Formação: COBOL

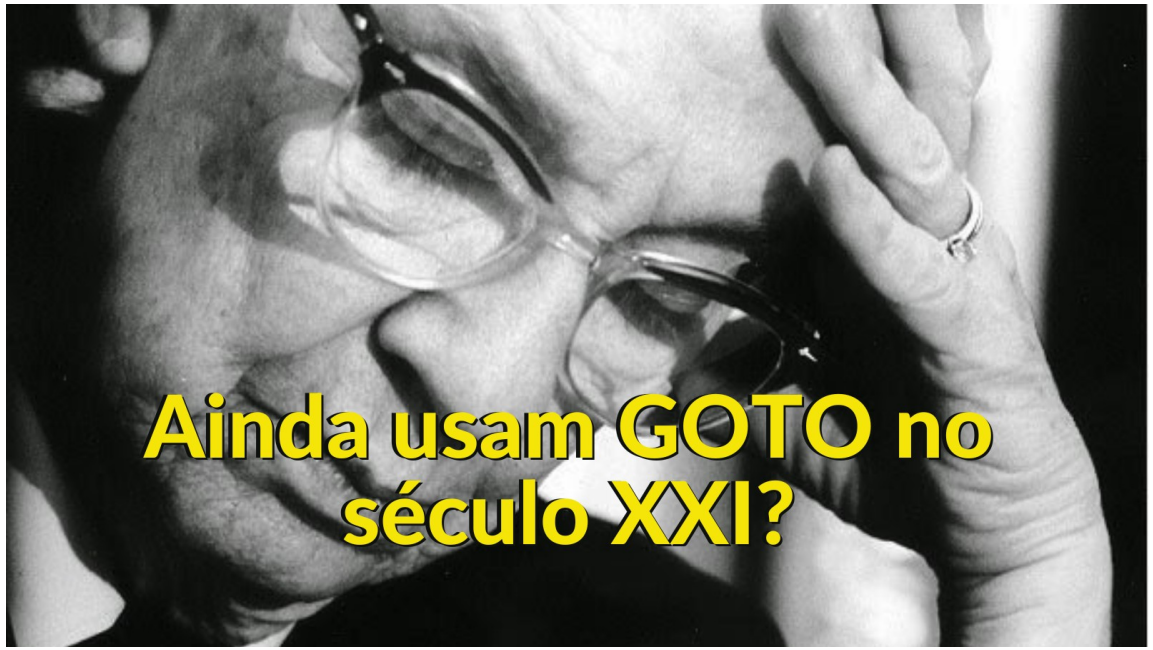
Professor: Vagner Bellacosa

Disciplina: Linguagem de Programação COBOL

COBOL

Modulo 06.02 – Variáveis modulo intermediário

1) GOTO
2) Procedure Division
3) Regras de Negócios
4) EBCDIC
5) Bits e Bytes
6) TSO - HEX
7) Mascara de Edição
8) Carácter ASA e os relatórios
9) Clausulas COMP
10) Numéricos COMP
11) Numéricos COMP-1
12) Numéricos COMP-2
13) Numéricos COMP-3
14) Numéricos COMP-4
15) Numéricos COMP-5



COBOL

```
GO TO   100-SAY-HELLO.
GO TO   200-GET-DATE.
GO TO   300-SAY-GOODBYE.
```

Sintaxe: GO TO <seção, paragrafo, label>

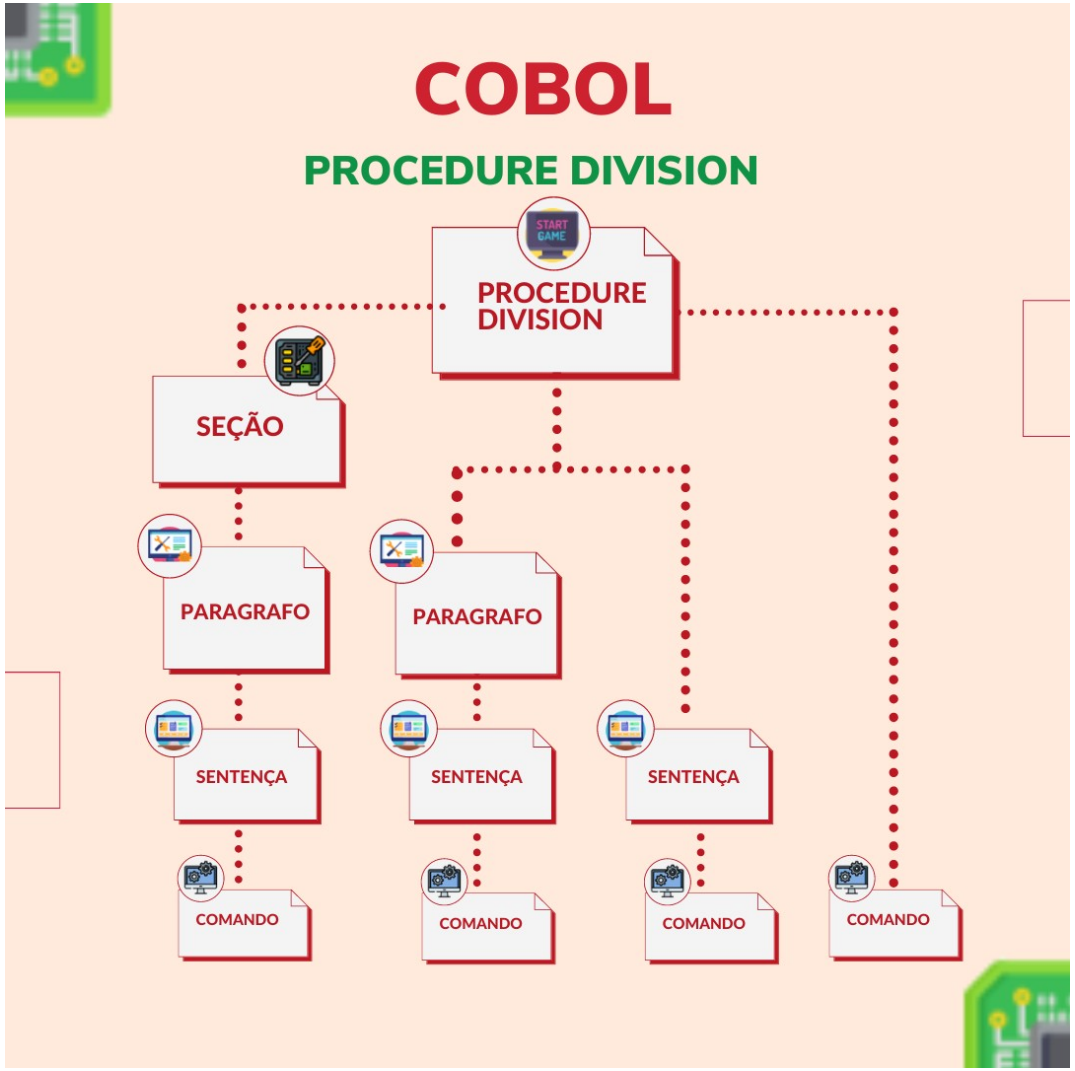
Comando GO TO

Aqui apresentado apenas para entender os programas Legados, famoso por criar os monstrengos, que tiram os cabelos dos programadores.

Associado ao Spaghett Code, ou programas longos e com saltos sem razão aparente, dificultando o Debug e as manutenções evolutivas e corretiva.

Não usem esse comando.

COBOL



Comando GO TO

Para entender sua sintaxe, devemos relembrar o conceito de Seção, Label e parágrafos.

Para nos situarmos estamos na **PROCEDURE DIVISION**.

Que é tem estruturas opcionais SECTIONs, Parágrafos, Sentenças e Comandos, uso destas estruturas ajudam a estruturar o programa para atender as regras do negocio.

COBOL

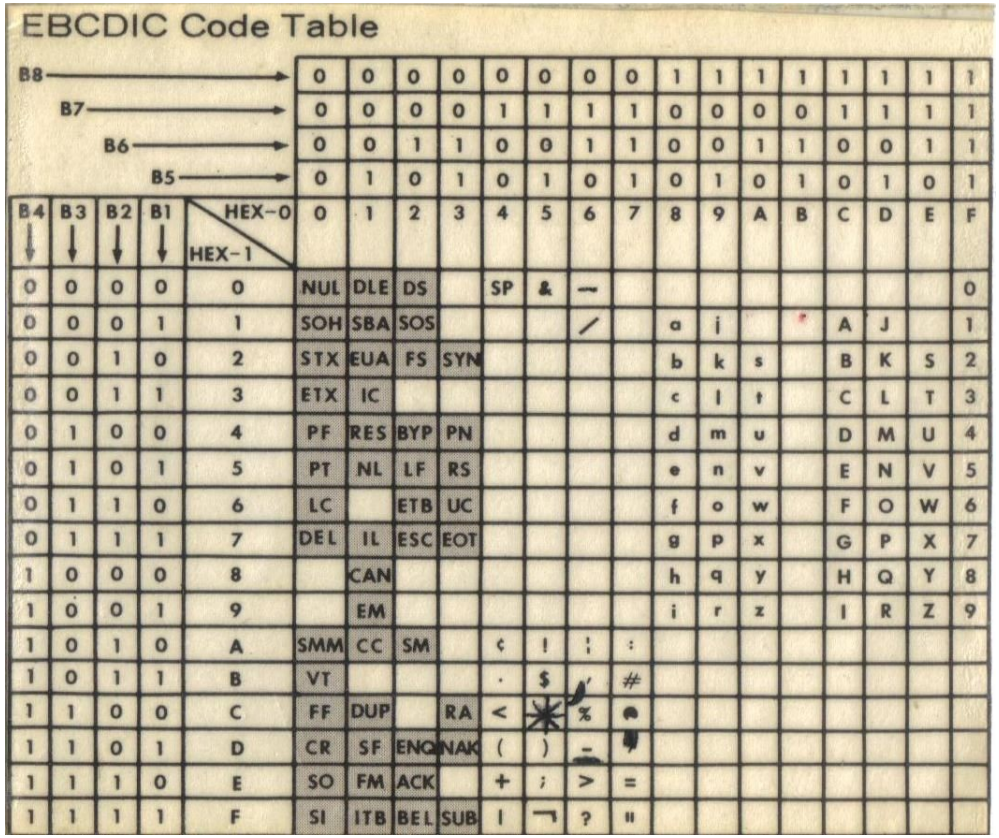
Regras de Negocio.

No decorrer das atividades muitas vezes vamos ouvir falar regras funcionais, lembrando que como programadores, nossa função é atender uma necessidade do usuário.

Essas necessidades surgem do surgimento de um novo produto, uma mudança legislativa, atender as necessidades dos stake holders e criar novas funcionalidades.

Criando manutenção evolutivas.

COBOL

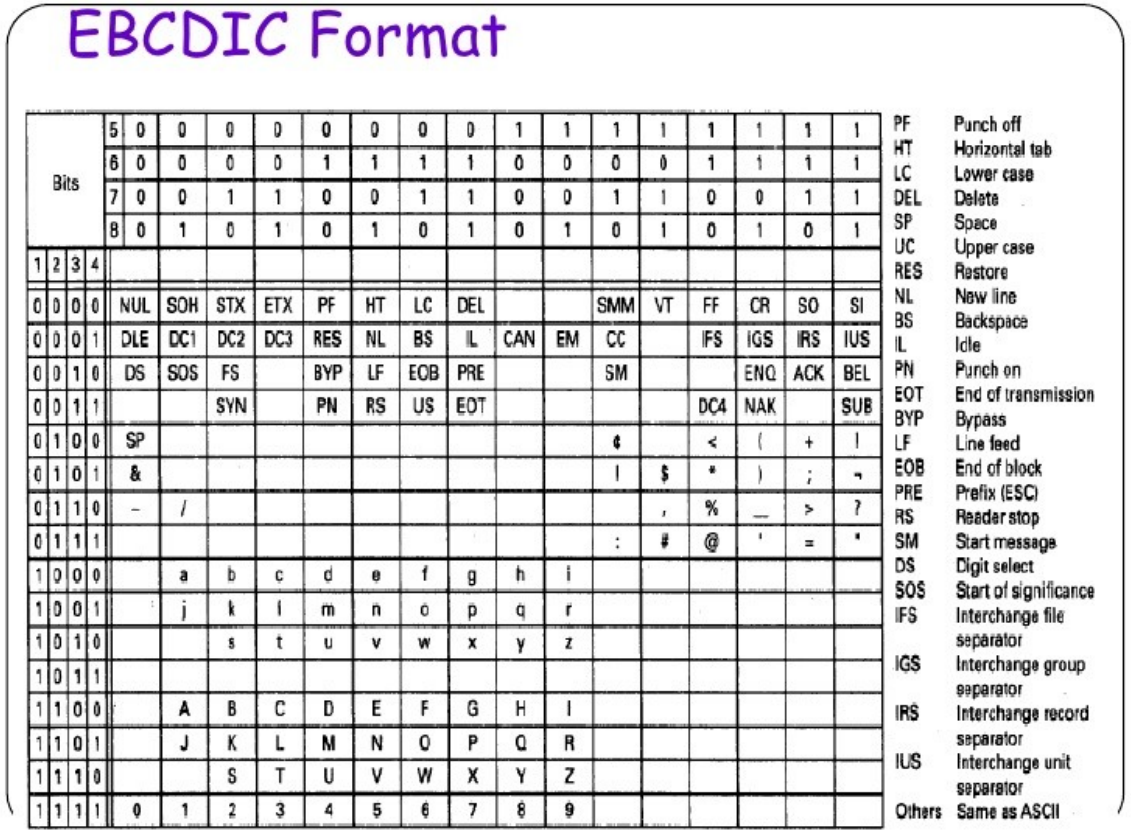


EBCDIC

Comentamos inúmeras vezes sobre a Tabela EBCDIC - Extended Binary Coded Decimal Interchange Code, traduzindo para a língua de Camões, pelo instituto tabajara, seria algo como Código de intercâmbio decimal codificado em binário estendido.

Em português simples, é a maneira que a IBM criou para humanos falarem com maquina. Onde cada caractere usado em processamento, tem seu equivalente em Hexadecimal e Binário

COBOL



EBCDIC Format

Bits				5	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
				6	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				7	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				8	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	2	3	4																	
0	0	0	0	NUL	SOH	STX	ETX	PF	HT	LC	DEL			SMM	VT	FF	CR	SO	SI
0	0	0	1	DLE	DC1	DC2	DC3	RES	NL	BS	IL	CAN	EM	CC		IFS	IGS	IRS	IUS
0	0	1	0	DS	SOS	FS		BYP	LF	EOB	PRE			SM			ENQ	ACK	BEL
0	0	1	1			SYN		PN	RS	US	EOT					DC4	NAK		SUB
0	1	0	0	SP										¢		<	(	+	|
0	1	0	1	&										!	$	*	)	;	¬
0	1	1	0	-	/										,	%	_	>	?
0	1	1	1											:	#	@	'	=	"
1	0	0	0		a	b	c	d	e	f	g	h	i						
1	0	0	1		j	k	l	m	n	o	p	q	r						
1	0	1	0			s	t	u	v	w	x	y	z						
1	0	1	1																
1	1	0	0		A	B	C	D	E	F	G	H	I						
1	1	0	1		J	K	L	M	N	O	P	Q	R						
1	1	1	0			S	T	U	V	W	X	Y	Z						
1	1	1	1	0	1	2	3	4	5	6	7	8	9						

PF	Punch off
HT	Horizontal tab
LC	Lower case
DEL	Delete
SP	Space
UC	Upper case
RES	Restore
NL	New line
BS	Backspace
IL	Idle
PN	Punch on
EOT	End of transmission
BYP	Bypass
LF	Line feed
EOB	End of block
PRE	Prefix (ESC)
RS	Reader stop
SM	Start message
DS	Digit select
SOS	Start of significance
IFS	Interchange file separator
IGS	Interchange group separator
IRS	Interchange record separator
IUS	Interchange unit separator
Others	Same as ASCII

EBCDIC

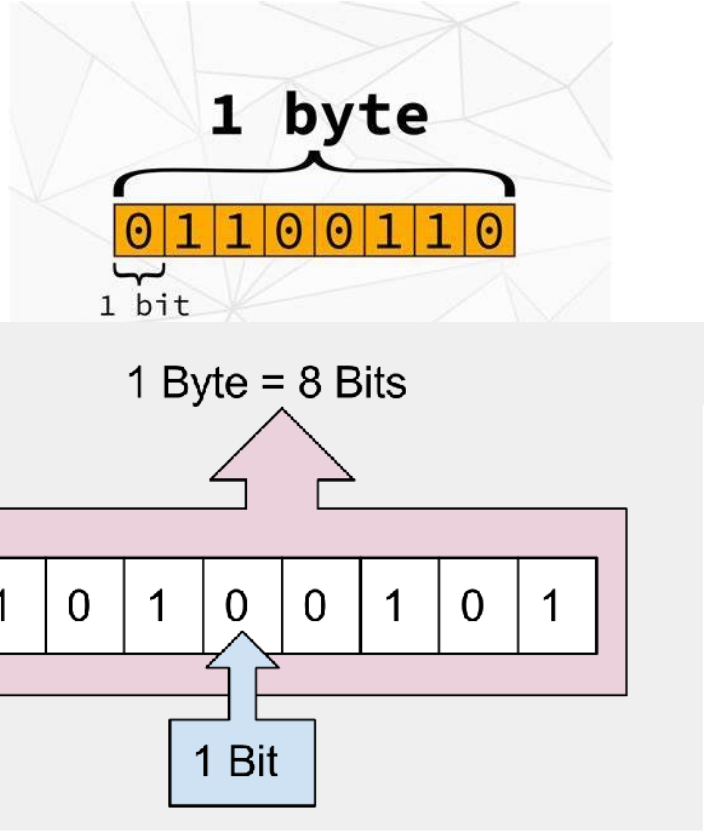
Usando meus poderes de clarividência, vejo surgir muitas perguntas, mas para que preciso aprender isso. WTF?

Nesta tabela vemos inúmeros comandos, tais comando TAB, New Line, Enter etc. E serve exatamente para isso, codificar e processar informações.

Mais adiante iremos ver os números e sua transformação nas variáveis PIC e COMP.

COBOL

Bits e Bytes



Resumão

Um bit é a menor unidade de valor processado em computadores de qualquer tipo. Traduzido pelo estado ligado/desligado. Eis a magia da computação.

Um grupo de oito bits, formam uma informação de alto nível chamada de byte.

O byte irá armazenar toda a informação necessária para conversar com humanos.

COBOL

Bits e Bytes

```
EDIT       KC02746.WORKBOOK.COBOL(HEXADECM) - 01.09      Member HEXADECM saved
Command ===> _____________________________________________ Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 ********************************************************************
000002 *** EBCDIC - EXEMPLO DE VALORES EM HEXADECIMAL                      ***
000003 ********************************************************************
000004
000005 A B C D E F G H I J
000006
000007 K L M N O P Q R S T
000008
000009 U V W X Y Z
000010
000011 0 1 2 3 4 5 6 7 8 9
****** **************************** Bottom of Data ****************************
```

```
 ----------------------------
 000005 A B C D E F G H I J
        C4C4C4C4C4C4C4C4C4D44
        10203040506070809010
 ----------------------------
```

```
 ----------------------------
 000007 K L M N O P Q R S T
        D4D4D4D4D4D4D4D4E4E4
        20304050607080902030
 ----------------------------
```

```
 ----------------------------
 000011 0 1 2 3 4 5 6 7 8 9
        F4F4F4F4F4F4F4F4F444
        00102030405060708090
 ----------------------------
```

Resumão

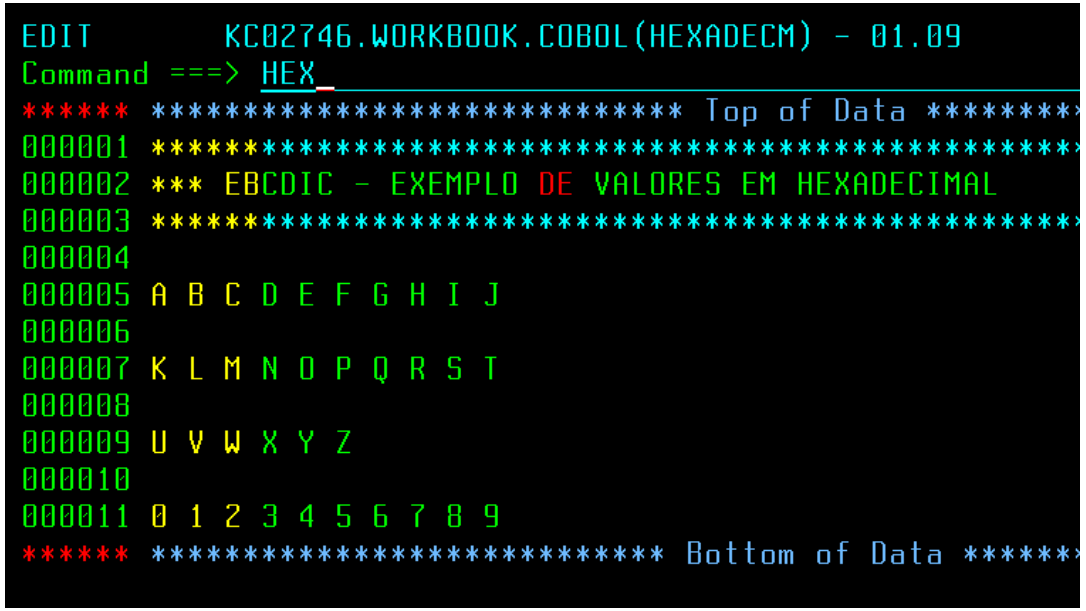
Toda letra, número e comandos elementares são armazenados num byte.

Estruturas mais complexa, usam N bytes, a titulo de exemplo temos o Kilobyte 102 bytes e o Megabyte 1024 kbs, mas isso é assunto para outra conversa.

Existem 256 combinações possíveis de bytes, chegaram onde eu queria. Exato a Tabela EBCDIC e mesmo a ASCII são as tabelas que armazenam o DE PARA.

TSO

Comando HEX



Alterna o modo de exibição na IDE do TSO para o formato Hexadecimal, assim conseguimos visualizar o valor real do byte, no registro.

Serve para depurar e editar massa de testes, em CICS e DB2, muitos erros inexplicáveis ocorrem por sujeira de byte em meio há uma string.

Sintaxe : **HEX [ON] [OFF]**

COBOL

Variáveis – uso de mascara de edição

```
000021 * MASCARA EDICAO NUMERICA                    EXEMPLO
000022   10 REL-COD-MASK            PIC ZZZ          123
000023   10 REL-COD-PT-BR           PIC Z.ZZ9        1.900
000024   10 REL-COD-DEC-PT-BR       PIC Z.ZZ9,99     543,55
000025   10 REL-COD-BLOQ            PIC *99,99       *77,77
000026   10 REL-COD-SPACE           PIC 9B9          6 6
000027   10 REL-COD-FAKE            PIC 99,00        88,00
000028   10 REL-COD-SIGNAL1         PIC +ZZ9         +876
000029   10 REL-COD-SIGNAL2         PIC 99+          56-
000030   10 REL-COD-SIGNAL3         PIC 99+          56+
000031   10 REL-COD-SIGNAL4         PIC +99          +56
000032   10 REL-COD-SIGNAL5         PIC -99          -56
000033   10 REL-COD-SIGNAL6         PIC 99-          56-
000034   10 REL-COD-SIGNAL7         PIC -99          56
000035   10 REL-COD-MONEY           PIC $9(4)        $8765
000036
```

Visto em detalhes no Tópico 002-002

Conceito

Quando estamos processando dados, nossa missão é informar ao usuario num front-end, o resultado do procedimento, seja um calculo, uma atualização ou mesmo uma simples consulta.

Em sistemas mainframe existem comandos que formatam as variaveis, de modo a torná-las legíveis e entendíveis aos seres humanos. Seja na tela do emuladores 3270, num aplicativo WEB ou Mobile, bem como em documentos impressos.

COBOL

Relatórios impressos

Caracteres ASA

```
000001 ************************************************
000002 *** COMANDO WRITE PARA RELATORIO
000003 ************************************************
000004
000005 *** PULA LINHAS ANTES OU DEPOIS
000006        WRITE REGISTRO [FROM NOME-DE-DADO]
000007            [{BEFORE ADVANCING INTEIRO LINES}]
000008            [{AFTER ADVANCING INTEIRO LINES}].
000009
000010 *** PULA PAGINAS ANTES OU DEPOIS
000011        WRITE REGISTRO
000012           [{BEFORE/AFTER ADVANCING } PAGE].
000013
```

Muito comum quando fazemos programas de geração de relatórios, necessitamos de comandos para avançar linhas, blocos e paginas.

São conhecidos como caractere ASA, de American Standart Association, neste momento estou apresentando o conceito e o seu uso das pictures.

Um relatório usa marcação especial no Working Storge Section e clausulas especiais no comando **WRITE.**

COBOL

Relatórios impressos

Caracteres ASA

Personagem ASA	Ação	Equivalente ASCII
em branco	Avance 1 linha antes de imprimir (espaçamento simples)	CR LF
1	Avançar para a próxima página antes de imprimir (alimentação de formulário)	CR FF
2–9, A, B, C	Avançar para parada de tabulação vertical ou canal de fita de controle de carro	CR VT (aproximadamente)
0	Avance 2 linhas antes de imprimir (espaçamento duplo)	CR LF LF
-	Avance 3 linhas (espaçamento triplo)	CR LF LF LF
+	Não avance nenhuma linha antes de imprimir, sobreponha a linha anterior com a linha atual	CR

Aqui chegamos onde desejo, oculto no arquivo gerado, existem comandos de 1 byte, que indicam os saltos da impressora.

O primeiro caractere de um registro, que é o byte de controle do carro da impressora, pode conter um byte ASCII ou um byte de máquina.

Necessário conhecer para evitar aborrecimentos na emissão dos relatórios e trabalhos com output de informação.

COBOL

Variáveis tipo COMPUTACIONAL

COMP ou Binary

```
VARIAVEL NUMERICA COMPACTADA
001
 A-COMP-1  :  .12345677E 17 LENGTH    : 000000004
002
 B-COMP-2  :  .12345678901234567E 17 LENGTH    : 000000008
003
 C-COMP-3  : 345678901234567800 LENGTH    : 000000010
004
 D-COMP-4  : 567890123456780000 LENGTH    : 000000008
005
 E-COMP-5  : 14614248257738682176 LENGTH    : 000000008
```

Quando precisamos trabalhar com números muito grandes, com sinais e casas decimais, ou mesmo, notação cientifica.

O COBOL fornece uma clausula para a definição de variáveis numéricas no programa. Usando COMP economizamos espaço em memoria e tempo de processamento, com menos ciclos de conversão, pois numéricas computacional são mais velozes em cálculos e movimentação, pois não são convertidas.

COBOL

Variáveis tipo BINARY

COMP ou Binary

```
PIC S9(4) USAGE IS BINARY
PIC S9(9) USAGE IS BINARY
PIC S9(18) USAGE IS BINARY
PIC 9(4) USAGE IS BINARY
PIC 9(9) USAGE IS BINARY
PIC 9(18) USAGE IS BINARY
```

Em algumas situações usaremos a notação **USAGE IS BINARY**, para trabalhar com 2 bytes (halfword), 4 bytes (fullword) e 8 bytes (doubleword).

São variáveis extramente velozes e utilizadas em caso de performance e melhoria, bem como cálculos com números científicos e de precisão.

Este conceito necessita bom entendimento da Arquitetura Mainframe e suas instruções em 32/64 bits, funcionamento de calculo e o uso do MIBS.

COBOL

Variáveis tipo BINARY

COMP-1

Recebe valores em formato de ponto flutuante e precisão simples, ocupando 4 bytes de armazenamento na memoria.

Aceitando variáveis com numeração positiva e negativa, sendo que o bit mais esquerda conterá o sinal e os restantes o numero.

COMP-1 não utiliza a clausula PIC.

```
05 COMPUTE-RESULT     USAGE COMP-1   VALUE 06.23E-24.
```

COBOL

Variáveis tipo BINARY

Picture	Storage representation	Numeric values
S9(1) through S9(4)	Binary halfword (2 bytes)	-32768 through +32767
S9(5) through S9(9)	Binary fullword (4 bytes)	-2,147,483,648 through +2,147,483,647
S9(10) through S9(18)	Binary doubleword (8 bytes)	-9,223,372,036,854,775,808 through +9,223,372,036,854,775,807
9(1) through 9(4)	Binary halfword (2 bytes)	0 through 65535
9(5) through 9(9)	Binary fullword (4 bytes)	0 through 4,294,967,295
9(10) through 9(18)	Binary doubleword (8 bytes)	0 through 18,446,744,073,709,551,615

COMP-2

Recebe valores em formato de ponto flutuante e precisão dupla, ocupando 8 bytes de armazenamento na memoria.

Aceitando variáveis com numeração positiva e negativa, sendo que o bit mais esquerda conterá o sinal e os restantes o numero.

Uso interno de ponto flutuante

COMP-2 não utiliza a clausula PIC.

COBOL

Variáveis tipo BINARY

HEX EBCDIC
F0 = 0
F1 = 1
F2 = 2
F3 = 3
F4 = 4
F5 = 5
F6 = 6
F7 = 7
F8 = 8
F9 = 9

COMP-3

Equivalente ao **PACKED-DECIMAL**

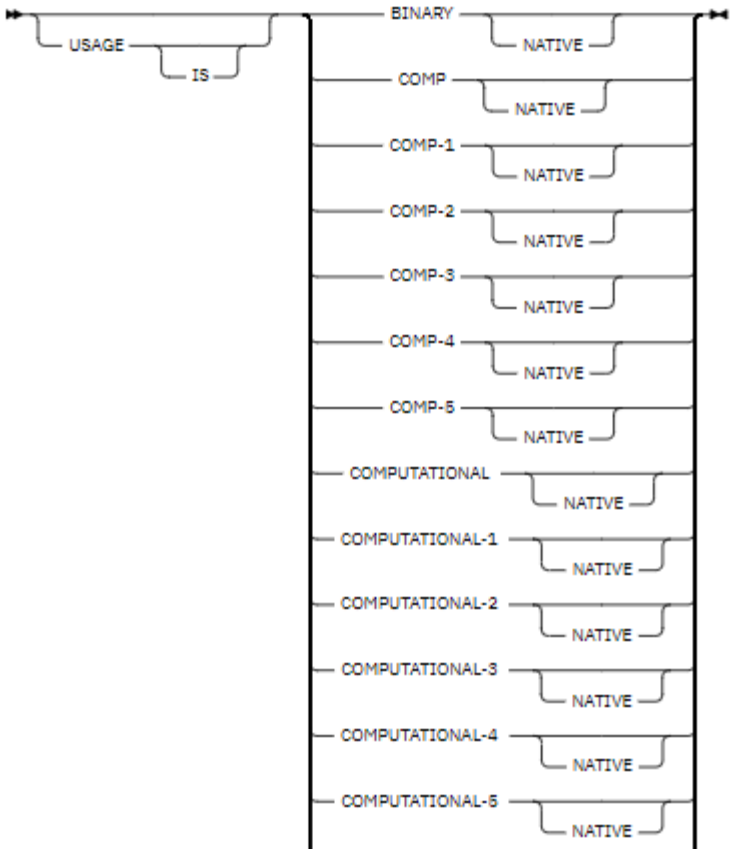
Este formato é o mais usado em programação COBOL, devido a sua versatilidade e rapidez em cálculos, para entendermos melhor, utilizando esta clausula em um único byte colocamos 2 dígitos do numero, economizando espaço e com melhoria de performance.

PIC S9 (7) COMP-3.
Quantidade de bytes = (1 + 7) / 2 = 8 / 2 = 4 bytes.

Detalhe ao visualizarmos um numero comp-3, o ultimo bit a direita, conterá o sinal, sendo C ou F para positivo e D para negativo.

COBOL

Variáveis tipo computacional



COMP-4

Equivalente ao **COMP**

Usado em cálculos científicos e de precisão, não se assuste, a não ser que trabalhes em automação bancaria, CICS ou NASA.

Raramente vera estes formatos em uso, é suficiente saber que otimizará programas gastadores de recurso, economizará espaço de armazenamento em disco e memoria.

COBOL

Variáveis tipo COMP-5 Phrase

COMP-5

Utilizado para armazenamento de dados binários na tabela abaixo veja a economia de espaço e o tamanhos de números comportados.

Table 1. Storage Representation for COMP-5 Data Items

Picture	Storage representation	Numeric values
S9(1) through S9(4)	Binary halfword (2 bytes)	-32768 through +32767
S9(5) through S9(9)	Binary fullword (4 bytes)	-2,147,483,648 through +2,147,483,647
S9(10) through S9(18)	Binary doubleword (8 bytes)	-9,223,372,036,854,775,808 through +9,223,372,036,854,775,807
9(1) through 9(4)	Binary halfword (2 bytes)	0 through 65535
9(5) through 9(9)	Binary fullword (4 bytes)	0 through 4,294,967,295
9(10) through 9(18)	Binary doubleword (8 bytes)	0 through 18,446,744,073,709,551,615

COBOL

Clausula USAGE

PIC editado

```
05 Price           Pic     9(5)v99.
05 Edited-price-D  Pic  $99,999.99
       Blank When Zero.
05 Edited-price-N  Pic  $99,999.99 Usage National
       Blank When Zero.
. . .
Move 0 to Price
Move Price to Edited-price-D
Move Price to Edited-price-N
Display Edited-price-D
Display Edited-price-N upon console
```

Em relatórios em algumas situações necessitamos exibir espaço em branco ou zero, quando a variavel não possui valor inicial.

A clausula **USAGE** auxilia nessa situação, vamos testar?

A clausula **UPPON CONSOLE** direciona a saída do display.

COBOL

Bônus MIPS

Muitas vezes nos corredores da empresa ou em reuniões, ouviram o anacronimo MIPS, mas afinal o que isso quer dizer?

Seu amigo professor te ajudará: Milhões de Instruções por Segundo, Millions of Instructions Per Second. É uma medida de desempenho do mainframe.

Avalia uma carga de trabalho
Densidade de I/O.
Uso do Sistema Operacional
Uso de Partição de Dados
Uso de Memoria etc...

COBOL

Variáveis aula prática

Programas COBOL014 até COBOL018

Para aprofundarmos estes conhecimentos, iremos treinar com os programas exemplos.

Atente no tamanho das variáveis

Experimente mover números inteiros e decimais.

Experimente números negativos e positivos.

Boa exploração.

COBOL

Duvidas????