD ERROR PROCEDURE ON ARQMAT, assume a verificação do encerramento do arquivo de leitura ARQMAT, tendo em vista que na especificação da instrução READ foi omitida a condição AT END.

```
EXHIBIT NAMED (LIN, 1) FD-NOME " " CAL-MAJ.
```

Ação: exibe na linha e coluna especificadas os campos FD-NOME e CAL-MAJ bem como o conteúdo dos mesmos.

# CAPÍTULO 11

## INSTRUÇÃO COPY

A instrução COPY é usada para compor logicamente o texto de um arquivo em disco (diferente do arquivo-fonte) com a codificação fonte que serve de entrada para o compilador COBOL. O formato da instrução COPY é:

**COPY** nome-de-texto.

onde nome-de-texto é o nome do arquivo em disco.

O comando deve terminar com um ponto e a parte final da entrada em que aparece até a coluna 72, deve ficar em branco. Pode aparecer em qualquer lugar de um programa-fonte.

# SEGMENTAÇÃO

A facilidade de segmentação do programa é fornecida para permitir a execução de programas COBOL que sejam maiores que a memória física. Quando é usada a segmentação (isto é, quando qualquer cabeçalho de seção no programa contém um número de segmento), toda a PROCEDURE DIVISION deve ser dividida em seções. A cada seção é atribuído um número-de-segmento por um cabeçalho de seção, da forma:

nome-de-seção **SECTION** [número-de-segmento].

O número-de-segmento deve ser um inteiro com um valor de 0 a 99. Se o número-de-segmento for omitido, assume valor zero. As seções DECLARATIVES devem ter números-de-segmentos menores que 50. Todas as seções que têm o mesmo número-de-segmento constituem um único segmento de programa e devem aparecer juntas no programa fonte. Além disso, todos os segmentos com números menores que 50 devem aparecer juntos no início da PROCEDURE DIVISION.

Os segmentos com números de 0 a 49 são chamados de segmentos fixos, e residem sempre na memória durante a execução.

Segmentos com números maiores que 49 são chamados independentes. Cada segmento independente é tratado como um "OVERLAY" do programa. Um segmento independente está em seu estado inicial, quando o controle é passado para ele pela primeira vez durante a execução de um programa, e também quando o controle é passado para aquela seção (implícita ou explicitamente) de um outro segmento de número-de-segmento diferente.

Especificamente, um segmento independente está em seu estado inicial quando é alcançado pelo fim de um segmento independente diferente ou fixo.

A segmentação causa as seguintes restrições no uso dos comandos ALTER e PERFORM:

1. um comando GO TO em um segmento independente não pode se referir a nenhum comando ALTER em qualquer outro segmento;

2. um comando PERFORM em um segmento fixo pode ter dentro de seu parágrafo apenas:

   *a.* seções e/ou parágrafos totalmente contidos dentro de segmentos fixos, ou;

   *b.* seções e/ou parágrafos totalmente contidos em um único segmento independente.

3. um comando PERFORM em um segmento independente pode ter dentro de seu parágrafo somente:

   *a.* seções e/ou parágrafos totalmente contidos dentro de segmentos fixos, ou;

   *b.* seções e/ou parágrafos totalmente contidos dentro do mesmo segmento independente com o comando PERFORM.

## Codificação do Programa

```
1   IDENTIFICATION  DIVISION.
2   PROGRAM-ID.     COPSEG.
3   ENVIRONMENT     DIVISION.
4   CONFIGURATION   SECTION.
5   SPECIAL-NAMES.  DECIMAL-POINT IS COMMA.
6   INPUT-OUTPUTSECTION.
7   FILE-CONTROL.
8       SELECT      ARQ01   ASSIGN TO DISK
9                   ORGANIZATION LINE SEQUENTIAL
10                  FILE STATUS FS.
11      SELECT      ARQ02   ASSIGN TO DISK
12                  ORGANIZATION LINE SEQUENTIAL
13                  FILE STATUS FS.
14      SELECT      ARQ03   ASSIGN TO DISK
15                  ORGANIZATION LINE SEQUENTIAL
```

```
16                      FILE STATUS FS.
17  DATA                DIVISION.
18  FILE                SECTION.
19  FD  ARQ01           LABEL RECORD STANDARD
20                      VALUE OF FILE-ID "ARQ01".
21          COPY R-ARQ01.
22 C01      REG-ARQ01.
23 C        02 CAT-01               PIC 99.
24 C        02 NUM-01               PIC 9(04).
25 C        02 ART-01.
26 C            03 NUM-ART          PIC 9(03).
27 C            03 QUANT            PIC 9(04).
28  FD  ARQ02   LABEL RECORD STANDARD
29              VALUE OF FILE-ID "ARQ02".
30          COPY R-ARQ02.
31 C01      REG-ARQ02.
32 C        02  CAT-02             PIC 99.
33 C        02  NUM-02             PIC 9(04).
34 C        02 ART-02.
35 C            03 NUM-ART          PIC 9(03).
36 C            03 QUANT            PIC 9(04).
37  FD ARQ03    LABEL RECORD STANDARD
38              VALUE OF FILE-ID "ARQ03".
39          COPY R-ARQ03.
40 C01      REG-ARQ03.
41 C02      CAT-03                 PIC 99.
42 C02      NUM-03                 PIC 9(04).
43 C02      ART-03.
44 C            03  NUM-ART        PIC 9(03).
45 C            03 QUANT            PIC 9(04).
46 C02 DATA-ARQ.
47 C            03 DIA-S            PIC 9(02).
48 C            03 MES-S            PIC 9(02).
49 C            03 ANO-S            PIC 9(02).
50 C
51  WORKING-STORAGE SECTION.
52  77  FS                         PIC X(02)    VALUE  SPACES.
53  77  CT-ARQ01                   PIC 9(04)    VALUE  ZEROS.
54  77  CT-ARQ02                   PIC 9(04)    VALUE  ZEROS.
55  77  CT-ARQ03                   PIC 9(04)    VALUE  ZEROS.
56  77  CT-DUPLO                   PIC 9(04)    VALUE  ZEROS.
57  01  CH-FIM.
58          02 CH-ARQ01             PIC X(04)    VALUE  SPACES.
59          02 CH-ARQ02             PIC X(04)    VALUE  SPACES.
60  01  DATA-SYS.
```

```
61          02 ANO              PIC 9(02)     VALUE   ZEROS.
62          02 MES              PIC 9(02)     VALUE   ZEROS.
63          02 DIA              PIC 9(02)     VALUE   ZEROS.
64      PROCEDURE DIVISION.
65      PRINCIPAL SECTION 01.
66      E-INICIALIZA.
67          PERFORM E-INICIO THRU S-INICIO.
68          PERFORM E-PROCESSA THRU S-PROCESSA.
69          PERFORM E-FINAL THRU S-FINAL.
70      S-INICIALIZA.
71          STOP RUN.
72
73      E-INICIO SECTION 50.
74
75      INICIALIZANDO.
76          DISPLAY (01 01) ERASE.
77          OPEN INPUT ARQ01.
78          IF FS = 00 NEXT SENTENCE
79          ELSE DISPLAY (23 30) "ERRO NO ARQ01 " FS STOP RUN.
80          OPEN INPUT ARQ02.
81          IF FS = 00 NEXT SENTENCE
82          ELSE DISPLAY (23 30) "ERRO NO ARQ02 " FS STOP RUN.
83          OPEN OUTPUT ARQ03.
84          IF FS = 00 NEXT SENTENCE
85          ELSE DISPLAY (23 30) "ERRO NO ARQ03 " FS STOP RUN.
86          ACCEPT DATA-SYS FROM DATE.
87          MOVE DIA TO DIA-S.
88          MOVE MES TO MES-S.
89          MOVE ANO TO ANO-S.
90          READ ARQ01 ADD 1 TO CT-ARQ01 MOVE NUM-01 TO CH-ARQ01.
91          READ ARQ02 ADD 1 TO CT-ARQ02 MOVE NUM-02 TO CH-ARQ02.
92      S-INICIO.
93          EXIT.
94
95      E-PROCESSA SECTION 80.
96
97      PROCESSANDO.
98          IF CH-ARQ01 > CH-ARQ02
99              MOVE REG-ARQ02 TO REG-ARQ03
100             PERFORM E-GRAVA THRU S-GRAVA
101             PERFORM E-LER2 THRU S-LER2
102         ELSE
103             IF CH-ARQ01 < CH-ARQ02
104                 MOVE REG-ARQ01 TO REG-ARQ03
105                 PERFORM E-GRAVA THRU S-GRAVA
```

```
106                 PERFORM E-LER1 THRU S-LER1
107             ELSE
108                 ADD 1 TO CT-DUPLO
109                 MOVE REG-ARQ01 TO REG-ARQ03
110                 PERFORM E-GRAVA THRU S-GRAVA
111                 PERFORM E-LER1 THRU S-LER1
112                 PERFORM E-LER2 THRU S-LER2.
113         IF CH-FIM = HIGH-VALUE GO TO S-PROCESSA
114             ELSE GO TO PROCESSANDO.
115 E-LER1.
116         READ ARQ01 AT END
117             MOVE HIGH-VALUE TO CH-ARQ01
118             GO TO S-LER1.
119         ADD 1 TO CT-ARQ01.
120         MOVE NUM-01 TO CH-ARQ01.
121 S-LER1.
122         EXIT.
123 E-LER2.
124         READ ARQ02 AT END
125             MOVE HIGH-VALUE TO CH-ARQ02
126             GO TO S-LER2.
127         ADD 1 TO CT-ARQ02.
128         MOVE NUM-02 TO CH-ARQ02.
129 S-LER2.
130         EXIT.
131
132 E-GRAVA.
133         WRITE REG-ARQ03.
134         ADD 1 TO CT-ARQ03.
135 S-GRAVA.
136         EXIT.
137 S-PROCESSA.
138         EXIT.
139
140 E-FINAL SECTION 90.
141
142 FINALIZANDO.
143         DISPLAY (10 30) "REGISTROS LIDOS EM ARQ01 = " CT-ARQ01.
144         DISPLAY (12 30) "REGISTROS LIDOS EM ARQ02 = " CT-ARQ02.
145         DISPLAY (14 30) "REGISTROS GRAVADOS EM ARQ03 = " CT-ARQ03.
146         DISPLAY (16 30) "REGISTROS DUPLOS..... = " CT-DUPLO.
147         DISPLAY (20 30) "Fim de Programa".
148         CLOSE ARQ01 ARQ02 ARQ03.
149 S-FINAL.
150         EXIT.
```

## Explanação e Instruções Introduzidas no Programa

```
COPY R-ARQ01.
COPY R-ARQ02.
COPY R-ARQ03.
```

Ação: os comandos COPY pegam as descrições completas dos registros REG-ARQ01, REG-ARQ02 e REG-ARQ03 que estão armazenadas nos arquivos R-ARQ01, R-ARQ02 e R-ARQ03, respectivamente. As linhas do programa-fonte que são trazidas por um COPY aparecem marcadas por um C ao lado da numeração.

A grande vantagem do COPY é que quando existem vários programas utilizando a mesma descrição de registro (ora como saída, ora como entrada), definimos este registro uma só vez para que o mesmo possa ser copiado por todos os programas.

```
PRINCIPAL SECTION 01.
```

Ação: é o segmento fixo do programa. Permite controlar a lógica geral enquanto os segmentos independentes estão sendo executados.

```
E-INICIO SECTION 50.
```

Ação: é o primeiro segmento independente. Este segmento é executado sem destruir o segmento fixo PRINCIPAL SECTION 01.

```
E-PROCESSA SECTION 80.
```

Ação: é o segundo segmento independente. Este segmento é carregado na memória, destruindo o segmento independente anterior e executando o mesmo até que sejam processados todos os registros existentes nos arquivos de entrada.

```
E-FINAL SECTION 90.
```

Ação: é o último segmento do programa. Este segmento é sobreposto ao segmento independente anterior e executado para realizar os procedimentos finais do programa.

# APÊNDICE A

## MENSAGENS DE ERRO

Este apêndice lista as mensagens de erro que podem ser detectadas durante a compilação e execução de um programa COBOL.

**Erros em tempo de compilação**

Os erros em tempo de compilação podem ser:

1. Erro de entrada de comando e erro de entrada/saída do sistema operacional. Estes erros são mostrados durante a compilação. Quando aparecer uma dessas mensagens, basta corrigir e recompilar.

2. Erros de sintaxe no programa COBOL fonte. Estas mensagens são colocadas no final da listagem do arquivo e são apresentadas também no vídeo. As mensagens consistem em:

- número da linha do programa-fonte, com 4 dígitos seguido por (:);

- uma explanação do erro. Se a explanação iniciar por um /F/ (uso incoerente do arquivo) ou um /W/ (advertência), então a mensagem é apenas uma advertência, se não, o erro é um erro sério que previne quando da execução do programa objeto.

No final da compilação é exibida uma mensagem mostrando o total de números de erros e advertências. Esta característica permite a realização da correção do programa-fonte.

## Lista de Mensagens de Erro dos Comandos de Entrada e Saída (E/S) do Sistema Operacional

?BAD FILENAME: um nome de arquivo não está especificado de acordo com as regras do sistema operacional.

?BAD SWITCH:/X: existe definição de um parâmetro Switch que o compilador não reconhece, (parágrafo SPECIAL-NAMES).

?CAN'T CREATE FILE: um arquivo de saída não foi aberto. Por exemplo, o arquivo de saída está protegido contra gravação.

?COMMAND ERROR: "X": existe um caractere (X) inválido no comando. Por exemplo, um nome de arquivo contém um @.

?COMPILER ERROR IN PHASE N AT ADDRESS: geralmente acontece por causa de um programa-fonte, compilador ou arquivo Overlay danificado. Nos dois últimos casos tentar com a cópia do compilador e seus Overlay.

?DISK X FULL: o disco do drive especificado está cheio. Se X estiver ausente, a mensagem se refere ao drive corrente.

?FILE NOT FOUND: o nome de arquivo especificado para entrada não existe.

?MEMORY FULL: ocorre quando a memória é insuficiente para todos os símbolos e outras informaçõoes obtidas do programa-fonte. Indica que o programa é grande e deve ser reduzido em tamanho ou dividido em módulos para serem compilados separadamente.

?OVERLAY N NOT FOUND: um dos arquivos Overlay do compilador COBOL (COBOLn.OVR) não se encontra no disco. Recompilando, depois de providenciar a gravação do arquivo Overlay em falta, elimina-se o problema.

## Erros de Sintaxe do Programa

A FILE-ID NAME IS UNDEFINED: um nome de dado especificado na cláusula VALUE OF FILE-ID não está definido.

A PARAGRAPH DECLARATION IS REQUIRED HERE: uma declaração EXIT não está seguida por uma Section ou parágrafo.

AREA A NOT BLANK IN CONTINUATION LINE: um caractere que não é o branco foi encontrado na área A na continuação de linha.

AREA-A VIOLATION; RESUMPTION AT NEXT PARAGRAPH / SECTION / DIVISION / VERB: uma entrada iniciada nas colunas de 8 a 12 não pode ser interpretada como nome de divisão, nome de seção, nome de parágrafo, descrição de arquivo ou um nível 01 ou 77.

CLAUSES OTHER THAN VALUE DELETED: na descrição de um dado de item de nível 88 foi mencionada uma outra cláusula que não o VALUE IS.

ELEMENT LENGTH ERROR: o comprimento de um literal é maior que 120 caracteres; ou um literal numérico é maior que 18 dígitos; ou um nome/identificador é maior do que 30 caracteres.

ERRONEOUS FILENAME IS IGNORED: uma entrada que não foi declarada com um nome de arquivo aparece onde um nome de arquivo é requisitado.

ERRONEOUS QUALIFICATION; LAST DECLARATION USED: a qualificação usada com um nome de dado está incorreto ou não é único.

ERRONEOUS SUBSCRIPTING; STATEMENT DELETED: falta ou há erro de subscrição em um nome de dado.

EXCESSIVE LITERAL POOL OR DISPLAY STRING LENGTH: o comprimento total do literal contido dentro de um parágrafo simples é maior do que 4096 bytes.

EXCESSIVE NUMBER OF FILES/4 KB WORKING-STORAGE BLOCKS: o total dos arquivos declarados + o tamanho da WORKING-STORAGE SECTION dividido por 4 com arredondamento para mais + as entradas de nível 01 e 77 da LINKAGE SECTION é maior do que 14.

EXCESSIVE OCCURS NESTING IS IGNORED: definição de mais de três níveis da cláusula OCCURS.

EXCESSIVE SEGMENT NUMBER: uma seção de um programa segmentado contém um número maior do que 99.

EXCESSIVE SEGMENT NUMBER IN DECLARATIVES: uma seção da Seção Declaratives contém um número de seção maior do que 49.

FILE CHARACTER CONFLICT: no formato 3 da instrução ACCEPT, SPACE-FILL e ZERO-FILL estão especificados.

FRACTIONAL EXPONENT OR NEGATIVE SCALED BASE (99P): no comando COMPUTE, o expoente é um literal numérico com um ponto decimal ou um item numérico descrito com um dígito à direita de um ponto decimal; ou a Picture de uma base de exponenciação contém um caractere P como dígito mais à direita.

GROUP SIZE GREATER THAN 4095; LENGTH SET TO 1: o tamanho do item de um nível diferente de 01 foi definido com um tamanho maior do que 4095 bytes.

ILLEGAL CHARACTER: foi detetado um caractere inválido.

ILLEGAL COPY FILENAME: é inválido o nome de arquivo da declaração COPY.

ILLEGAL MOVE OR COMPARISON IS DELETED: os operandos da instrução MOVE ou uma relação condicional são incompatíveis.

IMPERATIVE STATEMENT REQUIRED. STATEMENT DELETED: uma estrutura condicional está contida dentro de uma outra.

IMPROPER CHARACTER IN COLUMN 7: foi detectado um caractere inválido na coluna 7.

IMPROPER PICTURE. PIC X ASSUMED: foi detectado uma PIC inválida.

IMPROPER PONTUACTION: pontuação incorreta. Por exemplo, uma vírgula ou um ponto não foi seguido por um espaço.

IMPROPER REDEFINITION IGNORED: um nome de dado especificado na cláusula REDEFINES não possui o mesmo nível do nome de dado corrente.

IMPROPERLY FORMED ELEMENT: erro de sintaxe. Por exemplo, você pode ter terminado uma palavra com um hífen ou usado vários pontos decimais em um literal numérico.

INCOMPLETE (OR TOO LONG) STATEMENT DELETED: a instrução não está completa ou então o compilador não conseguiu interpretar o procedimento por ser muito grande.

INDEXED/RELATIVE REQUIRES DISK ASSIGNMENT: um arquivo assinalado para a PRINTER foi descrito com organização indexada ou relativa.

INVALID KEY SPECIFICATION: o campo-chave de um arquivo indexado ou relativo não pode ser subscritado, ou está inconsistente com a organização do arquivo na classe ou USAGE. Esta mensagem é expressa quando a instrução OPEN for processada.

INVALID QUOTED LITERAL: um literal de comprimento zero, construção imprópria, ou faltou fechar aspas.

INVALID SELECT-SENTENCE: a sintaxe da sentença SELECT no parágrafo FILE-CONTROL é incorreta.

INVALID VALUE IGNORED: o valor definido na frase VALUE IS não está correto como literal.

JUSTIFICATION CONFLICT: no formato 3 da instrução ACCEPT, estão especificados LEFT-JUSTIFY e RIGHT-JUSTIFY.

KEY DECLARATION OF THIS FILE IS NOT CORRECT: falta a cláusula RELATIVE KEY para um arquivo relativo, ou falta a cláusula RECORD KEY para um arquivo indexado.

KEYS MAY ONLY APPLY TO AN INDEXED/RELATIVE FILE: foi especificada a cláusula RELATIVE KEY ou RECORD KEY para um arquivo de organização seqüencial ou seqüencial em linha.

LITERAL TRUNCATED TO SIZE OF ITEM: o literal definido na frase VALUE IS é maior do que a declaração do nome-de-dado.

MISORDERED/REDUNDANT SECTION PROCESSED AS IS: a seção da IDENTIFICATION, ENVIRONMENT ou DATA DIVISION se encontra fora de ordem ou repetida.

NAME OMITTED; ENTRY BYPASSED: falta o nome de dado em uma descrição de entrada.

NON-CONTIGUOUS SEGMENT DISALLOWED: duas seções com o mesmo número, maior que 49, estão separados por uma ou mais seções com número diferente.

NO-PICTURE; ELEMENTARY ITEM ASSUMED TO BE BINARY: um item elementar não foi definido com PICTURE.

OCCURS DISALLOWED AT LEVEL 01/77, OR COUNT TOO HIGH: a cláusula OCCURS aparece na descrição de um nível 01 ou 77; ou o número de ocorrências especificado é maior que 1023.

OMITTED WORD 'SECTION' IS ASSUMED HERE: falta a definição da palavra SECTION em uma seção da DATA DIVISION.

PROCEDURE-NAME IS UNRESOLVABLE: a referência para um nome de seção ou parágrafo não está qualificada ou não é única.

PROCEDURE RANGE NOT IN CURRENT SEGMENT: o comando PERFORM em uma seção com o número de nível maior que 49 faz referência a um parágrafo em uma seção com o número diferente maior que 49.

PROCEDURE RANGE SPANS SEGMENTS: procedimentos enfileirados (parágrafo-1 THRU parágrafo-2) mencionados no comando PERFORM contêm parágrafos das seções com número diferente maior que 49.

REDUNDAND FD PROCESSED AS IS: o mesmo nome de arquivo aparece em mais de uma descrição.

REWRITE VALID ONLY FOR A DISK FILE: o nome de arquivo para a instrução REWRITE é um arquivo assinalado para impressora.

SEMANTICAL ERROR IN SCREEN DESCRIPTION: esta mensagem pode ser causada pelos seguintes problemas:

- a SCREEN SECTION não inicia a descrição com um nível 01;
- o nível 01 não especifica o nome da tela;
- o item de grupo da tela foi descrito com uma cláusula permitida somente para itens elementares;
- na descrição do item elementar da tela falta a cláusula FROM, TO, USING ou VALUE;
- a descrição de um item da tela contém dados inconsistentes (tais como USING e VALUE).

SIGN CLAUSE IGNORED FOR UNSIGNED ITEM: na PICTURE de um campo numérico descrito com USAGE IS DISPLAY sem sinal, aparece a cláusula SIGN IS.

SINGLE-SPACING ASSUMED DUE TO IMPROPER ADVANCING COUNT: o operando da frase BEFORE ou AFTER da instrução WRITE não é numérico, ou não está no intervalo 0-120.

SOURCE BYPASSED UNTIL NEXT FD/SECTION: um erro na descrição do arquivo impede mais análises.

STATEMENT DELETED BECAUSE INTEGRAL ITEM IS REQUIRED: um item numérico de PICTURE cujo dígito está à direita do ponto decimal está sendo usado onde um inteiro é exigido.

STATEMENT DELETED BECAUSE OPERAND IS NOT A FILENAME: o nome que aparece como nome de arquivo não pode ser declarado como tal.

STATEMENT DELETED DUE TO ERRONEOUS SINTAX: erro de sintaxe.

STATEMENT DELETED DUE TO NON-NUMERIC OPERAND: um item alfanumérico ou alfanumérico editado está sendo utilizado como um operando de uma operação aritmética; um item numérico editado está sendo utilizado como um operando que não é o resultado; ou um número contém mais que 18 dígitos.

SUBSCRIPT 0 OR OVER MAX. NO OCCURRENCES; 1 USED: o literal usado como subscritor está em desacordo com o campo associado à cláusula OCCURS, ou o subscritor contém um valor zero ou ultrapassou o limite especificado da tabela.

SUBSCRIPT OR INDEX-NAME IS NOT UNIQUE: um nome que requer qualificação é usado como um subscritor.

SINTAX ERROR IN SCREEN DESCRIPTION: um item de descrição da tela contém uma cláusula, não reconhecida, formulada indevidamente ou redundante.

UNRECOGNIZABLE ELEMENT IS IGNORED: falta uma palavra chave, ou um nome de dado ou um nome de procedimento não foi identificado.

USING-LIST ITEM LEVEL MUST BE 01/77: a lista do USING usado no cabeçalho PROCEDURE DIVISION não foi declarado no nível 01 nem 77.

VALUE DISALLOWED - OCCURS / REDEFINES / TYPE / SIZE CONFLICT: a cláusula VALUE está especificada para um item de dado descrito com (ou incluído dentro de um item descrito com) a cláusula OCCURS ou REDEFINES; ou o literal da cláusula VALUE não é compatível com a PICTURE do item.

VALUE OF FILE-ID REQUIRED: a cláusula VALUE OF FILE-ID não foi especificada na declaração do arquivo assinalado para DISK.

VARYING ITEM MAY NOT BE SUBSCRIPTED: o item de dado controlado pela frase VARYING de um PERFORM está subscritado.

## Erros no Manuseio de Arquivos

/F/ FILE NEVER CLOSED: não existe a instrução CLOSE para o arquivo.

/F/ FILE NEVER OPENED: não existe a instrução OPEN para o arquivo.

/F/ INCONSISTENT READ USAGE: existe uma instrução OPEN INPUT, mas não há procedimento READ; ou vice-versa.

/F/ INCONSISTENT WRITE USAGE: existe uma instrução OPEN OUTPUT, mas não há um procedimento WRITE; ou vice-versa.

## Erros de Advertência

/W/ BLANK WHEN ZERO IS DISALLOWED: a frase BLANK WHEN ZERO aparece na descrição de um item alfanumérico ou alfanumérico editado.

/W/ DATA DIVISION ASSUMED HERE: falta o cabeçalho DATA DIVISION.

/W/ DATA RECORDS CLAUSE WAS INACCURATE: o nome de registro definido na cláusula DATA RECORDS não está de acordo com a descrição do registro.

/W/ ERRONEOUS RERUN-ENTRY IS IGNORED: a cláusula RERUN do parágrafo I-O CONTROL contém um erro de sintaxe.

/W/ FD-VALUE IGNORED SINCE LABELS ARE OMITTED: a cláusula VALUE OF FILE-ID é usada na descrição de um arquivo assinalado para PRINTER.

/W/ FILE SECTION ASSUMED HERE: falta o cabeçalho FILE SECTION.

/W/ INVALID BLOCKING IS IGNORED: a cláusula BLOCK da FD contém um erro.

/W/ INVALID RECORD SIZE(S) IGNORED: a cláusula RECORD da FD contém um erro.

/W/ 'LABEL RECORD STANDARD' REQUIRED: a frase LABEL RECORD(S) STANDARD não está presente na FD de um arquivo assinalado para DISK.

/W/ LABEL RECORDS OMITTED ASSUMED FOR PRINTER FILE: falta a cláusula LABEL RECORD OMITTED em uma descrição de arquivo assinalado para PRINTER.

/W/ LEVEL 01 ASSUMED: o início da descrição de um registro é um nível diferente de 01.

/W/ RECORD ASSUMED AFTER PROCEDURE-NAME DEFINITION: a definição de seção ou de um parágrafo não termina com um ponto.

/W/ PICTURE IGNORED FOR INDEX ITEM: um item de dado descrito com a frase USAGE IS INDEX possui a palavra PICTURE.

/W/ PROCEDURE DIVISION ASSUMED HERE: falta o cabeçalho PROCEDURE DIVISION.

/W/ RECORD MAX DISAGREES WITH RECORD CONTAINS; LATTER SIZES PREVAIL o tamanho do registro especificado na cláusula RECORD CONTAINS de uma FD está em desacordo com o tamanho da descrição do registro.

/W/ REDUNDANT CLAUSE IGNORED: a mesma cláusula está especificada mais de uma vez na descrição do arquivo.

/W/ RIGHT PARENTHESIS REQUIRED AFTER SUBSCRIPTS: falta o fecha parênteses do subscritor.

/W/ TERMINAL PERIOD ASSUMED ABOVE: a descrição de um item ou parágrafo não termina com um ponto.

/W/ WORKING-STORAGE ASSUMED HERE: falta o cabeçalho WORKING-STORAGE SECTION.

## Erros em Tempo de Execução

CURSOR POSITION: tentativa de posicionar o cursor além do limite da linha ou coluna da tela. O formato 3 ou 4 do Accept ou um Display com especificação da posição ou nome de tela é o responsável pelo erro. Se for uma tela a ser exibida ou aceitada, um ou mais campos da tela inicia além do limite máximo da linha ou coluna.

DATA UNAVAILABLE: tentativa de referenciar um campo do registro de um arquivo que não foi aberto ou que encontrou a condição AT END.

DELETE; NO READ: tentativa de deletar um registro de um arquivo com acesso seqüencial quando a última operação READ não foi executada ou bem sucedida.

FILE LOCKED: tentativa de abrir um arquivo após a frase CLOSE WITH LOCK.

GO TO (NOT SET): tentativa de executar uma instrução GO TO que não foi alterada para referenciar um destinatário.

ILLEGAL DELETE: arquivo indexado ou relativo não aberto em I-O.

ILLEGAL READ: tentativa de ler um arquivo que não foi aberto no modo I-O ou INPUT.

ILLEGAL REWRITE: tentativa de regravar um registro que não foi aberto no modo I-O.

ILLEGAL START: arquivo não aberto em INPUT ou I-O.

ILLEGAL WRITE: tentativa de gravar um registro de um arquivo de acesso seqüencial que não foi aberto no modo OUTPUT, ou no modo OUTPUT ou I-O para arquivo com acesso randômico ou dinâmico.

INPUT/OUTPUT: um erro de I/O irrecuperável.

NEED MORE MEMORY: o arquivo indexado terminou anormalmente por causa de insuficiência de alocação de memória dinamicamente.

NON-NUMERIC DATA: quando o conteúdo de um item numérico não está de acordo com a PICTURE, este tipo de erro pode ocorrer.

OBJ. CODE ERROR: uma instrução não definida do programa objeto foi encontrado.

PERFORM OVERLAP: seqüência ilegal de PERFORM'S, como por exemplo, quando o parágrafo A é performado e outro PERFORM A é inicializado sobrepondo ao primeiro.

REDUNDANT OPEN: tentativa de abrir um arquivo que já está aberto.

REWRITE; NO READ: tentativa de regravar um registro com arquivo de modo de acesso seqüencial quando a última operação READ não foi executada ou bem sucedida.

SEG mn LOAD ERR: ocorrência de erro na tentativa de carregar um segmento independente.

SUBSCRIPT FAULT: o subscritor tem um valor ilegal. Este erro pode ser causado pelo indexador ou subscritor cujo valor é menor que 1.

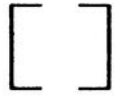
# APÊNDICE B

## FORMATO GERAL DO COBOL

Palavras em letras maiúsculas, sublinhadas, são obrigatórias.

Palavras em letras maiúsculas, não sublinhadas são opcionais.

Palavras em letras minúsculas, são codificadas pelo usuário.

Colchetes [ ] ou  indicam palavras e frases opcionais.

Chaves { } ou  indicam a possibilidade de escolha entre duas, ou mais formas possíveis.

Reticências, ..., indicam que o elemento precedente pode ser repetido.

## Identification Division

**PROGRAM-ID.** Nome do programa.

[**AUTHOR.** Nome do programador.]

[**INSTALLATION.** Local do sistema.]

[**DATE-WRITTEN.** Data em que foi escrito o programa.]

[**DATE-COMPILED.** Data de compilacao.]

[**SECURIY.** Especificações de proteção.]

## Environment Division

[**CONFIGURATION SECTION**].

[**SOURCE COMPUTER**. Nome do computador WITH **DEBUGGING MODE**]

[**OBJECT COMPUTER**. Nome do computador,

[**MEMORY** SIZE inteiro { **WORDS** / **CHARACTERS** / **MODULES** } ]

[PROGRAM COLLATING **SEQUENCE** IS **ASCII**] ]

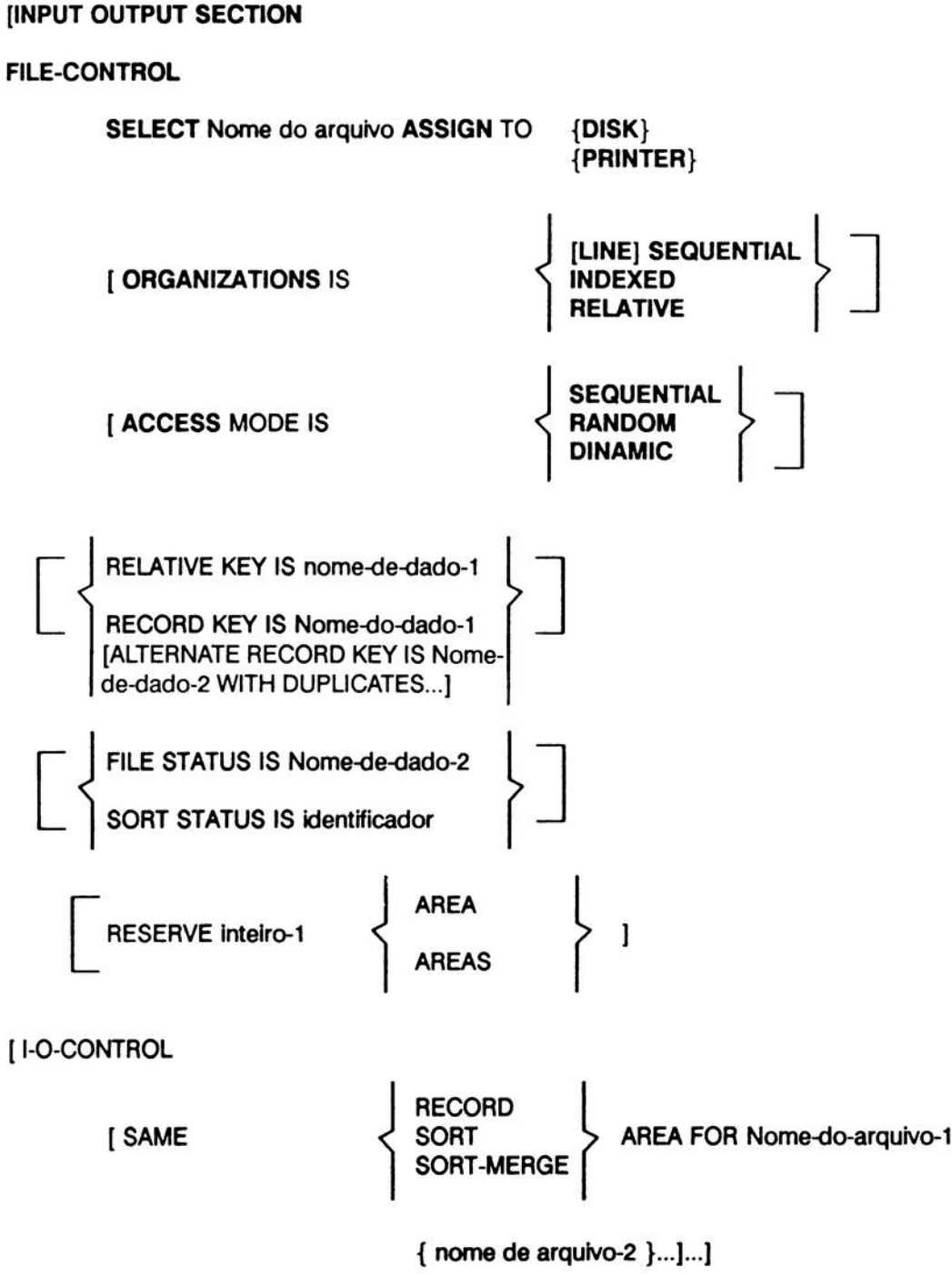
[SPECIAL-NAMES.

[**DECIMAL POINT** IS **COMMA**]

[**CURRENCY SIGN** IS literal]

[**PRINTER IS** mnemonico]

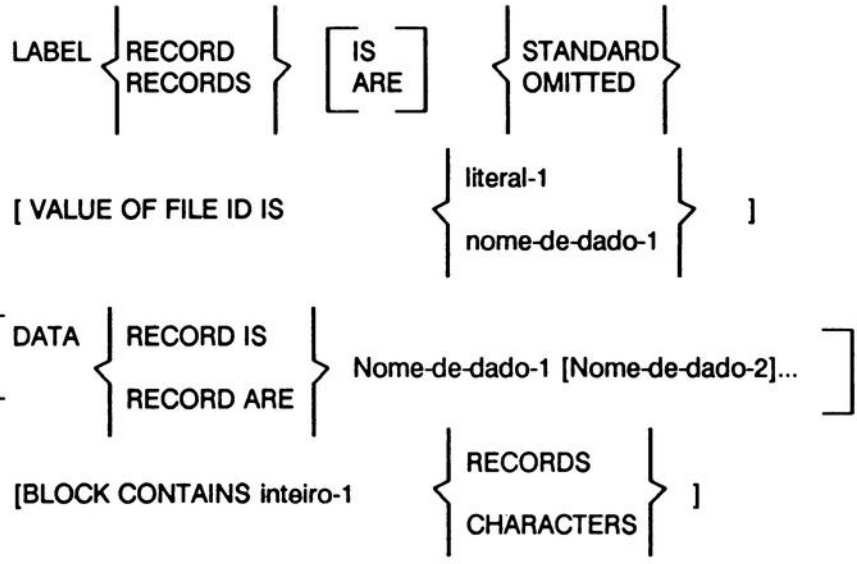
[**ASCII** IS {**STANDARD 1**} ] ]
{**NATIVE** } ]


[INPUT OUTPUT SECTION
FILE-CONTROL
SELECT Nome do arquivo ASSIGN TO
{DISK}
{PRINTER}
[ ORGANIZATIONS IS
[LINE] SEQUENTIAL
INDEXED
RELATIVE
[ ACCESS MODE IS
SEQUENTIAL
RANDOM
DINAMIC
RELATIVE KEY IS nome-de-dado-1
RECORD KEY IS Nome-do-dado-1
[ALTERNATE RECORD KEY IS Nome-de-dado-2 WITH DUPLICATES...]
FILE STATUS IS Nome-de-dado-2
SORT STATUS IS identificador
RESERVE inteiro-1
AREA
AREAS
]
[ I-O-CONTROL
[ SAME
RECORD
SORT
SORT-MERGE
AREA FOR Nome-do-arquivo-1
{ nome de arquivo-2 }...]...]

## Data Division

[ FILE SECTION

[ FD Nome-de-arquivo




[ CODE SET IS ASCII ]

{ entrada de descricao de registro }...]...

[ SD Nome-de-arquivo

[RECORD-clausula]
[DATA-RECORD(S)-clausula]
[VALUE-OF-clausula]

{ entrada de descricao de registro }...]...]

**[ WORKING-STORAGE SECTION**

[ entrada de descrição de nivel 77
entrada de descrição de registro ... ]

## FORMATO GERAL PARA ENTRADA DE DESCRIÇÃO DE DADOS

### Formato 1

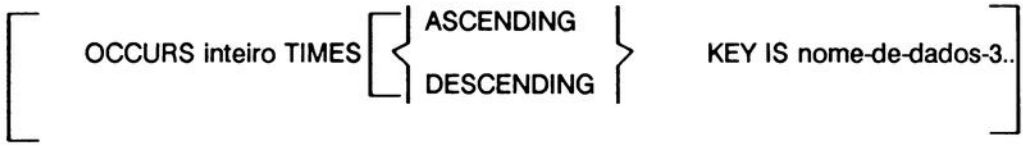
número de nível { nome-de-dado-1 / FILLER }

[ REDEFINES nome-de-dados-2 ]

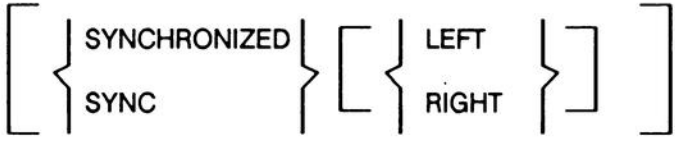
[ { PICTURE / PIC } IS cadeia de caracteres ]

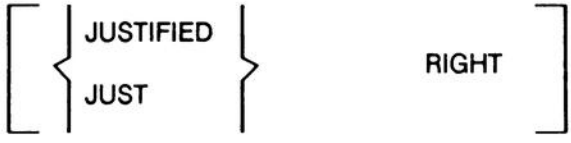
[ USAGE IS { { COMPUTATIONAL / COMP } / { COMPUTATIONAL-3 / COMP-3 } / DISPLAY / INDEX } ]

[ SIGN IS { LEADING / TRAILING } [ SEPARATE CHARACTER ] ]




[ INDEXED BY nome-de-indice-1 [nome-de-indice-2] .. ] ]




[ BLANK WHEN ZERO ]

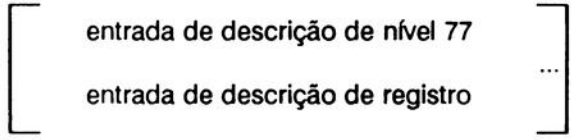
[ VALUE IS literal ]

## Formato 2

88 nome-de-condição **VALUE** IS literal-1 [ literal-2...]

88 nome-de-condição **VALUE** ARE literal-1 **THRU** literal 2

**[LINKAGE SECTION**



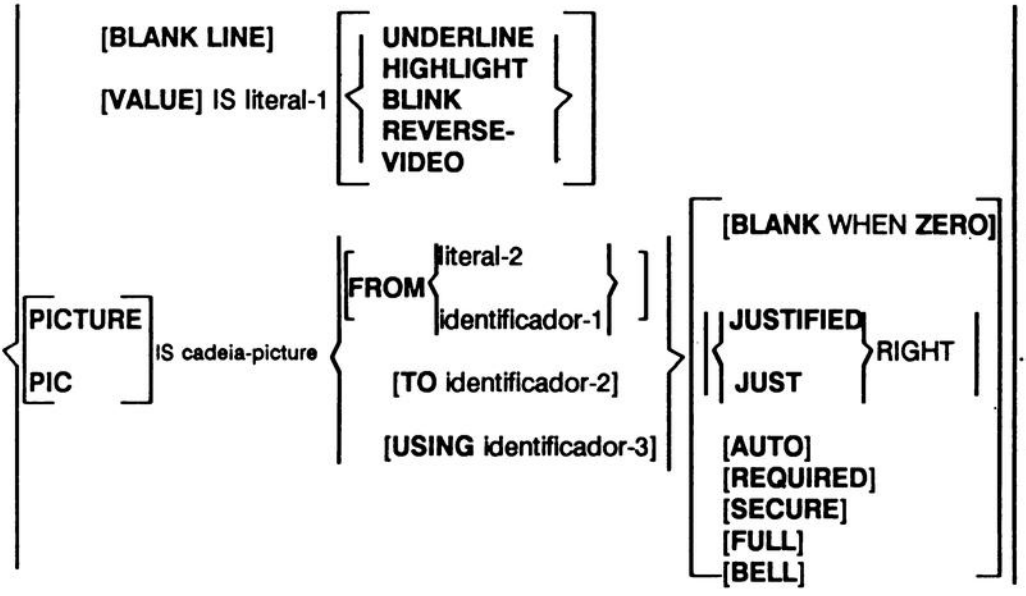

**[SCREEN SECTION**

número-de-nível nome-de-vídeo **[AUTO] [SECURE]**

[numero-de-nivel **BLANK SCREEN]**

[numero-de-nivel**[LINE NUMBER IS**[PLUS]inteiro-1]**[COLUMN NUMBER**

**IS**[PLUS]inteiro-2].....




## Formato 1

**PROCEDURE DIVISION** [**USING** nome-do-dado-1 [nome-do-dado2...]]

[**DECLARATIVES**.
{nome-da-seção **SECTION** [nome-do-segmento] sentença-declarativa
[nome-do-paragrafo [ sentença ]...]...}

[**END DECLARATIVES**]
{nome-de-seção **SECTION** [ número-de-segmento ]
[nome-de-parágrafo [ sentença ]...]...}...

## Formato 2

**PROCEDURE DIVISION** [**USING** nome-de-dado-1 [ nome-de-dado-2 ]...]

{nome-de-parágrafo [sentença]...}...
{nome-de-seção **SECTION** [número-de-segmento]
[nome-de-parágrafo [sentença]...]...}...

## Formato 3

**PROCEDURE DIVISION [ CHAINNING** nome-de-dado-1 [ nome-de-dado-2 ]...]

**ACCEPT** nome-de-dado

**ACCEPT** nome-de-dado **FROM** { **DATE** | **DAY** | **TIME** | **LINE NUMBER** | **ESCAPE KEY** }

**ACCEPT** posicionamento nome-de-dado .....

**[WITH** { **EMPTY-CHECK** | **SPACE-FILL** | **ZERO-FILL** | **LEFT-JUSTIFY** | **RIGHT-JUSTIFY** | **TRAILING-SIGN** | **PROMPT** | **UPDATE** | **LENGTH-CHECK** | **BEEP** | **AUTO-SKIP** | **NO-ECHO** } ...]

**ACCEPT** nome-de-vídeo **[ON ESCAPE** comando-imperativo]

**ADD** { nome-de-dado | literal-1 } [ { nome-de-dado-2 | literal-2 } ... ] { **TO** | **GIVING** } nome-de-dado-m[**ROUNDED**]

[ON **SIZE ERROR** comando-imperativo ...]

**ALTER** nome-de-parágrafo **TO** [PROCEED TO] nome-de-parágrafo-2

**CALL** { identificador-1 | literal } [**USING** nome-de-dado-1 [nome-de-dado-2]...]

**CHAIN** { literal | identificador-1 } [**USING** nome-de-dado-1 [nome-de-dado-2]...]

**CLOSE** nome-de-arquivo-1 [WITH **LOCK**]
[ nome-de-arquivo-2 [WITH **LOCK**]]...

**COMPUTE** nome-de-dado-1 [ **ROUNDED** ][ nome-de-dado-2 [ **ROUNDED** ]...]

{ expressão aritmetica / nome-de-dado / literal-numerico } [ON **SIZE ERROR** comando-imperativo...]

**DELETE** nome-de-arquivo RECORD [**INVALID** KEY comando-imperativo...]

**DISPLAY** { [posicionamento] { nome-de-dado / literal / **ERASE** }... [**UPON** mnemonico] }

**DIVIDE** { nome-de-dado-1 / literal-numerico-1 } { **INTO** nome-de-dado-2 / **BY** literal-numerico-2 } | **GIVING** nome-de-dado-3

[**ROUNDED**] [**ON SIZE ERROR** comando imperativo ...]

**EXIT** [**PROGRAM**]

**GO** TO nome-de-procedimento-1

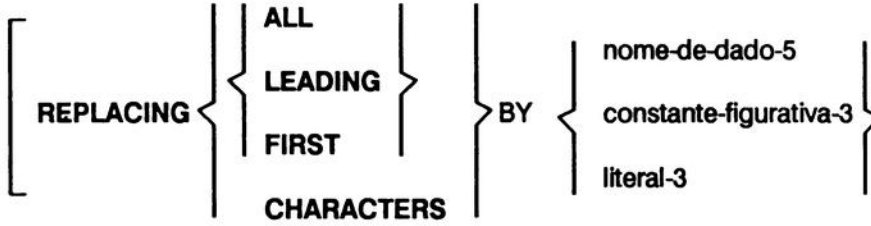
**GO** TO nome-de-procedimento-1 [nome-de-procedimento-2]...

**DEPENDING** ON nome-de-dado

IF condição { comando(s)-1 / **NEXT SENTENCE** } [ **ELSE** { comando(s)-2... / **NEXT SENTENCE** } ]

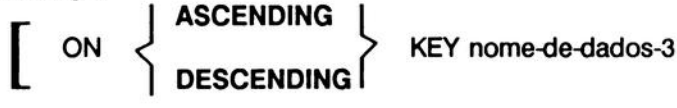
**INSPECT** nome-de-dado-1

[ **TALLYING** nome-de-dado-2 FOR { { **ALL** / **LEADING** } { nome-de-dado-3 / constante-figurativa-1 / literal-1 } / **CHARACTERS** } ]

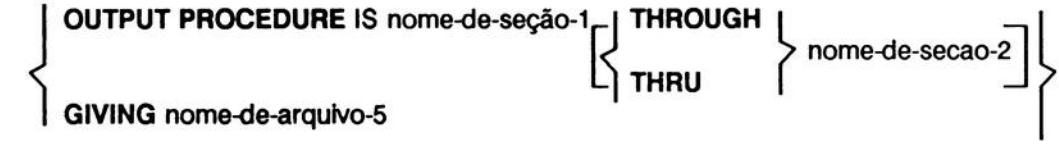


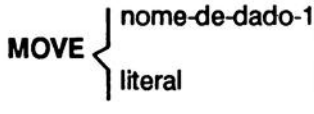

[nome-dados-2]...

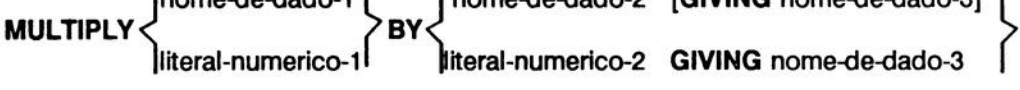



[nome-de-dados-4]...] ...

**USING** nome-de-arquivo-2, nome-de-arquivo-3 [nome-de-arquivo-4]...



**[ROUNDED]** [ON SIZE ERROR comando-imperativo..]

**OPEN** { **INPUT** / **I O** / **OUTPUT** / **EXTEND** } nome-de-arquivo-1 [nome-de-arquivo-2]...

**PERFORM** { nome-de-parágrafo / nome-de-seção / nome-de-procedimento-1 } [ **THRU** nome-de-procedimento-2 ]

[ { inteiro-1 / nome-de-dado-1 } **TIMES** ]

**PERFORM** { nome-de-parágrafo / nome-de-seção / nome-de-procedimento-1 } [ **THRU** nome-de-procedimento-2 ]

[**VARYING** { nome-de-índice / nome-de-dado } **FROM** inicial-1 **BY** incremento-1]

**UNTIL** condição-1

**PERFORM** { nome-de-parágrafo / nome-de-seção / nome-de-procedimento-1 } **THRU** nome-de-procedimento-2

[**VARYING** { nome-de-índice / nome-de-dado } **FROM** inicial-1 **BY** incremento-1]

**UNTIL** condição-1

[ **AFTER** { nome-de-índice / nome-de-dado } **FROM** inicial-2 **BY** incremento-2

**UNTIL** condição-2

[**AFTER** { nome-de-índice / nome-de-dado } **FROM** inicial-3 **BY** incremento-3

**UNTIL** condição-3 ]

**READ** nome-de-arquivo [**NEXT**] RECORD [**INTO** nome-de-dado]

[AT **END** comando-imperativo...]

**READ** nome-de-arquivo [**KEY IS** nome-de-dado-1] RECORD [**INTO** nome-de-dado-2]

[**INVALID** KEY comando imperativo...]

**RELEASE** nome-do-registro [FROM identificador]

**RETURN** nome-do-arquivo RECORD [**INTO** identificador]

AT **END** comando-imperativo

REWRITE nome-de-registro [**FROM** nome-de-dado]

REWRITE nome-de-registro [**FROM** nome-de-dado]

[**INVALID** KEY comando-imperativo]

**SET** { nome-de-dado-1 [,nome-de-dado-2... ] / nome-de-indice-1 [,nome-de-indice-1...] / item-indice-1 [item-indice-2... ] } **TO** { nome-de-dado-3 / nome-de-indice-3 / item-indice-3 / inteiro-1 }

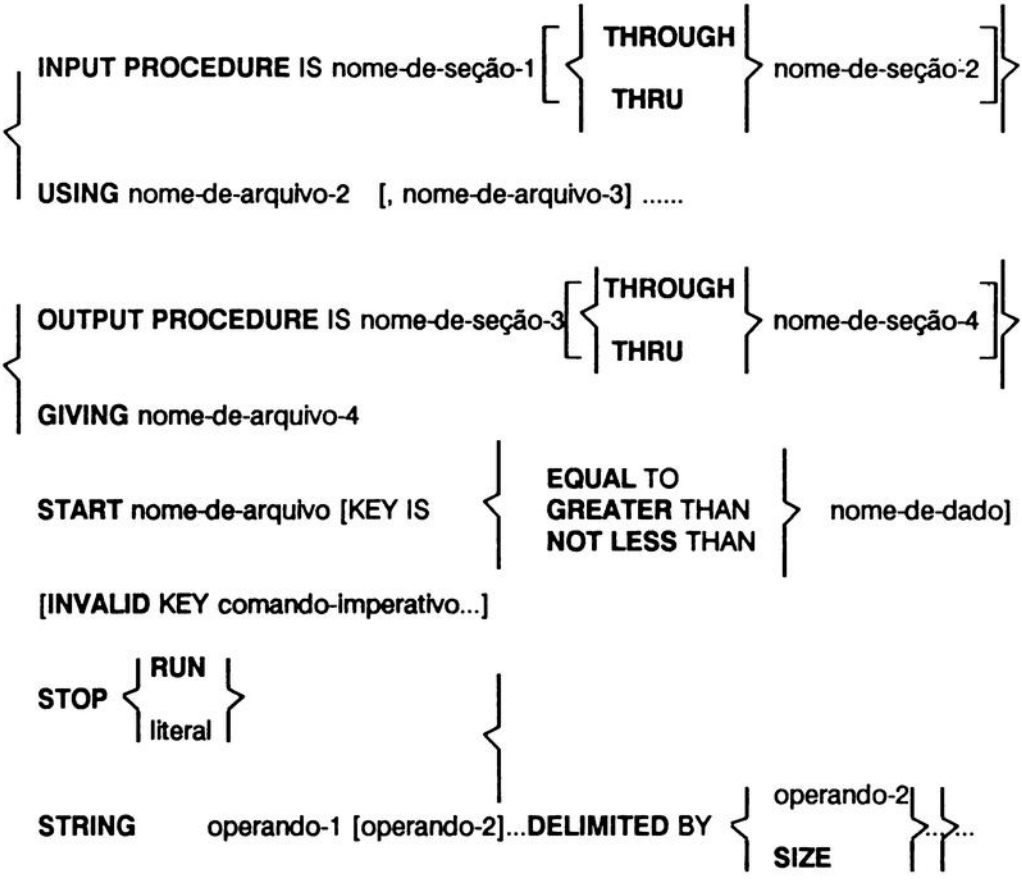
**SET** nome-de-indice-4 [nome-de-indice-5]... { **UP BY** / **DOWN BY** } { nome-de-dado-4 / inteiro-2 }

**SORT** nome-de-arquivo-1 ON { **ASCENDING** / **DESCENDING** } KEY nome-de-dados-1

[, nome-de-dado-2] ...

[ON { **ASCENDING** / **DESCENDING** } KEY nome-de-dados-3

[, nome-de-dado-4...]...


INPUT PROCEDURE IS nome-de-seção-1
THROUGH
THRU
nome-de-seção-2
USING nome-de-arquivo-2 [, nome-de-arquivo-3] .....
OUTPUT PROCEDURE IS nome-de-seção-3
THROUGH
THRU
nome-de-seção-4
GIVING nome-de-arquivo-4
START nome-de-arquivo [KEY IS
EQUAL TO
GREATER THAN
NOT LESS THAN
nome-de-dado]
[INVALID KEY comando-imperativo...]
STOP
RUN
literal
STRING operando-1 [operando-2]...DELIMITED BY
operando-2
SIZE
INTO identificador-1 [ WITH POINTER identificador-2 ]
[ ON OVERFLOW comando-imperativo]
SUBTRACT
nome-de-dado-1
literal-numérico-1
. . . FROM
nome-de-dado-m [ GIVING nome-de-dado-m]
literal-numérico-m GIVING nome-de-dado-m
[ ROUNDED ] [ ON SIZE ERROR comando-imperativo... ]

**UNSTRING** identificador-1

[ **DELIMITED** BY [ **ALL** ] operando-1 [ **OR** [ **ALL** ] operando-2 ] ... ]

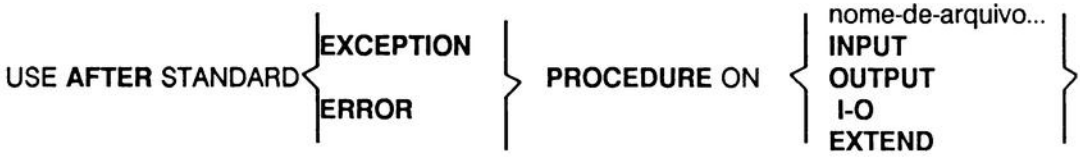
**INTO** { identificador-2 [ **DELIMITER** IN identificador-3 ]

[ **COUNT** identificador-4 ] } ...

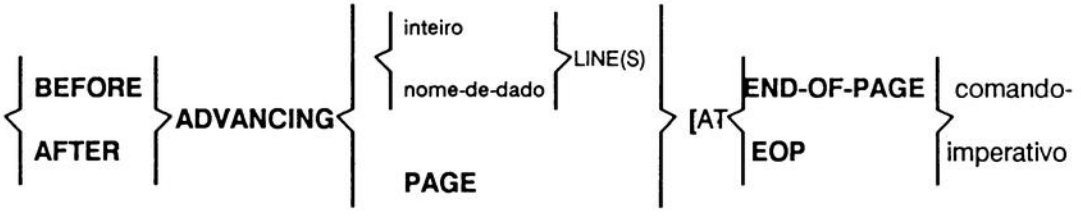
[ WITH **POINTER** identificador-5 ]

[ **TALLYING** IN identificador-6 ]

[ ON **OVERFLOW** comando-imperativo ]




**WRITE** nome-de-registro [ **FROM** nome-de-dado ]

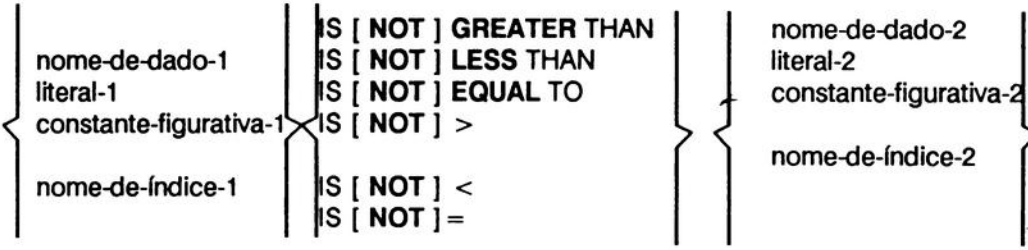



**WRITE** nome-de-registro [ **FROM** nome-de-dado ]

[ **INVALID** KEY comando-imperativo ... ]

# Condições no Cobol

## CONDIÇÃO RELACIONAL



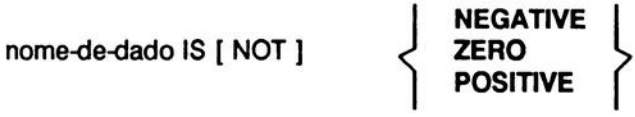

## CONDIÇÃO DE CLASSE


## NOME DE CONDIÇÃO (Nível 88)

nome-de-condição

## CONDIÇÃO COMPOSTA


## TESTE DE SINAL



# Outros Formatos

## COMANDO COPY

**COPY** nome-de-texto

## DEPURAÇÃO

**EXHIBIT NAMED** { [ especificação -da- posição ] } { identificador / literal / **ERASE** } }... [ **UPON** nome-mnemônico ]

{ **READY** / **RESET** } **TRACE**

## QUALIFICAÇÃO

nome-de-parágrafo [ { **OF** / **IN** } nome-de-seção ]

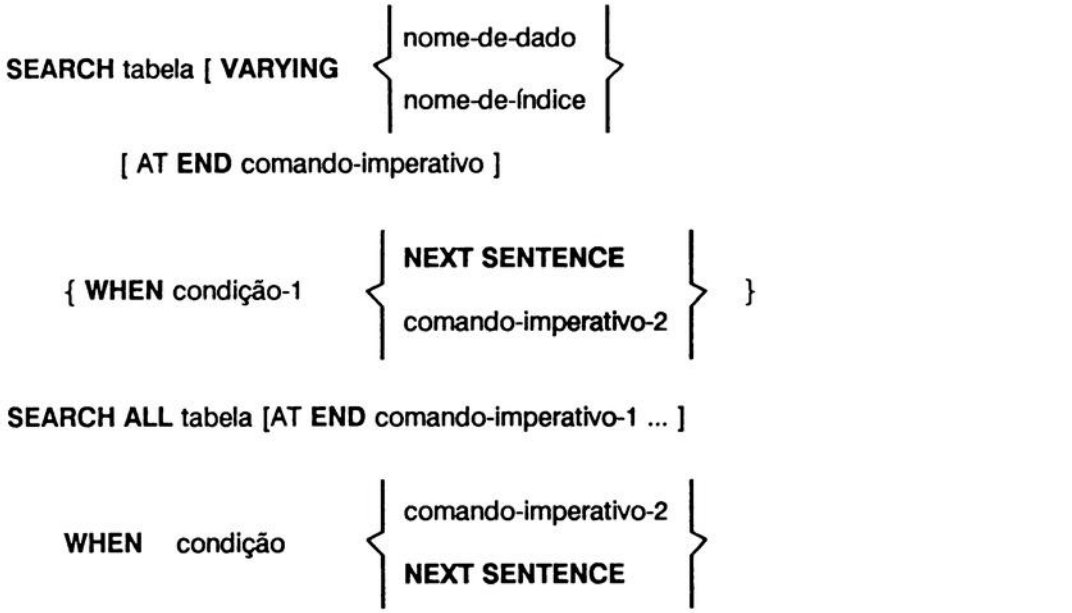
nome-de-dado-2 [ { **OF** / **IN** } nome-de-dado-1 ]

nome-de-condição [ { **OF** / **IN** } nome-de-dado ]

## SUBSCRIÇÃO

{ nome-de-dado / nome-de-condição } (subscrito-1 [, subscrito-2 [, subscrito-3 ] ] )

## Pesquisa em Tabelas



# APÊNDICE C

## PALAVRAS RESERVADAS DA LINGUAGEM COBOL

| | |
|---|---|
| ACCEPT | ACCESS |
| ADD | ADVANCING |
| AFTER | ALL |
| ALPHABETIC | ALSO |
| ALTER | ALTERNATE |
| AND | ARE |
| AREA(S) | ASCENDING |
| ASCII | ASSIGN |
| ASSIGN | AT |
| AUTHOR | AUTO-SKIP |
| BEEP | BEFORE |
| BLANK | BLOCK |
| BOTTOM | BY |
| CALL | CANCEL |
| CD | CF |

CH
CLOCK-UNITS
COBOL
CODE-SET
COLLATING
COMMA
COMP
COMPUTATIONAL-0
COMPUTATIONAL-3
COMPUTE
CONTAINS
COPY
COUNT
DATA
DATE-COMPILED
DAY
DEBUG-CONTENTS
DEBUG-LINE
DEBUG-SUB-1
DEBUG-SUB-3
DECLARATIVES
DELIMITED
DEPENDING
DESTINATION
DISABLE
DISPLAY
DIVISION
DUPLICATES
EGI
EMI
END
ENTER
EOP
ERASE
ESI
EXCEPTION
EXIT
FD

CHARACTER(S)
CLOSE
CODE
COL
COLUMN
COMMUNICATION
COMPUTATIONAL
COMP-0
COMP-3
CONFIGURATION
CONTROL(S)
CORR(ESPONDING)
CURRENCY
DATE
DATE-WRITTEN
DEBUGGING
DEBUG-ITEM
DEBUG-NAME
DEBUG-SUB-2
DECIMAL-POINT
DELETE
DELIMITER
DESCENDING
DE(TAIL)
DISK
DIVIDE
DOWN
DYNAMIC
ELSE
ENABLE
END-OF-PAGE
ENVIRONMENT
EQUAL
ERROR
EVERY
EXHIBIT
EXTEND
FILE

FILE-CONTROL
FILLER
FIRST
FOR
GENERATE
GO
GROUP
HIGH-VALUE(S)
IF
INDEX
INITIAL
INPUT
INSPECT
INTO
IS
I-O CONTROL
KEY
LAST
LEFT
LENGTH
LESS
LIN
LINAGE-COUNTER
LINE-COUNTER
LOCK
MEMORY
MESSAGE
MODULES
MULTIPLE
NATIVE
NEXT
NOT
NUMERIC
OCCURS
OFF
ON
OPTIONAL
ORGANIZATION
FILE-ID
FINAL
FOOTING
FROM
GIVING
GREATER
HEADING
IDENTIFICATION
IN
INDEXED
INITIATE
INPUT-OUTPUT
INSTALLATION
INVALID
I-O
JUST(IFIED)
LABEL
LEADING
LEFT-JUSTIFY
LENGTH-CHECK
LIMIT(S)
LINAGE
LINE(S)
LINKAGE
LOW-VALUE(S)
MERGE
MODE
MOVES
MULTIPLY
NEGATIVE
NO
NUMBER
OBJECT-COMPUTER
OF
OMITTED
OPEN
OR
OUTPUT

OVERFLOW
PAGE-COUNTER
PF
PIC(TURE)
POINTER
POSITIVE
PRINTING
PROCEED
PROGRAM-ID
QUEUE
RANDOM
READ
RECEIVE
REDEFINES
REFERENCES
RELEASE
REMOVAL
REPLACING
REPORTING
RESERVE
RETURN
REWIND
RF
RIGHT
ROUND
SAME
SEARCH
SECURITY
SEGMENT-LIMIT
SEND
SEPARATE
SEQUENTIAL
SIGN
SORT
SOURCE
SPACE(S)
SPECIAL-NAMES
STANDARD-1
PAGE
PERFORM
PH
PLUS
POSITION
PRINTER
PROCEDURE(S)
PROGRAM
PROMPT
QUOTE(S)
RD
READY
RECORD(S)
REEL
RELATIVE
REMAINDER
RENAMES
REPORT(S)
RERUN
RESET
REVERSED
REWRITE
RH
RIGHT-JUSTIFY
RUN
SD
SECTION
SEGMENT
SELECT
SENTENCE
SEQUENCE
SET
SIZE
SORT-MERGE
SOURCE-COMPUTER
SPACE-FILL
STANDARD
START

| | |
|---|---|
| STATUS | STOP |
| STRING | SUB-QUEUE-1.2.3 |
| SUBTRACT | SUM |
| SUPPRESS | SYMBOLIC |
| SYNC(HRONIZED) | TABLE |
| TALLYING | TAPE |
| TERMINAL | TERMINATE |
| TEXT | THAN |
| THROUGH | THRU |
| TIME | TIMES |
| TO | TOP |
| TRACE | TRAILING |
| TRAILING-SIGN | TYPE |
| UNIT | UNSTRING |
| UNTIL | UP |
| UPDATE | UPON |
| USAGE | USE |
| USING | VALUE(S) |
| VARYING | WHEN |
| WITH | WORDS |
| WORKING-STORAGE | WRITE |
| ZERO((E)S) | ZERO-FILL |
| + | - |
| * | / |
| ** | < |
| > | = |

# APÊNDICE D

## LOCKING (bloqueio) PARA ARQUIVO E REGISTRO

Um arquivo e um registro com a opção LOCKING para sistemas multiusuários podem ser implementados com o COBOL-MS.

Arquivo e registro com LOCKING assegura a proteção de arquivo de dados durante acessos simultâneos por múltiplos processos ou por uso exclusivo.

É uma extensão da linguagem e permite o processamento de arquivos SEQUENTIAL e LINE SEQUENTIAL com um LOCKING no modo EXCLUSIVE, processamento de arquivos INDEXED e RELATIVE com um LOCKING nos modos EXCLUSIVE, MANUAL ou AUTOMATIC.

A sintaxe para arquivo e registro com LOCKING sofre modificações nos seguintes pontos:

1. FILE-CONTROL;

2. comandos OPEN, READ, START e UNLOCK.

## Arquivo com LOCKING

Arquivo com LOCKING dá ao usuário exclusividade no uso desse arquivo. Neste caso, em todas as tentativas por outros processos para ler, gravar ou mesmo bloquear o arquivo haverá prevenção pelo sistema operacional. Se o outro processo é um programa COBOL, ocorrerá um erro. Arquivo com LOCKING é mais freqüentemente usado para processamento de arquivos seqüenciais.

## Registro com LOCKING

Registro com LOCKING é implementado apenas para arquivos indexados e relativos.

Selecionando a opção AUTOMATIC na cláusula LOCKING indica que os registros serão acessados no modo AUTOMATIC do registro com LOCKING. O registro com LOCKING AUTOMATIC pode automaticamente bloquear qualquer registro que é o objetivo do procedimento READ. O mesmo registro é automaticamente desbloqueado pela execução de um próximo READ, WRITE, REWRITE ou CLOSE.

Selecionando a opção MANUAL na cláusula LOCKING fica especificado que os registros serão acessados no modo LOCKING MANUAL. O modo LOCKING MANUAL bloqueia apenas um registro quando o verbo LOCK estiver presente no respectivo comando READ ou START. No modo LOCKING MANUAL, o registro é desbloqueado apenas por um procedimento UNLOCK ou quando o arquivo é fechado.

# CONSIDERAÇÕES SOBRE A SINTAXE

Explana-se a seguir o uso do arquivo e registro com o comando LOCKING na entrada FILE-CONTROL, e nos procedimentos de entrada e saída de arquivos: OPEN, READ, START e UNLOCK.

FILE-CONTROL (cláusula SELECT)

Arquivos SEQUENTIAL e LINE SEQUENTIAL

O seu formato é:

```
SELECT   [ OPTIONAL ] nome-de-arquivo ASSIGN TO { DISK /
         PRINTER } [ [ LOCKING ] EXCLUSIVE ]
         [ RESERVE inteiro [ AREA / AREAS ] ]
         [ ORGANIZATION IS [ LINE ] SEQUENTIAL ]
         [ FILE STATUS IS nome-de-dado-1 ].
```

O uso da opção EXCLUSIVE na cláusula SELECT especifica o uso exclusivo do arquivo. Aplica-se este modo somente para arquivos em disco.

A opção EXCLUSIVE deve ser usada com arquivos onde o modo e a organização não possua acesso randômico.

## ARQUIVOS INDEXADOS E RELATIVOS

O formato para arquivo indexado é:

```
SELECT nome-de-arquivo ASSIGN TO DISK
[ [ LOCKING IS ] { EXCLUSIVE / MANUAL / AUTOMATIC } ]
[ RESERVE inteiro [ AREA / AREAS ] ]
ORGANIZATION IS INDEXED
[ ACCESS MODE IS { SEQUENTIAL / RANDOM / DYNAMIC } ];
```

e para arquivo relativo é:

```
SELECT nome-de-arquivo ASSIGN TO DISK
[ [ LOCKING IS ] { EXCLUSIVE / MANUAL / AUTOMATIC } ]
[ RESERVE inteiro [ AREA / AREAS ] ]
ORGANIZATION IS RELATIVE
[ ACCESS MODE IS
```

```
{ SEQUENTIAL [ RELATIVE KEY IS nome-de-dado-1 ]
{ RANDOM     }
{ DYNAMIC    } RELATIVE KEY IS nome-de-dado-1 } ]

[ FILE STATUS IS nome-de-dado-2 ]
```

A opção EXCLUSIVE na cláusula SELECT especifica o uso exclusivo do arquivo.

A opção AUTOMATIC permite compartilhar os registros do arquivo. No entanto, um registro é automaticamente bloqueado quando acessado por um READ e automaticamente liberado quando acessado por um próximo READ, REWRITE, WRITE ou CLOSE. Apenas um registro é bloqueado e liberado neste intervalo.

Se a opção MANUAL for especificada, mais de um registro pode ser bloqueado ou liberado ao mesmo tempo. O READ e o START usando o verbo LOCK bloquearão o registro. O comando UNLOCK liberará todos os bloqueios presentes no arquivo.

As opções AUTOMATIC e MANUAL são usadas quando existem registros randômicos de arquivos relativos ou indexados abertos no modo OPEN I-O.

Instrução OPEN (para arquivos Sequential e Line Sequential)

O seu formato é:

```
OPEN [ [ LOCKING ] EXCLUSIVE ]
{ INPUT / OUTPUT / I-O / EXTEND } nome-de-arquivo .
```

Se a cláusula LOCKING não for especificada o modo LOCKING EXCLUSIVE é subentendido pelo OPEN EXTEND e OPEN OUTPUT. Arquivos abertos com a opção INPUT podem ser apenas lidos, em conseqüência não requerem bloqueio.

Instrução OPEN (para arquivos Indexed e Relative)

O seu formato é:

```
OPEN [ [ LOCKING IS ]
       { EXCLUSIVE / MANUAL / AUTOMATIC } ]
       { INPUT / OUTPUT / I-O } nome-de-arquivo.
```

Se a cláusula LOCKING não for especificada, o modo LOCKING EXCLUSIVE será subentendido pelo OPEN OUTPUT. O modo LOCKING AUTOMATIC será subentendido pelo OPEN I-O. Arquivos abertos com a opção INPUT só podem ser lidos e sendo assim não requerem bloqueio.

Instrução READ (no modo MANUAL)

O seu formato para arquivos indexados é:

```
READ nome-de-arquivo [ NEXT ] RECORD [ LOCK ] [ WAIT ]
     [ INTO identificador ] [ AT END comando-imperativo ].
READ nome-de-arquivo RECORD [ LOCK ] [ WAIT ]
     [ INTO identificador ] [ KEY is nome-de-dado ]
     [ INVALID KEY comando-imperativo ].
```

e para arquivo relativo é:

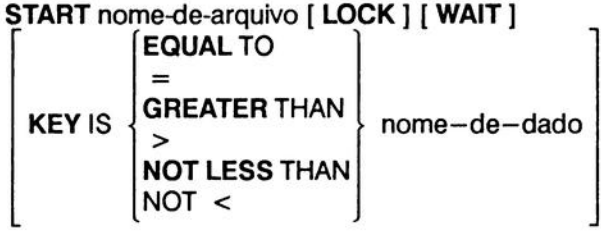
```
READ nome-de-arquivo [ NEXT ] RECORD [ LOCK ] [ WAIT ]
     [ INTO identificador ] [ AT END comando-imperativo ].

READ nome-de-arquivo RECORD [ LOCK ] [ WAIT ]
     [ INTO identificador ] [ INVALID KEY comando-imperativo ].
```

Os verbos LOCK e WAIT são opcionais. Se o verbo WAIT for usado e um registro bloqueado for encontrado durante a execução do programa, o procedimento ficará aguardando até que o registro especificado seja liberado, sendo então executada a instrução READ.

Se o verbo WAIT não for especificado e um registro bloqueado for encontrado durante a execução, a corrida retorna em uma condição de erro.

Instrução START (no modo AUTOMATIC e MANUAL)

A sintaxe do comando START para arquivo indexado e relativo é:




Os verbos LOCK e WAIT são opcionais. Se o verbo WAIT for usado e um registro bloqueado for encontrado durante a execução do programa, o processo aguardará até que o registro especificado seja liberado e será então executada a instrução START.

Se o verbo WAIT não for especificado e um registro bloqueado for encontrado durante a execução do programa, o mesmo retornará em uma condição de erro.

Diferente do comando READ, o qual bloqueia automaticamente um registro no modo LOCKING AUTOMATIC, o comando START no LOCKING AUTOMATIC não bloqueia um registro a não ser que o verbo LOCK esteja presente no procedimento.

Instrução UNLOCK.

O objetivo da instrução é liberar arquivos anteriormente bloqueados com um comando READ ou START usando a opção WITH LOCK, ou liberar automaticamente o último registro bloqueado.

O seu formato é:

```
UNLOCK nome-de-arquivo.
```

Nome-de-arquivo especifica o arquivo a ser manipulado por um comando READ ou START. O comando UNLOCK deve aparecer antes da execução do comando READ ou START cujo operando é o objeto da ação.

Exemplo:

```
UNLOCK ARQUIVO1.
```

# APÊNDICE E

## CARACTERÍSTICAS DO COBOL - 85*

Para o propósito da discussão, as alterações e incrementos incorporados no COBOL-85 estão classificados dentro de 2 grupos. O primeiro grupo contém as várias características para o melhoramento da linguagem COBOL. O outro grupo consiste de características específicas da programação estruturada. As alterações e incrementos na linguagem serão introduzidos em comparação com as características já existentes no Cobol-74 e outras versões.

Por outro lado nenhuma ordem está sendo seguida na apresentação das características individuais.

## ESTRUTURA DO PROGRAMA

Todo programa COBOL ínicia com um cabeçalho IDENTIFICATION DIVISION e termina dentro de um procedimento físico na PROCEDURE

* (BIBLIOGRAFIA: COBOL PROGRAMMING INCLUDING MS-COBOL AND COBOL-85, 2a. Edição, de M.K. ROY e D. GHOSH DASTIDAR, 1989.)

DIVISION. De acordo com o novo padrão, um terminador opcional END PROGRAM foi criado para marcar o fim físico do programa. O formato é:

```
END PROGRAM Nome-do-programa
```

O nome-do-programa é o nome dado no parágrafo PROGRAM-ID. O programa completo ao final deste capítulo ilustra o uso dessa característica.

O terminador END PROGRAM como tal não é uma grande facilidade. O incremento torna-se necessário porque é possível introduzir fisicamente o COBOL-85 dentro do escopo de um outro programa. O programa incluído pode tornar-se, ainda, incluso dentro de um outro programa. Um programa incluido é na realidade uma sub-rotina que está sendo chamada pelo programa principal. A estrutura de um programa está apresentada a seguir.

```
IDENTIFICATION DIVISION
PROGRAM-ID - Nome-do-programa-1
   outros parágrafos desta divisão
 [ ENVIRONMENT DIVISION ]
   conteúdo desta divisão
[ DATA DIVISION ]
    conteúdo desta divisão
PROCEDURE DIVISION
   conteúdo desta divisão
[ programa-fonte chamado ] ...
END PROGRAM nome-do-programa - 1
```

O primeiro ponto a ser observado é que o programa incluído (indicado como [programa-fonte chamado]) será colocado após o último procedimento físico incluído na PROCEDURE DIVISION. O programa também apresenta a estrutura acima. Outro aspecto que deve ser notado é que a ENVIRONMENT DIVISION bem como a DATA DIVISION são agora opcionais. Assim, se o programa não manipula qualquer arquivo, a entrada ENVIRONMENT DIVISION pode ser omitida. A DATA DIVISION é omitida apenas no caso de o programa a ser chamado aninhar-se no programa-fonte.

Na parte restante da seção, descreveremos o aninhamento do programa. Consideremos o esboço apresentado a seguir onde cada programa está contido dentro de um retângulo:

```
IDENTIFICATION DIVISION.
PROGRAM-ID A.
:
    IDENTIFICATION DIVISION.
    PROGRAM-ID  B.
    :
        IDENTIFICATION DIVISION.
        PROGRAM-ID C.
        :

        END PROGRAM C.

    END PROGRAM B.

    IDENTIFICATION DIVISION.
    PROGRAM-ID D.
    :
    END PROGRAM D.

:

END PROGRAM A.
```

Neste aninhamento, A é o programa mais extremo. B e D estão diretamente inclusos em A. O programa C está diretamente incluso em B e indiretamente incluso em A (porque B está incluso em A). Todos os programas juntos determinam uma única corrida.

Uma importante característica do aninhamento é que qualquer nome de dado, nome de arquivo, nome de registro, nome de condição, nome de indexador ou nome de relatório definido em um programa pode ser referenciado pelo programa diretamente ou indiretamente incluso, desde que provido de uma declaração para ser um nome global pelo uso da cláusula GLOBAL. Por exemplo, na ilustração anterior, o programa A poderia conter a seguinte descrição:

```
01 DATA-X  PIC  9(5) IS GLOBAL.
```

Então como resultado, DATA-X torna-se um nome global e pode ser referenciado nos programas B, C ou D. O formato para a cláusula é:

**IS GLOBAL**

e a cláusula só pode ser usada no nível 01, entradas FD e RD. Quando a cláusula GLOBAL é usada como item de grupo, aplica-se a declaração para todos os itens subordinados. Qualquer nome de condição ou nome de índice associado com o nome de dado global é automaticamente tratado como um item global.

É possível que um programa aninhado possa ter um nome de dado idêntico ao nome global descrito no programa considerado principal. Por exemplo, na ilustração anterior, descreve-se DATA-X como um nome global em A. Descreve-se também DATA-X em C, sem a cláusula GLOBAL. Agora, qualquer referência para DATA-X em C terá um DATA-X local (isto é, o DATA-X descrito em C). Entretanto, qualquer referência para DATA-X em B será tratado como DATA-X global. A regra para este caso é a seguinte:

O nome referenciado no programa é visto como o do mesmo programa. Se nenhuma descrição for encontrada para o nome no programa em referência, o nome do programa que mais diretamente o referencia será pesquisado. Por exemplo, supõe-se que DATA-X seja descrito separadamente como global em A bem como em B. Então, qualquer referência para DATA-X em A ou D será tratada como um item descrito em A. Note que referência para DATA-X em C será tratada como item global em B e não como item global em A porque B contém C mais diretamente.

Um item descrito em um programa incluso não é normalmente avaliado para o programa principal ou qualquer programa fora desse escopo. Entretanto, se um item é descrito com a cláusula EXTERNAL, pode ser referenciado por qualquer programa da presente execução. O formato da cláusula é:

**IS EXTERNAL**

e pode ser usada com a cláusula GLOBAL. Entretanto, a cláusula não poderá ser usada com a cláusula REDEFINES. Além disso, a cláusula valuy não poderá ser especificada como um item externo. Um item externo não é tratado como

parte do programa que contém esta descrição. De fato, todo o item externo é tratado juntamente em uma área comum para todos os programas da corrida.

Para aceitar o aninhamento dos programas a sintaxe do parágrafo PROGRAM-ID será estendida. O formato do parágrafo é:

```
                                    {|COMMON |}
PROGRAM-ID.  nome do programa [ IS {|         |} PROGRAM ].
                                    {| INITIAL |}
```

As duas linhas verticais indicam que COMMON e INITIAL podem ser usadas separadas ou juntas. Um programa aninhado fora da cláusula COMMON pode ser chamado apenas pelo programa que contém a cláusula. Se o programa aninhado tiver a cláusula COMMON, pode ser chamado por outro programa da corrida a qualquer outro programa contido dentro do programa geral. Ainda na ilustração, se B estiver declarado como comum através do uso da cláusula COMMON então poderá ser chamado pelo A bem como pelo D. Não poderá ser chamado pelo D a não ser que B esteja declarado como comum. A cláusula COMMON poderá ser usada apenas por programas aninhados. A cláusula INITIAL indica que quando o programa for chamado será adequado à condição inicial.

## ENTRADA DE DESCRIÇÃO DE DADOS SEM NOME

A entrada de descrição de dados poderá usar qualquer nome de dado. Por exemplo:

```
05  PIC 9(6) VALUE ZERO.
```

é agora uma entrada de descrição de dados válida. O compilador tratará como uma entrada FILLER. Assim a descrição acima é equivalente para

```
05  FILLER  PIC 9(6)  VALUE ZERO.
```

Além disso, a palavra Filler é permitida agora em um item de grupo. Por exemplo:

```
01  FILLER.
    02 DATA-1  PIC X(3)
    02 DATA-2  PIC 9(4)V99.
```

é totalmente válido.

### DE - EDIÇÃO

O COBOL-85 permite o movimento de item de dado numérico editado para um item de dado numérico através da instrução MOVE. Por exemplo, considere a seguinte entrada de descrição de dados:

```
02  CPO-EDITADO    PIC   ZZZ9.99.
02  CPO-NEDITADO   PIC   9(4)V99.
```

Agora, após a execução da seguinte instrução MOVE

```
MOVE  245.5  TO  CPO-EDITADO.
MOVE  CPO-EDITADO  TO  CPO-NEDITADO.
```

CPO-EDITADO e CPO-NEDITADO conterão 0245.50 e 0245.50 respectivamente. Note que tal movimento não era permitido nas outras versões do COBOL.

## ADD COM A FRASE GIVING

A palavra opcional TO foi juntada na sintaxe da instrução ADD com a opção GIVING. No caso simplificado (que é, sem as frases SIZE ERROR e ROUNDED) o formato é:

```
ADD  {identificador-1}          TO {identificador-2}
     {literal-1      }  ...        {literal-2      }
     GIVING identificador-3
```

Assim ADD A TO B GIVING C agora é válido. É equivalente a ADD A B GIVING C.

## CONDIÇÃO RELACIONAL

As relações abaixo foram adicionadas aos operadores relacionais já existentes.

IS **GREATER** THAN **OR EQUAL** TO ou > =
IS **LESS** THAN **OR EQUAL** TO ou < =

Assim:

```
a) IF SALÁRIO-BASE NOT < 500 MOVE 1 TO CODIGO-DO-SALÁRIO
b) IF SALÁRIO-BASE NOT >500 OR = 500 MOVE 1 TO CODIGO-DO-SALÁRIO
c) IF SALÁRIO-BASE NOT > = 500 MOVE 1 TO CODIGO-DO-SALÁRIO
```

são equivalentes. Note que o item c) não é permitido nas outras versões do COBOL. Além disso, note a diferença dos outros operadores relacionais: NOT não pode ser usado com esses operadores relacionais novos. Assim SALARIO-BASE NOT > = 500 (para SALARIO-BASE < 500 está incorreto).

## CONDIÇÃO DE CLASSE

Um dado alfabético consiste de letras maiúsculas, letras minúsculas e espaços. Para testar o conteúdo de um campo alfabético ou alfanumérico, três testes de classe foram incluídos no COBOL-85:

IS [ NOT ] **ALPHABETIC**
IS [ NOT ] **ALPHABETIC-UPPER**
IS [ NOT ] **ALPHABETIC-LOWER**

O teste ALPHABETIC checa as letras maiúsculas, as letras minúsculas e o espa ço. O ALPHABETIC-UPPER checa as letra maiúsculas e o espaço. Similarmente o ALPHABETIC-LOWER checa as letras minúsculas e o espaço. Os dois últimos são condições novas. Considere o seguinte procedimento para simplificar o novo teste:

```
IF CAMPO-A  IS NOT  ALPHABETIC-UPPER
    MOVE 9 TO CODIGO-DE-CLASSE .
```

O valor 9 será transferido para CODIGO-DE-CLASSE se CAMPO-A contiver qualquer caractere que não sejam letras maiúsculas e espaço.

## CLÁUSULA REDEFINES

No COBOL-85 a redefinição de um campo pode ser igual ou menor do que o item a redefinir. Note que nas outras versões do COBOL os dois deviam ser iguais.

Por exemplo:

```
02 DADO-A  PIC X(12).
02 DADO-B  REDEFINES DADO-A.
    03 CAMPO-A  PIC  9(3)V99.
    03 CAMPO-B  PIC  99.
```

agora é permitido.

## TABELAS

Há um número de trocas relativas à cláusula OCCURS e os elementos associados à linguagem.

A lista é a seguinte:

*a.* A tabela pode ter 7 dimensões. Nas outras versões do COBOL, a tabela estava limitada em 3 dimensões.

*b.* A cláusula VALUE pode aparecer dentro da cláusula OCCURS. Neste caso todos os elementos da tabela estão setados para o mesmo valor. Por exemplo:

```
01 TABELA.
   02 TAB-DADOS  OCCURS  10 TIMES  PIC 9(4) VALUE 1.
```

é agora permitido. Todos os elementos TAB-DADOS (1) ..... TAB-DADOS (10) serão inicializados em 1. Não era permitido nas outras versões do COBOL.

*c.* O número mínimo de ocorrências que pode ser especificado através de OCCURS com a frase DEPENDING é zero. Por exemplo, agora a seguinte descrição é permitida:

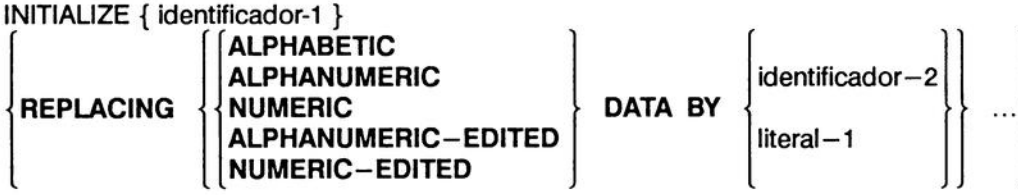
```
01 DESCRIÇÃO.
   02 A1  PIC  99.
   02 P1  PIC  XX.
   02 P2  PIC  99.
   OCCURS 0 TO 90 TIMES DEPENDING ON A1.
```

*d.* Nas outras versões do COBOL, o subscritor poderia ser um literal ou qualquer identificador com valor inteiro positivo. No COBOL-85, a subscrição relativa é também permitida. As expressões I + 1 ou J + 3 são agora permitidas como subscritores. Observe que a indexação relativa já era permitida nas outras versões do COBOL.

## VERBO INITIALIZE

Um novo verbo INITIALIZE foi introduzido.

A sintaxe é:




O propósito deste verbo é inicializar o identificador-1 que pode ser um item de grupo ou elementos. Quando identificador-1 especificar um item de grupo, apenas os itens que pertencem à categoria indicada pela frase REPLACING serão inicializados pelo valor indicado no identificador-2 / literal-1. Por exemplo, consideremos o seguinte exemplo de dados descritos no WORKING-STORAGE SECTTION:

```
01 A.
   02 A1  PIC  9(5).
   02 A2  PIC  X(4).
   02 A3  PIC  9(3).
   02 A4  PIC  Z(3)9.99.
```

Agora, o procedimento

INITIALIZE A REPLANCING NUMERIC DATA BY 50, serão inicializados somente A1 e A3 porque apenas esses campos são numéricos dentro do A. Entretanto, o seguinte procedimento:

```
INITIALIZE A
REPLACING  NUMERIC  DATA  BY 50
REPLACING  ALPHANUMERIC DATA  BY "A"
REPLACING  NUMERIC-EDITED  BY 54.2
```

serão inicializados todos os campos de A.

O processo de inicialização é equivalente à execução de seqüência de comandos MOVE onde são transferidos valores do identificador-2 / literal-1 para os itens elementares do identificador-1. Os campos são inicializados na seqüência que os mesmos aparecem dentro do item de grupo do identificador-1.

Assim o segundo exemplo INITIALIZE mencionado acima inicializará A1, A2, A3 e A4, respectivamente, para 0050, A000, 050 e 0054.20.

Quando o identificador-1 for um item elementar, então a inicialização será efetuada apenas se a categoria mencionada na frase REPLACING se igualar ao do identificador-1. Note que a frase REPLACING é opcional. Se for omitido, todos os campos numéricos serão inicializados com zeros e todos os outros campos com espaços. Assim

```
INITIALIZE A
```

serão inicializados A1 e A3 com valor zero, A2 e A4 com espaços. Os seguintes pontos complementares deverão ser observados:

*a.* item FILLER (ou sem nome) e nome de índice não são afetados pelo comando INITIALIZE;

*b.* se algum grupo indicado pelo identificador-1 contém um item de dado com a cláusula REDEFINES então o item de dado ou qualquer item de dado subordinado não serão inicializados. Por exemplo, considere a seguinte descrição de dados:

```
01 A.
   02 A1    PIC 9(05).
   02 A2    PIC X(10).
   02 B     REDEFINES A2.
   03 B1 PIC 9(04).
   03 B2 PIC 9(04)V99.
```

Agora, se escrevermos INITIALIZE A, A1 será preenchido com zeros e A2 será preenchido com espaços. Haverá dificuldade para que A2 seja preenchido com zeros porque B1 e B2 não são inicializados. Entretanto, identificador-1 pode ter a cláusula REDEFINES ou pode ser um item contendo a cláusula REDEFINES. Por exemplo:

```
INITIALIZE B.
```

serão colocados zeros em todas as posições de A2 porque B1 e B2 serão inicializados com valor zero.

Identificador-1 ou qualquer item subordinado pode também possuir a cláusula OCCURS. No caso todas as ocorrências serão inicializadas. Por exemplo, uma tabela definida como:

```
01  A.
    02 B PIC 9(05) OCCURS 20 TIMES.
```

Agora, INITIALIZE A colocará zeros em todos os elementos da tabela. O verbo INITIALIZE é funcionalmente similar ao verbo MOVE quando este último é usado para inicialização. Entretanto, um único procedimento INITIALIZE pode ser equivalente a vários comandos MOVE. A função do verbo INITIALIZE é também similar à cláusula VALUE. A diferença é que a inicialização através da cláusula VALUE ocorre somente uma vez no início da execução do programa ao passo que o procedimento INITIALIZE pode ser usado quando necessário.

## VERBO SET COM A FRASE TRUE

Em outras versões do COBOL não existem condições de setar um nome para TRUE ou FALSE exceto quando se tem um valor conveniente assinalado para uma variável condicional. Por exemplo, suponha o nome de condição descrito a seguir:

```
01  COD-TAXA  PIC 9.
    88 ISENTO VALUE ZERO.
```

Agora, para setar ISENTO para TRUE, devemos mover (explicitamente ou implicitamente) o valor zero para a variável COD-TAXA. Enquanto este processo indireto assinalando o valor TRUE para o nome de condição permanece fixo, o COBOL-85 possui a alternativa direta. Podemos usar o comando:

```
SET ISENTO TO TRUE
```

para este caso. Além de setar ISENTO para TRUE, também moverá o valor zero para COD-TAXA. A sintaxe do verbo SET é:

**SET** {nome-de-condição} ... **TO TRUE**

Devemos observar que o nome de condição pode ser setado para TRUE (e não para FALSE) pelo comando SET. Quando uma lista de valores é especificada no nome de condição da descrição de entrada, o primeiro valor dessa lista será movido para a variável condicional. Por exemplo, considere o seguinte:

```
05  COD-PROD    PIC 99.
    88 SOFT VALUE 10 THRU 19.
```

Agora, SET SOFT TO TRUE moverá 10 para COD-PROD. Note que este formato do verbo SET é uma nova opção. Isto não altera o verbo SET já existente que opera com nome de índice e itens de dados de índice.

## REFERÊNCIA MODIFICADORA

No COBOL-85, é possível referenciar parte de um campo pela especificação do caractere mais à esquerda e seu comprimento. Por exemplo, consideremos a seguinte descrição de dados:

```
05  CAMPO-DADO PIC X(10) VALUE "MATEMATICA".
```

Agora, aplicando-se o comando:

```
MOVE CAMPO-DADO (4: 2) TO CAMPO-VELHO.
```

moverá o valor "EM" para CAMPO-VELHO. No caso, CAMPO-DADO (4: 2) é uma referência modificadora indicando dois bytes consecutivos iniciando pelo

quarto byte (da esquerda) do CAMPO-DADO. A sintaxe da referência modificadora é:

```
nome-de-dado (posição do byte: [comprimento])
```

Note que a especificação do comprimento é opcional e, se omitida, todos os bytes restantes do campo de dados serão referenciados. Por exemplo: MOVE CAMPO-DADO (3: ) TO CAMPO-VELHO moverá o valor " TEMATICA" para CAMPO-VELHO.

Pode existir um espaço após o delimitador dois pontos (:). A posição do caractere mais à esquerda tanto quanto o comprimento podem ser um literal, um identificador ou uma expressão aritmética indicando um valor positivo inteiro.

O nome de dado cuja referência é modificada pode ser de qualquer classe ou categoria, mas deverá apresentar a usage DISPLAY. Quando uma referência modificadora é usada, o item de dado indicado pelo nome de dado é tratado como se fosse um item de dado alfanumérico. O item de dado criado pela referência modificadora é considerado como item elementar e pode ser usado em qualquer item de dado alfanumérico permitido. Entretanto, o mesmo não pode ser usado por um campo emissor do comando UNSTRING ou por um campo receptor do comando STRING.

A referência modificadora pode ser aplicada também em campos que utilizam a cláusula REDEFINES e RENAMES. Porém a sua finalidade básica é permitir criar e referenciar campos subordinados sem defini-los explicitamente na DATA DIVISION.

## VERBO INSPECT COM A OPÇÃO CONVERTING

Existem três formatos deste verbo nas versões anteriores do COBOL. No COBOL-85 foi incrementado o quarto formato.

O seu formato com a opção CONVERTING é:

```
INSPECT identificador-1 ONVERTING { identificador-2 } TO
                                  { literal-1      }

{ identificador-3 } { BEFORE } INITIAL
{ literal-2       } { AFTER  }

{ identificador-4 }
{ literal-3       } ...
```

Como no INSPECT com a opção REPLACING, esta opção pode também ser usada para alterar caracteres em identificador-1 com outros caracteres. Identificador- 2/literal-1 e identificador-3/literal-2 proporcionam as bases para a combinação e a alteração. Referenciaremos estes campos como campo subjetivo e campo objetivo respectivamente. Estes campos devem ter tamanhos idênticos de modo que correspondam caracteres do campo subjetivo com os do campo objetivo. Um caractere do identificador-1 é selecionado para alterar apenas se o caractere for encontrado no campo subjetivo. É então alterado o caractere correspondente do campo objetivo. Por exemplo, consideremos o comando:

```
INSPECT CAMPO-A CONVERTING "ABCDE" BY "EDCBA".
```

Agora, se o CAMPO-A contém "ABREVIATURA" antes da execução, conterá "EDRAVIETURE" após a execução do procedimento INSPECT. Em geral:

```
INSPECT A CONVERTING B TO C.
```

é equivalente a:

```
INSPECT A REPLACING ALL   B (1: 1) BY C (1: 1)
                          B (2: 1) BY C (2: 1)
                          ..... .... .....
                          B (n: 1) BY C (n: 1).
```

onde n é um literal indicando o tamanho de B bem como de C. Entretanto, INSPECT com a opção CONVERTING é mais conveniente. As opções BEFORE e AFTER têm o mesmo significado das outras opções do comando INSPECT. Todos os identificadores devem ter a usage DISPLAY.

## VERBO CONTINUE

O verbo CONTINUE é outro incremento do COBOL-85. Não indica operação. O formato é:

```
CONTINUE
```

Pode ser usado em qualquer procedimento condicional ou imperativo. Por exemplo:

```
READ ARQ-A RECORD AT END CONTINUE.
```

é usado quando o fim condicional do arquivo ocorrer durante a execução do comando READ.

O comando CONTINUE é funcionalmente similar ao comando EXIT, só que os mesmos têm objetivos diferentes. O EXIT deve ser usado para se ter um ponto final comum dentro de um parágrafo ou seção, o CONTINUE pode ser usado em qualquer parte quando um passo nulo for requerido. Pode também ser usado nos procedimentos IF na troca pela frase NEXT SENTENCE.

## MANIPULAÇÃO DE ARQUIVO

Várias alterações têm sido feitas nas características referentes à manipulação de arquivos. Não discutiremos todas essas alterações porque a manipulação de arquivos é bastante dependente dos sistemas operacionais. Portanto, nos restringiremos apenas às alterações mais importantes.

*a.* cláusula ASSIGN: a sintaxe da cláusula ASSIGN pode ser alterada para:

**ASSIGN** TO {nome-implementor / literal}

o literal é uma criação e indica a especificação do arquivo. Por exemplo:

```
SELECT ARQ ASSIGN TO "SALARIO".
```

selecionamos um arquivo seqüencial chamado SALARIO no disco do usuário. Esta forma de ASSIGN elimina a necessidade de uso da cláusula VALUE OF na entrada FD.

*b.* entrada FD: todas as cláusulas na entrada FD incluindo a cláusula LABEL RECORDS são opcionais. Uma nova forma da cláusula RECORD especificamente para arquivos com tamanho de registros variáveis foi incluída. O formato desta cláusula é:

**RECORD** IS **VARYING** IN SIZE [[FROM inteiro-1] [**TO** inteiro-2] CHARACTERS] [**DEPENDING** ON nome-de-dado-1]

Esta cláusula está incorporada à cláusula RECORD CONTAINS para registro de comprimento fixo. Permite ser especificada da seguinte maneira:

```
RECORD IS VARYING IN SIZE FROM 72 TO 96 CHARACTERS
DEPENDING ON CONTA-PALAVRA.
```

O tamanho atual do registro (que deve estar entre 72 e 96) é determinado pelo valor do CONTA-PALAVRA. Quando a cláusula DEPENDING é omitida, o tamanho é determinado pela descrição do registro que pode conter parte da variável definida através de OCCURS com a frase DEPENDING ON.

*c.* código de File Status: no COBOL-85, 25 novos códigos de status de arquivo foram definidos. Por exemplo, o código 04 verifica se o comprimento do registro a ser lido é compatível com a descrição do registro.

*d.* comandos SORT e MERGE: as INPUT e OUTPUT PROCEDURE do comando SORT e a OUTPUT PROCEDURE do comando MERGE não necessitam ser codificadas em uma ou mais seções consecutivas. Elas podem ser codificadas como parágrafos consecutivos porque as frases INPUT PROCEDURE e OUTPUT PROCEDURE podem agora referir-se também ao nome de parágrafo.

Uma nova frase opcional com a sintaxe:

```
WITH DUPLICATES IN ORDER
```

foi incluída no procedimento SORT. Quando usado, terá de estar no registro com a chave idêntica e a ordem original relativa será mantida. O exemplo abaixo apresenta o uso da frase:

```
SORT   ASORT ASCENDING  KEY DPTO
                DESCENDING KEY SAL-BAS
                WITH DUPLICATES IN ORDER
                USING ARQ-E GIVING ARQ-S.
```

Além disso, no COBOL-85, múltiplos nomes de arquivos podem ser dados na frase GIVING dos verbos SORT e MERGE para ter múltiplos arquivos de saída. Os nomes de arquivos das frases USING e GIVING do procedimento MERGE podem ser de organização seqüencial, relativa ou indexada.

*e.* procedimento START: para se tornar compatível com a introdução do novo operador relacional, o procedimento START foi também incrementado. O procedimento aceita agora a forma adicional na frase KEY.

**KEY** IS {**GREATER** THAN OR **EQUAL** TO } nome-de-dado.

Este formato tem o mesmo significado da frase KEY IS com a opção NOT LESS THAN ou NOT <

## Cláusula USAGE

Mais duas USAGE estão especificadas no COBOL-85. São elas BINARY e PACKED-DECIMAL. A exata representação interna dependerá obviamente do sistema operacional e do compilador. Na maioria dos casos, BINARY é o mesmo que COMP e PACKED-DECIMAL é o mesmo que a descrição de COMP-3.

## Sub-rotinas COBOL

No COBOL-85, parâmetros podem ser passados para uma sub-rotina por referência ou pelo conteúdo. Se o parâmetro é passado por referência, o programa que chama bem como o chamado referem-se à mesma área de trabalho para o valor do parâmetro. Mas, se o programa chamado alterar o valor do parâmetro formal, haverá reflexos no valor do parâmetro atual. Por outro lado, se o parâmetro é passado pelo conteúdo, o parâmetro formal e o atual não partilham da mesma área. Em tempo de chamada, uma cópia do valor corrente do parâmetro atual é passada para o parâmetro formal. Portanto, se o programa chamado alterar o valor do parâmetro formal, a troca não terá reflexos no valor do parâmetro atual. Um meio para ajustar esses dois tipos de passagem de parâmetros foi incrementado à frase USING do verbo CALL. O formato dessa frase é:

USING { [BY REFERENCE] identificador ... / BY CONTENT identificador ... }

Observe que BY REFERENCE é opcional. Se nenhum dos dois for especificado, será assumido REFERENCE. Cada especificação de REFERENCE ou CONTENT aplica-se a todos os identificadores sucessivamente até que uma frase diferente (REFERENCE/CONTENT) seja especificada.

Consideremos um exemplo para o procedimento CALL:

```
CALL "SUBROT" USING BY CONTENT DADO-X DADO-Y BY REFERENCE
DADO-Z.
```

O cabeçalho PROCEDURE DIVISION para a sub-rotina será:

```
PROCEDURE DIVISION USING DADO-A DADO-B DADO-C.
```

Obviamente, os parâmetros atuais DADO-X, DADO-Y e DADO-Z correspondem respectivamente aos parâmetros formais DADO-A, DADO-B e DADO-C. Aqui, DADO-X e DADO-Y são passados pelo conteúdo enquanto DADO-Z é passado pela referência. Isto significa que DADO-Z e DADO-C ocupam a mesma área de trabalho. Portanto, se a sub-rotina alterar o valor de DADO-C, o valor de DADO-Z será também modificado. Entretanto, o mesmo não é verdadeiro para DADO-X / DADO-A ou para DADO-Y e DADO-B. Quando o procedimento CALL for executado, o valor de DADO-X e DADO-Y é transferido, respectivamente, para DADO-A e DADO-B. Após isso cessam as relações entre os parâmetros formal e atual.

A questão importante que se apresenta agora é quando se passa um parâmetro pela referência e quando se passa pelo conteúdo. Como regra geral, se observarmos que a sub-rotina calcula o valor do parâmetro e retorna o valor para o programa principal, o parâmetro pode ser passado pela referência. Um outro meio é, se a sub-rotina usa apenas o valor do parâmetro e não altera o valor original do parâmetro atual, passamos pelo conteúdo.

Mais uma alteração na comunicação entre programas (sub-rotina) é o procedimento EXIT PROGRAM. Nas outras versões do COBOL, este comando podia ser codificado apenas como um procedimento no parágrafo. Agora esta restrição foi abandonada, podendo ser usado como último procedimento de qualquer parágrafo dentro da sub-rotina.

## Verbo DISPLAY

A sintaxe do verbo DISPLAY foi expandida para:

```
DISPLAY { literal-1      } ... [UPON nome-mnemônico]
        { identificador-1}
        [ WITH NO ADVANCING ]
```

A frase WITH NO ADVANCING pode ser usada como terminal interativo. Geralmente, após a execução do comando DISPLAY, o cursor move para a primeira posição da linha seguinte da tela.

Quando a frase WITH NO ADVANCING for usada, o cursor será posicionado após o último caractere exibido. Portanto, o procedimento ACCEPT será aceito após o último caractere exibido da mesma linha.

## CARACTERÍSTICAS DO COBOL-85 PARA PROGRAMAÇÃO ESTRUTURADA

A nova característica do COBOL-85 incorpora principalmente a facilidade da programação estruturada. Descreveremos a seguir os procedimentos dessas características um a um:

### Verbo IF

O verbo IF foi substancialmente incrementado no COBOL-85. Sua sintaxe é:

```
IF condicao THEN { { procedimento-1 } ... }
                 { NEXT SENTENCE       }

{ ELSE { procedimento-2 } ... END-IF }
{ ELSE NEXT SENTENCE                 }
{ END-IF                             }
```

Como nas outras versões do COBOL, a frase ELSE NEXT SENTENCE pode ser omitida se o procedimento terminar por um período (.). Assim, mesmo com as mudanças, qualquer procedimento IF escrito de acordo

com a sintaxe do COBOL de outras versões é válido. O procedimento possui agora a introdução da opção da palavra THEN e o escopo terminador END-IF. O propósito do THEN é melhorar a documentação do comando e não alterar o efeito da função. O uso do terminador END-IF facilita escrever as sentenças IF mais facilmente. Os exemplos abaixos ilustram o novo formato:

Exemplo 1:

```
IF PRECO IS GREATER THAN OR EQUAL TO 500,00
    THEN MOVE 1 TO CODIGO-PRECO
ELSE
    MOVE 2 TO CODIGO-PRECO  END-IF
ADD QUANTIA DO TOTAL.
```

No caso, se a condição especificada for verdadeira, 1 é movido para CODIGO-PRECO senão 2 será movido. O procedimento ADD QUANTIA TO TOTAL é executado em ambos os casos. Note que o procedimento IF foi encerrado por um terminador END-IF e não por um período (.). Se END-IF for omitido, o conjunto deve terminar por um período.

Exemplo 2:

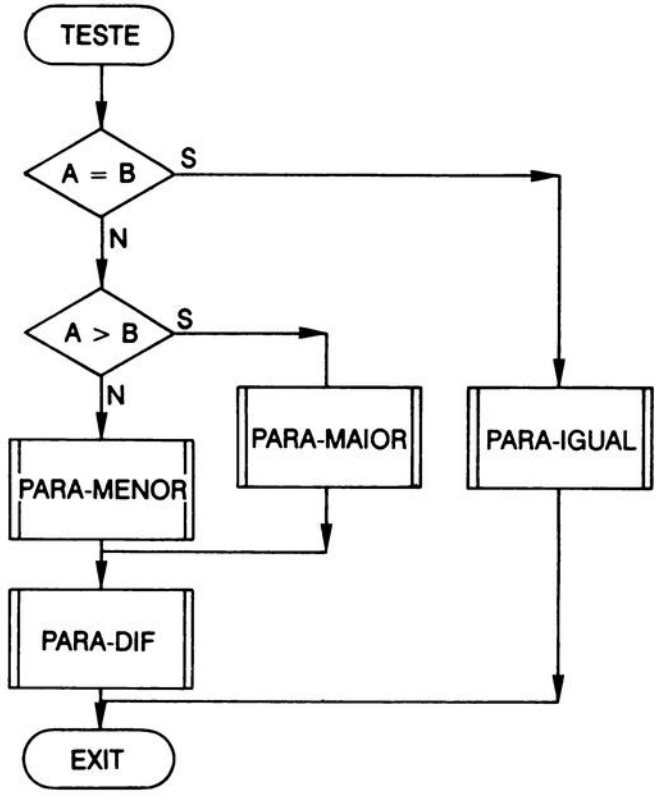
```
IF CODIGO = 1
    THEN IF DIAS IS GREATER THAN 175
            THEN MOVE 1 TO BONUS END-IF
         ELSE IF DIAS IS GREATER THAN 205
                 THEN MOVE 2 TO BONUS END-IF
END-IF.
```

Se CODIGO = 1 e DIAS > 175, BONUS será setado para 1. Se CODIGO for igual a 1 e DIAS > 205, BONUS será setado para 2. Em todos os outros casos, BONUS não será alterado. É conveniente observar o uso do terminador END-IF para mostrar a estrutura aninhada da sentença IF.

Exemplo 3:

Em diversas versões do COBOL, o terminador de um procedimento IF poderia ser um período (.) ou um ELSE do mesmo IF, o que determinava

algumas críticas para esta instrução. Agora, com a introdução do terminador END-IF o comando fica claro quando se descreve um ninho de IF.




```
TESTE.

IF A = B THEN PERFORM PARA-IGUAL
 ELSE IF A  > B
         THEN PERFORM PARA-MAIOR
     ELSE PERFORM PARA-MENOR
     END-IF
     PERFORM PARA-DIF
END-IF.
```

Por causa do terminador END-IF, foi possível terminar o escopo do procedimento IF antes do PERFORM PARA-DIF.

## REALCE PARA VERBOS CONDICIONAIS

A linguagem COBOL possui um grande número de verbos que implicitamente testam certas condições. A frase READ com AT END ou INVALID KEY, a frase ADD com ON SIZE ERROR são exemplos de procedimentos condicionais. No COBOL-85, dois tipos de realce foram incorporados para esses verbos condicionais. Primeiramente, o terminador END-verbo foi incluído para cada um desses verbos. Exemplos de terminadores END-verbo são END-READ, END-ADD, END-WRITE etc. Segundo, uma correspondência para cada frase que testa uma condição implícita, uma combinação da frase opcional pode ser fornecida dentro do escopo do procedimento. A combinação de frase testa a condição oposta. NOT AT END, NOT INVALID KEY são exemplos de combinação dessas frases. Os exemplos seguintes ilustram essas características:

Exemplo 1:

```
DIVIDE PRECO BY QTDE GIVING TAXA
    ON SIZE ERROR MOVE ZERO TO NOVA-TAXA
    NOT ON SIZE ERROR ADD 1 TAXA GIVING NOVA-TAXA
END-DIVIDE.
```

Este formato facilita o entendimento do mecanismo. No caso de erro de tamanho durante a divisão de PRECO por QTDE, NOVO-PRECO é setado para zero. O que deve ser visto no exemplo é o terminador END-DIVIDE e a combinação da frase NOT ON SIZE ERROR.

Exemplo 2:

```
READ ARQ-E RECORD
    AT END SET NAO-REGISTRO TO TRUE
    NOT AT END
        MOVE REG-E TO REG-S
        WRITE REG-S
END-READ.
```

Sempre que um registro do ARQ-E é lido no comando READ acima, os comandos imperativos na combinação da frase NOT AT END são

executados. No caso de fim de arquivo ser encontrado, o nome de condição NAO-REGISTRO é setado para verdadeiro. Observe a utilidade da frase NOT AT END. Se não usarmos esta frase, um procedimento IF será necessário. A codificação equivalente sem a frase NOT AT END seria:

```
READ ARQ-E RECORD
    AT END SET NAO-REGISTRO TO TRUE END-READ.
IF NOT NAO-REGISTRO
    MOVE REG-E TO REG-S
    WRITE REG-S  END-IF.
```

Verbos com condicional e suas combinações

| FRASE CONDICIONAL | FRASE COMBINADA | VERBO |
|---|---|---|
| ATEND | NOT AT END | READ, RETURN |
| AT {END-OF-PAGE/ EOP} | NOT AT {END-OF-PAGE / EOP} | WRITE |
| INVALID KEY | NOT INVALID KEY | READ, WRITE, REWRITE, DELETE, START |
| ON SIZE ERROR | NOT ON SIZE ERROR | ADD, SUBTRACT, MULTIPLY, DIVIDE, COMPUTE |
| ON OVERFLOW | NOT ON OVERFLOW | STRING, UNSTRING |
| ON OVERFLOW | --- | CALL |
| WHEN | --- | SEARCH |

A tabela acima apresenta os verbos, a condicional e as frases combinadas. A frase combinada, se usada, deve ser escrita após a frase condicional. Assim como a frase condicional, a frase combinada pode ser incluída apenas no modo imperativo dentro do mesmo. Cada verbo mencionado na tabela pode ter um terminador END-verbo. Quando um procedimento condicional estiver dentro de um terminador END-verbo, o procedimento pode ser chamado de delimitador. Um delimitador é tratado como um procedimento imperativo e pode ser usado em qualquer parte desse procedimento. Entretanto, existem compiladores COBOL (incorporando características do

COBOL-85) que não tratam como delimitadores dentro de um procedimento imperativo.

## Verbo EVALUATE

EVALUATE é um novo procedimento do COBOL-85. O mesmo implementa a estrutura "CASE". Na intenção de construir um comando poderoso, muitas opções foram criadas neste procedimento.

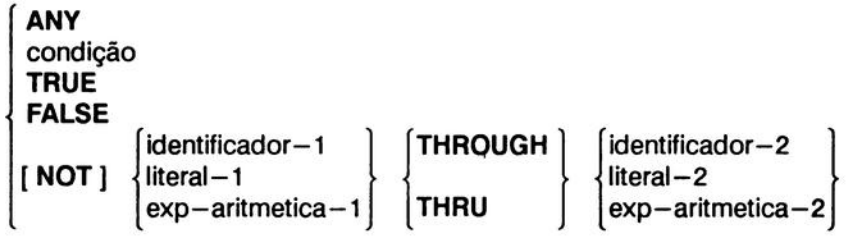
Seu formato é:

**EVALUATE** sujeito-1 [**ALSO** sujeito-2] ... { {**WHEN** objeto-1 [**ALSO** objeto-2] ... } ... } comando-imperativo-1} ... [**WHEN OTHER** comando-imperativo-2] [**END-EVALUATE**]

onde o sujeito é:

{identificador / literal / expressão / **TRUE** / **FALSE**}

e o objetivo é:




O propósito do procedimento é explorar as seguintes regras:

*a.* associar com o comando EVALUATE a lista de sujeitos e a de objetos;

*b.* a lista de sujeitos é especificada entre a palavra EVALUATE e o primeiro aparecimento do WHEN. O sujeito pode ser um identificador ou literal. Pode ser também uma expressão/condição

aritmética ou as palavras-chaves TRUE/FALSE. Assim, na avaliação, o sujeito conterá um valor numérico, um valor não numérico ou um valor condicional (mostrado pelo TRUE, FALSE ou expressão condicional). Consideremos alguns exemplos especificando o sujeito:

Exemplos:

1. TRUE – A lista de sujeitos consiste apenas de um elemento o qual avaliará a condição ao valor TRUE.

2. CODIGO-TRANS – Assume que CODIGO-TRANS é um identificador. Neste caso, também, a lista de sujeitos consiste de apenas um elemento que avaliará o valor de CODIGO-TRANS.

3. CODIGO-TRANS ALSO DATA-TRANS ALSO CODIGO-CLIENTE – Aqui a lista de sujeitos consiste de três elementos. São os valores de CODIGO-TRANS, DATA-TRANS e CODIGO-CLIENTE. Os valores dos sujeitos não necessitam ser da mesma classe. Por exemplo, CODIGO-TRANS pode ser numérico enquanto CODIGO-CLIENTE é alfanumérico. Quando a lista contiver mais de um sujeito, a posição do mesmo dentro da lista é importante. Esta posição é designada de **ORDINAL**.

*c*. Cada frase WHEN especifica uma lista de objetos. O valor do objeto pode ser um dos seguintes: um valor numérico/não numérico, uma faixa de valores numéricos/não numéricos, um valor condicional ou qualquer valor. Identificador-1/literal-1/expressão aritmética-1 indicam um valor simples numérico/não numérico quando a frase THROUGH/THRU é omitida. Quando a frase THROUGH/THRU for especificada, uma faixa de valores numérico/não numérico é indicada. Condição-1/TRUE/FALSE indica um valor condicional para o objeto. Alguns exemplos de especificação de lista dos objetos estão a seguir:

Exemplos:

1. 3 – O objeto consiste de apenas um elemento que especifica o valor 3.

2. 3 THRU 10 – Aqui também o objeto consiste de um elemento, sendo que este elemento avalia um conjunto de valores.

3. VAL-1 THRU VAL-2 ALSO ANY ALSO QTDE > 500 – A lista de objetos consta de três elementos. O primeiro avalia um conjunto de valores, o segundo avalia qualquer valor e o terceiro avalia uma condição de valor falso ou verdadeiro dependendo do valor de QTDE. Como no caso de uma lista de sujeitos, a posição ordinal de um objeto dentro da lista de objetos é importante.

4. Durante a execução do procedimento EVALUATE, os valores da lista de sujeitos são comparados com os valores da lista dos objetos na frase WHEN para estabelecer um "match" entre os dois. A comparação a ser processada é a seguinte:

– o valor do sujeito é comparado com o valor/conjunto de valores do objeto na correspondência da posição ordinal;

– no caso de um único objeto (numérico/não numérico), a comparação sujeito-objeto é feita de modo usual;

– quando um conjunto de valores for especificado para o objeto, a comparação sujeito-objeto resulta em verdadeiro, se o valor do sujeito pertencer ao conjunto;

– no caso de valores condicionais, a comparação sujeito-objeto resulta em verdadeiro, se ambos avaliam o mesmo valor (isto é, ambos são verdadeiros ou ambos são falsos);

– se ANY for especificado para um objeto, a comparação sujeito-objeto sempre resulta em verdadeiro;

— a lista de sujeitos é indicada para um "match" com a lista do objeto, se toda a correspondência de comparação sujeito-objeto resultar em verdadeiro.

5. Cada frase WHEN especifica uma lista de objetos. As frases WHEN são vistas como um "match" na ordem que eles aparecem dentro do procedimento EVALUATE. Entretanto, no resultado de um "match" o primeiro procedimento que segue a frase WHEN é selecionado para execução e o comando EVALUATE é encerrado. A frase WHEN OTHER, se especificada, é selecionada apenas se previamente o nome da frase WHEN não for escolhida.

O procedimento EVALUATE sem as frases ALSO torna-se muito simplificado porque é o caso em que existem um sujeito e um objeto (para cada WHEN). Esta forma de procedimento EVALUATE é útil na implementação da estrutura "CASE" sem o uso do procedimento GO TO conversão de tabela de decisões para o programa. Os exemplos abaixo ilustram o uso do procedimento EVALUATE.

Exemplo 1:

MES e NR-DIAS são campos de dois dígitos numéricos inteiros. Os valores 1, 2, 3 etc. para MES indicam respectivamente, Janeiro, Fevereiro, Março etc. Dependendo do valor do MES, queremos mover 30, 31 ou 28 para NR-DIAS. Por exemplo, se o valor do MES for 1, moveremos 31; se for 2, moveremos 28 e assim por diante. O comando EVALUATE para o caso é o seguinte:

```
EVALUATE TRUE
    WHEN MES = 4 OR 6 OR 9 OR 11
        MOVE 30 TO NR-DIAS
    WHEN MES = 2
        MOVE 28 TO NR-DIAS
    WHEN OTHER
        MOVE 31 TO NR-DIAS
END-EVALUATE.
```

Neste caso, estamos supondo que o campo MES tenha um valor correto.

Exemplo 2:

Vamos supor que NOTAS contém as notas obtidas pelos estudantes e GRAU é um campo alfanumérico de uma posição. Queremos calcular o GRAU de acordo com as seguintes regras:

| NOTAS | GRAU |
|---|---|
| 80-100 | A |
| 60-79 | B |
| 45-59 | C |
| 30-44 | D |
| 0-29 | E |

O comando EVALUATE para o caso é o seguinte:

```
EVALUATE NOTAS
    WHEN 80 THRU 100 MOVE "A" TO GRAU
    WHEN 60 THRU 79  MOVE "B" TO GRAU
    WHEN 45 THRU 59  MOVE "C" TO GRAU
    WHEN 30 THRU 44  MOVE "D" TO GRAU
    WHEN  0 THRU  29 MOVE "E" TO GRAU
    WHEN OTHER       MOVE "W" TO GRAU
END-EVALUATE.
```

O literal "W" é movido para GRAU no caso de NOTA ser diferente das mencionadas.

Exemplo 3:

Consideremos a seguinte tabela de decisões:

| | | | | | |
|---|---|---|---|---|---|
| TIPO-PRODUTO | = 1 | S | S | N | N |
| CATEGORIA | = 1 | S | N | S | N |
| COMISSAO | = 10% | S | | | |
| COMISSAO | = 8% | | S | | S |
| COMISSAO | = 12% | | | S | |

Observemos, agora, a seguinte codificação com o comando IF:

```
        IF TIPO-PRODUTO = 1 AND CATEGORIA = 1
            MOVE 10 TO COMISSAO GO TO FIM-TESTE.
        IF TIPO-PRODUTO = 1 AND CATEGORIA = 2
            MOVE  8 TO COMISSAO GO TO FIM-TESTE.
        IF TIPO-PRODUTO = 2 AND CATEGORIA = 1
            MOVE 12 TO COMISSAO GO TO FIM-TESTE.
        IF TIPO-PRODUTO = 2 AND CATEGORIA = 2
            MOVE  8 TO COMISSAO GO TO FIM-TESTE.
        IF TIPO-PRODUTO = 3 AND CATEGORIA = 1
            MOVE 10 TO COMISSAO GO TO FIM-TESTE.
        IF TIPO-PRODUTO = 3 AND CATEGORIA = 2
            MOVE 10 TO COMISSAO GO TO FIM-TESTE.
        MOVE ZERO TO COMISSAO.
FIM-TESTE.
```

... outros procedimentos.

Agora com o auxílio do comando EVALUATE, teremos:

```
        EVALUATE TIPO-PRODUTO ALSO CATEGORIA
                WHEN 1 ALSO 1 MOVE 10 TO COMISSAO
                WHEN 1 ALSO 2 MOVE  8 TO COMISSAO
                WHEN 2 ALSO 1 MOVE 12 TO COMISSAO
                WHEN 2 ALSO 2 MOVE  8 TO COMISSAO
                WHEN 3 ALSO 1 THRU MOVE 10 TO COMISSAO
                WHEN OTHER    MOVE ZERO TO COMISSAO
        END-EVALUATE.
```

Podemos verificar a elegância com que foi codificada a tabela de decisão acima com o comando EVALUATE.

Exemplo 4:

O comando EVALUATE do exemplo 3 pode ser escrito da seguinte maneira:

```
EVALUATE TIPO-PRODUTO ALSO CATEGORIA
    WHEN 1 ALSO 1
    WHEN 3 ALSO 1 THRU 2
        MOVE 10 TO COMISSAO
    WHEN 1 ALSO 2
    WHEN 2 ALSO 2
        MOVE 8 TO COMISSAO
    WHEN 2 ALSO 1
        MOVE 12 TO COMISSAO
    WHEN OTHER
        MOVE ZERO TO COMISSAO
END-EVALUATE.
```

Observe que quando a frase WHEN não possui um comando imperativo, o próximo procedimento é executado. Assim, se TIPO-PRODUTO = 1 e CATEGORIA = 1, 10 será movimentado para COMISSAO.

## Verbo PERFORM

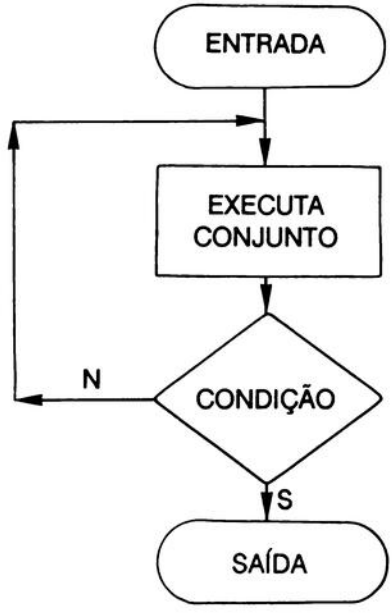
No COBOL-85, o verbo PERFORM foi modificado em dois casos. Para explanar essas duas alterações, consideremos PERFORM com a opção UNTIL como a representação do comando PERFORM. O novo formato do PERFORM com a opção UNTIL é:

**PERFORM** nome-de-proc-1 { [ **THROUGH** / **THRU** } nome-de-proc-2 ]

[ WITH **TEST** { **BEFORE** / **AFTER** } / **UNTIL** condicao ] [ comando-imperativo **END-PERFORM** ]

A frase TEST é uma opção nova. Se TEST BEFORE for especificada, a condição será testada no início de cada repetição do comando PERFORM. Se TEST AFTER for especificada, a condição será testada ao final de cada repetição do comando PERFORM. Assim, se a condição for verdadeira no início, todo o conjunto não será executado quando TEST BEFORE for

especificada. Em outras versões do COBOL, o comando PERFORM com a opção UNTIL é semelhante ao caso da frase TEST BEFORE. Como tal, TEST BEFORE é o default. O funcionamento do comando com a frase AFTER é mais bem ilustrado através da figura abaixo:




A frase TEST é também adequada para PERFORM com a opção VARYING bem como PERFORM com a opção VARYING ... AFTER. Entretanto, a frase TEST não é adequada para PERFORM com a opção TIMES posto que é também destinada para laços. Não há pergunta da frase TEST no caso de PERFORM simples. Os exemplos abaixo ilustram o uso da frase TEST.

Exemplo 1:

```
PERFORM ROT-LOOP WITH TEST BEFORE
        UNTIL CONTADOR > 30.
```

Esta instrução tem efeito como o procedimento das outras versões do COBOL:

```
PERFORM ROT-LOOP UNTIL CONTADOR > 30.
```

Exemplo 2:

Consideremos os dois procedimentos seguintes:

```
PERFORM ROT-LOOP WITH TEST BEFORE
    VARYING CONTADOR FROM 1 BY 1 UNTIL CONTADOR  >  30.

e

PERFORM ROT-LOOP WITH TEST AFTER
    VARYING CONTADOR FROM 1 BY 1 UNTIL CONTADOR  >  30.
```

A diferença essencial entre os dois é que no primeiro caso o laço é executado 30 vezes e no segundo caso é executado 31 vezes. Quando TEST AFTER é especificado, a condição é testada imediatamente após a execução do conjunto de instruções (no caso, o parágrafo ROT-LOOP).

Exemplo 3:

Vamos supor que existe um arquivo seqüencial indicado por ARQ-TESTE no programa. Além disso, a descrição de registro deste arquivo possui um campo inteiro chamado SINAL. Queremos saltar registros deste arquivo até que seja lido o valor de SINAL igual a 9. A codificação a seguir apresenta o uso do PERFORM usando as frases TEST BEFORE e TEST AFTER.

```
    PERFORM LEITURA WITH TEST AFTER
        UNTIL SINAL IS EQUAL TO 9.
    |
    |
LEITURA.
    READ ARQ-TEST AT END
        DISPLAY "REGISTRO NAO ENCONTRADO" STOP RUN.
```

A codificação acima é válida porque SINAL é testado apenas após a execução do parágrafo LEITURA. Entretanto, se trocarmos a frase TEST para TEST BEFORE, SINAL será testado antes da leitura de qualquer registro. Isto não é bom porque SINAL está definido dentro do próprio registro. Portanto,

para codificar o mesmo procedimento com a frase TEST BEFORE utilizamos a seguinte maneira:

```
        PERFORM LEITURA.
        PERFORM LEITURA WITH TEST BEFORE
            UNTIL SINAL IS EQUAL TO 9.
        |
        |
LEITURA.
        READ ARQ-TEST AT END
            DISPLAY "REGISTRO NAO ENCONTRADO" STOP RUN.
```

A estrutura do PERFORM ... UNTIL com TEST AFTER é conhecida como "DO UNTIL" em oposição à estrutura "DO WHILE". Muitas vezes a estrutura "DO UNTIL" é considerada como básica para a programação estruturada.

Uma outra revisão do procedimento PERFORM é a introdução do PERFORM "in-line". A maior dificuldade do comando PERFORM é que o parágrafo codificado, que é para ser executado, não pode ser incluído dentro do próprio comando, isto é, o processo é considerado "out-of line". Agora em todos os formatos do procedimento PERFORM, comandos imperativos podem ser incluídos dentro do escopo do procedimento que é encerrados com o delimitador END-PERFORM.

Exemplo 4:

```
PERFORM UNTIL I  > 30
    ADD TAB-A(I) TO TOTAL
    ADD 1 TO I
END-PERFORM.
```

A estrutura do comando PERFORM consiste de duas instruções ADD que são incluídas dentro do escopo do próprio PERFORM.

Exemplo 5:

```
PERFORM 30 TIMES
    ADD TAB-A(I) TO TOTAL
    ADD 1 TO I
END-PERFORM.
```

Este exemplo ilustra o PERFORM "in-line" com a opção TIMES.

Exemplo 6:

```
PERFORM VARYING I FROM 1 BY 1 UNTIL I  > 30
    ADD TAB-A(I) TO TOTAL
END-PERFORM.
```

Este é um exemplo do comando PERFORM "in-line" com opção VARYING.

Exemplo 7:

```
PERFORM
    MOVE REG-E TO REG-S
    WRITE REG-S
END-PERFORM.
```

Este exemplo ilustra o comando PERFORM "in-line" simples. Embora este tipo de codificação seja permitido, o procedimento PERFORM se torna ineficiente.

Os comentários abaixo deverão ser observados para o novo comando PERFORM.

*a.* quando o nome de procedimento (com ou sem a frase THROUGH/THRU) for especificado, o comando PERFORM é utilizado como PERFORM "out-of line". O comando imperativo e o delimitador END-PERFORM podem não ser especificados neste caso;

*b.* quando o nome de procedimento for omitido, o comando PERFORM é utilizado como PERFORM "in-line". O comando imperativo e o delimitador END-PERFORM deverão ser especificados neste caso;

*c.* um comando PERFORM "in-line" funciona exatamente como o comando PERFORM "out-of line" com a exceção de que o

comando imperativo contido dentro do escopo é executado ao em vez do conjunto "out-of line".

O procedimento PERFORM com a opção VARYING ... AFTER não tem limite de frases AFTER. Nas outras versões do COBOL era limitado em dois.

## EXEMPLOS DE PROGRAMAS

O primeiro programa usa os arquivos seqüenciais denominados ARQ-E e ARQ-S. O objetivo é ler os registros do ARQ-E, criptografar e gravar no arquivo ARQ-S. Para criptografar as letras foi usado o verbo INSPECT com a frase CONVERTING.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. CRIPTO.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
IDENTIFICATION DIVISION.
PROGRAM-ID. CRIPTO.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT ARQ-E ASSIGN TO DISK
        ORGANIZATION LINE SEQUENTIAL.
    SELECT ARQ-S ASSIGN TO DISK
        ORGANIZATION LINE SEQUENTIAL.
DATA DIVISION.
FILE SECTION.
FD  ARQ-E LABEL RECORD STANDARD
        VALUE OF FILE-ID ENTRADA-E.
01  REG-E PIC X(80).
FD  ARQ-S LABEL RECORD STANDARD
        VALUE OF FILE-ID SAIDA-S.
01  REG-S PIC X(80).
WORKING-STORAGE SECTION.
77  ENTRADA-E      PIC X(14).
```

```
77  SAIDA-S          PIC X(14).
77  STR-1            PIC X(26).
77  STR-2            PIC X(26).
77  STATUS           PIC 9.
    88 FIM-ARQ VALUE 1.
PROCEDURE DIVISION.
INICIO-DO-PROGRAMA.
    DISPLAY (01 01) ERASE.
    DISPLAY (05 10) "ARQUIVO DE ENTRADA:".
    ACCEPT  (05 30) ENTRADA-E.
    DISPLAY (07 10) "ARQUIVO DE SAIDA:".
    ACCEPT  (07 30) SAIDA-S.
    OPEN INPUT ARQ-E OUTPUT ARQ-S.
    DISPLAY (10 10) "ALFABETO PADRAO".
    ACCEPT  (10 30) STR-1.
    DISPLAY (12 10) "P/CRIPTOGRAFAR".
    ACCEPT  (12 30) STR-2.
    INITIALIZE STATUS.
    PERFORM ROT-GRAVA UNTIL FIM-ARQ.
    CLOSE ARQ-E ARQ-S.
    STOP RUN.
ROT-GRAVA.
    READ ARQ-E RECORD
        AT END SET FIM-ARQ TO TRUE
        NOT AT END INSPECT REG-E CONVERTING STR-1 TO
                                STR-2
                                WRITE REG-S FROM REG-E.
END PROGRAM CRIPTO.
```

O segundo programa apresenta a manipulação do verbo EVALUATE, atualizando um arquivo.

```
 IDENTIFICATION DIVISION.
PROGRAM-ID. P-MESTRE.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT ARQ-MESTRE  ASSIGN TO DISK
        ORGANIZATION LINE SEQUENTIAL.
    SELECT ARQ-TRANS   ASSIGN TO DISK
        ORGANIZATION LINE SEQUENTIAL.
    SELECT ARQ-NOVO    ASSIGN TO DISK
        ORGANIZATION LINE SEQUENTIAL.
```

```
DATA DIVISION.
FILE SECTION.
FD  ARQ-MESTRE LABEL RECORD STANDARD
            VALUE OF FILE-ID "MESTRE".
01  REG-MESTRE.
    02 CH-MESTRE PIC X(05).
    02           PIC X(45).
FD  ARQ-TRANS    LABEL RECORD STANDARD
                 VALUE OF FILE-ID "TRANS"
01  REG-TRANS.
    05 COD-TRANS PIC 9.
    88 INSERCAO  VALUE 1.
    88 DELECAO   VALUE 2.
    88 MODIFICA  VALUE 3.
    05 DADOS-TRANS.
       10 CH-TRANS   PIC X(05).
       10           PIC X(45).
FD  ARQ-NOVO LABEL RECORD STANDARDDD
                VALUE OF FILE-ID "NOVO".
01  REG-NOVO          PIC X(50).
WORKING-STORAGE SECTION.
01  NOMES-DE-CONDICAO.
    02 FIM-MESTRE     PIC X(05).
    88 MESTRE-MAIS-REG VALUE "FALSE".
    88 MESTRE-NAO-REG  VALUE "TRUE".
    02 FIM-TRANS      PIC X(05).
    88 TRANS-MAIS-REG VALUE "FALSE".
    88 TRANS-NAO-REG VALUE "TRUE".
    02 FIM-AMBOS      PIC X(05).
    88 FIM                 VALUE "TRUE".
01  FLAG               PIC X.
PROCEDURE DIVISION.
INICIO-DO-PROGRAMA.
    OPEN INPUT ARQ-MESTRE ARQ-TRANS
       OUTPUT ARQ-NOVO.
    INITIALIZE NOMES-DE-CONDICAO REPLACING
       ALPHANUMERIC DATA BY "FALSE".
    DISPLAY (01 01) ERASE.
    PERFORM LER-MESTRE. PERFORM LER-TRANS.
    PERFORM AVALIAR UNTIL FIM.
    CLOSE ARQ-MESTRE ARQ-TRANS ARQ-NOVO WITH LOCK.
    STOP RUN.
AVALIAR.
    EVALUATE CH-MESTRE  > CH-TRANS
       ALSO CH-MESTRE = CH-TRANS
```

```
                ALSO MESTRE-MAIS-REG
                ALSO TRANS-MAIS-REG
                ALSO TRUE
            WHEN  TRUE ALSO FALSE ALSO TRUE ALSO TRUE ALSO
                  INSERCAO
                  WRITE REG-NOVO FROM DADOS-TRANS
                  PERFORM LER-TRANS
            WHEN  FALSE ALSO TRUE ALSO TRUE ALSO TRUE ALSO
                  DELECAO
                  PERFORM LER-MESTRE
                  PERFORM LER-TRANS
            WHEN  FALSE ALSO TRUE ALSO TRUE ALSO TRUE ALSO
                  MODIFICA
                  WRITE REG-NOVO FROM DADOS-TRANS
                  PERFORM LER-MESTRE
                  PERFORM LER-TRANS
            WHEN  FALSE ALSO FALSE ALSO TRUE ALSO TRUE ALSO ANY
                  WRITE REG-NOVO FROM REG-MESTRE
                  PERFORM LER-MESTRE
             WHEN  ANY ALSO ANY ALSO TRUE ALSO FALSE ALSO ANY
                  PERFORM COPIA-MESTRE
            WHEN  ANY ALSO ANY ALSO FALSE ALSO TRUE ALSO
                  INSERCAO
                  PERFORM COPIA-TRANS
            WHEN   ANY ALSO ANY ALSO FALSE ALSO FALSE ALSO ANY
                  SET FIM TO TRUE
            WHEN  OTHER
                  PERFORM ACAO-ERRO
        END-EVALUATE.
    LER-MESTRE.
        READ ARQ-MESTRE RECORD
            AT END SET MESTRE-NAO-REG TO TRUE.
    LER-TRANS.
        READ ARQ-TRANS RECORD
            AT END SET TRANS-NAO-REG TO TRUE.
    COPIA-MESTRE.
        PERFORM UNTIL MESTRE-NAO-REG
            WRITE REG-NOVO FROM REG-MESTRE
            PERFORM LER-MESTRE
        END-PERFORM.
    COPIA-TRANS.
        PERFORM UNTIL TRANS-NAO-REG
            WRITE REG-NOVO FROM REG-TRANS
            PERFORM LER-TRANS-COM-TESTE
        END-PERFORM.
```

```
LER-TRANS-COM-TESTE.
    PERFORM LER-TRANS.
    PERFORM ACAO-ERRO WITH TEST BEFORE
        UNTIL TRANS-NAO-REG OR INSERCAO.
ACAO-ERRO.
    DISPLAY (10 10) "Erro quando transacao = "
    CH-TRANS
    DISPLAY (11 10) "Transacao ignorada".
    DISPLAY (14 10) "Pressione uma tecla p/continuar".
    ACCEPT  (14 40) FLAG.
    PERFORM LER-TRANS.
END PROGRAM P-MESTRE.
```

Lógica do verbo EVALUATE

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| CH-MESTRE < CH-TRANS | N | N | N | S | | | | |
| CH-MESTRE > CH-TRANS | S | N | N | N | | | | |
| CH-MESTRE = CH-TRANS | N | S | S | N | | | | E |
| MESTRE-MAIS-REG | S | S | S | S | S | N | N | L |
| TRANS-MAIS-REG | S | S | S | S | N | S | N | S |
| INSERÇÃO (COD-TRANS = 1) | S | N | N | | | S | | E |
| DELEÇÃO (COD-TRANS = 2) | N | S | N | | | N | | |
| MODIFICA (COD-TRANS = 3) | N | N | S | | | N | | |
| WRITE FROM REG-MESTRE | | | | X | | | | |
| WRITE FROM REG-TRANS | X | | | | | | | |
| WRITE REG MODIFICADO FROM REG-MESTRE/REG-TRANS | | | X | | | | | |
| COPIA REG DO ARQ-MESTRE | | | | | | | | |
| COPIA REG DO ARQ-TRANS | | | | | X | | | |
| READ ARQ-MESTRE | | X | X | X | | X | | |
| READ ARQ-TRANS | X | X | X | | | | | |
| FIM | | | | | | | X | X |
| AÇÃO-ERRO | | | | | | | | X |

O terceiro programa ilustra o aninhamento de três programas.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PRINCIPAL.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  N        PIC 999.
01  LISTA GLOBAL.
    05 A     PIC 9(4) OCCURS 100 TIMES.
01  I        PIC 999.
PROCEDURE DIVISION.
INICIO-E-FIM-DO-PROGRAMA.
    DISPLAY "Digite o valor de N " WITH NO ADVANCING.
    ACCEPT N.
    PERFORM VARYING I FROM 1 BY 1 UNTIL I  >  N
        ACCEPT A(I)
    END-PERFORM.
    CALL "ASORT" USING N  END-CALL.
    PERFORM VARYING I FROM 1 BY 1 UNTIL I  > N
        DISPLAY A(I)
    END-PERFORM.
    STOP RUN.
IDENTIFICATION DIVISION.
PROGRAM-ID. ASORT.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  I        PIC 999.
LINKAGE SECTION.
01  N        PIC 999.
PROCEDURE DIVISION USING N.
INICIO-E-FIM-PGM-ASORT.
    IF N  >  1
        PERFORM VARYING I FROM 2 BY 1 UNTIL I  > N
            CALL "INCLUIR" USING I END-CALL
        END-PERFORM
    END-IF.
    EXIT PROGRAM.
IDENTIFICATION DIVISION.
PROGRAM-ID. INCLUIR.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  TEMP   PIC 9(4).
01  K        PIC 999.
LINKAGE SECTION.
```

```
01  J          PIC 999.
PROCEDURE DIVISION USING J.
INICIO-DE-INCLUIR.
    MOVE J TO K.
    PERFORM UNTIL K  < = 1
        IF   A(K)  < A(K - 1)
             MOVE A(K)       TO TEMP
             MOVE A(K - 1)   TO A(K)
             MOVE TEMP       TO A(K - 1)
             SUBTRACT 1 FROM KQ
        ELSE MOVE 1 TO K
        END-IF
    END-PERFORM.
    END PROGRAM INCLUIR
    END PROGRAM ASORT.
    END PROGRAM PRINCIPAL.
```

# ÍNDICE ANALÍTICO

# CADASTRO PARA MALA DIRETA

## Favor preencher todos os campos

NOME (não abreviar):

END. RESIDENCIAL:

CIDADE: UF: CEP: FONE:

EMPRESA:

SEXO: F( ) M( ) NASCIMENTO:_____/____
MÊS ANO

**1. Cargo:** ( ) 1. Digitador ( ) 2. Programador
( ) 3. Analista ( ) 4. Supervisor
( ) 5. Gerente ( ) 6. Diretor
( ) 7. Outros (especificar):_____

**2. Escolaridade:** ( ) 1. 1º grau ( ) 2. 2º grau
( ) 3. 3º grau

**3. Eu geralmente, compro livros:**
( ) 1. Em livrarias ( ) 2. Por telefone
( ) 3. Em feiras ( ) 4. Por Reembolso Postal
( ) 5. Outros (especificar):_____

**4. Costumo comprar em média, a seguinte quantidade de livros por ano:**
( ) 1. 01 ( ) 2. 03 ( ) 3. 05 ( ) 4. mais

**5. Estou interessado em livros das seguintes áreas:**
( ) 1. Informática ( ) 2. Negócios ( ) 3. Controle Qualidade
( ) 4. Informática na Educação ( ) 5. Ensino Inglês ( ) 6. Psicologia
( ) 7. Sociologia ( ) 8. Metodologia ( ) 9. Medicina
( ) 10. Eng. Civil ( ) 11. Eng. Elétrica/ Eletrônica ( ) 12. Eng. Mecânica
( ) 13. Eng. Química ( ) 14. Física ( ) 15. Matemática
( ) 16. Outras (especificar):_____

**6. Utiliza microcomputador:**
( ) Sim ( ) Não

**7. Qual?**
( ) 1. PC ( ) 2. Apple ( ) 3. MSX
( ) 4. Outros (especificar):_____

**8. Softwares mais utilizados:**
( ) 1. Banco de Dados ( ) 2. Planilha ( ) 3. Proc. Texto
( ) 4. Outros (especificar):_____

**9. Linguagens:**
( ) 1. C ( ) 2. Cobol
( ) 3. Clipper/dBase ( ) 4. Assembler
( ) 5. Outras (especificar):_____

_____

**Obra:** Cobol para Microcomputadores
**Autoria:** Mutsuo Ono

**Editora McGRAW-HILL Ltda**
"Padrão em Livros de Qualidade"
Rua Tabapuã, 1105 – Itaim Bibi – São Paulo
04533 – Tels.: (011) 829-8604 / 820-8528

Impressão e Acabamento
GRÁFICA E EDITORA FCA
AV. HUMBERTO DE ALENCAR CASTELO BRANCO, 3972 - TEL.: 419-0200
SAO BERNARDO DO CAMPO - CEP 09700 - SP

# COBOL 80/COBOL MS/COBOL MB

## Características do Livro

a) Teorias desenvolvidas com intercalação de programas resolvidos e programas propostos.

b) Explanação dos comandos e instruções utilizadas em cada um dos programas resolvidos.

c) Instruções e comandos apresentados na seqüência da necessidade e do grau de dificuldade.

d) Programas rodados em **Cobol MS** da Microsoft que é por sua vez compatível com o **Cobol 80** e o **Cobol MB,** com exceção somente na formatação das telas.

## Mutsuo Ono

*Professor de Matemática e Física graduado pela Faculdade de Filosofia, Ciências e Letras Integrada de Guarulhos.*

*Possui cursos na área de Processamento de Dados tais como: Análise de Sistemas, Programação COBOL, Programação STV-1600, 1100 DEMAND/USER WORKSHOP, Cobol Aplicado ao SO 1100, Diretivas do DOS III do Equipamento HP, Symbolic Stream Generator/UNIVAC-1100, Open Access II, entre outros.*

*É professor de Linguagem de Programação do Colégio Comercial Brasil e Programador no Centro de Informática n.º 2.*

0-07-450248-4

McGraw Hill