1 – Introdução

COBOL (acrônimo de “Common Business-Oriented Language” – Linguagem comum orientada a negócios) é uma das linguagens de programação mais antigas, pertencendo à segunda geração das linguagens de programação. É muito utilizada em aplicações voltadas para o mundo financeiro, devido à sua precisão e rapidez na aritmética de ponto flutuante.

2 – História

A primeira especificação COBOL foi desenvolvida e apresentada na primeira metade de 1959 por Grace Hopper, almirante da marinha americana e uma das pioneiras no desenvolvimento de linguagens de programação, durante um evento de informática na Universidade da Pensilvânia (EUA). Após este evento, o departamento de defesa americano decidiu patrocinar e assumir o projeto, criando três comitês (de curto, médio e longo prazo) para criar uma especificação oficial para a linguagem, a fim de viabilizar o uso de uma linguagem comum voltada para negócios.

O comitê de curto prazo era formado por três agências governamentais e seis fabricantes de computadores. No fim, um subcomitê derivado deste, composto de apenas seis pessoas, desenvolveu e apresentou a especificação oficial COBOL em dezembro de 1959. Essa especificação foi, em grande parte, inspirada em outras linguagens como a FLOW-MATIC (desenvolvida anos antes também por Grace Hopper), a COMTRAN da IBM (desenvolvida por Bob Berner) e a FACT , da Honeywell.

O comitê de médio prazo foi formado, mas não foi utilizado e o de longo prazo não chegou a ser criado.

Em 1960, foram implementados os primeiros compiladores de COBOL. Em 6 e 7 de dezembro de 1960, o mesmo programa COBOL foi rodado em dois computadores diferentes (um da RCA e um Univac da Remington), provando assim a compatibilidade da linguagem.

Desde 1959, várias versões de COBOL foram desenvolvidas. Em 1968, o “American National Standards Institute” (ANSI – Instituto Americano de Padrões Nacionais) desenvolveu uma nova especificação, a fim de resolver problemas de incompatibilidades entre as diferentes versões de COBOL e os sistemas que as utilizavam. Mais duas versões de COBOL foram desenvolvidas pelo ANSI, em 1974 e 1985.

Em 1991, os padrões de COBOL passaram a ser desenvolvidos pela “International Organization for Standardization” (ISO – Organização Internacional de Normalização),

que lançou a especificação COBOL 2002, além de várias emendas para implementar novas funcionalidades e corrigir problemas. Uma revisão completa da especificação COBOL está sendo feita pela ISO e deve ser aprovada como nova especificação padrão para a linguagem em 2010.

3 – Controvérsias

> COBOL
> /koh'bol/, n.
>
> [Common Business-Oriented Language] (Synonymous with evil.) A weak, verbose, and flabby language used by code grinders to do boring mindless things on dinosaur mainframes. Hackers believe that all COBOL programmers are suits or code grinders, and no self-respecting hacker will ever admit to having learned the language.
> Its very name is seldom uttered without ritual expressions of disgust or horror. One popular one is Edsger W. Dijkstra's famous observation that "The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offense." (from Selected Writings on Computing: A Personal Perspective) See also fear and loathing, software rot.

COBOL é uma linguagem muito controversa no mundo da informática. Embora seja muito boa para executar operações em lote (batch operation), sua estrutura não permite o desenvolvimento de aplicações interativas e nem voltadas para web. Perante os desenvolvedores é considerada uma linguagem obsoleta e inferior, um vestígio da idade média da computação. Então por que simplesmente a linguagem não morre?

A resposta para isso é simples: o COBOL hoje em dia é utilizado para escrever programas que permitem a reutilização e suporte de sistemas antigos em novos mainframes, eliminando a necessidade de desenvolver novos sistemas e cortando custos com isso. Ao migrar programas COBOL para Windows, Unix ou sistemas distribuídos, os programas permanecem em COBOL porque é a solução mais barata e menos arriscada para a integridade e confiabilidade do sistema.

4 –COBOL como linguagem de programação

COBOL é uma linguagem simples e inteiramente procedural, que não possui suporte a diversos recursos que as atuais linguagens de alto nível possuem como variáveis locais, recursividade, alocação dinâmica de memória e programação estruturada ou orientada a objeto. Seu aprendizado é difícil, já que, por não possuir bibliotecas, possui um número extenso de palavras reservadas. Nas revisões sofridas pela linguagem, muitas dessas funções foram implementadas.

5 – Estrutura básica de um programa COBOL

Desta seção em diante, utilizaremos o COBOL-85 como padrão.

Um programa COBOL consiste em quatro divisões básicas:

- IDENTIFICATION DIVISION: É a parte do programa na qual se fornece informações sobre o programa, tais como nome do programa, autor, comentários do autor e informações de uso para o usuário final.

- ENVIROMENT DIVISION: É a parte que faz a interface do programa com o ambiente operacional, possibilitando a obtenção e gravação de dados em arquivos em uma unidade de armazenamento físico.

- DATA DIVISION: É a seção na qual se declaram as variáveis, as constantes e todos os tipos de dado que irão alocar memória no decorrer do processo.

- PROCEDURE DIVISION: É a seção na qual se colocam os procedimentos que manipulam os dados contidos na *DATA DIVISION*. Esta divisão possui uma estrutura hierárquica dividida em seções, parágrafos, sentenças e comandos. Todo programa COBOL deve conter pelo menos um parágrafo, uma sentença e um comando.

5.1 – Parágrafos da IDENTIFICATION DIVISION:

Obrigatório:

PROGRAM-ID. (nome do programa). – É o parágrafo onde o nome do programa é especificado com, no máximo, oito posições alfanuméricas (regra válida apenas para mainframes antigos).

Opcionais:

AUTHOR. (nome). – É onde se especifica o nome do autor do programa.

DATE-WRITTEN. (texto). – É onde se deve indicar a data em que o programa foi escrito.

DATE-COMPILED. (texto). – É onde se deve indicar a data em que o programa foi compilado.

SECURITY. (texto). – É onde se dá as informações relativas à segurança e acesso do programa.

REMARKS. (texto). – É onde se faz comentários sobre o uso do programa e se dá informações relevantes para o utilizador.

5.2 – Seções e parágrafos da ENVIROMENT DIVISION:

Seção 1 – CONFIGURATION SECTION:

SOURCE-COMPUTER. (nome). – Identifica em qual computador o programa foi feito.

OBJECT – COMPUTER. (nome). – Identifica o computador no qual o programa está.

SPECIAL-NAMES. (comandos). – É onde se especifica comandos que atribuem características como sinal monetário, tipo de ponto decimal e caracteres simbólicos necessários para o bom funcionamento da aplicação.

Seção 2 – INPUT-OUTPUT SECTION:

FILE-CONTROL. (comandos). – É onde se especificam os arquivos que o programa irá acessar.

I-O-CONTROL. (comandos). – É onde é especificado como o programa deve receber as informações por outros meios de entrada, como cartões perfurados.

Lembrando que todos estes parágrafos devem ser obrigatoriamente especificados se o programa fizer uso de algum recurso que os exija.

5.3 – Seções da DATA DIVISION:

Seção 1 – FILE SECTION:

Aqui são definidas todas as variáveis e constantes utilizadas pelo programa e que realmente serão gravadas em arquivo.

Seção 2 – WORKING-STORAGE SECTION:

Aqui são definidas todas as variáveis e constantes utilizadas pelo programa que serão apenas utilizadas em tempo de execução (tais como variáveis utilizadas para armazenar valores temporários e operadores lógicos).

Lembrando que todas estas seções devem ser obrigatoriamente especificadas se o programa fizer uso de algum recurso que as exija.

6 – Variáveis em COBOL

COBOL não possui tipos definidos de variáveis. Todas as variáveis em COBOL são expressas em "Picture clause".

6.1 – Picture clause

"Picture clause" é um elemento em programação utilizado para descrever um tipo de dado, utilizando caracteres simples que indicam o tipo de item contido na variável e o seu tamanho.

Uma "picture clause" é feita de vários caracteres de formato, cada um dos quais representa uma porção dos dados. Cada caractere de formato pode se repetir ou vir acompanhado de um número indicativo de repetição, o qual especifica o número de vezes que o item formatado ocorre na variável de dados.

A tabela a seguir apresenta os caracteres de formato:

Caractere	Descrição
A	Caractere alfabético (A-Z, a-z ou em branco)
B	Caractere em branco (espaço)
CR	Indicador de sinal ('CR' se negativo, em branco se positivo) – Crédito
DB	Indicador de sinal ('DB' se negativo, em branco se positivo) – Débito
E	Expoente de ponto flutuante
G	Caractere gráfico ou alfanumérico double-byte
N	Caractere double-byte
S	Sinal implícito (não exibido)
V	Ponto decimal implícito (não exibido)
X	Qualquer caractere (alfanumérico, símbolo, qualquer coisa)
Z	Dígito numérico, mas suprimindo zeros à esquerda
0	Dígito '0'
9	Qualquer dígito numérico
/	Caractere '/'
,	Separador de dígitos
.	Separador de casas decimais
+	Sinal ('-' se negativo, '+' se positivo)
-	Sinal ('-' se negativo, em branco se positivo)
$	Símbolo da moeda
*	Preenchedor de dígitos

Assim, podemos declarar uma variável qualquer como "PIC 999", indicando que a variável em questão irá conter um valor de 3 dígitos quaisquer ou "PIC X(8)" indicando que a variável receberá qualquer string de até 8 caracteres.

6.2 – Declaração de variáveis

Todas as variáveis e constantes utilizadas devem ser declaradas dentro da *DATA DIVISION* e nas seções corretas, de acordo com as suas finalidades no programa.

Toda variável deve ser declarada da seguinte forma:

```
<número de nível>    <nome da variável>    <picture clause>.
```

6.2.1 – Números de nível

Em COBOL, é possível hierarquizar as variáveis, a fim de facilitar a vida dos usuários do sistema. Os números de nível vão de 01 a 49 e os números 77 e 88 são níveis especiais que permitem um uso diferenciado para as variáveis.

OBS: Variáveis que servem como raiz para outras variáveis que não tenham nível 77 ou 88 NÃO possuem "picture clause".

Um exemplo:

```
01      REGISTRO.       (não possui picture clause porque é nível raiz)
        02          NOME        PIC A(10).
        02          TELEFONE    PIC 9999999.
```

O nível 88 é utilizado para especificar os chamados "condition-names". Um "condition-name" é utilizado para mostrar e especificar um valor específico ou uma faixa de valores específicas para uma determinada variável. Para entender com uma maior facilidade, acompanhe a seguinte declaração de variável:

```
01      CHAVE.          PIC 9.
        88          CHAVE-ON    VALUE 1 THRU 9.
        88          CHAVE-OFF   VALUE ZERO.
```

Em caso de necessidade de se trabalhar com a variável *"CHAVE"* no programa, podemos apenas trabalhar apenas com as condições, facilitando assim a leitura do código do programa. Por exemplo, uma instrução *"IF CHAVE = 0"* tem o mesmo efeito de uma instrução *"IF CHAVE-OFF"*. Essa é uma maneira de emular uma estrutura variável booleana em COBOL.

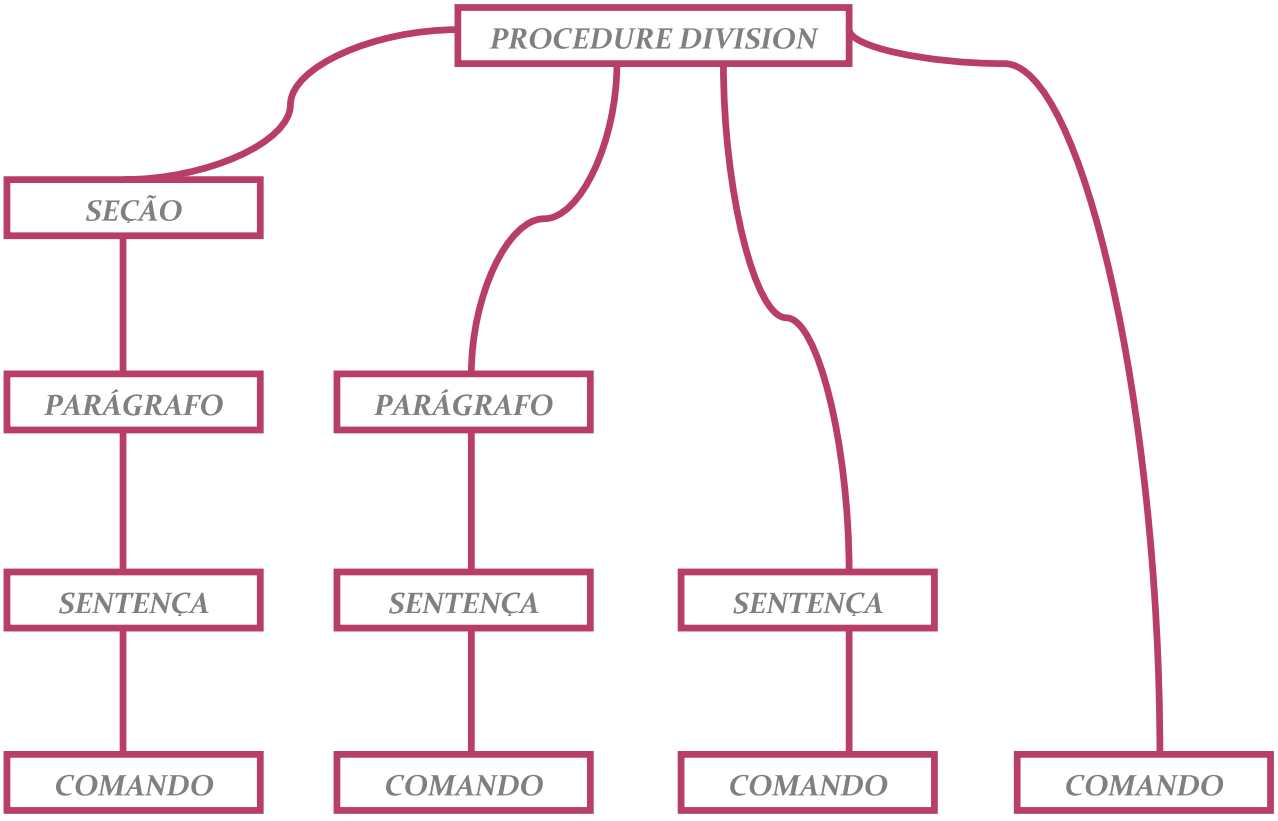
7 – Sintaxe COBOL

COBOL é uma linguagem que apresenta uma sintaxe simples baseada em palavras reservadas que executam ações específicas (conhecidas também como verbos COBOL).

7.1 – A PROCEDURE DIVISION

Como já foi dito anteriormente, essa é a parte do programa responsável por realizar as operações necessárias para o funcionamento do programa e é dividida em seções, parágrafos, sentenças e comandos.

Hierarquicamente, a *PROCEDURE DIVISION* se divide de acordo com o organograma abaixo:



Uma seção contém pelo menos um parágrafo, que por sua vez contém pelo menos uma sentença e esta, por sua vez, contém um ou mais comandos. Um programa COBOL deve conter pelo menos um comando.

7.1.1 – Seções

Uma seção é um trecho do código que contém um ou mais parágrafos. Cada seção é identificada por um nome seguido pela palavra reservada *SECTION*. Uma seção termina apenas no início da próxima seção e duas seções não podem ser identificadas pelo mesmo nome.

7.1.2 – Parágrafos

Um parágrafo é um trecho do código que contém uma ou mais sentenças. Cada parágrafo é identificado por um nome seguido da palavra reservada *PARAGRAPH*. Um parágrafo termina apenas no início do próximo parágrafo ou seção. Dois parágrafos só podem ter o mesmo identificador se estiverem em parágrafos diferentes.

7.1.3 – Sentenças

Uma sentença é um trecho do código que contém um ou mais comandos. Pode se estender por mais de uma linha e é terminada por um ponto final.

7.1.4 – Comandos

Um comando contém apenas um verbo COBOL, um sujeito e um nome de objeto opcional. Exemplos: *"MOVE A TO B"* ou *"PERFORM E"*.

Podemos executar determinados trechos do programa com os seguintes comandos:

PERFORM <seção>
PERFORM <parágrafo>
PERFORM <parágrafo 1> THRU <parágrafo 2>
GO TO <parágrafo>

7.2 – Estrutura condicional "IF"

A estrutura condicional *"IF"* do COBOL consiste no seguinte trecho de código:

```
IF <condição>
      <sentença>.
ELSE
      <sentença>.
END-IF.
```

Podemos aninhar várias estruturas condicionais da seguinte maneira:

```
IF <condição>
      IF <condição>
            <sentença>.
      ELSE
            <sentença>.
      END-IF.
ELSE
      <sentença>.
END-IF.
```

Lembrando que, em COBOL:

1 – Todo "IF" precisa ter um "END-IF".
2 – Um "IF" não precisa ter um "ELSE".
3 – Um "ELSE" vale pro último "IF" sem "ELSE" dentro de um "END-IF" dentro do ninho.

7.3 – Estrutura condicional "EVALUATE WHEN"

O comando *"EVALUATE WHEN"* é o comando equivalente no COBOL para a estrutura condicional *"switch"* de outras linguagens. Ela representa uma alternativa mais simples para a criação de um ninho de instruções "IF". Consiste no seguinte trecho de código:

```
EVALUATE <variável>
WHEN <condição 1>       <sentença>.
WHEN <condição 2>       <sentença>.
WHEN <condição 3>       <sentença>.
END-EVALUATE.
```

7.4 – Estrutura de repetição "PERFORM UNTIL"

A principal estrutura de repetição do COBOL é dada pelo comando *"PERFORM UNTIL"*. Sua sintaxe básica é dada por:

```
PERFORM VARYING <variável> FROM <valor1> TO <valor2> UNTIL
```

O parâmetro *"VARYING <variável> FROM <valor 1> TO <valor 2>"* permite que a estrutura se comporte como uma estrutura *"for"* de outras linguagens. A omissão do mesmo faz com que a estrutura se comporte como uma estrutura de repetição *"do while"*.

7.5 – Outros comandos básicos

- *DISPLAY:* É responsável pela exibição de texto para o usuário do sistema.

- *ACCEPT X FROM Y:* É responsável pela entrada de dados para uma variável.

- *MOVE X TO Y:* É o comando que atribui valores a uma variável.

- *ADD X TO Y:* Adiciona um valor a uma variável.

- *SUBTRACT X FROM Y:* Subtrai um valor de uma variável.

- *GO TO/PERFORM:* Executa um trecho específico do programa.

- *COMPUTE:* É utilizado quando o usuário precisa fazer cálculos matemáticos com uma maior complexidade.

8 – Exemplos de programa:

8.1 – Hello, World!

```
IDENTIFICATION DIVISION.
PROGRAM ID. Hello-world.          // Define o nome da aplicação
PROCEDURE DIVISION.
     DISPLAY "Hello, World!"      // Imprime na tela a string
"Hello, World!"
     STOP RUN.               // Fim do programa.
```

Esse é o menor programa possível em COBOL.

8.2 – 99 Bottles of Beer

```
IDENTIFICATION DIVISION.
       PROGRAM-ID. 99-Bottles-of-Beer-On-The-Wall.
       AUTHOR. Joseph James Frantz.
DATA DIVISION.
       WORKING-STORAGE SECTION.
       01 Keeping-Track-Variables.
          05 Bottles                      PIC S99   VALUE 0.
          05 Remaining-Bottles            PIC S99   VALUE 0.
          05 Counting                     PIC 99    VALUE 0.
          05 Start-Position               PIC 99    VALUE 0.
          05 Positions                    PIC 99    VALUE 0.
       PROCEDURE DIVISION.
       PASS-AROUND-THOSE-BEERS.
        PERFORM VARYING Bottles FROM 99 BY -1 UNTIL Bottles = -1
          DISPLAY SPACES
          SUBTRACT 1 FROM Bottles GIVING Remaining-Bottles
          EVALUATE Bottles
            WHEN 0
              DISPLAY "No more bottles of beer on the wall, "
                      "no more bottles of beer."
              DISPLAY "Go to the store and buy some more, "
                      "99 bottles of beer on the wall."
            WHEN 1
              DISPLAY "1 bottle of beer on the wall, "
                      "1 bottle of beer."
              DISPLAY "Take one down and pass it around, "
                      "no more bottles of beer on the wall."
            WHEN 2 Thru 99
              MOVE ZEROES TO Counting
              INSPECT Bottles,
                TALLYING Counting FOR LEADING ZEROES
              ADD 1 TO Counting GIVING Start-Position
              SUBTRACT Counting FROM 2 GIVING Positions
              DISPLAY Bottles(Start-Position:Positions)
                      " bottles of beer on the wall, "
                      Bottles(Start-Position:Positions)
                      " bottles of beer."
              MOVE ZEROES TO Counting
              INSPECT Remaining-Bottles TALLYING
                Counting FOR LEADING ZEROES
              ADD 1 TO Counting GIVING Start-Position
              SUBTRACT Counting FROM 2 GIVING Positions
              DISPLAY "Take one down and pass it around, "
                      Remaining-Bottles(Start-Position:Positions)
                      " bottles of beer on the wall."
          END-EVALUATE
        END-PERFORM
       STOP RUN.
```