Outro material de apoio: na **Turma-TPI3,** no tópico ***UFCD 0807*** - Programação em Cobol – Fundamentos, da Classroom, do Google drive.

Linguagem COBOL (Common Business Oriented Language) é uma linguagem de programação de terceira geração, que foi originalmente desenvolvida para negócios, finanças e necessidades dos sistemas administrativos (empresas e governos).

O COBOL foi criado em 1959 durante o *CODASYL (Conference on Data Systems Language)*, um dos três comitês propostos numa reunião no Pentágono em maio de 1959, organizado por Charles Phillips do Departamento de Defesa dos Estados Unidos. O CODASYL foi formado para recomendar as diretrizes de uma linguagem para negócios.

Os compiladores de COBOL geralmente baseavam-se no COBOL Padrão Nacional Americano (ANSI), que adotou o COBOL como uma linguagem padrão. A última revisão foi concluída em 2002, incluindo a padronização de OO (Object Oriented).

Atualmente usada por empresas públicas e privadas, de todos os setores e portes em todo o mundo, e, principalmente, da área financeira, o Cobol é uma linguagem viva e que deverá manter alta demanda de profissionais nos próximos anos.
Diariamente são processadas 200 vezes mais transações que utilizam linguagem de programação Cobol do que os acessos a pesquisas realizadas em alguns mecanismos de busca na internet. No entanto, a falta de programadores especializados já é um dos grandes desafios para as empresas.

Caraterísticas do COBOL

- A linguagem Cobol foi projetada para ser entendido por não-programadores, como sejam, supervisores, gerentes e meros utilizadores. Por isso o COBOL usa frases normais da língua inglesa, e a sua estrutura é semelhante a um texto com verbos, cláusulas, secções, divisões e parágrafos. Este objetivo fez do COBOL a linguagem de programação mais legível, compreensível e auto documentada, em uso hoje em dia;

- O COBOL é uma linguagem muito simples, sem apontadores, sem tipos de dados definidos pelo utilizador e com um número limitado de funções;

- É a única linguagem que aceita hífens em nomes e variáveis;

- Utiliza comandos separados para cada operação matemática básica e comandos para as fórmulas matemáticas;

- As variáveis são divididas por níveis, podendo uma variável ser parte de outra.

Depois de escrito o programa Cobol (ou programa fonte), é necessário traduzí-lo para a linguagem interna do computador (linguagem máquina), convertendo-se então num programa objeto. Esta conversão é feita pelo próprio computador, usando o programa compilador de Cobol.

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Europeu de Desenvolvimento Regional

Regras de codificação:

Qualquer programa escrito na linguagem Cobol possui algumas regras a serem seguidas.
Uma destas regras refere-se ao formato das linhas de comando (instruções) dentro do seu editor do programa fonte.
Uma linha de comando Cobol pode ter até 80 carateres, de acordo com o formato abaixo:

Sequência		I	Área A		Área B		Comentário	
1	6	7	8	11	12	72	73	80

Colunas de 1 a 6: Área de numeração sequêncial
Coluna 7: Área de indicação
Colunas de 8 a 11: Área A
Colunas de 12 a 72: Área B
Colunas de 73 a 80: Comentários

2.1 Área de numeração sequêncial - sequência

Normalmente consiste em seis dígitos em ordem crescente que normalmente são utilizados para numerar as linhas do programa fonte. Segundo as regras no ANS85 pode-se também colocar comentários nesta área. Além disso, podemos colocar um asterisco na coluna 1 (um) ou qualquer outro caráter com valor ASCII menor do que 20 (ESPAÇO), fazendo com que a linha inteira seja considerada como um comentário.

Pode-se também deixar esta área em branco.

2.2 Área de indicação . I

Um hífen (-) nesta posição indica que existe uma continuação de um texto que foi iniciado na linha anterior.

Um asterisco (*) nesta posição indica que toda a linha deve ser tratada como um comentário.

Uma barra (/) nesta posição, além de marcar a linha como comentário fará com que ao se imprimir este caráter, haverá um salto de página após esta linha.

2.3 Área A

Posição a partir do qual se iniciam divisões, parágrafos ou secções.

2.4 Área B

Posição a partir da qual se escrevem as instruções Cobol.
As instruções são finalizadas com ponto final (.)

Nota: Quando o código for compilado, será verificado se o código adere ao agrupamento de colunas especificado para layout. Caso haja uma violação da regra o compilador irá dar erro.

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Europeu de Desenvolvimento Regional

Exemplo 1:

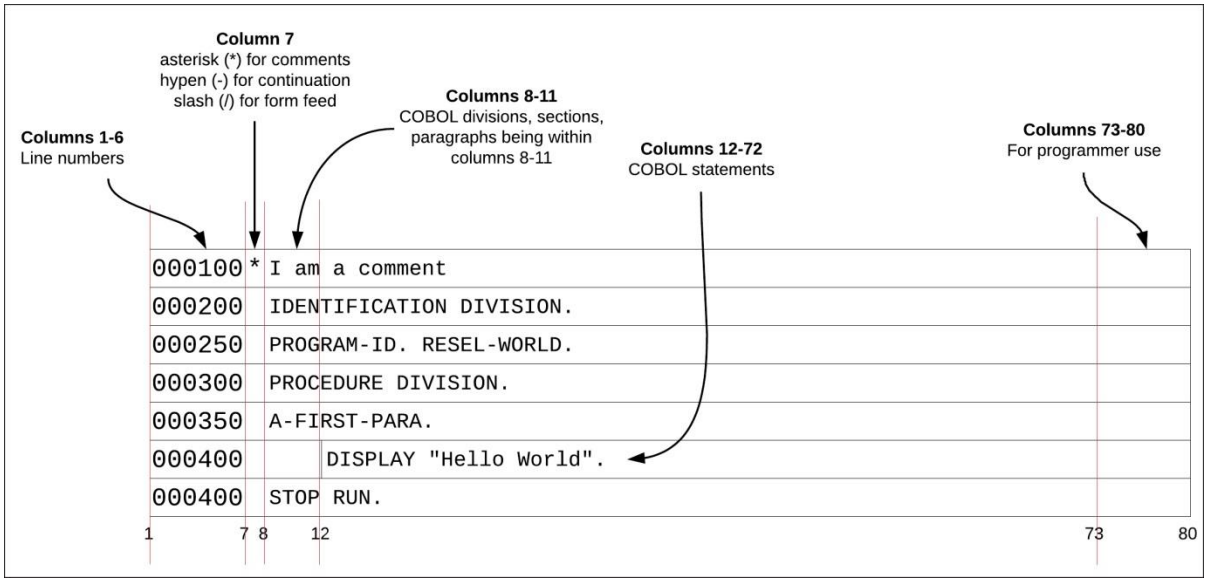


Figura 1 - Cada coluna num ficheiro de COBOL é definida para servir um propósito específico

A estrutura hierárquica do COBOL

A hierarquia de um programa em COBOL consiste numa estrutura de tópicos do documento, com uma única posição de nível superior, seguida de níveis subordinados.
As unidades organizacionais que compõem a hierarquia são programa, divisão/division, secção/section, parágrafo/paragraph, frase/sentence, instrução/statement e caráter/character.

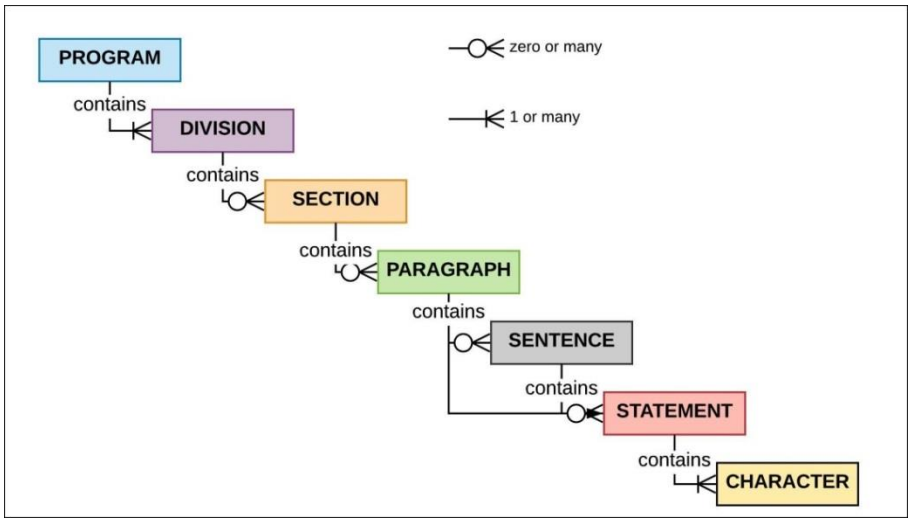


Figura 2 - A hierarquia estrutural das entidades do programa COBOL

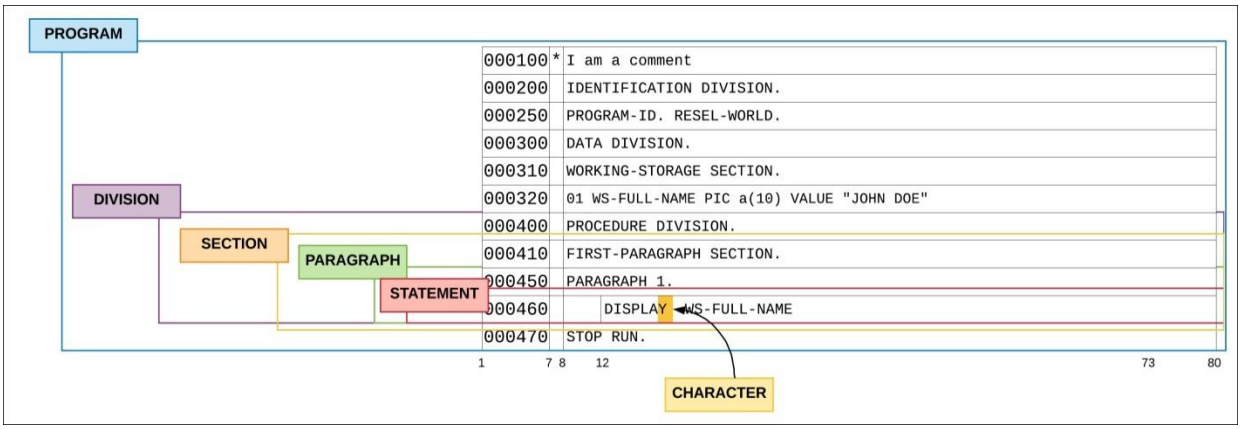


Figura 3 - Estrutura hierárquica de um programa em Cobol

DIVISION

A divisão é um bloco de código que contém uma ou mais secções ou conjuntos de secções ou conjuntos de frases ou um bloco de instruções.
Cada divisão começa com o seu nome e termina no início da próxima divisão ou no final do programa.
Todas as divisões devem ser codificadas na área de A.

As divisões existentes são:

IDENTIFICATION DIVISION. (*divisão de identificação)*

ENVIRONMENT DIVISION. *(divisão de ambiente)*

DATA DIVISION. *(divisão de dados)*

PROCEDURE DIVISION. *(divisão de procedimentos)*

IDENTIFICATION DIVISION

É a divisão de identificação do programa. Aqui as informações têm o objetivo principal de documentação do código fonte. Esta divisão não possui secções. Este é o cabeçalho do programa.

IDENTIFICATION DIVISION.	**DIVISÃO DE IDENTIFICAÇÃO.**
PROGRAM-ID. TESTE1.	**NOME DO PROGRAMA. TESTE1.**
AUTHOR. FULANO DE TAL.	**AUTOR. FULANO DE TAL.**
DATE-WRITTEN. NOVEMBRO/2009.	**DATA EM QUE FOI ESCRITO. NOVEMBRO/2009.**
DATE-COMPILED.	**DATA EM QUE FOI COMPILADO.**
INSTALATION. EMPRESA ACME.	**INSTALAÇÃO. EMPRESA ACME.**
SECURITY. CADASTRO DE TESTE APENAS.	**COMENTÁRIO.**
REMARKS. EXEMPLO DO CURSO – NOV/2009.	**COMENTÁRIO.**

Elemento obrigatório:

- PROGRAM-ID (o nome do programa deve ter no máximo 8 carateres)

Elementos opcionais:

- AUTHOR
- DATE-WRITTEN
- DATE-COMPILED
- INSTALATION
- SECURITY
- REMARKS

NOTAS: Cada elemento acima é delimitado por um ponto, assim como os respetivos dados.

Síntaxe:

```
[Mandatory]     IDENTIFICATION DIVISION.
[Mandatory]     PROGRAM-ID. NameOfProgram.
[Optional]      AUTHOR. NameOfProgrammer.
[Optional]      INSTALLATION. Development-center.
[Optional]      DATE-WRITTEN. mm/dd/yy.
[Optional]      DATE-COMPILED. mm/dd/yy. HH:MM:SS.
[Optional]      SECURITY. Program-type.
```

ENVIRONMENT DIVISION

É a divisão de ambiente, onde se especifica o método de expressão dos aspetos do processamento de dados que dependem das caraterísticas físicas do computador. E pode ser dividida em duas secções:

CONFIGURATION SECTION

INPUT-OUTPUT SECTION

Síntaxe:

```
ENVIRONMENT DIVISION.
[Optional]      [CONFIGURATION SECTION.
[Optional]      [SOURCE-COMPUTER. {source-computer-entry}]
[Optional]      [OBJECT-COMPUTER. {object-computer-entry}]
[Optional]      [SPECIAL-NAMES.   {special-names-entry}]]

[Optional]      [INPUT-OUTPUT SECTION.
[Optional]      [FILE-CONTROL.    {file-control-entry}]
[Optional]      [I-O-CONTROL.     {i-o-control-entry}]]
```

CONFIGURATION SECTION - Trata das caraterísticas do computador fonte e do computador objeto. É subdividida em 3 parágrafos (cláusulas) opcionais:

- **SOURCE-COMPUTER**: descreve a configuração na qual é gerado o código intermediário do programa, após compilação.
- **OBJECT-COMPUTER**: descreve a configuração na qual o programa será executado.
- **SPECIAL-NAMES**: Pode ser utilizado para definir carateres especiais dos alfabetos, ou qual o caráter para definir o ponto decimal, ou caraterísticas como o caráter representativo de moeda.

Por exemplo:

DECIMAL POINT IS COMMA : define que o separador decimal será a vírgula (,) e não o ponto (.), que é o padrão.

CÓDIGO:
```
CONFIGURATION SECTION.
    SPECIAL-NAMES. DECIMAL POINT IS COMMA.
```

INPUT-OUTPUT SECTION - Trata com todas as informações necessárias para controlar a transmissão e manipulação de dados entre meios externos e o PROGRAMA e possui dois **parágrafos** **FILE-CONTROL e I-O CONTROL**:

FILE-CONTROL. Dá os nomes aos ficheiros do programa e associa-os aos meios externos, inclui gravar, excluir e ler ficheiros. Não é obrigatório estar no programa desde que não se realize nenhuma operação com ficheiros físicos.

No parágrafo FILE-CONTROL usamos uma instrução SELECT para cada arquivo descrito. A sintaxe da instrução SELECT é a seguinte:

```
SELECT nome-do-arquivo ASSIGN TO dispositivo-do-computador.
```

No exemplo abaixo mostramos uma ENVIRONMENT DIVISION de um programa que irá aceder a um ficheiro CLIENTES.

```
*
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SPECIAL-NAMES.
           DECIMAL-POINT IS COMMA.
      *
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT CLIENTES ASSIGN TO DA-S-CLIENTES.
      *
```

Neste exemplo escolhemos como *nome-do-arquivo* dentro da instrução SELECT a palavra CLIENTES. O programador pode usar qualquer palavra com até 30 carateres como *nome-de arquivo*, mas uma vez definido um *nome-de-arquivo* na instrução SELECT, deverá usar sempre este nome no programa quando se referir a este arquivo.

O formato da cláusula *dispositivo-do-computador* varia conforme o computador (micro, mainframe, etc), mas no caso do mainframe usa-se o formato mostrado no exemplo, composto de 3 segmentos separados por hífen:

Tipo de dispositivo –

UR - para dispositivos de registro fixo (impressoras, cartão).
UT - para dispositivos de registro variável (fitas).
DA - para dispositivos de acesso aleatório (discos).

Modo de acesso –

S – Sequencial.
D – Direto (Random).
I – Indexado.

Nome externo do arquivo –

Nome pelo qual o operador do computador reconhece o arquivo. O nome externo geralmente está associado aos comandos JCL na execução do programa.

Para arquivos abertos para leitura (veremos OPEN INPUT e OPEN I-O mais adiante), pode-se especificar a clausula OPTIONAL no SELECT. Com esta clausula, se o ficheiro a abrir não existir, ele é automaticamente criado vazio. A sintaxe do SELECT fica então:

SELECT OPTIONAL nome-do-ficheiro ASSIGN TO dispositivo

I-O CONTROL. Este parágrafo define técnicas de controlo especiais a serem usadas pelo programa (não é utilizado com frequência porque o dispositivo usado – fita magnética caiu em desuso)

DATA DIVISION

Todos os dados com que o programa vai trabalhar são definidos nesta divisão.

Aqui na **DATA DIVISION**, são armazenadas todas as informações em tempo de execução do programa.

Variáveis, conteúdo de ficheiros físicos e estrutura de base de dados ficam armazenadas aqui.

Esta divisão é dividida em 4 secções:

FILE SECTION. SEÇÃO DE FICHEIRO

WORKING-STORAGE SECTION. SEÇÃO DE ARMAZENAMENTO DE TRABALHO

LINKAGE SECTION. SEÇÃO DE VÍNCULO, CONEXÃO, LIGAÇÃO.

SCREEN SECTION. SEÇÃO DE ECRÃ (a deixar para mais tarde).

- **FILE SECTION** Aqui são declaradas as estruturas dos ficheiros que são incluídos em **file-control**. Vamos abordar essa secção mais detalhadamente quando passarmos a trabalhar com arquivos de dados.
- **WORKING-STORAGE SECTION**

  Nesta secção serão definidas as variáveis e as constantes que serão utilizadas no programa. Estas variáveis somente existirão em memória RAM, pois não serão armazenadas em nenhum outro local.

  Exemplo:

```
DATA DIVISION.
    WORKING-STORAGE SECTION.
        01 DataNascimento.
            02 Ano.
                03 Seculoant PIC 99.
                03 AnoNasc PIC 99.
            02 MesNasc PIC 99.
            02 DiaNasc PIC 99.
```

- **LINKAGE SECTION**

  É semelhante à **working-storage section**, porém é utilizada para passar parâmetros de um programa para o outro.

  As variáveis são declaradas aqui da mesma forma que são na working-storage section, mas sem o valor inicial declarado.

  Nessa secção não devemos declarar constantes, apenas variáveis.

Conceitos da linguagem que são utilizados nessa divisão:

a) CONSTANTES FIGURATIVAS

São palavras reservadas e referem-se a valores constantes específicos. Algumas dessas palavras são descritas abaixo. Pode-se usar tanto o plural quanto o singular[1]. Não deve ser escrita entre apóstrofos, para não ser considerada uma constante não numérica:

ZERO, ZEROS, ZEROES Representa o valor numérico "zero" ou uma ou mais ocorrências do caráter 0. Dependendo do contexto pode também ser numérico ou não numérico.

Exemplo:

```
MOVE ZERO   TO DATA-NAME1
MOVE ZEROS  TO DATA-NAME1
```

SPACE, SPACES Representa um ou mais espaços, tratados como um item alfanumérico.

Exemplo:

[1]Por exemplo, se DATA-NAME-1 é um item de dados de cinco carateres, cada uma das instruções seguintes move 5 espaços para DATA-NAME-1:

```
MOVE SPACE       TO  DATA-NAME-1
MOVE SPACES      TO  DATA-NAME-1
MOVE ALL SPACES  TO  DATA-NAME-1
```

HIGH-VALUE, HIGH-VALUES. Representa um ou mais carateres com valores altos (representação em hexadecimal do maior valor na sequência de carateres na representação EBCDIC). Usualmente é o hexadecimal "FF". A constante figurativa HIGH-VALUE é tratado como um literal não numérico.

Exemplo:

```
MOVE HIGH-VALUE TO DATA-NAME1
```

LOW-VALUE, LOW-VALUES. Representa um ou mais caracteres com valores baixos. Usualmente é o binário 0 (representação em hexadecimal do menor valor na sequência de caracteres na representação EBCDIC –tabela da IBM de codificação de carateres). A constante figurativa LOW-VALUE é tratado como um literal não numérico.

Exemplo:

```
MOVE LOW-VALUE TO  DATA-NAME1
MOVE LOW-VALUES TO DATA-NAME1
```

ALL *Literal* Representa um conjunto de carateres pré definido que não deve ser numérico. Quando uma constante figurativa é usada, a palavra ALL é redundante.

Exemplo:

```
MOVE ALL ' '      TO DATA-NAME1
MOVE SPACES       TO DATA-NAME1
MOVE ALL '*'      TO DATA-NAME1
```


POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Europeu de Desenvolvimento Regional

QUOTE, QUOTES. Utiliza-se esta constante quando é necessário imprimir o apóstrofo. O apóstrofo é o carater delimitador de strings. O QUOTES substitui o apóstrofo por outro caráter delimitador, deixando-o livre para ser usado como uma string.

NULL, NULLS Representa o valor numérico "zero". Também representa um endereço inválido de memória quando usado em conjunto com tipos de dados POINTER.

b) DEFINIÇÃO DE DADOS

As regras para nomes de variáveis são:

- No máximo 30 carateres;
- Pode começar com um número, mas precisa ter pelo menos uma letra;
- Não podem ser utilizadas palavras reservadas da linguagem;
- Pode conter apenas os carateres de "A" até "Z", de "a" a "z", de "0" até "9" , o hífen "-", o travessão baixo (underscore) "_" e nenhum outro tipo de caráter é permitido;
- Não pode terminar por um hífen "-";
- Não se podem usar espaços nem carateres acentuados.

Nota: FILLER é um nome genérico atribuído a campos que não têm necessidade de serem especificados, pois não serão utilizados no processamento (como por exemplo campo nulo).

Hífen usado como sinal de subtração: os operadores matemáticos devem, obrigatoriamente, estar separados por espaços das variáveis que constituem os seus operados, caso contrário, gera um erro de compilação (a-b ✗ a - b ✓)

- Quando declarares uma variável/constante, procura ser simples e objetivo. Evita o uso de nomes complicados e que não se assemelhem aos objetos que queres apresentar, pois pode-se não conseguir entender o que representam no programa ou qual sua função.

c) TIPO DE DADOS

Tipos de dados usados nas instruções:

- Numérico (sinalizado ou não, com decimal ou inteiros);
- Alfabético;
- Alfanumérico;
- Sinal operacional;
- Ponto decimal assumido;
- Constantes figurativas.

REGRAS NA UTILIZAÇÃO DOS TIPO DE DADOS

NUMÉRICOS (constantes ou variáveis)

- Ter no máximo 18 dígitos (ou seja, só aguenta um número até 18 bytes);
- Pode colocar-se um sinal de mais (+) ou menos (-) à esquerda do número (ocupando um byte a mais);
- Um ponto decimal, ou vírgula, pode ser colocado no meio. Este ponto decimal não pode ser o último caráter do número (é apenas uma convenção, porque pode declarar um número real sem casas decimais depois da vírgula, sendo a mesma coisa que declarar um inteiro).

Exemplos: 1528, 1528.95, -1528, +1528

NÃO NUMÉRICO OU ALFANUMÉRICOS

- Pode conter no máximo 160 carateres, incluindo espaços;
- Deve utilizar apóstrofos (‘) ou aspas (“), para delimitar o literal;
- Pode-se colocar, dentro do literal, qualquer caráter, menos o caráter delimitador (as aspas ou apóstrofos).

Exemplos: “Anabela”, “1528”, “-1528”, “1528.95”

d) CONDIÇÕES DE CLASSE

A condição de classe é usada para testes, onde se deseja saber se uma variável é formada ou não por um tipo particular de dados.

NUMERIC Numérico, carateres de 0 a 9 (sinalizado ou não, como decimal ou inteiro).
ALPHABETIC Alfabético, carateres de A - Z, de a – z e espaços.
ALPHABETIC-UPPER Alfabético, carateres de A - Z, e espaços.
ALPHABETIC-LOWER Alfabético, carateres de a - z, e espaços.

Síntaxe:

```
IF VARIAVEL IS [NOT] {NUMERIC } {ALPHABETIC } {ALPHABETIC-UPPER}
{ALPHABETIC-LOWER}      {nome-de-
classe }
```

Exemplo:

```
IF NOME IS NOT ALPHABETIC-UPPER
    DISPLAY MESSAGE BOX “INSIRA APENAS DE A – Z E
ESPAÇOS” END-IF
```

e) Cláusula PICTURE

PICTURE ou PIC é uma cláusula na linguagem COBOL que descreve o conteúdo de cada dado elementar, permitindo a descrição do seu tamanho e do seu tipo. Uma “picture clause” é feita de vários caracteres de formato.

Um item de dado elementar é aquele dado indivisível para o programa. Quando é necessário agrupar vários itens elementares define-se um item de grupo.

A tabela a seguir apresenta os carateres de formato:

Símbolo	Descrição
9	Qualquer dígito numérico
A	Caráter alfabético ou espaço (A-Z ou a-z ou " ")
B	Separador de campos - caráter de inserção de espaço
E	Expoente de ponto flutuante
G	Caráter gráfico ou alfanumérico double-byte
N	Caráter double-byte
S	Sinal operacional assumido (n.º com sinal) -/+
X	Qualquer caráter (alfanumérico, símbolo, qualquer coisa)
Z	Dígito numérico, mas suprimindo zeros à esquerda (substitui o 9) – só funciona se o valor dado for inteiro
*	Imprime um asterisco por cada zero à esquerda. Funciona como símbolo de proteção de valores
$	Cifrão - Símbolo da moeda
.	Ponto decimal - separador de casas decimais
/	Caráter de inserção de barras (/)
P/V	Ponto decimal assumido (.)
+	Sinal positivo
-	Sinal negativo
,	Vírgula - separador de dígitos
CR	Indicador de sinal ('CR' se negativo, em branco se positivo) – Crédito
DB	Indicador de sinal ('DB' se negativo, em branco se positivo) – Débito
0	(zero)

Exemplos:

1. **Tipo alfanumérico:** PIC XXXXXXX = PIC **X**(7) indica um campo alfanumérico de sete posições.

2. **Tipo numérico:** PIC 9999 = PIC **9**(4) indica um campo numérico (qualquer dígito de 0 a 9) de quatro posições.

3. PIC 9(n), com n < 18.

4. PIC 999**V**99 indica que após a terceira posição de um dado numérico existe um ponto decimal **implícito**.

5. PIC **S**9(4) indica que o número possui sinal (+ ou -).

6. Exemplos de pictures de edição com o que será apresentado ao utilizador:

Picture	Valor inserido	Valor exibido	Picture	Valor exibido
S9(5)v99	0035488	+35488.00	ZZZZ9.99	35488.00
9(7)v99	654321	0654321.00		
9(5)v99	0135862	35862.00		
9(8)	12052009	12052009	99/99/9999	12/05/2009
9(14)	76080738001069	76080738001069		
S9(3)V9	-56	-056.0		
$9(4).99	005698.7	5698.70	$9(4)V99	$ 569870
9(5)	98763	98763	9099099	9087063
A(6)	"ABACAXI"	ABACAX		
A(8)	"ABACAXI"	ABACAXI		

f) Cláusula VALUE (Valor)

É uma cláusula opcional que é usada para inicializar os itens de dados, quando o programa é executado. Os valores podem ser numérico (inteiro ou com casas decimais), alfanumérico, ou constante figurativa.

g) NÍVEIS DE DADOS

A declaração das variáveis no COBOL é feita utilizando níveis hierárquicos. Devem ser utilizados pela ordem crescente.

Podemos criar pequenos (ou grandes) grupos de dados dentro da estrutura do programa.

O **nível mais baixo** é o nível **01**. Todos os grupos devem ser iniciados a partir dele. O nível **mais alto** é o nível **49**. Existem porém outros níveis a considerar. Níveis existentes no Cobol:

01 a 49 – Definição de itens de estruturas.
66 – Itens para renomeação
77 – Itens elementares, não fazem parte de nenhuma hierarquia
78 – Constante
88 – Nomes de condição

Os **níveis** são usados para descrever a hierarquia existente entre os itens de grupo e os elementares, e mostram quais os itens elementares que compõem cada item de grupo:

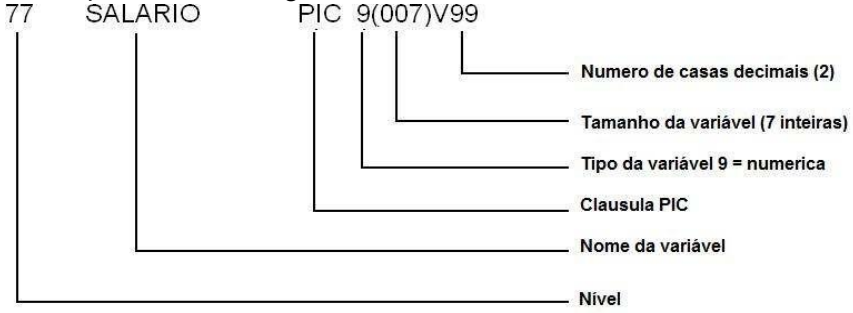
- **Itens elementares** . Número do nível, nome dos dados, cláusula imagem e valor (opcional) são usados para descrever um item elementar.
- **Os itens do grupo** consistem em um ou mais itens elementares. Número do nível, nome dos dados e cláusula Valor (opcional) são usados para descrever um item de grupo e cujo número do nível é sempre 01.

O exemplo a seguir mostra Grupo elementar e itens:

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WS-NAME     PIC X(25).                          ---> ELEMENTARY ITEM
01 WS-CLASS    PIC 9(2)  VALUE  '10'.              ---> ELEMENTARY ITEM
01 WS-ADDRESS.                                     ---> GROUP ITEM
   05 WS-HOUSE-NUMBER    PIC 9(3).                 ---> ELEMENTARY ITEM
   05 WS-STREET          PIC X(15).                ---> ELEMENTARY ITEM
   05 WS-CITY            PIC X(15).                ---> ELEMENTARY ITEM
   05 WS-COUNTRY         PIC X(15)
```

Para descrever os dados em COBOL podemos ter a seguinte estrutura:

- Nível
- Nome
- PIC
- Tipo
- Tamanho
- Decimal
- Casas decimais
- Valor inicial



e outras definições (OCCURS, EXTERNAL, USAGE "comp, comp-3, comp-1"...)

Níveis e Subníveis de Variáveis (Hierarquia dos dados)

Nível	O1	O2	O3
	DataNasc	AnoNasc	Seculo
			Ano
		MesNasc	
		DiaNasc	

```
01 DataNasc.
   02 Ano.
       03 SeculoAnt  PIC99.
       03 AnoNasc    PIC99.
   02 MesNasc        PIC99.
   02 DiaNasc        PIC99.
```

DataNasc							
Ano				MesNasc		DiaNasc	
seculoAnt		AnoNasc					
1	9	4	5	1	2	2	5

Modelo de memória para os itens de dados declarados no exemplo

Exemplo 1:

```
WORKING-STORAGE SECTION.
77 SALARIO            PIC 9(007)V99.
77 NOME               PIC X(030).
77 SALDO              PIC S9(010)V99.
77 INDICE             PIC 9(007)V9(008).
01 DATA-DE-HOJE.
     05  ANO          PIC 9(004).
     05  MES          PIC 9(002).
     05  DIA          PIC 9(002).
```

```
SALARIO : 0000000.00
NOME    : .000000
SALDO   : +0000000000.00
INDICE  : 0000000.00000000
ANO     : 0000
MES     : 00
DIA     : 00
```

Exemplo2:

```
01 WS-NUM1 PIC S9(3)V9(2).
01 WS-NUM2 PIC PPP999.
01 WS-NUM3 PIC S9(3)V9(2) VALUE -123.45.
01 WS-NAME PIC A(6) VALUE 'ABCDEF'.
01-WS-ID   PIC X(5) VALUE 'A121$'.
```

```
WS-NUM1 : +000.00
WS-NUM2 : .000000
WS-NUM3 : -123.45
WS-NAME : ABCDEF
WS-ID : A121$
```

Exemplo3:

```
01 ws-data.
   03 ws-dia pic 9(002) value 11.
   03 filler     pic x(001) value "/".
   03 ws-mes pic 9(002) value 02.
   03 filler pic x(001) value "/".
   03 ws-ano pic 9(004) value 2018.
```

Se mandasse imprimir o identificador **ws-data** no ecrã, aparecia: **11/02/2018**
Se mandasse imprimir apenas o identificador **ws-ano**, ficaria: **2015**

3. Podemos criar um grupo maior e mais organizado:

```
01 ws-trabalho.
   03 ws-data-sistema.
      05 ws-dia                 pic 9(002) value 11.
      05 filler                 pic x(001) value “/”.
      05 ws-mes                 pic 9(002) value 02.
      05 filler                 pic x(001) value “/”.
      05 ws-ano                 pic 9(004) value 2018.
   03 filler              pic x(001) value spaces.
   03 ws-utilizador-sistema.
      05 ws-nome-utilizador     pic x(006) value “Maria”.
      05 filler                 pic x(001) value spaces.
      05 ws-senha-utilizador    pic 9(006) value 123456.
```

Imprimindo o identificador **ws-trabalho** teremos: **11/02/2018 Maria 123456**

4.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
DATA DIVISION.
   WORKING-STORAGE SECTION.
   01 WS-NUM1 PIC 99V9 VALUE IS 3.5.
   01 WS-NAME PIC A(6) VALUE 'ABCD'.
   01 WS-ID PIC 99 VALUE ZERO.
```

```
WS-NUM1 : 03.5
WS-NAME : ABCD
WS-ID   : 00
```

Todos os dados declarados acima possuem valores pré-definidos.

Os valores destas variáveis podem ser mudados a qualquer momento, dentro do programa.

Níveis Especiais

66 – renames

Semelhante ao comando REDEFINES, pode-se utilizá-lo para dar outro nome a uma variável, sem redefinir sua estrutura ou modificar suas propriedades. Qualquer alteração na variável original em tempo de execução, ocorrerá também neste identificador 66.

Exemplo:

```
01 ws-trabalho.
   03 ws-nome-utilizador          pic x(010) value spaces.
   66 ws-espelho-utilizador   renames ws-nome-utilizador.
```

Atribuindo “Maria Luísa” a **ws-nome-utilizador**, irá atribuir também a **ws-espelho-utilizador** a mesma string. Não utilizamos com frequência este recurso no COBOL, pois a cláusula REDEFINES é mais completa e mais útil.

77 – standalone

É utilizado para variáveis que não fazem parte de nenhum grupo, por isso o nome “standalone”.

Exemplo:

```
   77 ws-nome-utilizador    PIC x(010) value spaces.
```


POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Europeu de Desenvolvimento Regional

78 – constante

Este nível também é "standalone" e não pode ser declarado dentro de um grupo. Quando o programa é executado, a constante recebe um valor inicial e não pode ser alterado. Podemos declarar números inteiros, decimais, strings…

Exemplos:

```
78 c-nome-utilizador     value "Maria Luísa".
78 c-codigo-operadora value 47.
78 PI value 3.14159.
78 c-salario-base     value 995.50.
```

Usando o exemplo acima, podemos usar a constante **c-nome-utilizador** para declarar outra variável, utilizando como valor inicial de **ws-nome-utilizador** o conteúdo de **c-nome-utilizador**.

Exemplo:

```
77 ws-nome-utilizador    pic x(010) value c-nome-utilizador.
```

88 – condição

Definimos um identificador nível 88 quando queremos testar um valor TRUE ou FALSE:

```
01 ws-tipo-utilizador   pic 9(001) value zeros.
   88 ws-tipo-utilizador-master value 1.
   88 ws-tipo-utilizador-comum  value 0.
```

Quando **ws-tipo-utilizador** for igual a 1, **ws-tipo-utilizador-master** será TRUE, **ws-tipo-utilizador-comum** será FALSE (e vice versa). Se o valor for diferente das 2 condições declaradas, ambas serão FALSE.

Este nível é utilizado em itens de grupo, por exemplo:

```
01 FormaPagamento PIC 9(1) VALUE zeros.
    88 Cartao VALUE 1.
    88 Dinheiro VALUE 2.
    88 Cheque VALUE 3.
```

Na situação inicial desta declaração, Cartão, Dinheiro e Cheque têm valor FALSE, devido ao value zeros de FormaPagamento.
Quando queremos que o programa identifique qual a opção de pagamento desejada pelo cliente basta mover 1 (por exemplo) para FormaPagamento, ficando Cartão igual a TRUE e Dinheiro e Cheque iguais a FALSE.
Se FormaPagamento for alterado para 3, a variável Cheque ficará com o valor TRUE e Cartão e Dinheiro ficarão com FALSE.
Pode-se utilizar também um comando do compilador chamado **SET** (atribui um valor a uma referência em COBOL) , por exemplo:

```
SET CARTÃO TO TRUE.
```

Nomes condicionais com múltiplos valores

THROUGH/THRU indica que cada nível pode abranger uma faixa de valores para se tornar verdadeiro. Executa uma serie de valores, dados o 1,º e último valores da sequência.

Sintaxe:

```
88 nome_da_condicao { VALUE/VALUES} { literal low values { THROUGH/THRU } High-values}
```

Exemplo 1:

```
77 HORA      PIC 9(002).
88 HORA-OK VALUES 0 THRU 23.
IF NOT HORA-OK
      DISPLAY MESSAGE BOX “HORA INVALIDA !!”
END-IF
```

Exemplo 2:

```
…
DATA DIVISION.
WORKING-STORAGE SECTION.
01 InputChar PIC X.
       88 Vogal     VALUE "A","E","I","O","U".
       88 Consoante VALUE "B" THRU "D", "F","G","H"
                          "J" THRU "N", "P" THRU "T"
                          "V" THRU "Z".
       88 Digit VALUE "0" THRU "9".
       88 ValidChar VALUE "A" THRU "Z", "0" THRU "9".
PROCEDURE DIVISION.
BEGIN.
DISPLAY "Insira um caráter:- " WITH NO ADVANCING
ACCEPT InputChar            *> accept recebe um valor do teclado a atribuir à variável InputChar
IF ValidChar
       DISPLAY "Input OK!"
ELSE
       DISPLAY "Entrada de caráter inválido!"
END-IF
IF Vogal
       DISPLAY "Entrada de vogal"
END-IF
IF Digit
       DISPLAY "Entrada de dígito"
END-IF
STOP RUN.
```

Output 1:

```
Insira um caráter:-g
Entrada de caráter
inválido!"
```

Output 2:

```
Insira um caráter:-5
Input OK!
Entrada de dígito
```

Output 3:

```
Insira um caráter:-E
Input OK!
Entrada de vogal
```

Regras Básicas

Devemos obedecer algumas regras para declarar as variáveis.

- Utilizar acentuação nos identificadores. Dependendo do compilador, não ocorrerá erro na compilação, porém isso está fora do padrão da escrita COBOL.

- Não utilizar carateres especiais nos identificadores.

78 constante-não-ok value “N”. ✗

- Não utilizar espaços nos identificadores.

78 constante nao ok value “N”. ✗

- Quando declarares uma variável/constante, procura ser simples e objetivo. Evita o uso de nomes oblíquos e complicados, pois pode-se não conseguir entender o que representam no programa ou qual sua função.

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Europeu de Desenvolvimento Regional

- Os programadores utilizavam siglas no início de cada variável/constante, para facilitar a procura destes elementos dentro dos programas. Esta regra foi assumida e muito bem aceite, pois deixa o código fonte mais organizado e legível.

- *WORKING-STORAGE SECTION*, utiliza-se o prefixo **ws-**

```
01 ws-nome-aluno pic x(010) value spaces.
```

- *LINKAGE SECTION*, utiliza-se o prefixo **lnk-**

```
01 lnk-nome-aluno pic x(010).
```

- *LOCAL-STORAGE SECTION*, utilizamos o prefixo **ls-**

```
01 ls-nome-aluno pic x(010) value spaces.
```

Exercícios:

1. Tendo em conta as seguintes linhas de código indica o seu output:
   a) **`01 VALOR pic 9(005)v99      value 456.89.`**

   b) **`01 VALOR pic ZZZZ9.99.`**

   c) **Suponhamos ser introduzido o valor 76080738001069.**
      c.1) **`01  COD   pic 9(014).`**

      c.2) **`01  COD   pic ZZ,ZZZ,ZZ9/9999B99.`**

Bibliografia/webgrafia:
https://mainframesupport.wordpress.com/2012/07/15/figurative-constants/
http://www.mainframestechhelp.com/tutorials/cobol/
Beginning COBOL for Programming, Michael Coughlan, Editora Apress
Linguagem Cobol, Marcio Adroaldo da Silva em www.controlsyst.com
Mainframe Apostila de Cobol, G &P Treinamentos em http://www.csis.ul.ie/cobol/
https://www.apostilando.com/apostila/2962/manual-pratico-de-programacao-em-cobol
https://www.tutorialspoint.com/cobol/
https://www.apostilando.com/apostila/2638/apostila-de-cobol

Compilador de cobol online: https://www.jdoodle.com/execute-cobol-online
IDE para instalação: **https://launchpad.net/cobcide/+download**

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Europeu de Desenvolvimento Regional