

FACULDADE DE TECNOLOGIA DE SÃO PAULO
- FATEC · SP -

ASSUNTO:

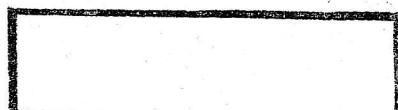
COBOL
LTP-II E III

DISCIPLINA:

DEPTO: PROCESSAMENTO DE DADOS

PROFESSOR:

ANO



Nº DA APOSTILA

SUMÁRIO	Pági
CAPÍTULO 1:	
1. INTRODUÇÃO A LINGUAGEM COBOL	05
1.1. HISTÓRICO	
1.2. FOLHA DE CODIFICAÇÃO COBOL	07
CAPÍTULO 2:	
2. IDENTIFICATION DIVISION	08
CAPÍTULO 3:	
3. ENVIRONMENT DIVISION	09
3.1. CONFIGURATION SECTION	
3.1.1. SOURCE-COMPUTER	
3.1.3. OBJECT-COMPUTER	
3.1.3. SPECIAL-NAMES	10
3.1.3.1. DECIMAL-POINT IS COMMA	
3.2. INPUT-OUTPUT SECTION	12
3.2.1. FILE-CONTROL	
3.2.1.1. NÚMERO DO SISTEMA	13
3.2.1.2. CLASSIFICAÇÃO	
3.2.1.3. NÚMERO DO PERIFÉRICO	
3.2.1.4. MODO DE ACESSO	15
CAPÍTULO 4:	
4. DATA DIVISION	17
4.1. FUNÇÕES DA DATA DIVISION	
4.2. ESTRUTURA GERAL DA DATA DIVISION	
4.3. FILE SECTION	18
4.3.1. BLOCK CONTAINS	
4.3.2. RECORDS CONTAINS	19
4.3.3. LABEL RECORDS	
4.3.4. DATA RECORDS	20
4.3.5. VALUE OF ...	
4.4. RECORD DESCRIPTION	21
4.4.1. PICTURE DE DADOS	
4.4.2. OUTRAS CLAUSULAS DE DEFINIÇÃO DE DADOS	
4.5. USAGE	
4.5.1. USAGE IS DISPLAY	22
4.5.2. USAGE IS COMPUTACIONAL	
4.5.3. USAGE IS COMP-3	
4.6. CLÁUSULA OCCURS	
4.7. CLÁUSULA REDEFINES	

CAPÍTULO 5:

5. WORKING-STORAGE SECTION	25
5.1. ÍTEIS INDEPENDENTES DA WORKING-STORAGE SECTION	26
5.2. REGRAS PARA CONTINUAÇÃO DE LITERAIS	27
5.3. NOMES CONDICIONAIS	32
5.4. PICTURE DE EDIÇÃO	35

CAPÍTULO 6:

6. SCREENS SECTION	36
6.1. LINE E COLUMN	
6.2. TO / FROM / USING	

CAPÍTULO 7:

7. PROCEDURE DIVISION	38
7.1. SUBDIVISÕES DA PROCEDURE	
7.2. OPEN	39
7.3. CLOSE	
7.4. READ	40
7.5. WRITE	41
7.6. IF-ELSE	42
7.7. RELATION CONDITION	44
7.8. CLASS CONDITION	
7.9. SIGNAL CONDITION	
7.10. COMPOUND CONDITION	
7.11. MOVE	45
7.12. GO TO	47
7.13. PERFORM	48
7.14. STOP RUN	48
7.15. EXPRESSÕES ARITMÉTICAS	50
7.15.1. ADD	
7.15.2. SUBTRACT	
7.15.3. MULTIPLY	
7.15.4. DIVIDE	
7.15.5. COMPUTE	52
7.16. DISPLAY	53
7.16.1 DISPLAY EM TELAS	54
7.17. ACCEPT	
7.17.1 ACCEPT EM TELAS	55
7.18. EXIT	
7.19. CURSOR	
7.20. FILE STATUS	
7.21. TECLAS DE FUNÇÃO	57

CAPÍTULO 8:

8. MANIPULAÇÃO DE TABELAS	60
8.1. SUBSCRITOR	
8.2. SET	62

CAPÍTULO 9:

9. ARQUIVOS INDEXADOS	63
9.1. ARQUIVO INDEXADO	
9.2. MODO INDEXADO	
9.3. DEFINIÇÃO DE UM ARQUIVO INDEXADO	
9.4. COMANDOS DE ARQUIVOS INDEXADOS	64
9.4.1. OPEN	
9.4.2. READ	
9.4.3. WRITE	
9.4.4. REWRITE	
9.4.5. DELETE	
9.4.6. START	

CAPÍTULO 10:

10. UTILIZANDO O 'MICROSOFT' VERSAO 4.5	65
10.1. USANDO O "SOFTWARE" - PWB	
10.1.1. EDITANDO TEXTO	
10.1.2. COMPILANDO PROGRAMA	
10.1.3. LINKANDO PROGRAMA	
10.1.4. EXECUTANDO PROGRAMA	
10.1.5. RECONSTRUINDO/REORGANIZANDO ÍNDICES	66
10.2. "SCREENS"	
10.2.1. DEFININDO UMA TELA	
10.2.1.1. DESENHANDO A MOLDURA	67
10.2.1.2. DEFININDO CAMPOS	
10.2.1.3. PINTANDO CAMPOS	
10.2.1.4. DEFININDO GRUPOS (OCORRÊNCIAS)	68
10.2.1.5. MOVENDO/COPIANDO CAMPOS	
10.2.1.6. ALTERANDO A ORDEM DO ACCEPT	
10.2.1.7. GERANDO O COBOL	
10.2.1.8. SALVANDO A TELA	
10.2.1.9. DELETANDO ATRIBUTOS DE CAMPOS	69
10.2.1.10. DELETANDO ATRIBUTOS DE GRUPOS	
10.2.1.11. DELETANDO LINHAS	
10.2.1.12. INSERINDO LINHAS	
10.2.1.13. SAINDO DA "SCREENS"	
10.2.2. ARQUIVOS GERADOS	
10.2.3. ATRIBUTOS DE CAMPOS	

CAPÍTULO 11:

11. EXEMPLOS DE PROGRAMAS	70
11.1. PROGRAMA 1 LÊ-GRAVA	
11.2. PROGRAMA 2 LÊ-IMPRIME	72

11.3. PROGRAMA 3 LÊ-PROCESSA-GRAVA

75

CAPÍTULO 1

1. INTRODUÇÃO A LINGUAGEM COBOL

1.1. HISTÓRICO

O COBOL (Common Business Oriented Language) é uma linguagem de programação de alto nível, projetada para aplicações comerciais.

O objetivo do COBOL é oferecer uma linguagem de programação que atenda aos seguintes requisitos:

- Interdependência de tipo de equipamento
- Facilidade de aprendizado
- Boa documentação dos programas

Existem quatro grandes divisões em um programa COBOL:

IDENTIFICATION DIVISION	
ENVIRONMENT DIVISION	
DATA DIVISION	
PROCEDURE DIVISION	

EXEMPLO DE PROGRAMA EM COBOL:

IDENTIFICATION DIVISION.
PROGRAM-ID. EX01.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. UNISYS.
OBJECT-COMPUTER. UNISYS.
INPUT-OUTPUT SECTION.
FILE-CONTROL.

SELECT ARQ-ENT ASSIGN TO TAPE.
SELECT RELAT ASSIGN TO PRINTER.

DATA DIVISION.

FILE SECTION.

FD ARQ-ENT

LABEL RECORD ARE STANDARD
BLOCK CONTAINS 1260 CHARACTERS
VALUE OF TITLE IS "CADASTRO".

01 REG-ENT.

05 CODIGO PIC 9(05).
05 NOME PIC X(30).
05 FILLER PIC X(49).

FD RELAT

LABEL RECORD ARE OMITTED.

01 REG-REL PIC X(132).

WORKING-STORAGE SECTION.

77 CH-FIM PIC X(03) VALUE "NAO".
77 CT-LIN PIC 9(02) VALUE 31.

01 DETALHE.

02 FILLER PIC X(32) VALUE SPACES.
02 CODIGO-REL PIC X(05).
02 FILLER PIC X(32) VALUE SPACES.
02 NOME-REL PIC X(30).
02 FILLER PIC X(33) VALUE SPACES.

01 CAB-01.

02 FILLER PIC X(33) VALUE SPACES.
02 FILLER PIC X(65) VALUE "R E L A C A O D E N O M
- "È S D A E M P R E S A D T C".
02 FILLER PIC X(34) VALUE SPACES.

PROCEDURE DIVISION.

00-EX01.

PERFORM 10-INICIO.
PERFORM 30-PROCESSO UNTIL CH-FIM EQUAL "SIM".
PERFORM 60-FINALIZA.
STOP RUN.

10-INICIO.

OPEN INPUT ARQ-ENT
OUTPUT RELAT.
PERFORM 20-LEITURA.

20-LEITURA.

READ ARQ-ENT
AT END MOVE "SIM" TO CH-FIM.

30-PROCESSO.

PERFORM 40-MOVE-IMPRIME.
PERFORM 20-LEITURA.

40-MOVE-IMPRIME.

IF CT-LIN GREATER THAN 30
PERFORM 50-CABECALHO.
MOVE CODIGO TO CODIGO-REL.
MOVE NOME TO NOME-REL.
WRITE REG-REL FROM DETALHE AFTER ADVANCING 1 LINE.
ADD 1 TO CT-LIN.

50-CABECALHO.

WRITE REG-REL FROM CAB-01 AFTER ADVANCING PAGE,
MOVE ZEROES TO CT-LIN.

60-FINALIZA.

CLOSE ARQ-ENT
RELAT.

1.2. FOLHA DE CODIFICAÇÃO COBOL

Os programadores de COBOL normalmente escrevem seus programas em formulários especiais denominados Formulários de Codificação que tem a descrição abaixo.

DESCRÍÇÃO DA FOLHA DE PROGRAMAÇÃO

Basicamente temos 72 colunas disponíveis para a escrita dos comandos COBOL.

As colunas compreendidas entre 73 a 80 são destinados à identificação do programa. As demais colunas possuem as seguintes finalidades:

COLUNAS de 1 a 6

Usadas para numerar as linhas de um programa. A numeração é em ordem crescente. Opcionalmente podem deixar de ser preenchidas.

COLUNA 7

É usada para um "hífen", que significa continuação de literais não numéricos, e o símbolo "*" usado para comentários.

Exemplo 1:

```
....  
02 FILLER PIC X(63) VA."RELATORIO DA CO  
- "MPANHIA TEND-TUDO".  
....
```

COLUNA 8 a 72

Usada para todas as entradas do programa. Note, no entanto que a coluna 8 tem a letra A e a coluna 12 tem a letra B. Estas são as margens. Algumas entradas devem começar na margem A, outras na margem B.

Os seguintes ítems devem ser iniciados na margem A:

- Cabeçalho de entrada nas descrições de arquivos;
- Cabeçalhos de divisão;
- Cabeçalhos de seção;
- Cabeçalhos de parágrafo.

CAPÍTULO 2

2. IDENTIFICATION DIVISION

É a menor, mais simples e menos importante divisão de um programa COBOL. É usada para identificar o programa.

Ela não possui seções, ao invés disso consiste de parágrafos.

FORMATO

A
IDENTIFICATION DIVISION.

PROGRAM-ID. Nome do programa.

AUTHOR. Nome do autor.

INSTALLATION. Local de uso.

DATE-WRITTEN. Data em que foi escrito o programa.

DATE-COMPILED. Data em que foi compilado o programa.

SECURITY. Comentário sobre a segurança.

REMARKS. Comentários gerais sobre o programa.

Por ser o nome de uma divisão, a IDENTIFICATION DIVISION é codificada na margem A. Os nomes dos parágrafos também são codificados na margem A, e cada um deles é seguido de um ponto final.

A única entrada obrigatória na IDENTIFICATION DIVISION é o PROGRAM-ID. O nome que se segue a PROGRAM-ID deve ser codificado na margem B. Este nome deve conter no máximo 30 caracteres (letras, dígitos e hifens), mas os 8 primeiros são os que realmente identificam o programa ao computador.

EXEMPLO NO UNISYS:

IDENTIFICATION DIVISION.

PROGRAM-ID. IDENT.

AUTHOR. MARCO.

INSTALLATION. FATEC-SP.

DATE-WRITTEN. 08-10-91.

DATE-COMPILED.

SECURITY. ALTERACAO SOMENTE AUTORIZACAO DO AUTOR.

*REMARKS. PROGRAMA MODELO.

OBS: Não é necessário colocar a data no parágrafo DATE-COMPILED, pois o UNISYS colocará a data do sistema operacional.

É preciso colocar o sinal de comentário, asterisco (*), na coluna 7 do parágrafo REMARKS para o compilador.

CAPÍTULO 3

3. ENVIRONMENT DIVISION

As funções da divisão de Equipamento são descrever as características do computador a ser utilizado e definir os arquivos usados no programa, podendo também descrever técnicas e controles especiais para entrada ou saída de dados.

Esta divisão é composta de duas seções:

- Configuração : CONFIGURATION SECTION
- Entrada e de saída: INPUT-OUTPUT SECTION

3.1. CONFIGURATION SECTION

A seção de configuração fornece informações sobre o computador e é dividida em três parágrafos:

SOURCE-COMPUTER
OBJECT-COMPUTER
SPECIAL-NAMES

Tanto a seção de configuração como os parágrafos a ela associados são opcionais; os parágrafos SOURCE-COMPUTER e OBJECT-COMPUTER são tratados como comentários.

3.1.1. SOURCE-COMPUTER

O parágrafo SOURCE-COMPUTER descreve em qual computador o programa-fonte será compilado, e é apenas documentacional, sendo tratado como comentário pelo compilador COBOL. Apresenta o seguinte formato:

SOURCE-COMPUTER. Nome-do-computador.

3.1.2. OBJECT-COMPUTER

O parágrafo OBJECT-COMPUTER descreve o computador no qual o programa objeto será executado. Apresenta o seguinte formato:

OBJECT-COMPUTER. Nome-do-computador.

3.1.3. SPECIAL-NAMES

Neste parágrafo podem-se definir técnicas especiais para a programação, como, por exemplo, a alteração da simbologia padronizada pela linguagem. Apresenta o formato:

SPECIAL-NAMES.

[Nome-de-função IS nome-simbólico]
[DECIMAL-POINT IS COMMA].

Os nomes de funções válidos são:

SYSIPT
SYSPUNCH
SYSLST
CONSOLE
C01 a C12 (Controla o canal)
CSP (Suprime espaço)
S01 a S05 (Seleciona escaninho)

Os nomes SYSIPT (leitora de cartões), SYSPUNCH (perfuradora de cartões), SYSLST (impressora) e CONSOLE podem ser associados a nomes-simbólicos (definidos pelo programador) para serem utilizados nos comandos ACCEPT e DISPLAY.

Os nomes C01 e C12 se referem às perfurações na "fita de carro", conhecidas como canais. E por fim, os nomes CSP, S01, a S05 da leitora/perfuradora de cartões.

3.1.3.1. DECIMAL-POINT IS COMMA

Esta cláusula inverte a função do ponto decimal (usado nos países anglo-saxônicos) para vírgula, nos literais numéricos e campos numéricos editados.

Exemplo:

Inglês	Português
1,000.00	1.000,00
\$20,000,000.00	\$20.000.000,00

O uso dessa cláusula permite ao programador definir sua edição com o formato em Português.

Exemplo no UNISYS:

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. UNISYS.
OBJECT-COMPUTER. UNISYS.
SPECIAL-NAMES. DECIMAL-POINT IS COMMA.

3.2. INPUT-OUTPUT SECTION

A seção de entrada e saída define os arquivos utilizados pelo programa COBOL, assim como efetua as criações entre o programa e o equipamento da máquina. Ela está dividida em dois parágrafos:

FILE-CONTROL

I-O CONTROL (Não será dada)

3.2.1. FILE-CONTROL

É neste parágrafo que cada arquivo é selecionado e designado a uma unidade de entrada ou saída. Aqui são usadas as seguintes cláusulas, que devem ser codificadas a partir da margem B:

SELECT nome-do- arquivo ASSIGN TO especificação do periférico

As especificações dos periféricos variam entre os fabricantes. Considere o seguinte formato de uma cláusula SELECT, necessário na maioria dos compiladores ANS.

SELECT nome do arquivo ASSIGN TO número do sistema UR UT número DA do periférico S D I

Exemplos:

FILE-CONTROL.

SELECT CARD-IN ASSIGN TO SYS001-UR-2540R-S.

SELECT TAPE-OUT ASSIGN TO .SYS002-UT-2400-S.

O parágrafo FILE-CONTROL pode parecer desnecessariamente complexo, por enquanto. As entradas são padronizadas em cada instalação. O único termo fornecido pelo programador é o nome do arquivo.

O nome de arquivo dado a cada dispositivo deve concordar com as regras de formação de nomes atribuídos pelo programador.

Nome do arquivo deve ser único; isto quer dizer que não deve haver outro dado com o mesmo nome ou programa.

Para cada periférico usado no programa, a cláusula SELECT deve ser especificada. Se um programa precisa de cartões como entrada e imprime listagens como saída, devem ser especificadas duas cláusulas SELECT. Um nome deve ser dado ao arquivo em cartões e outro ao arquivo de impressão.

Quando for usado o segundo formato, cada dispositivo tem:

1. número de sistema
2. classificação - UR para UNIT RECORD
UT para UTILITY
DA para DIRECT ACCESS
3. número do periférico
4. modo de acesso - S para SEQUENTIAL
D para DIRECT
I para INDEXED.

3.2.1.1. NÚMERO DO SISTEMA

O número do sistema depende da instalação, pode variar em cada centro de processamento e deve, portanto, ser fornecido por cada centro de processamento de dados. Cada unidade física na sala do computador deve ter um único número do sistema. O número de sistema usado é um nome externo.

3.2.1.2. CLASSIFICAÇÃO

As classificações que podem ser usadas são entradas padronizadas. Há três tipos de classificações de periféricos:

- UNIT-RECORD
- UTILITY
- DIRECT-ACCESS

A impressora, perfuradora, e leitora de cartões são periféricos UNIT-RECORD. Isto quer dizer que cada registro associado com qualquer destes registros é de tamanho fixo. Um cartão, por exemplo, é um documento de UNIT-RECORD, já que cada linha do cartão tem sempre o mesmo número de posições impressas. Por isso os arquivos em cartões e listagens sempre têm a classificação UR, de UNIT-RECORD.

A fita não é um documento UNIT-RECORD, uma vez que os registros em fita podem ser de qualquer tamanho. Uma fita é classificada como UT para um periférico UTILITY. Unidades de memória auxiliar, como discos, tambores e células de dados, são classificados como DA para os periféricos DIRECT-ACCESS.

3.2.1.3. NÚMERO DO PERIFÉRICO

O número do periférico é dado pelo fabricante do computador. A IBM, por exemplo, utiliza os seguintes números de dispositivos para as unidades de sistema 360:

TAPE 2400

READER 2540R
PRINTER 1403
PUNCH 254OP

3.2.1.4. MODO DE ACESSO

O modo de acesso sempre será S para cartões, fitas ou arquivos de impressão, porque nestes arquivos o acesso às informações é de modo sequencial.

Note que a cláusula de especificação do dispositivo que segue a ASSIGN TO contém entradas separadas interligadas por hífens. A omissão destes hífens na codificação causará erros no programa.

Vamos examinar os seguintes exemplos usando números de sistemas arbitrários, pois estes dependem da instalação.

Exemplo 1:

Um arquivo em cartões contendo dados sobre uma transação pode ser assim (IBM):

```
SELECT TRANS-FILE ASSIGN TO SYS004-UR-2540-R-S.
```

O nome, TRANS-FILE, é dado pelo programador. O resto da declaração é necessária quando se utiliza a leitora. A leitura é uma unidade UNIT-RECORD com o número 2540R, do sistema SYS004.

Observe que as cláusulas SELECT são codificadas na margem B.

Exemplo no UNISYS de um arquivo em cartões:

```
SELECT TRANS-FILE ASSIGN TO READER.
```

Exemplo 2:

Um arquivo em fita contendo os dados de um empregado pode ser assim (IBM):

```
SELECT EMPLOYEE-FILE ASSING TO SYS007-UT-2400-S.
```

EMPLOYEE-FILE é um nome dado ao arquivo em fita. Todas as palavras posteriores a ASSIGN TO são dadas pela instalação para indicar uma unidade específica de fita.

Observe que a entrada importante na cláusula SELECT é o nome dado ao arquivo. Este nome é usado na DATA DIVISION para reservar a área de entrada ou saída utilizada pelo arquivo. É novamente usado na PROCEDURE DIVISION para ter acesso ao arquivo. As outras entradas na cláusula SELECT dependem do computador. Os números do sistema, no entanto, devem ser obtidos de cada instalação particular.

Exemplo no UNISYS:

FILE-CONTROL.

```
SELECT ARQ-ENT    ASSIGN TO DISK.  
SELECT ARQ-SAI    ASSIGN TO TAPE.  
SELECT ARQ-RELAT  ASSIGN TO PRINTER.
```

Especifica-se os arquivos ARQ-ENT, ARQ-SAI, ARQ-RELAT respectivamente para disco, fita, impressora.

Exemplo da ENVIRONMENT DIVISION:

```
ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
SOURCE-COMPUTER. UNISYS.  
OBJECT-COMPUTER.UNISYS.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.
```

```
SELECT ARQ-ANT ASSIGN TO DISK.  
SELECT ARQ-NOVO ASSIGN TO DISK.
```

Na SELECT é determinada a organização e o modo de acesso do arquivo. A organização pode ser sequencial, indexada ou relativa e o modo de acesso pode ser sequencial ou direto (randômico), se a organização e o acesso não forem especificada admite-se sequencial para ambos. Neste caso, os registros serão armazenados no dispositivo designado na ordem em que forem processados, um após o outro (organização sequencial) e serão acessados também na ordem em que foram armazenados.

CAPÍTULO 4

4. DATA DIVISION

É a divisão do COBOL onde codificamos os comandos de definição e especificação dos arquivos de dados, registros de dados e áreas de trabalho.

4.1. FUNÇÕES DA DATA DIVISION

- Definir nome, tamanho e tipo de todos arquivos usados pelo programa.
- Definir nome, tamanho, tipo e demais características dos locais de memória onde cada registro será lido ou formatado (nível 01).
- Definir o nome, tipo, tamanho e demais características dos campos e sub-campos de cada registro (níveis 02 a 49).

Os itens acima são especificados na FILE SECTION:

- Definir nome, tipo, tamanho e demais características das áreas de trabalho utilizados pelo programa (WORKING-STORAGE SECTION).

4.2. ESTRUTURA GERAL DA DATA DIVISION

DATA DIVISION.

FILE SECTION.

FD (descrição do ARQUIVO1)
01 (descrição dos registros do ARQUIVO1)

..

..

..

FD (descrição do arquivo N)
01 (descrição dos registros do arquivo N)

Informações necessárias para especificar um arquivo, utilizando COBOL.

As informações são dadas em três etapas:

- Na ENVIRONMENT DIVISION (cláusula SELECT)
- Na DATA DIVISION (cláusula FD)
- Na DATA DIVISION (nível 01 Descrição do registro)

Cada arquivo definido através da cláusula SELECT deve ter sua respectiva cláusula FD.

4.3. FILE SECTION

Função:

Descrever aspectos gerais dos arquivos.

- Nome
- Quantidade registros lógicos por bloco
- Tipos de identificação
- Nome dos registros
- Quantidade de caracteres por registro lógico
- Modo de gravação (Fixo, variável)
- Quantidade de registros por arquivo

FILE SECTION

```
FD nome-do-arquivo
[BLOCK CONTAINS [inteiro-1 TO] inteiro-2 {CHARACTERS}]
[                                (RECORDS )]
{ RECORDS CONTAINS [inteiro-3 TO] inteiro-4 CHARACTERS]
  LABEL {RECORDS ARE STANDARD}
        (RECORDS IS OMITTED }

  [DATA RECORD IS nome-reg1 {nome-reg2}]
  [      RECORD ARE

  VALUE OF nome-dado1 is {nome-dado2}
        (LITERAL )
```

4.3.1. BLOCK CONTAINS

Para um melhor aproveitamento do espaço em disco e para aumentar a velocidade de processamento dos arquivos, costuma-se blocar os registros dos arquivos contidos em discos ou fitas. Ao blocar, faz-se uma distinção entre registro lógico, processado um por vez pelo programa e um registro físico, isto é, a unidade de informação transferida para um dispositivo de entrada/saída.

Existem três possibilidades:

1. Arquivo não blocado

Neste caso a cláusula BLOCK CONTAINS é omitida do parágrafo FD.

2. Blocos de comprimento fixo

Utilize:

BLOCK CONTAINS inteiro-2 CHARACTERS ou
BLOCK CONTAINS inteiro-2 RECORDS

3. Blocos de comprimento variável

Ocorre quando os registros lógicos de um arquivo são de tamanho diferentes. Utilize:

BLOCK CONTAINS inteiro-1 TO inteiro-2 CHARACTERS
ou RECORDS CONTAINS inteiro-1 TO inteiro-2 CHARACTERS

No UNISYS utilize:

BLOCK CONTAINS 1260 CHARACTERS

Neste caso, o bloco tem 1260 caracteres.

Não se esqueça que a blocagem somente é aplicável para discos, fitas ou similares. Para arquivo em impressora a blocagem deve ser omitida.

4.3.2. RECORDS CONTAINS

É uma cláusula opcional porque o compilador, a partir do nível-01 da descrição do registro, pode determinar seu comprimento. A sua utilização é de interesse para uma boa documentação e porque alguns compiladores emitem uma mensagem de erro quando a soma dos campos não confere com o valor em RECORDS CONTAINS.

No UNISYS coloque:

RECORD CONTAINS 84 CHARACTERS

4.3.3. LABEL RECORDS

São registros que opcionalmente são criados no início e no final do arquivos em disco e fita. Estes registros não são manuseados pelo programa e contêm informações de identificação do arquivo.

OPÇÃO OMITTED - Especifica que o arquivo não contém LABEL RECORDS. Deve ser especificados para arquivos em cartão e relatórios.

OPÇÃO STANDARD - Especifica que o arquivo tem LABEL RECORDS, no formato padrão.

4.3.4. DATA RECORDS

Especifica o nome dos registros do arquivo.
Não é obrigatória a sua codificação. Tem efeito somente documentacional.

4.3.5. VALUE OF nome-dado1 is nome-dado2

Especifica o nome interno do programa,

No UNISYS utilize:

VALUE OF TITLE IS "nome-do-arquivo".

No MICROSOFT 4.5 utilize:

VALUE OF FILE-ID IS "nome-do-arquivo".

Exemplo da utilização das cláusulas no UNISYS:

FD DISK-FILE

LABEL RECORDS ARE STANDARD
BLOCK CONTAINS 1260 CHARACTERS.
RECORD CONTAINS 84 CHARACTERS
DATA RECORD IS REG-REL
VALUE OF TITLE IS "CADASTRO".

4.4. RECORD DESCRIPTION

Depois de definirmos um arquivo por uma FD, seguem-se as entradas RECORD DESCRIPTION para cada registro do arquivo. Ela vai indicar que itens aparecem no registro, a ordem em que eles aparecem e como esses itens são relacionados entre si. Um nome de registro é codificado no nível 01. Qualquer campo de dado de um registro é codificado em nível subordinado a 01 (Podem ser de 02 a 49).

Exemplo no UNISYS:

NOME	DEPTO	SALÁRIO	FILLER
DIV	SEC		
X(30)	9(02)	9(03) 9(09)V99	X(38)

01 REG-EMPREGADO.

05 NOME PIC X(30).
05 DEPTO.
10 DIVISÃO PIC 9(02).
10 SEÇÃO PIC 9(03).
05 SALÁRIO PIC 9(09)V99.
05 FILLER PIC X(38).

Item elementar - Item que não tem subdivisão.
(nome, divisão, seção e salário)

Item de grupo - Item que sofre subdivisões (depto)

OBS: Se a entrada RECORDS CONTAINS 84 CHARACTERS for colocada é preciso completar os 84 caracteres na descrição do registro (no UNISYS)

4.4.2. PICTURE DE DADOS

9 - REPRESENTA CARACTER NUMERICO
A - REPRESENTA CARACTER ALFABETICO
X - REPRESENTA CARACTER ALFANUMERICO
S - INDICA OCORRENCIA DE SINAL (POSITIVO/NEGATIVO)
V - INDICA POSICAO DO PONTO IMPLICITO
P - REPRESENTA ESCALA DO VALOR NUMERICO

REPETICAO DO A, X, E 9.:

A(10) = AAAAAAAA

X(5) = XXXXX

9(3) = 999

4.4.3 OUTRAS CLAUSULAS PARA DEFINIR DADOS:

FILLER - Define campo sem atribuir nome.

JUSTIFIED RIGHT/LEFT - Justifica o valor do dado a direita ou a esquerda do campo.

4.5. USAGE (VÁLIDO PARA EQUIPAMENTO IBM)

Especifica a representação interna que deve ser utilizada para um item de dado. É muitas vezes omitida porque a USAGE DISPLAY, o default, é aquela frequentemente mais apropriada.

4.5.1. USAGE IS DISPLAY

O item de dado é representado no código de caracteres do sistema de computador (EBCDIC ou ASCII), em que cada caracter alfanumérico é representado por um código binário que requer um byte de memória.

4.5.2. USAGE IS COMPUTACIONAL (COMP)

USAGE do COBOL para ítems numéricos a serem utilizados em cálculos.

A tabela abaixo fornece a quantidade de bytes ocupados por número de dígitos na PICTURE.

n.o de dígito na PICTURE	OCUPAÇÃO NA MEMÓRIA
1 a 4	2 bytes
5 a 9	4 bytes
10 a 18	8 bytes

4.5.3. USAGE IS COMP-3

Cada item é armazenado na memória no formato decimal compactado. Há dois dígitos por byte, sendo que o sinal fica definido pelo último meio byte mais à esquerda.

4.6. CLÁUSULA OCCURS

Formato para tabela unidimensionais:

número-nível nome-dado OCCURS inteiro TIMES.

A cláusula OCCURS pode ser utilizada em qualquer nível com exceção dos níveis 01 e 77.

Exemplo:

FILE SECTION.

01 REG-ALUNO.

 02 NUMERO PIC 9(06).
 02 NOME PIC X(20).
 02 CURSO PIC X(02).
 02 DISCIPLINA OCCURS 4 TIMES.
 05 MATERIA PIC X(10).
 05 NOTA PIC 9(02)V99.

Disciplina é uma tabela de 4 elementos formado por matéria e nota. Cada entrada da tabela terá $10 + 4 = 14$ bytes de comprimento, e a tabela toda terá $14 * 4 = 56$ bytes.

4.7. CLÁUSULA REDEFINES

E utilizada para fornecer uma descrição alternativa para um item de dado de grupo ou elementar, isto é, fazer com que um campo possa ser referenciado por mais de um nome.

FORMATO:

(N.O DE NÍVEL) data-name2 REDEFINES data-name1

IMPORTANTE:

1. o número de nível do data-name1 deverá ser o mesmo do data-name2.
2. Não deverá ser usado no nível 01 da FILE SECTION pois a redefinição é automática.
3. As entradas que serão redefinidas não podem conter a cláusula OCCURS.

Exemplo:

```
01 REG-CLIENTE.  
    02 CODIGO PIC X(03).  
    02 NOME PIC X(30).  
    02 ENDERECO PIC X(30).  
    02 TEL PIC 9(07).  
    02 FONE REDEFINES TEL PIC X(07).  
    02 FILLER PIC X(14).
```

A cláusula REDEFINES faz com que os campos TEL e FONE ocupem o mesmo espaço de memória. Esse conteúdo tem duas interpretações diferentes: TEL é numérico ao passo que FONE é alfanumérico.

Outro exemplo:

```
01 REG-EMPRESA.  
    05 DIV-DEPTO PIC 9(04).  
    05 DIV-DEPTO-RED REDEFINES DIV-DEPTO.  
        07 DIV PIC 9(02).  
        07 DEPTO PIC 9(02).
```

Neste exemplo, existe um único campo no registro que poderá ser utilizado como DIV-DEPTO e contém o valor da divisão e do departamento em único campo, ou poderá ser utilizado como DIV e DEPTO separadamente. É o mesmo espaço de memória sendo referenciado de dois modos diferentes.

CAPÍTULO 5

5. WORKING-STORAGE SECTION

A DATA DIVISION como um todo, contém todas as áreas de armazenamento definidas a serem alocados para o processamento. A WORKING-STORAGE SECTION contém todos os campos que não pertençam à entrada ou à saída, mas que são necessários ao processamento. Quaisquer constantes, totais intermediários, ou áreas de trabalho que não façam parte dos arquivos e que afetem o processamento são alocados na WORKING-STORAGE SECTION.

Esta seção pode conter duas categorias de campos:

- itens independentes e
- itens de campo, que requeiram subdivisão

Os itens independentes, definidos na WORKING-STORAGE SECTION, são campos individuais, cada um executando uma função independente e não relacionada a qualquer outro item.

A WORKING-STORAGE SECTION, como todas as seções, é codificada na margem A. Esta entrada vem após a FILE SECTION e todas as suas descrições, itens independentes recebem o nível 77 e também são codificados na margem A.

Os nomes atribuídos aos itens desta seção obedecem às regras de formação de nomes criados pelo programador. Todos os itens independentes devem ter suas cláusulas PICTURE correspondentes. Indicamos o formato ou o modo de um campo na WORKING-STORAGE como especificado anteriormente:

X denota campos alfanuméricos

9 denota campos numéricos

A denota campos alfabéticos

Exemplo:

WORKING-STORAGE SECTION.

77 TOTAL-INTERMEDIÁRIO PIC 9(5)V99.

77 CONSTANTE-1 PIC X(4).

77 FLOA PIC A(3).

Os itens independentes na WORKING-STORAGE SECTION são geralmente inicializados, ou seja, a eles são dados valores iniciais pela cláusula VALUE. Esta cláusula deve ser usada na FILE SECTION da DATA DIVISION.

É importante recordar que os computadores não limpam automaticamente a memória, quando lêem um novo programa. Uma área que é especificada na DATA DIVISION tem um valor indefinido quando começa a execução do programa. A menos que o programador indique um valor inicial para o campo, não se pode considerar que o campo esteja limpo com brancos ou zeros.

Para nos certificarmos de que os registros ou campos de saída especificados na FILE SECTION estão limpos no começo do programa, movemos SPACES para essas áreas na PROCEDURE DIVISION antes de iniciar o processamento. Na WORKING-STORAGE SECTION no entanto, podemos inicializar um item independente usando a cláusula VALUE.

Exemplo:

```
77 TOTAL      PIC 9(3)  VALUE ZERO.  
77 CONSTANTE-1 PIC X(4)  VALUE SPACES.
```

Uma cláusula VALUE não precisa ser especificada para um item independente. Tendo omitida, no entanto, não se pode saber o conteúdo inicial do campo. Se a cláusula VALUE não for indicada, é melhor usar a instrução MOVE na PROCEDURE DIVISION para obter um estudo inicial no campo.

São usadas quatro entradas (três das quais são obrigatórias) para indicar um item independente na WORKING-STORAGE SECTION.

5.1. ITENS INDEPENDENTES NA WORKING-STORAGE SECTION

1. Nível 77, codificado na margem A indica ao computador que um item de dado independente vai ser definido.
2. Um nome dado pelo programador define o campo.
3. O tamanho de um campo e seu formato ou modo são definidos pela cláusula PICTURE.
4. Um valor inicial deve ser armazenado no campo pela cláusula VALUE.

A cláusula VALUE conterá uma constante literal ou figurativa para ser colocada no campo. Deve estar no mesmo modo que a cláusula PICTURE. Se a PICTURE indica um campo numérico, o valor deve ser um literal numérico ou ZERO.

Exemplo:

```
77 TOTAL      PICTURE 9(04)  VALUE ZERO.  
77 CONSTANTE-X PICTURE 9(04)  VALUE 76.
```

Note que as cláusulas VALUE para inicializar campos não devem ser usadas na FILE SECTION da DATA DIVISION. Somente as entradas na WORKING-STORAGE podem ter tais cláusulas VALUE.

A cláusula VALUE de um campo numérico conterá um literal numérico:

```
77 FLDA PICTURE 9(2)V9(2)  VALUE 12,34.
```

Os literais numéricos não podem exceder a 18 dígitos de comprimento. Nesse caso, o VALUE de um item numérico na WORKING-STORAGE SECTION não pode ter mais de 18 dígitos.

Uma cláusula VALUE não numérica, como um literal não numérico deverá aparecer entre aspas e contém um máximo de 120 caracteres.

Exemplo:

77 CH-AUX PICTURE X(03) VALUE "NAO".

Caso haja necessidade de continuar a cláusula VALUE de uma linha da folha de codificação para outra devem ser codificadas as regras relacionadas a seguir.

5.2. REGRAS PARA CONTINUAÇÃO DE LITERAIS

1. Começar o literal entre aspas.
2. Continuar o literal até o fim da linha (não feche com aspas).
3. Coloque um traço (-) na posição de continuação da linha seguinte (7 colunas).
4. Continue o literal na margem B da próxima linha, começando com aspas.
5. Termine o literal com aspas.

As regras para continuação de literais na PROCEDURE DIVISION são as mesmas.

Os ítems de grupo podem ser armazenados na WORKING-STORAGE SECTION. Um ítem de grupo é aquele que é subdividido em dois ou mais ítems elementares.

Exemplo: Um nome de campo subdividido em primeiro e último nome, um nome de data, subdividido em dia, mês e ano, etc.

Os ítems de grupo na WORKING-STORAGE SECTION são codificados como registros no nível 01. Todos os ítems de grupo seguem os ítems elementares de nível 77 e são codificados na margem A, no nível 01.

Exemplo:

WORKING-STORAGE SECTION.

77 TOTAL PICTURE 9(05) VALUE ZEROES.
77 CONST PICTURE X(04) VALUE "CODE".
77 SAVE-AREA PICTURE X(03) VALUE SPACES.

01 ENDEREÇO-1.

 02 NÚMERO PICTURE 9(04).

 02 RUA PICTURE X(20).

 02 CIDADE PICTURE X(25).

 02 ESTADO PICTURE X(03).

01 DATA-ENT.

 02 MES PICTURE 9(02) VALUE 06.

 02 ANO PICTURE 9(02) VALUE 75.

A WORKING-STORAGE SECTION consiste de duas partes. Todos os ítems independentes aparecem no nível 77 e são seguidos pelos ítems de grupo que aparecem no nível 01. Ambas as entradas 77 e 01 são codificadas na margem A, os níveis 02 e 49, se usados, são codificados na margem B.

Os ítems de grupo na WORKING-STORAGE SECTION têm sua utilização máxima no armazenamento de grupos de campos de entrada que devem ser salvos para futuro processamento.

Exemplo:

Cartões de controle, cartões de entrada etc.

Uma outra forma de uso de ítems de grupo na WORKING-STORAGE SECTION é para a acumulação de dados de saída. Até agora temos escrito programas onde os dados de saída têm sido acumulados na área de saída da FILE-SECTION. Uma instrução WRITE (nome do registro) transmitirá o dado armazenado para o correspondente dispositivo de saída.

Os dados de saída, no entanto, podem ser armazenados na WORKING-STORAGE. Eles devem ser movidos para uma área de saída antes que a instrução WRITE possa ser executada.

Uma pergunta válida, neste ponto, é porque acumular os dados de saída na WORKING-STORAGE SECTION. A resposta é que os valores podem ser inicializados com a cláusula VALUE na WORKING-STORAGE SECTION, ao passo que as cláusulas VALUE não pode ser usadas na FILE SECTION.

Consideremos o seguinte registro de impressão válido para equipamento IBM:

```
FD PRINT-FILE
RECORDING MODE IS F
LABEL RECORDS ARE OMITTED
RECORD CONTAINS 133 CHARACTERS
DATA RECORD IS PRINT-REC.
```

```
01 PRINT-REC.
02 FILLER    PIC X(001).
02 INITIAL1  PIC X(001).
02 CONST1    PIC X(001).
02 INITIAL2  PIC X(001).
02 CONST2    PIC X(001).
02 LAST-NAME PIC X(018).
02 FILLER    PIC X(004).
02 MONTH     PIC 9(002).
02 CONST3    PIC X(001).
02 YEAR      PIC 9(002).
02 FILLER    PIC X(101).
```

Além de ler a primeira inicial, a segunda inicial, o último nome, mês e ano de um documento de entrada e movê-los para PRINT-REC, as seguintes operações MOVE são necessárias, para manter um relatório legível e estético:

```
MOVE SPACES TO PRINT-REC.
MOVE "."   TO CONST1.
MOVE ":"   TO CONST2.
MOVE "/"   TO CONST3.
```

O relatório de saída deve ter as constantes apropriadas:

J.E. SILVA 07/44
R.A. SOUZA 05/41

Se, no entanto, STORED-PRINT-LINE for dividida na WORKING-STORAGE como se segue, as operações anteriores serão desnecessárias:

```
01 STORED-PRINT-LINE.  
02 FILLER    PIC X(001) VALUE SPACES.  
02 INITIAL1  PIC X(001).  
02 CONST1    PIC X(001) VALUE ". ".  
02 INTIAL2   PIC X(001).  
02 CONST2    PIC X(001) VALUE ". ".  
02 LAST-NUM  PIC X(018).  
02 FILLER    PIC X(004) VALUE SPACES.  
02 MONTH     PIC 9(002).  
02 CONST3    PIC X(001) VALUE ". ".  
02 YEAR      PIC 9(002).  
02 FILLER    PIC X(101) VALUE SPACES.
```

Uma vez que as cláusulas VALUE são permitidas na WORKING-STORAGE SECTION, as constantes podem dar valores iniciais melhor do que movendo os literais apropriados para estes campos na PROCEDURE DIVISION.

Para imprimir os dados em STORED-PRINT-LINE depois que os campos de entrada forem movidos para o registro dizemos:

```
MOVE STORED-PRINT-LINE TO PRINT-REC.  
WRITE PRINT-REC AFTER ADVANCING 1 LINE.
```

Por isso, quando são necessários valores específicos num registro de saída, um item de grupo na WORKING-STORAGE pode ser estabelecido com as cláusulas VALUE apropriadas. Esta entrada pode ser então movida para uma área de saída, antes que o registro seja escrito. O método visto acima é considerado mais eficiente do que colocar campo na FILE-SECTION e efetuar operações MOVE independentes para cada literal desejado.

O uso de ítems de grupo na WORKING-STORAGE para armazenar dados de saída é ainda mais interessante quando se cria registros de cabeçalho na impressora.

CLÁUSULA [VALUE IS literal]

Esta cláusula têm efeito na WORKING-STORAGE SECTION.

EXEMPLO: Seja na WORKING-STORAGE SECTION:

```
01...  
    02 DIA PIC 9(02) VALUE 5.
```

Exemplo de registro de impressão no UNISYS:

Especificar na FILE SECTION o arquivo de relatório:

```
FD RELATORIO
LABEL RECORD ARE OMITTED.
01 REG-REL PIC X(132).
```

Na WORKING-STORAGE SECTION especificar o cabeçalho do relatório:

01 DETALHE.

```
02 FILLER    PIC X(03)  VALUE SPACES.
02 CODIGO    PIC 9(05).
02 FILLER    PIC X(03)  VALUE SPACES.
02 NOME      PIC X(30).
02 FILLER    PIC X(03)  VALUE SPACES.
02 CODIGO2   PIC 9(05).
02 FILLER    PIC X(05)  VALUE SPACES.
02 NOME2     PIC X(30).
02 FILLER    PIC X(48)  VALUE SPACES.
```

Na PROCEDURE DIVISION:

```
WRITE REG-REL FROM DETALHE AFTER ADVANCING LINE.
```

(Não pode ser usado com a cláusula OCCURS)

5.3. NOMES CONDICIONAIS

NOMES CONDICIONAIS são nomes fornecidos pelo programador na DATA DIVISION. Um nome condicional dá nome a um valor específico que um item de dado pode assumir. Na DATA DIVISION é codificado no nível especial 88. Todas as entradas de nível 88 são nomes condicionais que indicam valores de itens de dados específicos. Veja o seguinte exemplo:

02 MARITAL-STATUS **PIC 9.**

Suponha que i no campo chamado MARITAL-STATUS indique um status solteiro. Podemos usar um nome condicional, SINGLE (solteiro), para indicar este valor:

02 MARITAL-STATUS **PIC** **9(1).**

88 SINGLE **VALUE 1.**

Quando o campo chamado MARITAL-STATUS é igual a 1, chamamos essa condição de SINGLE. O item de nível 88 não é o nome do campo, mas o nome da condição. O item de nível 88 se refere somente ao item elementar que o precede imediatamente. SINGLE é o nome condicional aplicado ao campo chamado MARITAL-STATUS, já que MARITAL-STATUS precede diretamente o item de nível 88. A condição SINGLE existe se MARITAL-STATUS = 1.

Um nome condicional segue as regras de formação de nomes dados pelo programador. É sempre codificado no nível 88 e só tem uma cláusula VALUE associado a ele. Já que um nome condicional não é o nome do campo de dado, ele não conterá uma cláusula PICTURE.

Os ítems de nível 88 têm o seguinte formato:

O nome condicional se refere somente ao item elementar que o precede e, portanto, deve ser único. O VALOR do nome condicional deve ser um literal compatível com o tipo de dado do campo que o precede:

```
02 FLDX          PIC      X(02).
  88 CONDITION-A  VALUE    "12".
```

A declaração acima é válida, já que o valor é um literal não numérico e o campo é definido alfanuméricamente.

Nomes condicionais se referem somente a itens elementares da DATA DIVISION. O item de dado ao qual o nome condicional se refere deve conter uma cláusula PICTURE. Os itens de nível 77 na WORKING-STORAGE SECTION podem ter nomes condicionais associados a eles.

Nomes condicionais são definidos na DATA DIVISION para facilitar o processamento na PROCEDURE DIVISION. Um nome condicional é um método alternativo de expressar uma relação simples na PROCEDURE DIVISION, usando as seguintes entradas da DATA DIVISION:

```
02 MARITAL-STATUS   PIC      9(01).
  88 DIVORCED       VALUE    0.
```

Podemos usar qualquer um dos seguintes testes na PROCEDURE DIVISION.

IF MARITAL-STATUS EQUAL ZEROES GO TO DIVORCE-RTN.

OU

IF DIVORCED GO TO DIVORCE-RTN.

O nome condicional DIVORCED fará um teste para determinar se MARITAL-STATUS tem, de fato, o valor de 0.

Podemos usar vários nomes condicionais para um campo de dado:

```
02 GRADE          PIC      X.
  88 EXCELLENT     VALUE    "A".
  88 GOOD          VALUE    "B".
  88 FAIR          VALUE    "C".
  88 POOR          VALUE    "D".
  88 FAILING       VALUE    "F".
```

Admitindo que os valores acima são os únicos válidos, o teste da PROCEDURE DIVISION pode ser como se segue:

IF EXCELLENT OR GOOD OR FAIR OR POOR OR FAILING
NEXT SENTENCE

ELSE

GO TO ERROR-RTN.

Observe que também podemos dizer IF NOT FAILING GO TO PASS-RTN.

Suponha outro exemplo de utilização do nível 88:

```
01 DATA-ENTRADA.  
 02 ANO PIC 9(02).  
 02 MES PIC 9(02).  
    88 MES-30 VA 4 6 9 11.  
    88 MES-31 VA 1 3 5 7 8 10 12.  
    88 MES-VALIDO YA 1 THRU 12.  
 02 DIA PIC 9(02).  
    88 DIA-30 VA 1 THRU 30.  
    88 DIA-31 VA 1 THRU 31.  
    88 DIA-28 VA 1 THRU 28.
```

NA PROCEDURE DIVISION:

```
:  
  
IF NOT MES-VALIDO  
  MOVE "DATA INVALIDA" TO MENSAGEM-DATA  
ELSE  
  IF NOT(DIA-30 OR DIA-31 OR DIA-28)  
    MOVE "DATA-INVALIDA" TO MENSAGEM-DATA.  
  ELSE  
    IF NOT(MES-30 OR MES-31))  
      MOVE "DATA INVALIDA" TO MENSAGEM-DATA
```

Nomes condicionais podem ser usados na PROCEDURE DIVISION, a critério do programador para facilitar a programação. Há vezes em que o uso de nomes condicionais é necessário. Uma condição de overflow que testa o término da página, por exemplo, pode ser testada somente com um nome condicional.

5.4. PICTURE DE EDIÇÃO.

Caracteres e sinais usados na edição:

- Z - supressão de zeros à esquerda.
- "+" ou "-" - edita o sinal de "+" ou "-".
- "," - insere vírgula na posição indicada.
- "*" - substitui zeros à esquerda por "*".
- 0 ou B - insere zero ou espaço na máscara.
- ".." - edição do ponto decimal explícito.
- CE ou DB - coloca CR ou DB à esquerda ou à direita se o valor interno do dado for negativo.

Exemplos:

DADO	ENTRADA	PICTURE	SAIDA	EDICAO
571		9(03)V99	\$ZZ.299,99	\$Bpp571,00
(-)571		S99V99	\$*99,99	\$*57,10
(-)571		S99V99	\$*99,99DB	\$*57,10DB
(+)571		99V9	\$*99,99DB	\$*57,10
(-)571		S99V9	+99,99	-57,10
650		9(2)V9(3)	\$***9,99	\$***0,65
01MAI94		X(07)	99BXXB99	01BMAIB94
010594		9(06)	99/99/99	01/05/94

B = branco

CAPITULO 6

6. SCREEN SECTION

6.1. LINE e COLUMN.

As telas que serão usadas em programas são definidas após a Working-Storage Section na SCREEN SECTION. Nesta seção, informamos os valores constantes da tela e os campos variaveis pela intersecção de LINHAS x COLUNAS, que são definidas através de números de níveis.

Exemplo de uma tela:

SCREENS SECTION.

```
01 Tela.  
    05 Blank Screens.  
    05 Line 1 column 1 value "-----".  
    05 Line 2 column 1 value " Bom dia ".  
    05 Line 3 column 1 value " ".  
    05 Line 2 column 10 pic X(10) from nome.
```

6.2. FROM / TO / USING.

Se na working-storage o campo NOME estivesse definido como:

77 Nome PIC X(10) VALUE SPACES.

E na Procedure Division fossem dados os comandos:

Move "FATEC" to Nome. (move "FATEC" para o campo NOME)
Display Tela. (exibe a tela).

Apareceria no monitor na hora da execução do programa:

Bom dia FATEC

Ou seja, o conteúdo do campo NOME seria movido da working-storage section para o campo NOME da tela.

OBS: Blank Screens limpa tela.

Se ao invés de definirmos o campo Nome na Tela como FROM, usássemos TO; após um ACCEPT TELA, (comando que aceita entrada de dados via digitação através da tela) o conteúdo do campo Nome, digitado pelo usuário, seria movido da Tela para o campo Nome, que poderia ter sido definido na Working-Storage como um campo de trabalho ou na File Section como um campo pertencente a um registro, eliminando um comando de movimentação de campo dentro da Procedure Division, pois seria feita automaticamente.

O USING funciona como um FROM, em um display e como TO em um accept para um mesmo campo definido na tela, ou seja, serve ao mesmo tempo como entrada e como saída de dados.

7. PROCEDURE DIVISION

É a mais importante divisão do COBOL.

Contém todas as instruções a serem executadas pelo computador.

Toda lógica do programa está contida nestas instruções.

É nesta DIVISION, que o dado é lido, processado e onde se produz a informação de saída.

APRENDEREMOS NESTE CAPÍTULO A:

1. Abrir arquivo de entrada e saída.
2. Ler e escrever informações.
3. Realizar simples operações de mover e desviar.
4. Realizar operações específicas de finalização.

7.1. SUBDIVISÕES DA PROCEDURE

- A PROCEDURE DIVISION está dividida em parágrafos. Cada parágrafo define uma rotina independente, em uma série de instruções designadas para realizar uma função específica.

- Cada parágrafo, ainda, é subdividido em declarações ou sentenças. Uma declaração é uma instrução COBOL para o computador. Uma sentença é uma declaração ou grupo de declarações de um parágrafo. Cada declaração, salvo no caso de testar uma condição, começa com um verbo ou operação.

- Uma declaração geralmente acaba com um ponto final, que deve ser seguido de pelo menos um espaço em branco. Várias declarações podem ser escritas em uma linha de codificação COBOL, mas as palavras não podem ser divididas quando termina a linha.

- Cada declaração pode também ser escrita em linhas separadas. A fim de que fique mais clara a apresentação, esta forma é frequentemente a mais preferida.

- Todas as declarações são executadas na ordem em que são escritas, a menos que uma situação de desvio transfira o controle para alguma parte do programa.

- Todas as declarações na PROCEDURE DIVISION são codificadas na margem B. Somente os nomes de parágrafos são escritas na margem A.

7.2. OPEN (DECLARAÇÕES SIMPLIFICADAS)

Antes que um arquivo de entrada ou saída possa ser lido ou escrito devemos primeiramente, OPEN ("abrir") o arquivo. O computador é instruído para ter acesso ao arquivo pela declaração OPEN.

FORMATO

```
OPEN INPUT ( nome(s) de arquivo(s) )
          OUTPUT ( nome(s) de arquivo(s) )
```

O formato acima especifica que:

- (a) OPEN, INPUT e OUTPUT são palavras reservadas do COBOL.
- (b) Todos os nomes de arquivo de entrada e saída são palavras fornecidas pelo programador.

Para cada cláusula SELECT na ENVIRONMENT DIVISION, um nome de arquivo é definido e um periférico é alocado.

Em resumo, são realizadas duas funções básicas pela declaração OPEN.

1. Indica quais os arquivos que servem como entrada e quais os que servem como saída.
2. Ela coloca o arquivo à disposição de processamento.

7.3. CLOSE

Os arquivos devem ser abertos por uma declaração OPEN antes que os dados possam ser lidos ou escritos. Uma declaração CLOSE é necessário no final do programa para fechar estes arquivos.

O formato é:

```
CLOSE ( nome(s) de arquivo(s) )
```

Todos os arquivos que foram abertos precisam ser fechados no final do processamento.

Exemplo:

a. FIM.

CLOSE CARTAO FITA IMPRESSORA.

b. FIM.

CLOSE CARTAO.

CLOSE FITA.

CLOSE IMPRESSORA.

As duas rotinas são equivalentes, porém a menos que os arquivos sejam fechados em diferentes partes do programa, o segundo método (b) é considerado ineficiente.

7.4. READ (DECLARAÇÕES SIMPLIFICADAS)

Depois que um arquivo de entrada foi aberto, pode ser lido. Uma declaração READ transmite um dado do dispositivo de entrada, alocado na ENVIRONMENT DIVISION para a área de entrada de memória definida na FILE SECTION da DATA DIVISION.

FORMATO

READ (nome-do-arquivo) AT END (declaração).

O nome do arquivo especificado na declaração READ aparece em três outros lugares do programa.

1. Na cláusula SELECT, indicando o nome e o dispositivo alocado para o arquivo. Se, por exemplo, a leitora de cartões é o dispositivo alocado, a declaração READ transmite o dado do cartão de entrada para a área de entrada.
2. Na entrada FD, descrevendo o arquivo.
3. Na declaração OPEN, que dá acesso ao arquivo.

A principal função da declaração READ é transmitida aos dados para a memória.

A cláusula AT END na declaração READ testa o fim dos dados de entrada.

Uma declaração AT END junto com a declaração READ instrui o computador sobre o que fazer no caso de não haver mais dados para serem lidos.

7.5. WRITE

A instrução WRITE pega os dados acumulados na área de saída da DATA DIVISION e os transmite para os dispositivos especificados na ENVIRONMENT DIVISION.

FORMATO 1

WRITE (nome do registro)

Deve ser notado um ponto importante, embora os arquivos sejam lidos, nós escrevemos registros. O nome do registro aparece no nível 01 e é geralmente subdividido em campos.

A expansão WRITE (nome do registro) é usada para criar todos os registros de saída do dispositivo alocado.

- Exemplo:

WRITE REG-SAI.

OBS: REG-SAI é nome do registro definido na DATA DIVISION da FD nível 01.

FORMATO 2

WRITE registro [FROM nome do dado]

Exemplo:

WRITE REG FROM CAB.

É o mesmo que mover CAB para REG e gravar REG.

MOVE CAB TO REG.

WRITE REG.

FORMATO 3

WRITE registro [BEFORE ADVANCING inteiro LINES.]
[AFTER ADVANCING]

Exemplo:

WRITE LINHA BEFORE ADVANCING 2 LINES.

NOTA:

BEFORE - Imprime depois avança.

AFTER - Avança depois imprime.

FORMATO 4

WRITE REGISTRO AFTER ADVANCING PAGE.

Procedimento para pular de folha.

Neste caso, ocorre o salto de folha e o conteúdo de registro é impresso.

7.6. IF/ELSE

É a estrutura de seleção no COBOL.

FORMATO:

IF condição { instrução-1 } [ELSE instrução-2]
[NEXT SENTENCE] [ELSE NEXT SENTENCE]

EXEMPLO 1:

```
IF A > B  
    MOVE A TO MAIOR  
ELSE  
    MOVE B TO MAIOR.
```

Neste exemplo, se a condição A maior que B for satisfeita, o valor de A será armazenado na variável MAIOR. Se A for menor que B, então a instrução seguinte ao ELSE é executada e o valor de B é armazenado em MAIOR.

A cláusula ELSE na instrução IF é opcional. Se neste exemplo não tivéssemos a cláusula ELSE, a instrução IF seria executada quando a condição fosse válida, para condições inválidas, o programa passaria para as instruções seguintes ao ponto do IF.

Convém salientar que poderemos ter tantos comandos quantos forem necessários entre o comando IF e a cláusula ELSE, assim como entre o ELSE e o ponto final da instrução IF-ELSE.

EXEMPLO 2:

```
IF NOTA-ALU > 7  
    MOVE "APROVADO" TO DESCRICAO-REL  
    MOVE NOTA-ALUNO TO NOTA-REL  
    PERFORM GRAVA-APROVADO  
ELSE  
    MOVE "REPROVADO" TO DESCRICAO-REL  
    PERFORM GRAVA-REPROVADO.
```

EXEMPLO 3:

```
IF FALTA-ALU < 4
    NEXT SENTENCE
ELSE
    MOVE "REPROVACAO POR FALTA" TO DESC-REL.
```

Neste exemplo, se a condição FALTA-ALU. > 4 for satisfeita, então NEXT SENTENCE é realizado e o programa passa para as instruções seguintes ao ponto do IF-ELSE. Caso contrário, a instrução seguinte ao ELSE é realizado.

Neste exemplo, a utilização da sentença NEXT SENTENCE não é primordial, já que poderíamos ter feito a lógica omitindo esta passagem, entretanto em alguns casos de instruções IF-ELSE "aninhados" esta instrução será essencial para o fluxo de seleção.

COMANDO IF COM COMPARAÇÃO DE DOIS ÍTEMS

Quando o comando IF envolver comparação entre dois itens, ela poderá ser alfanumérica ou algébrica, dependendo das definições de tais ítems.

IMPORTANTE:

1. Na COMPARAÇÃO ALFANUMÉRICA, os dois ítems são comparados byte-a-byte, da esquerda para a direita. Se forem de tamanhos diferentes, o menor é completado com brancos à direita (APENAS para efeito de comparação) até se igualarem em tamanho (Usando registradores especiais).
2. Na COMPARAÇÃO ALGÉBRICA, os dois ítems são comparados pelo valor algébrico que representam.
3. A COMPARAÇÃO será ALFANUMÉRICA se um dos ítems for um literal não numérico (literal entre aspas), um item de grupo, ou um item elementar picture X ou A. Caso contrário será algébrica.

OTIMIZAÇÃO

Sempre que possível, deve-se empregar comparação alfanumérica, pois em geral, é mais rápida que uma correspondente algébrica.

7.7. RELATION CONDITION

```
IF (identifier-1) ([NOT] GREATER (>) THAN (identifier-2))  
    ([NOT] LESS (<) THAN )  
    ([NOT] EQUAL (=) TO )
```

7.8. CLASS CONDITION

```
IF (identifier-1) (IS [NOT] NUMERIC )  
    (IS [NOT] ALPHABETIC)
```

OBS: 1. Um campo numérico consiste dos dígitos de 0 a 9, com ou sem sinal de operação.

2. Um campo alfabético consiste dos caracteres A até Z, e do "SPACE".

7.9. SIGNAL CONDITION

```
IF (identifier-1) (IS [NOT] NEGATIVE)  
    } IS [NOT] POSITIVE)
```

7.10. COMPOUND CONDITION

AND ou OR

IMPORTANTE:

AND prevalece sobre o OR.

A OR B AND C.

A OR (B AND C).

Assim, os parênteses envolvendo o AND é desnecessário.
As duas opções são equivalentes.

TABELA AND E OR

1.o condição 2.o condição resultado (AND) resultado (OR)

V	V	V	V
V	F	F	V
F	V	F	V
F	F	F	F

7.II. MOVE

"MOVE" é um verbo enganoso, pois o objetivo da instrução MOVE é de copiar o conteúdo de um item de dado para outro(s) item(s) de dado(s).

O campo origem é inalterado pela execução de uma instrução MOVE.

O(s) campo(s) de destino(s) tem seu conteúdo original substituído pelo conteúdo do campo de origem, o conteúdo original do(s) campo(s) de destino é destruído.

FORMATO

MOVE [(IDENTIFICADOR-1)] TO (IDENTIFICADOR-2)
[LITERAL]

EXEMPLO:

01 ITEM-MOVER-AMOSTRA.

05 A PIC X(05).
05 B PIC X(05).
05 C PIC X(03).
05 D PIC X(07).

Supondo que o conteúdo do campo A seja "HELLO" então:

EXEMPLO 1:

MOVE A TO B. antes A = HELLO após A = HELLO
 B = ????? B = HELLO

EXEMPLO 2:

MOVE A TO C. antes A = HELLO após A = HELLO
 C = ??? C = HEL

EXEMPLO 3:

MOVE A TO D. antes A = HELLO após A = HELLO
 C = ??????? C = HELLObbb

EXEMPLO 4:

MOVE A TO B C D. antes A = HELLO após A = HELLO
 B = ????? B = HELLO
 C = ??? C = HEL
 D = ??????? D = HELLObbb

EXEMPLO 5:

MOVE SPACES TO A. antes A = HELLO após A = bbbbb

MOVE NUMÉRICO (REGRAS)

1. Quando se move um campo transmissor completo ou uma porção do campo transmissor para o campo receptor, o movimento é da direita para a esquerda. Todas as posições mais à esquerda não preenchidos do campo receptor são preenchidas com zeros.
2. Quando move uma porção decimal ou fracionária do campo transmissor para o campo receptor, o movimento é de esquerda para a direita, começando do ponto decimal implícito. As posições decimais não preenchidas (mais à direita) são preenchidas com zeros.

MOVE ALFANUMÉRICO (REGRAS)

Num movimento alfanumérico, o dado é transmitido do campo transmissor para o campo receptor da esquerda para a direita. As posições de mais baixa ordem do campo receptor, que não estão preenchidas são substituídas por espaços ou brancos.

DECLARAÇÃO MOVE CORRESPONDING

FORMATO:

MOVE CORRESPONDING (item de grupo) TO (item de grupo)

A opção MOVE CORRESPONDING é usada para substituir uma série de instruções MOVE simples. Todos os campos no campo transmissor são movidas para os campos de mesmo nome no campo receptor. Todas as regras para operações MOVE são também usadas para a opção MOVE CORRESPONDING.

7.12. INSTRUÇÃO GO TO

FORMATO:

GO TO (nome-do parágrafo)

A declaração GO TO transfere o controle para o nome do parágrafo indicado. Em algum ponto do parágrafo tem que haver um parágrafo com este nome. Ele é um comando incondicional de desvio, portanto, o programador deve estar atendo ao seu uso.

EXEMPLO:

PROCEDURE DIVISION.

OPEN INPUT ARQUIVO1
OUTPUT ARQUIVO2.

LEITURA.

READ ARQUIVO1 AT END GO TO FIM.

.

.

.

FIM.

CLOSE ARQUIVO1 ARQUIVO2.

7.13. PERFORM

Esta instrução permite desviar o programa para uma posição diferente daquela que normalmente executaria dentro da sequência normal, executando um parágrafo inteiro uma ou mais vezes.

A instrução mais simples da instrução PERFORM é:

SINTAXE 1

PERFORM nome-parágrafo

Onde nome-parágrafo é o nome de um parágrafo da PROCEDURE DIVISION definida pelo programador. A instrução faz com que todas as instruções existentes no parágrafo indicado sejam executados na sequência normal. A sequência normal das instruções é reassumida na instrução imediatamente posterior à instrução PERFORM.

EXEMPLO:

PROCEDURE DIVISION.

OPEN PARA-2.

PARA-2.

ADD A TO B.

SINTAXE 2

PERFORM nome-parágrafo UNTIL condição.

A utilização desta versão faz com que o parágrafo indicado seja executado até que a "condição" seja satisfeita.

EXEMPLO:

MOVE ZEROES TO TOTAL.

PERFORM RTN1 UNTIL B = 0.

RTN1.

ADD 1 TO TOTAL.

SUBTRACT 1 FROM B.

SINTAXE 3

PERFORM nome-procedimento1 [THROUGH] nome-procedimento2
[THRU]

Permite que n parágrafos sejam executados, desde nome-parágrafo1 até nome-parágrafo2.

EXEMPLO:

MOVE ZEROES TO TOTAL.
PERFORM READ-RTN THRU PRINT-RTN.

READ-RTN.

READ CARD-FILE AT END CLOSE CARD-FILE PRINT-FILE
STOP RUN.

PRINT-RTN.

MOVE TOTAL TO EDIT1.
WRITE PRINOT-OUT AFTER ADVANCING 2 LINES.

SINTAXE 4

PERFORM nome-procedimento1 [THROUGH] nome-procedimento2
[identifier1] TIMES
[inteiro1]

Faz com que sejam exécutados os parágrafos de nome-procedimento1 até nome-procedimento2 tantas quantas forem o valor de identifier1 ou inteiro1.

EXEMPLO:

PERFORM FIRST-RTN THRU PRINT-RTN TIMES.

FIRST-RTN.

MOVE ZEROES TO TOTAL.
PERFORM READ-RTN 10 TIMES.
GO TO PRINT-RTN.

READ-RTN.

READ CARD-FILE AT END CLOSE CARD-FILE PRINT-FILE
STOP RUN.

ADD AMT TO TOTAL.

PRINT-RTN.

MOVE TOTAL TO EDIT1.
WRITE PRINOT-OUT AFTER ADVENCING 2 LINES.

SECOND-RTN.

•
•

SINTAXE 5

```
PERFORM nome-procedimento1 [(THROUGH)
                           [(THRU      )
                            nome-procedimento2]
                           UNTIL     condição1
```

É o mesmo caso do PERFORM anterior com a diferença da condição que é até a "condição1" ser satisfeita.

EXEMPLO:

```
PERFORM LEITURA.
PERFORM GRAVA UNTIL LEITURA UNTIL CH-FIM = "S".
```

GRAVA.

```
WRITE ARQSAI FROM ARQENT.
```

LEITURA.

```
READ ARQENT AT END MOVE "S" TO CH-FIM.
```

SINTAXE 6

```
PERFORM nome-procedimento1 [THRУ (nome-procedimento2)
                           VARYING (nome-de-dado1) FROM [(nome-de-dado2)]
                           BY [(nome-de-dado3)] UNTIL (condição)
```

Exemplo:

```
PERFORM ROTINA VARYING CTR FROM
                           0 BY 1
                           UNTIL CTR=20.
```

O contador CTR é inicializado de 0, a condição CTR=20 é testada e se não for atendida, CTR é incrementado de 1. A sequência é sempre esta: Primeiro a inicialização seguida do teste de condição. Se ocorrer erro, a instrução logo após o PERFORM é executado, caso contrário ocorre o incremento, novamente a condição de erro é testada e assim até que a condição seja desfeita.

7.14. STOP RUN

O comando STOP RUN instrui o computador para terminar o programa.

Todos os programas devem terminar com uma declaração STOP RUN.

O formato é:

STOP RUN.

É obrigatório a colocação de uma instrução STOP RUN por programa, sem a qual os erros são imprevisíveis.

7.15. EXPRESSÕES ARITMÉTICAS

a. Símbolos de operações numéricas usadas no COBOL:

- ADIÇÃO (+)
- SUBTRAÇÃO (-)
- MULTIPLICAÇÃO (*)
- DIVISÃO (/)

b. As expressões aritméticas são resolvidas pelo COBOL, na ordem matemática.

- PARÊNTESES
- MULTIPLICAÇÃO E DIVISÃO
- SOMA E SUBTRAÇÃO

7.15.1. ADD

ADD A TO B.

É o mesmo que $B = B + A$.

ADD A B C GIVING D.

É o mesmo que $A + B + C = D$.

ADD A B C TO D.

É o mesmo que $A + B + C + D = D$.

7.15.2. SUBTRACT

SUBTRACT X Y Z FROM A.

É o mesmo que $A = A - (X + Y + Z)$.

SUBTRACT X Y Z FROM A GIVING B.

É o mesmo que $B = A - (X + Y + Z)$

7.15.3. MULTIPLY

MULTIPLY A BY B.

É o mesmo que $B = B * A$.

MULTIPLY A BY B GIVING C.

É o mesmo que $C = A * B$.

7.15.4. DIVIDE

DIVIDE A INTO B.

É o mesmo que $B = B / A$.

DIVIDE A INTO B

GIVING C REMAINDER D.

É o mesmo que $C = B / A$.

DIVIDE A BY B

GIVING C REMAINDER D.

É o mesmo que $C = A / B$.

OBS: 1. D armazena o RESTO da divisão.

2. Só se usa BY com o comando GIVING.

7.15.5. COMPUTE

FORMATO:

COMPUTE (nome-do-dado) (ROUNDED) = [literal]
[expressão aritmética]
[nome-do-dado]
ON SIZE ERROR (comando-imperativo)

EXEMPLO 1:

COMPUTE A = (B + C) / D * E.

EXEMPLO 2:

Para arredondar os resultados na declaração COMPUTE para as especificações do campo receptor:

COMPUTE A ROUNDED = B + C + D

EXEMPLO 3:

Para testar estouro de memória aritmética quando faltam posições inteiras suficientes no campo receptor para o resultado:

COMPUTE A = 105 - 3 ON SIZE ERROR GO TO ERROR-RTN.

OBS: Neste caso, sendo A um campo numérico de 2 dígitos ocorreria um truncamento do dígito da centena.

7.16. DISPLAY

A instrução DISPLAY é planejada para pequenos volumes de impressão em arquivos especiais do sistema ou no console do operador.

SINTAXE:

DISPLAY [identificador-1] [identificador-2]...[UPON nome-literal-1] [literal-2] especial]

É útil para "testes de mesa" de programas para conhecer o valor armazenado numa variável.

Exemplo:

DISPLAY "CH-AUX = " CH-AUX.

Se CH-AUX contiver o valor 2 aparecerá: CH-AUX = 2.

DISPLAY EM TELAS.

1 - DISPLAY [campo] UPON [command-line/con].

2 - DISPLAY [campo] AT LLCC.

3 - DISPLAY [campo].

7.17. ACCEPT

SINTAXE:

ACCEPT identificador FROM (DATE)
 (TIME)
 (DAY)

Exemplo:

Para importar a data do sistema operacional:

Na WORKING-STORAGE coloque:

01 DATA-SISTEMA.
02 AA-SIS PIC 9(02).
02 MM-SIS PIC 9(02).
02 DD-SIS PIC 9(02).

Na PROCEDURE DIVISION dê o comando ACCEPT para transferir a data do sistema operacional para o item de grupo DATA-SISTEMA. Geralmente os sistemas operacionais armazem a data no formato ANO/MES/DIA, logo é importante definir o item de grupo obedecendo esta ordem.

7.19. CURSOR - DEFININDO E POSICIONANDO.

Usado em telas para indicar sua posição física.
EXEMPLO:

DEFININDO:

- Na ENVIRONMENT DIVISION.

SPECIAL-NAMES.

CURSOR IS w-cursor.

- Na WORKING-STORAGE SECTION.

01 W-CURSOR.

 05 LINHA PIC 99.

 05 COLUNA PIC 99.

USANDO:

- Na PROCEDURE DIVISION.

MOVE 10 TO LINHA.

MOVE 15 TO COLUNA.

ACCEPT campo.

OBS: FUNCIONA APÓS UM ACCEPT.

QUANDO SE DÁ UM ACCEPT EM UM CAMPO INFORMANDO A LINHA E COLUNA, O CURSOR TAMBÉM SE POSICIONA NO INÍCIO DO CAMPO.

EXEMPLO.

ACCEPT campo AT 1015.

Assim:

ACCEPT DATA-SISTEMA FROM DATE.

Agora é só movimentar o conteúdo do DATA-SISTEMA para os campos específicos do seu programa.

ACCEPT EM TELAS.

1 - ACCEPT [campo] FROM {time/date/day-of-week/con}.

2 - ACCEPT [campo] AT LLCC.

onde LL = linha e CC = coluna.

3 - ACCEPT [campo].

7.18. EXIT.

E usado como ponto de retorno de um paragrafo usado como subprograma atraves do uso do comando PERFORM. Nem sempre é obrigatorio, normalmente ha o retorno ao ponto de chamada apos a execucao do ultimo comando do paragrafo usado. Entretanto, quando o paragrafo possui mais de uma alternativa de ramificacao proporcionada por comandos condicionais IF, passa a existir mais de um ponto de retorno que devem terminar, entao, em um paragrafo formado pelo comando EXIT.

7.20. FILE STATUS.

FILE STATUS INDICA O ESTADO DE UM ARQUIVO AO SER ACESSADO (ABERTO, FECHADO, LIDO, GRAVADO...).

DEFININDO:

- NA ENVIRONMENT DIVISION:

SELECT ARQUIVO ASSIGN TO DISK

FILE STATUS IS f-s.

- NA WORKING-STORAGE SECTION:

77 F-S PIC XX.

USANDO:

- NA PROCEDURE DIVISION:

READ ARQUIVO.

IF F-S EQUAL "00"

MOVE "SUCESSO" TO MENSAGEM

ELSE

IF F-S EQUAL "10"

MOVE "FIM DE ARQUIVO" TO MENSAGEM

ELSE

IF F-S EQUAL "23"

MOVE "REGISTRO NÃO ENCONTRADO".

7.21. TECLAS DE FUNÇÃO:

São as teclas ESC, F1, F2... que podem ser definidas para uso específico dentro de um programa com telas.

DEFININDO:

- NA ENVIRONMENT DIVISION.

SPECIAL-NAMES.

CRT STATUS IS status-chave.

- NA WORKING-STORAGE SECTION - DEFININDO CAMPOS:

01 STATUS-CHAVE.

05 TIPO-CHAVE PIC X.

05 CODIGO1-CHAVE PIC 9(02) COMP-X.

05 CODIGO2-CHAVE PIC 9(02) COMP-X.

(onde : TIPO-CHAVE = "0" - accept com término normal.
"1" - término definido pelo programador.
"2" - terminado pelo ADIS.
"3" - término por chave de dados de 8 bits.
"4" - término por chave de dados de 16 bits.
"9" - erro.)

01 BIT-SETADO

PIC 9(02) COMP-X VALUE 1.

01. CONTROLE-CHAVES.

05 ATIVA-DESATIVA PIC 9(02) COMP-X.

05 FILLER PIC X VALUE "1".

05 PRIMEIRA-CHAVE PIC 9(02) COMP-X.

05 QTDE-CHAVES PIC 9(02) COMP-X.

(onde: ATIVA-DESATIVA = 0 - desativa.

1 - ativa.

PRIMEIRA-CHAVE = número da primeira tecla
de função desejada.

QTDE-CHAVES = quantidade de teclas
desejadas à partir da
primeira-chave.)

- NA PROCEDURE DIVISION

USANDO:

(Movendo valores para as chaves)

MOVE 1 TO ATIVA-DESATIVA.

MOVE 0 TO PRIMEIRA-CHAVE.

MOVE 11 TO QTDE-CHAVES.

(Ativandando chaves)

CALL X"AF", USING BIT-SETADO CONTROLE-CHAVES.

(Testando as chaves)

ACCEPT campo AT 0101.

IF TIPO-CHAVE = "1"

EVALUATE CODIGO1-CHAVE

WHEN 0

DISPLAY "ESCAPE FOI PRESSIONADO"

WHEN 1

DISPLAY "F1 FOI PRESSIONADO"

WHEN 2

DISPLAY "F2 FOI PRESSIONADO"

.

.

.

WHEN 10

DISPLAY "F10 FOI PRESSIONADO".

END-EVALUATE

END-IF.

CAPÍTULO 8:

8. MANIPULAÇÃO DE TABELAS

TABELAS DE UMA DIMENSÃO

FORMATO para a definição de tabelas de comprimento fixo:

número-nível nome-dado OCCURS inteiro TIMES.

8.1. SUBSCRITOR

Para referenciar os elementos em uma tabela onde as descrições são idênticas, faz-se uso dos subscritores formados por um número inteiro positivo descritos entre parênteses após o nome do item da tabela.

Exemplo:

Considere o caso na qual lemos de uma entrada conhecida o valor do mês em forma de item numérico (definido como PIC 9) e precisamos imprimir o seu conteúdo na forma extendida. Assim, se o item lido foi 9 devemos imprimir SETEMBRO.

Poderíamos utilizar vários IF's indentados em nosso programa:

```
IF MES = 1
    MOVE "JANEIRO" TO MES-EXTENSO
ELSE
    IF MES = 2
        MOVE "FEVEREIRO" TO MES-EXTENSO
    ELSE
        IF MES = 3
            MOVE "MARCO" TO MES-EXTENSO
        .
        .
        .
    IF MES = 12
        MOVE "DEZEMBRO" TO MES-EXTENSO.
```

Há uma forma mais prática e estética de fazer isto no COBOL que vem da utilização de tabela.

Definimos as tabelas na WORKING-STORAGE SECTION.

01 TABELA-MESES.

```
03 FILLER PIC X(11) VALUE "01JANEIRO".
03 FILLER PIC X(11) VALUE "02FEVEREIRO".
02 FILLER PIC X(11) VALUE "03MARCO".
02 FILLER PIC X(11) VALUE "04ABRIL".
02 FILLER PIC X(11) VALUE "05MAIO".
02 FILLER PIC X(11) VALUE "06JUNHO".
02 FILLER PIC X(11) VALUE "07JULHO".
02 FILLER PIC X(11) VALUE "08AGOSTO".
02 FILLER PIC X(11) VALUE "09SETEMBRO".
02 FILLER PIC X(11) VALUE "10OUTUBRO".
02 FILLER PIC X(11) VALUE "11NOVEMBRO".
02 FILLER PIC X(11) VALUE "12DEZEMBRO",
```

01 TABELA-MESES-RED REDEFINES TABELA-MESES.

02 LINHA-MESES OCCURS 12 TIMES.

05 NUMERO-MES PIC 9(02).

05 MES-EXTENSO PIC X(09).

Na primeira etapa damos valores a cada entrada do mês e a seguir a redefinimos em um item de grupo para cada entrada na qual o primeiro item é PIC 9 e fornece o número do mês e o segundo item relaciona o mês na forma extendida.

Para referenciar os elementos da tabela devemos declarar o subscritor no nível 77:

77 SUB PIC 9(02) VALUE ZEROES.

Agora basta fazermos uma lógica na PROCEDURE DIVISION na qual pesquisamos o número do mês lido na tabela e a seguir imprimimos o seu conteúdo por extenso.

PROCEDURE DIVISION.

MOVE "NAO" TO ACHOU.

MOVE 1 TO SUB.

PERFORM PESQUISA UNTIL ACHOU = "SIM" OR SUB > 12.

PESQUISA.

IF MES EQUAL NUMERO-MES(SUB)

MOVE MES-EXTENSO(SUB) TO EXTEENO

MOVE "SIM" TO ACHOU

PERFORM IMPRIMIR.

ADD 1 TO SUB.

8.2. COMANDO SET.

Fornece um valor qualquer, incrementa ou decrementa os valores de um índice definido por USAGE IS INDEX na Working-Storage Section.

Exemplos:

Na Working-Storage Section.:

77 INDEX-X usage is index.

01 tabela.

02 dados occurs 100 times INDEXED BY INDEX-X.

03 numero pic 9(05).

Na Procedure Division

SET index-x to 2.

SET index-x UP BY 1. (incrementa de 1)

SET index-x DOWN BY 2. (decrementa de 2).

CAPITULO 9

9.1 ARQUIVO INDEXADO.

Em um arquivo, os dados podem ser colocados ou organizados de diversas maneiras, a fim de poder aproveitar ao máximo as vantagens oferecidas pelo tipo físico. (fita magnética, disco magnético, tambor magnético etc.) do dispositivo que forma o arquivo.

9.2 MODO INDEXADO.

Neste caso, a posição de cada dado no arquivo é determinada por índices ou KEYS mantidos pelo sistema operacional do computador que, usando uma técnica apropriada de pesquisa e busca de dados, coloca ou busca os dados no arquivo. Os arquivos de dados com organização indexada devem ser criados em dispositivos de acesso direto como o disco magnético e são especificados pela cláusula

ORGANIZATION IS INDEXED na ENVIRONMENT DIVISION.

9.3. DEFININDO UM ARQUIVO INDEXADO:

- NA ENVIRONMENT DIVISION:

```
SELECT [ARQUIVO] ASSIGN TO [PERIFÉRICO]
ORGANIZATION IS INDEXED
ACCESS MODE IS (SEQUENTIAL)
          (RANDOM)
          (DYNAMIC)
RECORD KEY IS [CAMPO-CHAVE].
```

OBS - ORGANIZACAO: É como o arquivo foi organizado, criado:

INDEXADO - Atraves de indices chamados de chaves.

- MODO DE ACESSO: Como os dados podem ser extraídos.

SEQUENTIAL - sequencialmente,

RANDOM - randomicamente (por chaves).

DYNAMIC - dinamicamente (sequencialmente e randomicamente no mesmo programa).

- NA FILE DESCRIPTION

Idem ao arquivo sequencial.

9.4. COMANDOS DE ARQUIVOS INDEXADOS.

9.4.1. OPEN:

OPEN (INPUT)
(OUTPUT)
(I-O)

Na forma I-O ele pode ser tanto de entrada como de saída dentro de um mesmo programa.

9.4.2. READ:

A - SEQUENCIALMENTE:

READ [ARQUIVO] NEXT RECORD [INTO]
[ARQUIVO1] AT END [COMANDO].

B - RANDOMICAMENTE:

MOVE [DADO] TO CHAVE.
READ [ARQUIVO] RECORD [INTO]
KEY IS CHAVE
[NOT] INVALID KEY [COMANDO]

9.4.3. WRITE:

WRITE [REGISTRO] [FROM] [REGISTRO1]
[NOT] INVALID KEY [COMANDO]

9.4.4. REWRITE:

REWRITE [REGISTRO] [FROM] [REGISTRO2]
[NOT] INVALID KEY [COMANDO]

9.4.5. DELETE: (DELEÇÃO LÓGICA)

DELETE [ARQUIVO] RECORD
[NOT] INVALID KEY [COMANDO]

9.4.6. START: (POSICIONA PONTEIRO DE REGISTRO):

START [ARQUIVO] KEY IS (=) [DADO]
(<)
(>)
[NOT] INVALID KEY [COMANDO]

CAPITULO 10:

UTILIZANDO O 'MICROSOFT' VERSAO 4.5.

10.1. USANDO O "SOFTWARE" - PWB. (Programmer's WorkBench).

Para se utilizar o "SOFTWARE", digite PWB dentro do diretório COBOL, onde ele se encontra instalado. Irá apresentar uma tela com um MENU que poderá ser acessado através da tecla ALT - pressione a letra em destaque para acessar a opção desejada ou com o cursor sobre a palavra, pressione ENTER e será aberta uma janela de opções. Para sair, use a janela FILE, opção EXIT.

10.1.1. EDITOR DE TEXTO.

Para editar textos ou acessar arquivos, use a PWB na opção FILE, esta janela dá acesso a criação de novos arquivos, recuperação dos já criados, e salvamento automático toda vez que um arquivo for alterado.

10.1.2. COMPILEANDO UM PROGRAMA.

Para compilar, use a PWB na opção MAKE, ou na linha de comando do DOS, dentro do diretório COBOL digite - COBOL, será automaticamente pedido o nome do arquivo que contém o programa fonte (.CBL), e os nomes dos arquivos aonde serão gerados o programa objeto (.OBJ), a listagem de compilação (.LST) e a listagem do objeto, se solicitados.

10.1.3. LINKANDO UM PROGRAMA.

Para Linkar um programa e gerar o executável, use a PWB na opção OPTIONS ou na linha de comando do DOS digite - LINK nome-do-programa.OBJ + ADIS + ADISINIT + ADISKEY + ADISDYNA + EXTFH, e será pedido o nome do arquivo aonde será gerado o programa executável (EXE).

10.1.4. EXECUTANDO UM PROGRAMA.

Para executar um programa, use a PWB na opção MAKE ou na linha de comando do DOS digite o nome do arquivo que contém o PROGRAMA EXECUTÁVEL (.EXE).

10.1.5. RECONSTRUINDO/REORGANIZANDO ÍNDICES DE UM ARQUIVO.

Para reconstruir/reorganizar os índices de um arquivo, use a PWB na opção MAKE ou na linha de comando do DOS digite:

Para reconstruir : REBUILD in-file.DAT [k...] [/i] [/v] [/n] [/e] [/c].

Para reorganizar : REBUILD in-file.DAT,out-file.DAT [/x] [/i] [/v] [/e].

/C = Especifica o tipo de compactação a ser feita no arquivo.

/E = Previne duplicidade de chaves gerando um aborto.

/I = Exibe informações.

/K = Define estrutura de chaves do arquivo de saída.

/N = Exibe informações sem processar o arquivo.

/V = Exibe contador de registros da execução.

/X = Indica a chave de referência.

10.2. "SCREENS".

Na linha de comando do DOS digite - SCREENS [arquivo].

Aparecerá uma tela em branco com um MENU no rodapé. Pressionando e segurando a tecla ALT aparecerá um segundo menu, com a tecla CTRL, um terceiro.

10.2.1. DEFININDO UMA TELA.

SEJA ESTE O MODELO:

1			1
1	FACULDADE DE TECNOLOGIA DE SÃO PAULO		1
1			1
1	TESTE	DATA / /	1
1			1
1	INCLUSÃO		1
1			1
1	NOME . . . :		1
1			1
1	ENDEREÇO:		1
1			1
1	MENSAGEM:		1
1			1

10.2.1.1. DESENHANDO MOLDURA.

PRESSIONANDO CTRL + F6 = (DRAW - DESENHO)
aparecerá um menu no rodapé com as opções:

- F2 [ERASE/MOVE/DRAW] = (APAGA/MOVE/DESENHA)
Pressione F2 várias vezes até aparecer na linha de menu, à esquerda no rodapé, a palavra DRAW.
- F3 = TIPO DE MOLDURA
Pressione F3 e escolha a moldura desejada, simples ou dupla.
- COM AS SETAS DE DIREÇÃO TRACE A MOLDURA NA TELA.
- ESC = ENCERRA.

10.2.1.2. DEFININDO CAMPOS.

- Digite o texto da tela (cabecalho, nome de campo).
- Defina os campos variáveis digitando ^ (acento circunflexo) para cada posição do campo.
ex: se nome é um campo com 30 (trinta) posições, digite 30 (^).
- Pressionando F3 sobre o primeiro (^) do campo variável, abrirá uma janela com opções de tipos de atributos, escolha o tipo com as setas de direção e fixe com F2. (F2 novamente desmarca).
- Usando a opção TO, FROM ou USING será pedido o nome do campo que será utilizado como entrada e/ou saída, informe o nome e dê ENTER.
- ESC - ENCERRA.

10.2.1.3. PINTANDO CAMPOS.

- ESCOLHENDO COR DE FRENTES X FUNDO)
Pressione ALT + F9, será aberto um menu com opções de cores; com as setas de direção escolha as cores e fixe-as com uma das teclas F2 à F7, sendo que F7 será a cor definida como padrão. Dê ESC para sair.
- F6 (SELECCIONANDO COR FIXADA PARA FRENTES E FUNDO)
Escolha uma das cores fixadas teclando sucessivamente F6, verificando a opção na amostragem do rodapé.
- COR DE FRENTES
Ao se digitar algum caractere na tela, este sairá com a cor definida como FRENTES.
- F5 (PINTANDO FUNDO DA TELA)
Tecle F5 sobre a extensão do campo para se pintar o fundo deste com a cor escolhida com F6.
- ATIVANDO/DESATIVANDO CORES)
Para ativar/desativar opções de cores, ficando a padrão, pressione ALT + F2.

10.2.1.4. DEFININDO GRUPOS (OCORRÊNCIAS).

- F2 (MARCA/DESMARCA CAMPO).
Sobre o primeiro (^) do campo variável tecle F2, ele aparecerá em destaque, ande com o cursor até o final da extensão do campo;
- F4 (DEFINE OCORRÊNCIAS).
Pressione F4, aparecerá um menu de opções, para repetir o campo verticalmente, pressione F4 quantas vezes quiser que este campo se repita.

10.2.1.5. MOVENDO / COPIANDO TEXTO E DADOS DA TELA.

- F2 - idem anterior.
- F7 - SEPARA PARTE DA TELA.
abre novo menu no rodapé.
- Com a seta de direção posicione o cursor no local de inserção.
- F2 - MOVE campo.
- F3 - COPIA campo,

10.2.1.6. ALTERANDO A ORDEM DO ACCEPT.

- F10 - (DEFINE A ORDEM DE ENTRADA DOS CAMPOS).
Quando esta tecla é acionada, todos os campos variáveis aparecem com "5" sobre a sua extensão. Para alterar, digite sobre o primeiro "5" de cada campo a ordem de prioridade de acesso, se quiser que seja acessado antes, digite "4" na primeira posição do campo, se quiser que seja depois, digite "6" e assim sucessivamente.
- F10 - (SAI da opção SALVANDO a ordem).

10.2.1.7. GERANDO O COBOL.

- ALT + F5 (GERA O COBOL).
Será pedido o nome do arquivo, informe e dê ENTER
- F3 (ESQUELETO).
Após o ALT + F5 um menu, tecle F3 e aparecerá o esqueleto da codificação do programa.CBL

10.2.1.8. SALVANDO A TELA.

- ALT + F4 (OPÇÃO PARA SALVAR TELA).
Será pedido o nome da tela (.SRN é default)
- ENTER (SALVA TELA).

10.2.1.9. DELETANDO DEFINIÇÕES DE ATRIBUTOS DE CAMPOS.

CTRL + F9.

10.2.1.10. DELETANDO DEFINIÇÕES DE ATRIBUTOS DE GRUPOS.

CTRL + F10.

10.2.1.11. DELETANDO LINHAS EM BRANCO.

ALT + F7.

10.2.1.12. INSERINDO LINHAS.

ALT + F8.

10.2.1.13. SAINDO DA "SCREENS".

ESC -
Y (yes).

10.2.2. ARQUIVOS GERADOS:

Após gerar e salvar o programa e a tela, serão criados os seguintes arquivos em disco:

TELA.CBL - Esqueleto do programa fonte.

TELA.SRN - Desenho da tela.

TELA.SS - Screen Section (codificação da tela).

TELA.WKS - Working-Storage Section. (das variáveis).

OBS. O programa TELA.CBL inclui os arquivos TELA.SS e TELA.WKS através do comando:

COPY "tela.xxx".

À partir do arquivo TELA.CBL fazer as devidas alterações e incluir a lógica do programa.

10.2.3. ATRIBUTOS DE CAMPOS.

AUTO - SALTA AO FINAL DE UM CAMPO, PARA O COMEÇO DO PRÓXIMO.

ZERO FILL - PREENCHE CAMPO NUMÉRICO COM ZEROS.

REQUIRED - CAMPO DE PREENCHIMENTO OBRIGATÓRIO

JUST RIGHT - ALINHA CONTEÚDO DO CAMPO À DIREITA.

FULL - OBRIGATÓRIO O PREENCIMENTO COMPLETO.

BLANK WHEN ZERO - QUANDO FOR ZEROS, TROCA POR BRANCOS.

BELL - TOCA SINO.

BLINK - CAMPO PISCANTE

SIZE - AJUSTE AUTOMÁTICO DE TAMANHO DE CAMPO PELO MAIOR CONTEÚDO CONTIDO.

CAPÍTULO 11: PROGRAMA CADASTRO DE PESSOAS - FATEC/SP

Exemplos de programas:

IDENTIFICATION DIVISION.

PROGRAM-ID. TESTE01.

AUTHOR. ATN.

INSTALLATION. FATEC/SP.

DATE-WRITTEN. 26-AGOSTO-1991.

DATE-COMPILED.

SECURITY. ESTE PROGRAMA SOMENTE PODERA SER MODIFICADO COM AUTORIZACAO DO AUTOR.

*REMARKS. PROGRAMA QUE LE UM REGISTRO DO ARQUIVO DE ENTRADA E GRAVA NUM ARQUIVO DE SAIDA EM DISCO.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. UNISYS.

OBJECT-COMPUTER. UNISYS.

SPECIAL-NAMES. DECIMAL-POINT IS COMMA.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT CADASTRO01 ASSIGN TO DISK.

SELECT CADASTRO02 ASSIGN TO DISK.

DATA DIVISION.

FILE SECTION.

FD CADASTRO01

LABEL RECORD ARE STANDARD

VALUE OF TITLE IS "CADASTRO01"

BLOCK CONTAINS 1260 CHARACTERS.

01 REG-CAD01.

02 CODIGO PIC 9(05).

02 NOME PIC X(30).

02 IDADE PIC 9(02).

02 SALARIO PIC 9(07)V9(02).

02 FILLER PIC X(38).

FD CADASTRO02

LABEL RECORD IS STANDARD

BLOCK CONTAINS 1260 CHARACTERS.

01 REG-CAD02.

02 CODIGO-SAI PIC 9(05).

02 NOME-SAI PIC X(30).

02 SALARIO-SAI PIC 9(07)V99.

02 FILLER PIC X(40).

WORKING-STORAGE SECTION.

77 FIM-ARQ PIC X(03) VALUE "NAO".

PROCEDURE DIVISION.

TESTE01.

PERFORM INICIO.

PERFORM PRINCIPAL UNTIL FIM-ARQ EQUAL "SIM".

PERFORM FIM.

STOP RUN.

INICIO.

OPEN INPUT CADASTRO01

OUTPUT CADASTRO02.

PERFORM LEITURA.

LEITURA.

READ CADASTRO1

AT END MOVE "SIM" TO FIM-ARQ.

PRINCIPAL.

PERFORM GRAVACAO.

PERFORM LEITURA.

GRAVACAO.

MOVE CODIGO TO CODIGO-SAI.

MOVE NOME TO NOME-SAI.

MOVE SALARIO TO SALARIO-SAI.

WRITE REG-CADO2.

FIM.

CLOSE CADASTRO01

CADASTRO02 SAVE.

ESSAO.

PERFORM CADASTRO01

MOVE CODIGO

MOVE NOME

MOVE IDADE

MOVE SALARIO

WRITE REG-CADO2

ADD 1 TO CT-REG

Exemplo 2:

IDENTIFICATION DIVISION.
PROGRAM-ID. TESTE02.
INSTALLATION. FATEC/SP.
DATE-WRITTEN. 26-AGOSTO-1991.
DATE-COMPILED.
SECURITY. PROGRAMA PASSIVEL DE MODIFICACAO.
*REMARKS. PROGRAMA QUE LE UM ARQUIVO DE ENTRADA DE IMPRIME
UM RELATORIO COMO SAIDA.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. UNISYS.
OBJECT-COMPUTER. UNISYS.
SPECIAL-NAMES. DECIMAL-POINT IS COMMA.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
 SELECT CADASTRO02 ASSIGN TO DISK.
 SELECT RELAT ASSIGN TO PRINTER.

DATA DIVISION.
FILE SECTION.
FD CADASTRO02
 LABEL RECORD IS STANDARD
 VALUE OF TITLE IS "CADASTRO02"
 BLOCK CONTAINS 1260 CHARACTERS.

01 REG-CAD02.
 05 CODIGO PIC 9(05).
 05 NOME PIC X(30).
 05 IDADE PIC X(02).
 05 SALARIO PIC 9(07)V99.
 05 FILLER PIC X(38).

FD RELAT
 LABEL RECORD IS OMITTED.
01 REG-REL PIC X(132).

WORKING-STORAGE SECTION.
77 FIM-ARQ PIC X(03) VALUE "NAO".
77 CT-LIN PIC 9(02) VALUE 11.
77 CT-PAG PIC 9(02) VALUE ZEROES.

01 CAB-01.
02 FILLER PIC X(01) VA SPACES.
02 FILLER PIC X(07) VA "RELEMO1".
02 FILLER PIC X(04) VA SPACES.
02 FILLER PIC X(08) VA "FATEC-SP".
02 FILLER PIC X(31) VA SPACES.
02 FILLER PIC X(28) VA "P R O G R A M A T E S T E".
02 FILLER PIC X(30) VA SPACES.
02 FILLER PIC X(01) VA SPACES.
02 FILLER PIC X(04) VA "PAG.". .
02 VAL-PAG PIC Z(03).
02 FILLER PIC X(15) VA SPACES.

01 DETALHE.
05 FILLER PIC X(05) VA SPACES.
05 CODIGO-REL PIC 9(05).
05 FILLER PIC X(10) VA SPACES.
05 NOME-REL PIC X(30).
05 FILLER PIC X(10) VA SPACES.
05 IDADE-REL PIC 9(02).
05 FILLER PIC X(10) VA SPACES.
05 SALARIO-REL PIC Z.ZZZ.ZZZ,99.

PROCEDURE DIVISION.

TESTE02.

PERFORM INICIO.
PERFORM PRINCIPAL UNTIL FIM-ARQ EQUAL "SIM".
PERFORM FIM.
STOP RUN.

INICIO.

OPEN INPUT CADASTRO02
OUTPUT RELAT.
PERFORM LEITURA,

LEITURA.

READ CADASTRO02
AT END MOVE "SIM" TO FIM-ARQ.

PRINCIPAL.

PERFORM IMPRESSAO.
PERFORM LEITURA.

IMPRESSAO.

IF CT-LIN GREATER THAN 10
PERFORM CABECALHO.
MOVE CODIGO TO CODIGO-REL.
MOVE NOME TO NOME-REL.
MOVE IDADE TO IDADE-REL.
MOVE SALARIO TO SALARIO-REL.
WRITE REG-REL FROM DETALHE AFTER ADVANCING 1 LINE.
ADD 1 TO CT-LIN.

CABECALHO.

ADD 1 TO CT-PAG.

MOVE CT-PAG TO VAL-PAG.

WRITE REG-REL FROM CAB-01 AFTER ADVANCING PAGE.

MOVE ZEROES TO CT-LIN.

FIM.

CLOSE CADASTRO02

RELAT.

Exemplo 3:

IDENTIFICATION DIVISION.
PROGRAM-ID. TESTE03.
AUTHOR. ATN.
INSTALLATION. FATEC-SP.
DATE-WRITTEN. 26-AGOSTO-1991.
DATE-COMPILED.
SECURITY. PROGRAMA PODERA SER MODIFICADO PARA MELHORIAS NO
FUTURO.
*REMARKS. PROGRAMA QUE SELECCIONA OS FUNCIONARIOS COM
SALARIO MAIOR OU IGUAL A Cr\$30.000,00 GRAVANDO-OS
NUM ARQUIVO DE SAIDA.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. UNISYS.
OBJECT-COMPUTER. UNISYS.
SPECIAL-NAMES. DECIMAL-POINT IS COMMA.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT CADASTRO03 ASSIGN TO DISK.
SELECT CADASTRO04 ASSIGN TO DISK.

DATA DIVISION.
FILE SECTION.
FD CADASTRO03
LABEL RECORD ARE STANDARD
VALUE OF TITLE IS CADASTRO03
BLOCK CONTAINS 1260 CHARACTERS.
01 REG-CAD03.
05 CODIGO03 PIC X(05).
05 NOME03 PIC X(30).
05 IDADE03 PIC 9(02).
05 SALARIO03 PIC 9(07)V99.
05 FILLER PIC X(38).

FD CADASTRO04
LABEL RECORD ARE STANDARD
BLOCK CONTAINS 1260 CHARACTERS,

01 REG-CAD04.
05 CODIGO04 PIC 9(05).
05 NOME04 PIC X(30).
05 IDADE04 PIC 9(02).
05 SALARIO04 PIC 9(07)V99.
05 FILLER PIC X(38).

WORKING-STORAGE SECTION.

77 CH-NOTA PIC 9(02)V99.

77 FIM-ARQ PIC X(03) VA "NAO".

PROCEDURE DIVISION.

TESTE03.

PERFORM INICIO.

PERFORM PRINCIPAL UNTIL FIM-ARQ EQUAL "SIM".

PERFORM FIM.

STOP RUN.

INICIO.

OPEN INPUT CADASTRO03

OUTPUT CADASTRO04.

PERFORM LEITURA.

LEITURA.

READ CADASTRO03

AT END MOVE "SIM" TO FIM-ARQ.

PRINCIPAL.

IF SALARIO03 GREATER THAN 30000 OR

SALARIO03 EQUAL 30000

PERFORM GRAVACAO.

PERFORM LEITURA.

GRAVACAO.

MOVE CODIGO03 TO CODIGO04.

MOVE NOME03 TO NOME04.

MOVE IDADE03 TO IDADE04.

MOVE SALARIO03 TO SALARIO04

WRITE REG-CAD04.

FIM.

CLOSE CADASTRO03

CADASTRO04 SAVE.