PROGRAMAÇÃO

Resumo de Linguagem COBOL

INTRODUÇÃO

A palavra COBOL é a abreviação de Commom Busines Oriented Language. Esta é uma linguagem de computador orientada para negócios. As regras que comandam o uso da linguagem a fazem aplicável a problemas comerciais. Criada em 1959, tem passado por grandes e constantes aperfeiçoamentos, inclusive com versões WINDOWS.

Todas as instruções são codificadas em inglês, em vez de códigos complexos. São programas mais extensos, porém mais claros e de mais rápidos entendimento e assimilação, não só da linguagem como dos programas escritos nela.

AS DIVISÕES

Todo programa COBOL consiste, obrigatoriamente, em 4 divisões separadas. Cada divisão é escrita em inglês, para diminuir o esforço e facilitar a compreensão do programa por pessoas alheias ao processamento de dados. Cada uma das 4 divisões tem funções específicas.

1. A IDENTIFICATION DIVISION serve para identificar o programa no computador e também proporciona informações documentais que são de suma importância para pessoas que não entendem nada de processamento e queiram analisar superficialmente o programa.
2. A ENVIRONMENT DIVISION descreve o computador e os periféricos que serão utilizados pelo programa.
3. A DATA DIVISION descreve os arquivos de entrada e saída que serão processados pelo programa, especificando seus formatos. Também define as áreas de trabalho e constantes necessárias para o processamento dos dados.
4. A PROCEDURE DIVISION contém as instruções e o curso lógico e necessário para chegar-se ao resultado final.

As divisões devem sempre aparecer nesta ordem, dentro de um programa.

As divisões podem ser divididas em seções (SECTIONS) e estas em parágrafos. Todas as outras instruções do programa são consideradas declarações COBOL.

REGRAS BÁSICAS

Os nomes de divisões, seções e parágrafos devem ser codificados na margem A (coluna 8). Todas as outras declarações são codificadas na margem B (coluna 12).
Cada declaração termina com um ponto final, que deve ser seguido de um espaço em branco.

Os nomes de divisão e seção devem aparecer na linha sem nenhuma outra entrada. Nomes de parágrafos podem aparecer na mesma linha de uma ou mais declarações, sempre seguidos de ponto e espaço em branco.

Regras para formação de nomes:

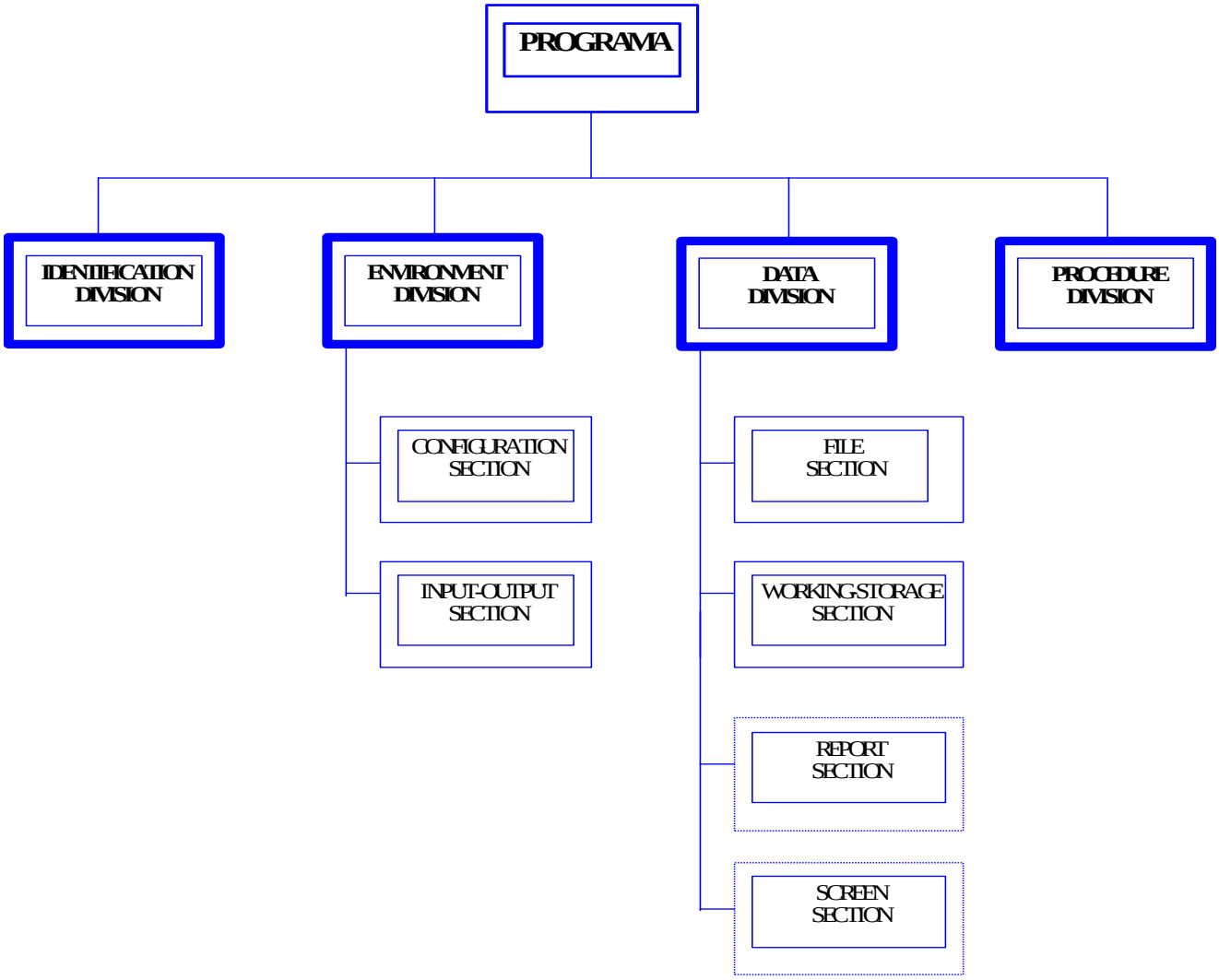
1. Arquivos: de 1 até 30 caracteres;
   nenhum caracter especial;
   nenhum caracter branco no meio;
   pelo menos um caracter alfabético.

2. Dados (registros, campos, etc): de 1 até 30 caracteres;
   não podem começar nem terminar com hífen;
   pelo menos um caracter alfabético.
   não podem ser palavra reservada do COBOL.
   podem conter letras, números ou hífens e mais nenhum caracter especial;

3. Literais Numéricos: máximo de 18 dígitos;
   sinal (“+” ou “-”) à esquerda do número;
   ponto decimal, que não pode ser o último caractere.

4. Literais Não Numéricos: máximo de 120 caracteres, incluindo espaços branco.
   qualquer caractere especial;
   devem estar entre aspas, normalmente simples.

ESTRUTURAS BÁSICAS DA DIVISÕES



Layout: palavras em MAIÚSCULAS - reservadas do COBOL.
palavras em minúsculas - definidas pelo programador.
palavras entre colchetes [ ] - declaração opcional.
palavras entre chaves { } - mutuamente exclusivas.
palavras entre asteriscos * * - comentários da apostila.

```
7  8       12
   IDENTIFICATION DIVISION.
   PROGRAM-ID. nome do programa.
   [AUTHOR. nome do programador.]
   [INSTALLATION. nome da empresa ou local de geração do programa.]
   [DATE-WRITTEN. data em que o programa foi escrito.]
   [DATE-COMPILED. data em que o programa foi compilado.]
   [SECURITY. comentários sobre a segurança do programa e/ou seus arquivos.]
   [REMARKS. comentários adicionais sobre o programa.]
```

7 8 12

ENVIRONMENT DIVISION.

[CONFIGURATION SECTION.]

[SOURCE-COMPUTER. computador a ser utilizado na compilação do programa.]

[OBJECT-COMPUTER. computador onde o programa será executado.]

[SPECIAL NAMES

[CURRENCY SIGN IS literal] obs 1

[DECIMAL-POINT IS COMMA.] obs 2

[PRINTER IS nome externo (mnemônico) para referenciação da impressora.]]

[INPUT-OUTPUT SECTION.]

[FILE-CONTROL

[SELECT nome do arquivo ASSIGN TO {DISK}

{PRINTER}] obs 3

[ORGANIZATION IS {SEQUENTIAL}

{INDEXED}

{RELATIVE}] obs 4

[ACCESS MODE IS {SEQUENTIAL}

{RANDOM}

{DINAMIC}] obs 5

[RECORD KEY IS nome de um campo ou conjunto de campos índice do arquivo]

[FILE STATUS IS nome de uma variável para armazenamento do *status* do arquivo.]]

obs 1.: substitui na cláusula PICture (DATA DIVISION) o sinal corrente "$" pela literal especificada (por exemplo: "R$").

obs 2.: substitui o ponto decimal utilizado na notação americana, pela vírgula utilizada na notação nacional.

obs 3.: DISK: se o arquivo de leitura ou gravação for direcionado para disco;
PRINTER : se for um arquivo a ser impresso.

obs 4 : SEQUENTIAL os registro só podem ser pesquisados um após o outro;

e obs 5.: INDEXED/ RANDOM: os registros são pesquisados sem uma ordem aparente, através de um arquivo de índices relacionado ao arquivo principal.

RELATIVE/ DINAMIC: idem.

7 8 12

DATA DIVISION.

[FILE SECTION : * descreve o conteúdo e a organização interna de todos os arquivos utilizados programa.

uma para cada arquivo descrito na cláusula SELECT da CONFIGURATION SECTI da ENVIRONMENT DIVISION. *

```
   FD  <nome de arquivo>
          LABEL RECORD {OMMITED} * para arquivos de impressão *
                       {STANDARD} * para arquivos em disco *
          VALUE OF FILE-ID  "c:\nome-externo.extensão"  * máximo 8 caracteres no nome-externo *
          [BLOCK CONTAINS  <número-inteiro>  RECORD]
          [RECORD  CONTAINS  <número-inteiro> CHARACTERS] * soma do tamanho de todos
                                                           campos do registro *
          [DATA RECORD  <nome-registro>.] ]
   01     <nome-registro>.  * item de grupo *
          03 <nome-campo1>                    PIC X(<número-inteiro>).  * item elementar *
          03 <nome-campo2>.  * item de grupo *
             05 <nome-campo21>                PIC 9(<número-inteiro>).  * item elementar *
             05 <nome-campo22>                PIC 999999.  * item elementar *
          03 <nome-campo3>                    PIC 9(<número-inteiro>)V(<número-inteiro>)
          03 <nome-campo4>                    PIC 9999V99
          03 <nome-campo5>                    PIC XXXXXXXX.
          03 <nome-campo6>                    PIC A(<número-inteiro>).
          03 <nome-campo6>                    PIC AAAAAAAAA.
          03 FILLER                           PIC X(10).
*
*         * asterisco na coluna 7 marca comentários  do programador, desconsiderados pelo programa *
*
   [SD    <nome-arquivo>  * de "sort" (classificação) *
          VALUE OF FILE-ID  "c:nome-externo.extensão".
   01     <nome-registro>.
          03 <nome-campo1>                    PIC X(<número-inteiro>).
          03 <nome-campo2>.
             05 <nome-campo21>                PIC 9(<número-inteiro>).
             05 <nome-campo22>                PIC 999999.
          03 <nome-campo3>                    PIC 9(<número-inteiro>)V(<número-inteiro>).
          03 FILLER                           PIC X(5).
          03 <nome-campo4>                    PIC 9999V99.
          03 <nome-campo5>                    PIC XXXXXXXX.
          03 <nome-campo6>                    PIC A(<número-inteiro>).
          03 <nome-campo6>                    PIC AAAAAAAAA.
7  8    12
*
          WORKING-STORAGE                    utilizada para descrever registros e campos auxiliares
          SECTION.                           trabalho, tais como: totalizadores, contadores, flags etc. A
                                             pode ser definido também a formatação de relatóri
                                             cabeçalhos e linhas detalhes.
   77     <campo-aux>                        PIC X(9)       [VALUE]  "COBOL".
   01     <campo-cont>.                      PIC 999        [VALUE]  ZEROS, ZEROES ou 0
   01     <campo-total>.                     PIC 9999v99    [VALUE]  ZEROS, ZEROES ou 0
   77     <campo-flag>                       PIC 9          [VALUE]  0.
   01     <reg-aux>.
          03 <campo1>                        PIC X(40).
```

```
        03 <campo2>                          PIC 9.
           88 .<campo21>                     VALUE <valor> * pode ser assumido pelo campo *.
           88  <campo22>                    VALUES <valor1>, <valor2>, <volorn>.
        03 <campo3>                          PIC 9(12).
        03      <campo4>          REDEFINES
        <campo3>.
           05 <campo41>                      PIC X(02).
           05 <campo42>                      PIC 9(08).
           05 <campo43>                      PIC X(02).
*
    01  <reg-cabecalho1>.
        03 <campo1>                          PIC X(<número inteiro>)  VALUE "<constante>".
        03 FILLER                            PIC X(5)    VALUE SPACES.
        03 <campo2>                          PIC 99/99/99B(02).
        03 <campo3>                          PIC ZZZ9.
*
    01  <reg-detal>.
        03 FILLER                            PIC X(5)    VALUE SPACES.
        03 <campo1>                          PIC X(40).
        03 FILLER                            PIC X(5)    VALUE SPACES.
        03 <campo2>                          PIC ZZ999B(5).
        03 <campo3>                          PIC ZZ9.99B(5).
        03 <campo4>                          PIC.ZZ9.99B(10).
```

ESTRUTURA DE COMANDOS DE UM PROGRAMA

PROCEDURE DIVISION

- UTILIZAÇÃO DE COMANDOS E PALAVRAS RESERVADAS.

O uso das palavras do COBOL possuem um significado especial para o compilador COBOL. Tais palavras não podem ser usadas como nome de dados ou nome de parágrafo. Algumas podem não ser reservadas para computadores específicos. É aconselhável, entretanto, não utilizar qualquer das palavras da lista na formação de nomes criados pelo programador.
Estas palavras geralmente se apresentam no singular, no entanto, o plural de qualquer destas palavras não deve ser usada para definir nomes de dados ou nome de parágrafos.
Ex.: ACCEPT, ACTUAL, AND, BLANK, BLOCK, COBOL , COMMA, etc...

Os comandos podem ser:

CONDICIONAIS: especifica o valor de uma condição (verdadeiro/falso).
Ex.: READ <arquivo-entrada> **AT END** MOVE 1 TO FIM-ARQ.
IMPERATIVOS: indica uma ação incondicional a ser tomada pelo programa.
Ex.: READ <arquivo-entrada>

- CATEGORIA DE COMANDOS

A - COMANDOS ARITMÉTICOS: ADD, SUBTRACT, MULTIPLY, DIVIDE, COMPUTE.
B - COMANDOS DE ENTRADA/SAÍDA: OPEN, CLOSE, READ, WRITE, ACCEPT, DISPLAY.
C - COMANDOS DE MANIPULAÇÃO DE DADOS: MOVE, SET.
D- COMANDO DE DESVIOS : PERFORM
E - COMANDOS DE PARADA: STOP
F - COMANDOS DE CONDIÇÃO : IF THEN ELSE

A - COMANDOS ARITMÉTICOS

CLÁUSULA ADD - soma

FORMATO*: ADD <nome-dado1> TO <nome dado2>*

ADD <nome-dado1>, <nome-dado2> GIVING <nome dado3>

EX.: ADD A TO B → A + B = B

ADD A, B GIVING C ⟶ A + B = C

CLÁUSULA SUBTRACT - subtração

FORMATO: *SUBTRACT <nome-dado1> FROM <nome-dado2>*
SUBTRACT <nome-dado1>, <nome-dado2> FROM <nome-dado3>
SUBTRACT <nome-dado1> FROM <nome-dado2> GIVING <nome-dado3>

EX.: SUBTRACT A, B FROM ⟶ C - (A + B)
SUBTRACT A FROM B GIVING C ⟶ C = B - A

CLÁUSULA MULTIPLY - multiplicação

FORMATO: *MULTIPLY <nome-dado1> BY <nome-dado2>*
MULTIPLY <nome-dado1> BY <nome-dado2> GIVING <nome-dado3>

EX.: MULTIPLY A BY B ⟶ A = B x A
MULTIPLY A BY B GIVING C ⟶ C = B x A

CLÁUSULA DIVIDE - divisão

FORMATO: *DIVIDE <nome-dado1> BY <nome-dado2>*
DIVIDE <nome-dado1> BY <nome-dado2> GIVING <nome-dado3>

EX.: DIVIDE A INTO B ⟶ A = B / A
DIVIDE A INTO B IVING C ⟶ C = B / A

CLÁUSULA COMPUTE - cálculo

Utiliza os símbolos aritméticos para fazer as representações de fórmulas matemáticas.

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

EX.: COMPUTE A = B - (C + D * * 2) ⟶ A = B - (C + D^2)
COMPUTE J = 10 / 2 + 8 ⟶ J = 10 / 2 + 8

OBS.: os parenteses '( )' determina a ordem de prioridade de execução na cláusula.

B - COMANDOS DE ENTRADA / SAÍDA

CLÁUSULA OPEN - abrir

FORMATO: *OPEN [INPUT <nome-arquivo>] - permite leitura do arquivo*
[OUTPUT <nome-arquivo>] - permite gravação de registros no arquivo
[EXTEND <nome-arquivo>] - permite adicionar registros em arquivos seqüênciais
[I-O <nome-arquivo>] - permite leitura/gravação no arquivo

EX.: OPEN INPUT CADASTRO OUTPUT RELATO.

CLÁUSULA CLOSE - fechar

FORMATO: *CLOSE <nome-arquivo1>*
<nome-arquivo2> - fará ofechamento dos arquivos abertos anteriormente pelo comando OPEN.

EX.: CLOSE CADASTRO RELATO.

CLÁUSULA READ - ler

FORMATO: *READ <nome-arquivo1>*
[INTO] <nome-arqaux> faz com que o registro lido seja transferido para uma área definida na WS Section.
[AT END] - detecta o fim de um arquivo sequencial.
[INVALID KEY] - usado para arquivo de acesso indexado ou randômico para validar o campo chave do arquivo

EX.: READ CADASTRO.
READ MOV INTO MOV-WS.
READ CADASTRO AT END MOVE 1 TO FIM-CAD.
READ FUNC INVALID KEY PERFORM ROTERRO.

CLÁUSULA WRITE - gravar

FORMATO: WRITE <nome-reg1> - registro do arquivo a ser gravado.
[FROM] <nome-reg2> - de onde será gravado.
[AFTER] <número de linhas> ou <minemônico> - para arquivos associados a impressora.

[BEFORE] - ‘ ‘ ‘ ‘.

[INVALID KEY] - *usado para arquivo de acesso indexado ou randômico para validar o campo chave do arquivo.*

EX.: WRITE REG-CAD INVALID KEY PERFORM ROTERRO.
WRITE REG-REL FROM CABEC01 AFTER SALTO.
WRITE REG-REL FROM CABEC02 AFTER 2.

CLÁUSULA ACCEPT - aceitar.

Obtém dados de fora do programa.

FORMATO: *ACCEPT <dados> FROM [DATE/TIME/DAY/SCAPE KEY]*.
ACCEPT (L,C) <dados>

EX.: ACCEPT DATAW FROM DATE.
ACCEPT (L,C) RESP.

CLÁUSULA DISPLAY - aceitar.

Obtém dados de fora do programa.

FORMATO: *DISPLAY (L,C) [<literal>] <dados>*.

EX.: DISPLAY (10,50) ‘DATA HOJE’ DATAW.
DISPLAY (12,50) ‘NOME : ‘ NOME
.

C - COMANDOS DE MANIPULAÇÃO

CLÁUSULA SEARCH - pesquisar.

Pesquisa uma tabela para localizar um elemento que satisfaça determinada condição.

FORMATO: SEARCH <nome-tabela> [VARYNG <index1> <index2>]
[AT END <sentença>]
[WHEN <condição sentença>].

EX.: SEARCH CAMPO AT END PERFORM ROT1
WHEN CODIGO(X) = COD-LIDO

DISPLAY NOME(X) UPON CONSOLE.

CLÁUSULA MOVE - mover

Transfere dados de uma área para outra área na memória principal.

FORMATO: *MOVE [CORRESPONDING] <nome-dado1> TO <nome-dado2 > ..... <nome-dadon>*

EX.: MOVE CPF TO CPFW.
MOVE CORR DATA-HOJE TO DATA-CAB.

CLÁUSULA SET - atribuir.

Transfere dados de uma área para outra área na memória principal.

FORMATO: *SET <nome-dado1> TO <valor > - atribui um valor.*
SET <indice1> UP BY <numero-inteiro> - acréscimo do indice por um valor.
SET <indice1> DOWN BY <numero-inteiro> - decréscimo do indice por um valor.

EX.: SET IND TO 1.
SET IND 2 TO M.
SET IND UP BY 2.
SET IND2 DOWN BY 3.

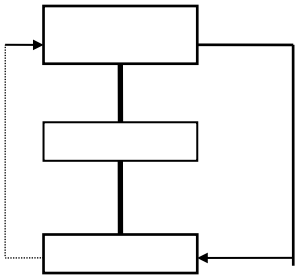
D - COMANDO DE DESVIOS

CLÁUSULA PERFORM - executar

Executa os comandos de um parágrafo. Quando todas as instruções são executadas o controle é transferido para instrução que segue imediatamentes cláusula PERFORM.

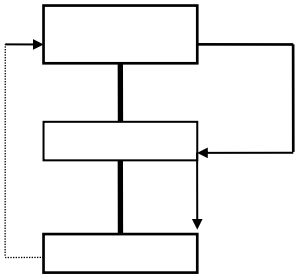
FORMATO :*PERFORM <nome-parágrafo>*.

EX.: INICIO.
PERFORM LER2.



: *PERFORM <nome-parágrafo1> THRU <nome-parágrafo2>*.

EX.: INICIO
PERFORM LER1 THRU LER2.



: PERFORM <nome-parágrafo> <nome-dado> TIMES.
<número-inteiro>

EX.: PERFORM PROCESSA 20 TIMES.

: *PERFORM <nome-parágrafo> UNTIL <condição>*.

EX.: PERFORM PROCESSAMENTO UNTIL FIM-ARQ = 'S'.

CLÁUSULA EXIT - saída

É o ponto comun de finalização para uma série de procedimentos.
Deve ser precedida por um nome de parágrafo.

FORMATO 1: PERFORM <nome-parágrafo>.

EX.: PERFORM A THRU B.

A.
ADD...
MOVE...
B. **EXIT**.

E - COMANDO DE PARADA

CLÁUSULA STOP - executar

É usado para parar o processamento temporariamente ou definitivamente.

FORMATO : STOP [RUN] - parada definitiva.
[<literal>] - parada temporária.

```
EX.:  ROT-PROC.
          MOVE...
          PERFORM ....
          STOP  'PARADA' .

      ROT-FIM
          CLOSE ......
          STOP RUN.
```

F - COMANDO DE CONDIÇÃO

CLÁUSULA IF THEN ELSE - se / então / senão

É qualquer sentença que executa uma ou mais de uma operação dependendo da ocorrência de uma ou mais de uma condição.

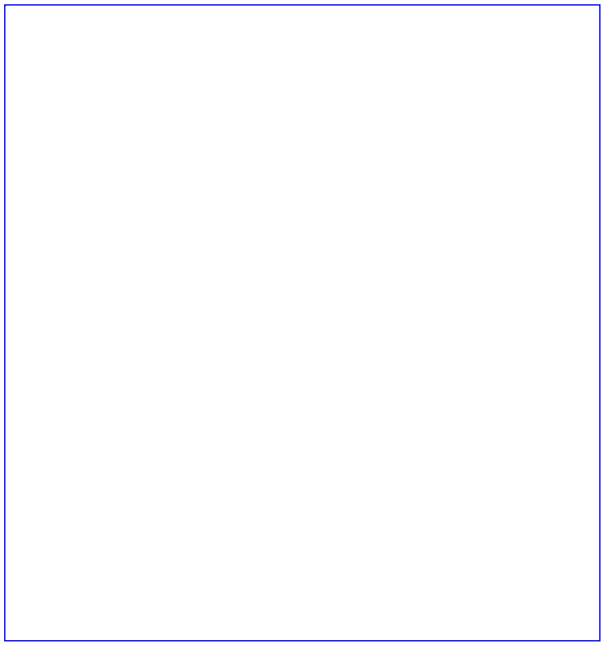
FORMATO : *IF <condição1>...<condiçãon>*
THEN <sentença1>...<sentençaN> - se condição verdadeira.
ELSE <sentença1>...<sentençaN>. - se condição falsa.

```
EX.:   IF A > B
          THEN ADD A TO B
                  ADD A TO C
           ELSE  ADD B TO C.
```

OBS.: TESTES DE CONDIÇÃO >, < E =. Estes sinais equivalem às seguintes palavras reservadas:

> → GREATER THAN
< → LESS THAN
= → EQUAL TO

PROCEDURE DIVISION
ESQUEMA BÁSICO



```
7       8       12
        PROCEDURE DIVISION.
        000-ROTINA BÁSICA.
              PERFORM  100-INICIO-PROC.
              PERFORM  500-PROCESSAMENTO UNTIL <campo-flag> =  1.
              PERFORM  900-FINAL-PROC.
              STOP RUN.
     *
        100-INICIO-PROC.
             OPEN   INPUT     CADNOTA
                    OUTPUT  RELATO.
              READ  CADNOTA  AT END MOVE 1 TO FIM-CAD.
              MOVE ZEROS   TO  <campo-aux1> <campo-aux2> .....<campo-auxn)
              MOVE SPACES TO  <campo-aux1> <campo-aux2> .....<campo-auxn).
              ACCEPT DATAH FROM DATE.
     *
        500-PROCESSAMENTO.
             MOVE NOME IN REG-CAD TO NOME IN REG-SAI
             MOVE CPF-CAD TO CPF-DET.
             PERFORM 510-CALC-GRAU.
             PERFORM 520-VERIF-SITUACAO.
             READ  CADNOTA AT END MOVE 1 TO FIM-CAD.

        510-CALC-GRAU.
```

```
      COMPUTE NP  ROUNDED = ( VE + 2 * VC) / 3.
520-VERIF-SITUACAO.
       IF NP > 4,9
            MOVE ‘APROV’ TO SITUACAO
       ELSE
            MOVE ‘RECUP’ TO SIOTUACAO.
900-FINAL-PROC.
     DISPLAY (10,30) ‘FINAL PROCESSAMENTO’.
     CLOSE CADNOTA RELATO.
999-FIM-PGM.
```