1 INTRODUÇÃO

O COBOL é uma linguagem voltada exclusivamente para ambientes comerciais, ela possui uma arquitetura de programação muito fácil e clara, tornando os programas auto-documentáveis.

Para executar os exercícios e compilar os programas exemplos dados neste material é necessário que tenhamos um compilador COBOL no seu computador.

O objetivo desta disciplina é conhecer e aprender a linguagem COBOL.

1.1 **COBOL**

COBOL é uma linguagem de programação cuja sigla significa ***Co****mmon* ***B****usiness* ***O****riented* ***L****anguage* (Linguagem Comum Orientada aos Negócios), que tem como objetivo principal lidar com sistemas comerciais, financeiros e administrativos.

1.2 **ORIGEM**

O **COBOL** foi criado em 1959 durante o *CODASYL (Conference on Data Systems Language)*, 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.

Foi constituído por membros representantes de seis fabricantes de computadores e três órgãos governamentais, a saber:

1) IBM;
2) Burroughs Corporation;
3) Minneapolis-Honeywell (Honeywell Labs);
4) RCA;
5) Sperry Rand;
6) Sylvania Electric Products;
7) Força Aérea dos Estados Unidos;
8) David Taylor Model Basin;
9) Agência Nacional de Padrões (*National Bureau of Standards* ou NBS).

O COBOL foi desenvolvido num período de seis meses. A primeira versão (*COBOL 60*), não durou muito tempo devido a numerosos erros que foram rapidamente corrigidos na versão *COBOL 61.*

A versão *COBOL 61* serviu como base para outra versão, que foi lançada em 1962 e foi nomeada de *COBOL-61 - Versão Estendida*.

Compiladores COBOL geralmente se baseiam no Padrão Nacional Americano (ANSI). O primeiro padrão foi divulgado em 1968 e em seguida em 1974, 1985 e 1989. A última revisão foi concluída em 2002.

1.3 **MANTENDO-SE FORTE**

Embora o COBOL tenha sido proposto originalmente como solução para resolver problemas de programação do governo e das forças armadas americanas, os programas COBOL ainda continuam em uso na maioria das empresas comerciais em todo o mundo, notadamente nas instituições financeiras, e em praticamente todos os sistemas operacionais, incluindo o IBM z/OS, o Microsoft Windows e a família Unix/Linux.

A base global de código é imensa e os aplicativos, de tempos em tempos, são sujeitos a manutenção.

O custo de reescrever um aplicativo COBOL, já depurado, em uma nova linguagem não justifica os benefícios que possa eventualmente trazer.

No fim dos anos 90 o Gartner Group, uma empresa de pesquisa na área de informática, estimou que dos 300 bilhões de linhas de código-fonte existentes no mundo, 80% - ou cerca de 240 bilhões de linhas - eram em COBOL.

Eles também reportaram que mais de metade dos novos aplicativos de missão crítica ainda estavam sendo desenvolvidos usando o COBOL.

Ao se aproximar o fim do século XX houve uma febre de atividade de programadores COBOL para corrigir os efeitos do bug do milênio, em certos casos em sistemas desenvolvidos por estes mesmos programadores há décadas.

Este problema foi mais crítico no código COBOL porque as datas são primordiais em aplicativos comerciais, e as maiorias dos aplicativos comerciais foram escritos em COBOL.

O COBOL provou ser durável e adaptável.

O padrão atual do COBOL suporta conveniências modernas como Unicode, geração de XML e convenção de chamadas de/para linguagens como o **C**, inclusão como linguagem de primeira classe em ambientes de desenvolvimento como o .NET da Microsoft e a capacidade de operar em ambientes fechados como Java e acesso a qualquer base SQL.

No Brasil a área financeira e de seguros são os principais mercados de COBOL e ainda continua aquecido devido grandes compras e fusões das instituições.

As empresas que possuem programas em COBOL e que gostariam de migrar para outras linguagens, justificam e usam como principal artifício, o fato de muitas vezes esta migração não ocorrer devido ao alto custo e ao risco deste processo.

O tempo da migração também é um fator que mantém o COBOL no mercado.

2 EXPLANAÇÃO DE UM PROGRAMA SIMPLES FONTE COBOL

Todo programa escrito na linguagem COBOL possuí algumas regras rígidas a serem seguidas como uma redação que possui início, meio e fim.

Todo programa COBOL, possui quatro divisões que devem ser utilizadas nesta ordem:

IDENTIFICATION DIVISION;
ENVIRONMENT DIVISION;
DATA DIVISION;
PROCEDURE DIVISION.

2.1 IDENTIFICATION DIVISION

A IDENTIFICATION DIVISION possui informações documentais, como nome do programa, quem o codificou e quando essa codificação foi realizada.

2.2 ENVIRONMENT DIVISION

A ENVIRONMENT DIVISION descreve o computador e os periféricos que serão utilizados pelo programa.

2.3 **DATA DIVISION**

A DATA DIVISION descreve os arquivos de entrada e saída que serão usadas pelo programa. Também define as áreas de trabalho e constantes necessárias para o processamento dos dados.

2.4 **PROCEDURE DIVISION**

A PROCEDURE DIVISION contém o código que irá manipular os dados descritos na DATA DIVISION. É nesta divisão que o desenvolvedor descreverá o algoritmo do programa.

Estas divisões devem ser escritas seguindo algumas regras de posicionamento:

....|....1....|....2....|....3....|....4....|....5....|....6....|....7....|...

Colunas de 1 a 6: Área de numeração sequencial
Coluna 7: Área de indicação (comentário ou continuação)
Colunas de 8 a 11: Área A ou margem A
Colunas de 12 a 72: Área B ou margem B

Pode-se digitar até a coluna 80, mas tudo que for digitado entre 73 e 80 será considerado como um comentário.

2.5 **Áreas de numeração sequencial**

Coluna de 1 a 6

Normalmente consiste em seis dígitos em ordem crescente que normalmente são utilizados para numeração sequencial de identificação do programa fonte.

2.6 **Áreas de indicação**

Coluna 7

Coloca-se um asterisco(*) na coluna 7 (SETE), fazendo com que a linha inteira seja considerada como um comentário.

Um hífen (-) nesta posição indica que existe uma continuação de uma cadeia de caracteres que foi iniciada na linha anterior.

Opcionalmente podemos colocar o texto inteiro na linha de baixo para não termos que quebra-lo.

2.7 **Áreas A ou margem A**

Colunas de 8 a 11

As divisões, seções, declaração de variáveis e parágrafos do programa são os únicos que podem utilizar a área A.

2.8 **Áreas B ou margem B**

Colunas de 12 a 72

São usados para escrever todos os outros comandos.

2.9 **Exemplo EX01.CBL**

Vamos digitar e analisar o código fonte a seguir:

```
IDENTIFICATION DIVISION.

program-id.       EX01.
author.           Hiromasa Nagata.
Installation.     FATEC-SP.
Date-written.     04/03/2016.
Date-compiled.  04/03/2016.
Security.         Não há, pois é um programa teste.

DATA DIVISION.
working-storage section.
77 LARGURA          pic 9(003) value zeros.
77 ALTURA           pic 9(003) value zeros.
77 AREA-RESULT      pic 9(006) value zeros.
```

```
PROCEDURE DIVISION.

INICIO.

    display erase.
    display "Calculo de area (Retangulos)" at 0521.
    display "Largura: " at 1010.
    display "Altura : "  at 1210.
    accept  LARGURA  at 1019.
    accept  ALTURA    at 1219.
    Compute AREA-RESULT = LARGURA * ALTURA.
    display "Area : "    at 1410 AREA-RESULT.
    stop run.
```

3. IDENTIFICATION DIVISION

Divisão de identificação do programa.
Esta divisão possui a seguinte estrutura:

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     nome-programa.
AUTHOR.         comentário.
INSTALLATION.   comentário.
DATE-WRITTEN.   comentário.
DATE-COMPILED.  comentário.
REMARKS.        comentário.
SECURITY.       comentário.
```

Todos os comandos que possuem a palavra comentário na frente, não possuem nenhum efeito na aplicação, são apenas parâmetros opcionais para documentação do programa.
Porém, caso se queira utilizar algum comentário, ela deve ser escrita corretamente para que não cause erro de compilação.

Atenção: Alguns compiladores NÃO ACEITAM o comando REMARKS.

4. DATA DIVISION

Divisão voltada única e exclusivamente à definição de estruturas de registros, variáveis e constantes do programa, ou seja, uma área de alocação de memória para todo o espaço necessário ao seu programa.

Esta divisão possui a **WORKING-STORAGE SECTION**. Esta seção da **DATA DIVISION** é voltada para a declaração das variáveis e constantes do programa.

Todos os nomes utilizados no programa devem ser exclusivos, existem meios de quebrar esta regra conforme vermos posteriormente.

Os nomes devem ter no **máximo trinta caracteres**, caso o nome de algum elemento ultrapasse estes trinta caracteres, simplesmente o compilador ignorará o seu excedente.

77 largura pic 9(003) value zeros.

Esta linha está declarando uma variável chamada **largura** de 3 posições numéricas e lhe atribui como valor inicial: zero. O número 77 à frente indica o nível da variável. Maiores detalhamento serão explanados em aulas posteriores.

5. PROCEDURE DIVISION

Esta divisão controla a execução do programa, Aqui que colocamos os comandos a serem executados em uma ordem lógica. Esta execução é controlada por parágrafos (existe um chamado INICIO no programa exemplo) eles funcionam como identificações de blocos de comandos.

Explanação dos comandos do programa **EX01.**

5.1. **display erase**

O comando **display** é utilizado para exibir informações na tela em ambientes caracteres.

A palavra reservada **erase** é utilizada em conjunto com display para limpar a tela.

5.2. **display "Calculo de area (Retangulos)" at 0521**

Esta forma de utilização do **display** irá exibir na tela a cadeia de caracteres entre as aspas (“ ”) na posição especificada por **at 0521**, ou seja, linha 5 e coluna 21 que são compreendidas entre linhas de 1 a 25 e colunas de 1 a 80.

5.3. **accept largura at 1019**

Este comando é utilizado para aceitarmos alguma informação, neste caso estaremos esperando que usuário informe algo na posição 1019 (Seguem as mesmas regras do comando display).

Atenção: O usuário indica para a aplicação que terminou de fornecer estas informações pressionando a tecla ENTER.

5.4. **multiply largura by altura giving area-result**

O comando multiply é um dos comandos aritméticos do COBOL, ele é utilizado para funções de multiplicação. Neste caso ele irá multiplicar o conteúdo numérico da variável largura por altura movendo o resultado para a variável área-result.

5.5. **stop run**

Este comando encerra a aplicação, na verdade o **run** é um parâmetro do comando stop onde podemos substituir o run por uma cadeia de caracteres (string). Por exemplo: stop "Teste".

Este comando irá exibir a mensagem e esperar por um ENTER para que seja finalizado.

Todo parágrafo deve possuir um ponto final ou o último comando deve possuir um ponto.

6. **COMPILAÇÃO**

Para efetuarmos a compilação de um programa em COBOL, siga os seguintes passos:

1) Salve inicialmente o programa fonte com a extensão **".CBL**". Por exemplo **TELA01.CBL**;
2) Vá para o ambiente DOS;
3) Digite: COBOL <enter>
4) Source file-name: Nome do programa fonte <enter>
5) Object file-name: nome do progama <enter>
6) Source listing: nome do programa <enter>
7) Object listing: nome do programa <enter>

7. **LINKEDIÇÃO**

Para efetuarmos a LINKEDIÇÃO de um programa em COBOL, que contem os comandos accept e/ou display siga os seguintes passos:

1) Vá para o ambiente DOS;
2) Digite: link nomedopgm.obj+adis+adiskey+adisinit+extfh;<enter>
3) Este processo criará um módulo executável;

ATENÇÃO: para realização da linkedição, o programa obrigatoriamente deverá ter passado pelo processo de compilação.

8. **EXERCÍCIOS**

1) Digite o programa ilustrado em EX01.CBL e execute-o.

2) Copie o programa EX01.CBL e salve o novo programa como EX02.CBL fazendo com que todos os display's e accept's fiquem duas linhas mais abaixo na sua apresentação e execute-o.

3) Fazer um programa EX03.CBL que calcule e exiba a somatória de dois números inteiros digitados. Este programa deverá ter a seguinte formatação da tela:

CALCULO DA SOMATORIA DE DOIS VALORES

Entre com o primeiro valor:

Entre com o segundo valor:

Resultado da somatória e: