

Programação para Mainframe



Curso de Análise de Desenvolvimento de Sistemas

Aula 02

Prof. Claudio Benossi

1. Unidade

**Criando nosso Primeiro
Programa em Cobol**

Introdução ao COBOL

Nosso objetivo na aula de hoje é criar um programa em Cobol e discutir sobre o ambiente de desenvolvimento e teste da Linguagem Cobol.

Existem algumas ferramentas disponíveis, que tornam o COBOL utilizável em computadores modernos como a utilização de outras linguagens de programação mais modernas. Hoje vamos usar um projeto de código aberto que usam COBOL em computadores com Mac OS, Linux ou Windows.

Introdução ao COBOL

Nesse primeiro momento vamos usar o OpenCobolIDE



OpenCobolIDE

<https://launchpad.net/cobcide/+download>

Introdução ao COBOL

OpenCobolIDE

Você pode escrever seu código fonte usando um editor de texto de sua escolha, mas ambientes de desenvolvimento integrados (IDE's) tornam a nossa vida mais fácil.

O OpenCobolIDE é um IDE concebido para trabalhar com o COBOL.

Introdução ao COBOL

OpenCobolIDE

Fornecendo ótimas ferramentas para programação, como:

- Marcador de sintaxe;
- Esquemas de cores escuras e temas;
- Oferece a opção de compilar como um programa (.exe) ou como um subprograma (.so / .dll);
- dentre outras opções.

Introdução ao COBOL

OpenCobolIDE é um programa muito bem concebido e sua interface é simples e não deve exigir muito esforço para se acostumar, mas se o usuário precisar de ajuda, a documentação do **OpenCobolIDE** fornece uma boa visão geral da IDE.

Introdução ao COBOL

Agora vamos começar nossos primeiros passos para criar um programa em Coobol.

- ▶ Criar as Divisões do Cobol (4 divisões);
- ▶ Identificação do nosso Programa (Deve ser feito no início e fim do Programa;
- ▶ Definir as seções;
- ▶ Desenvolver o programa de acordo com a necessidade ou problema a ser resolvido.



Introdução ao COBOL

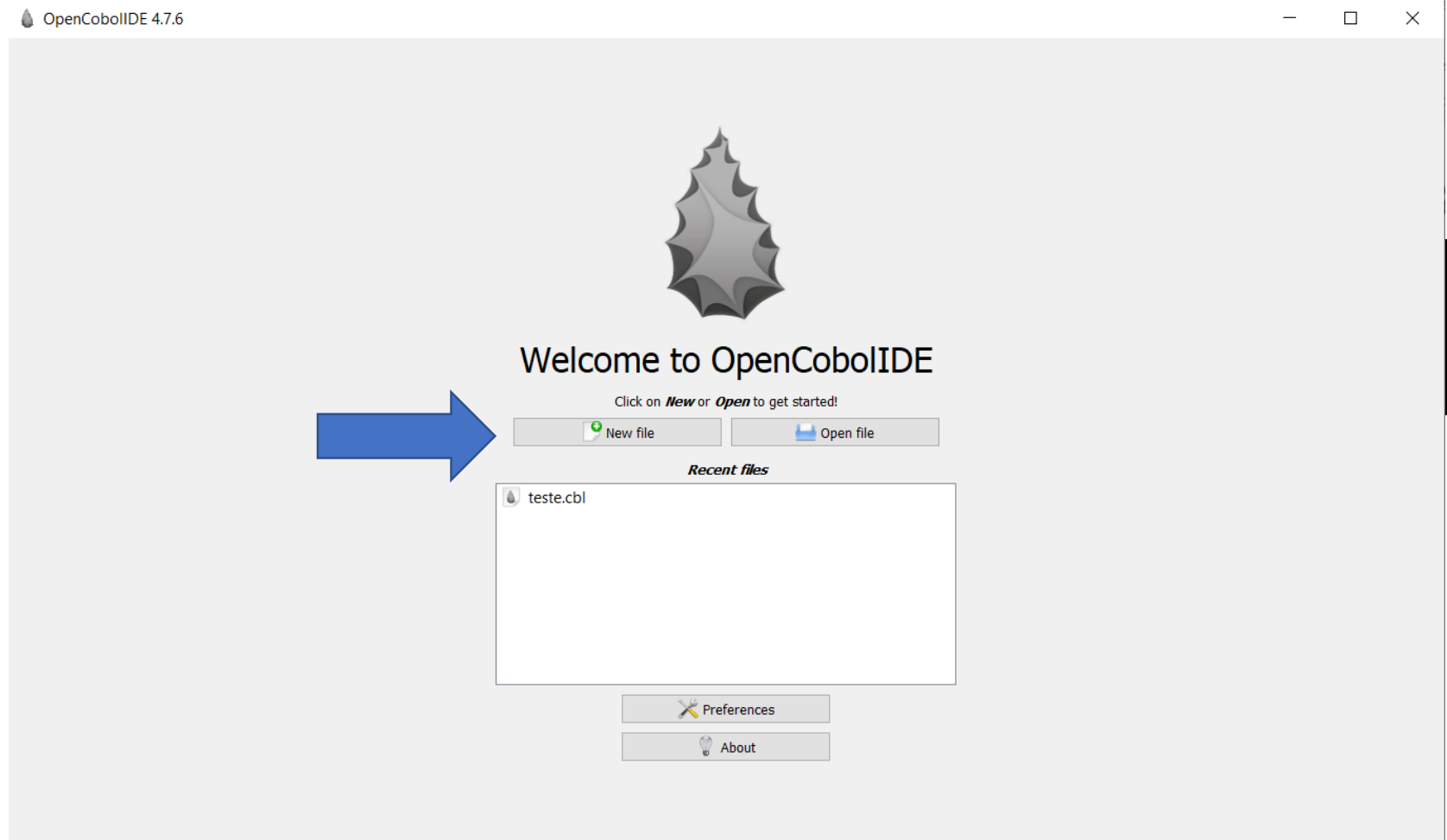
E um dos aspectos muito importante é usar as ...



BOAS PRÁTICAS

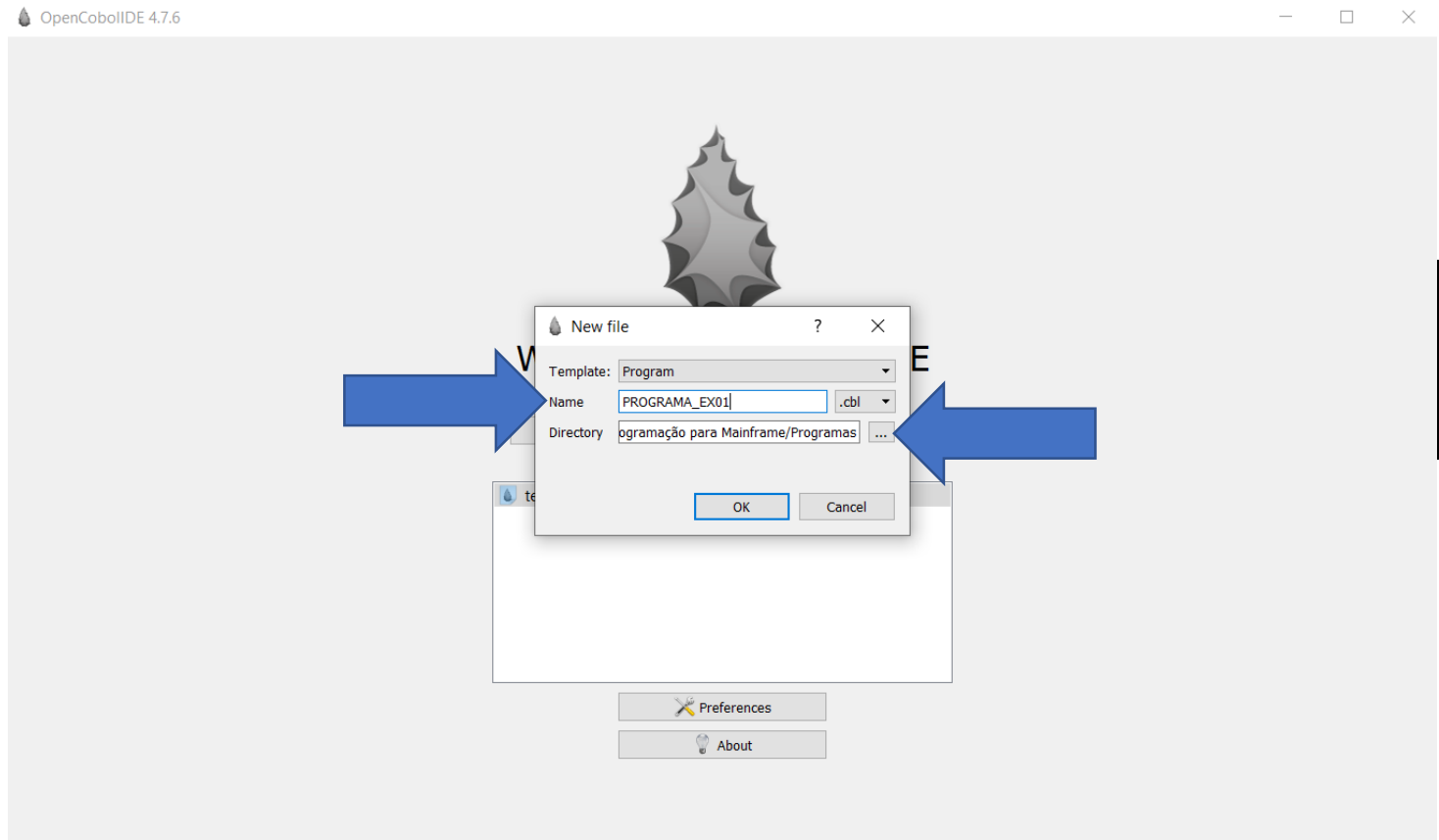
Introdução ao COBOL

O OpenCobolIDE



Introdução ao COBOL

Vamos colocar o nome do nosso primeiro programa como **PROGRAMA_EX01** e selecionar o local onde deve ser salvo.





File system

Programas

PROGRAMA_EX01.cbl

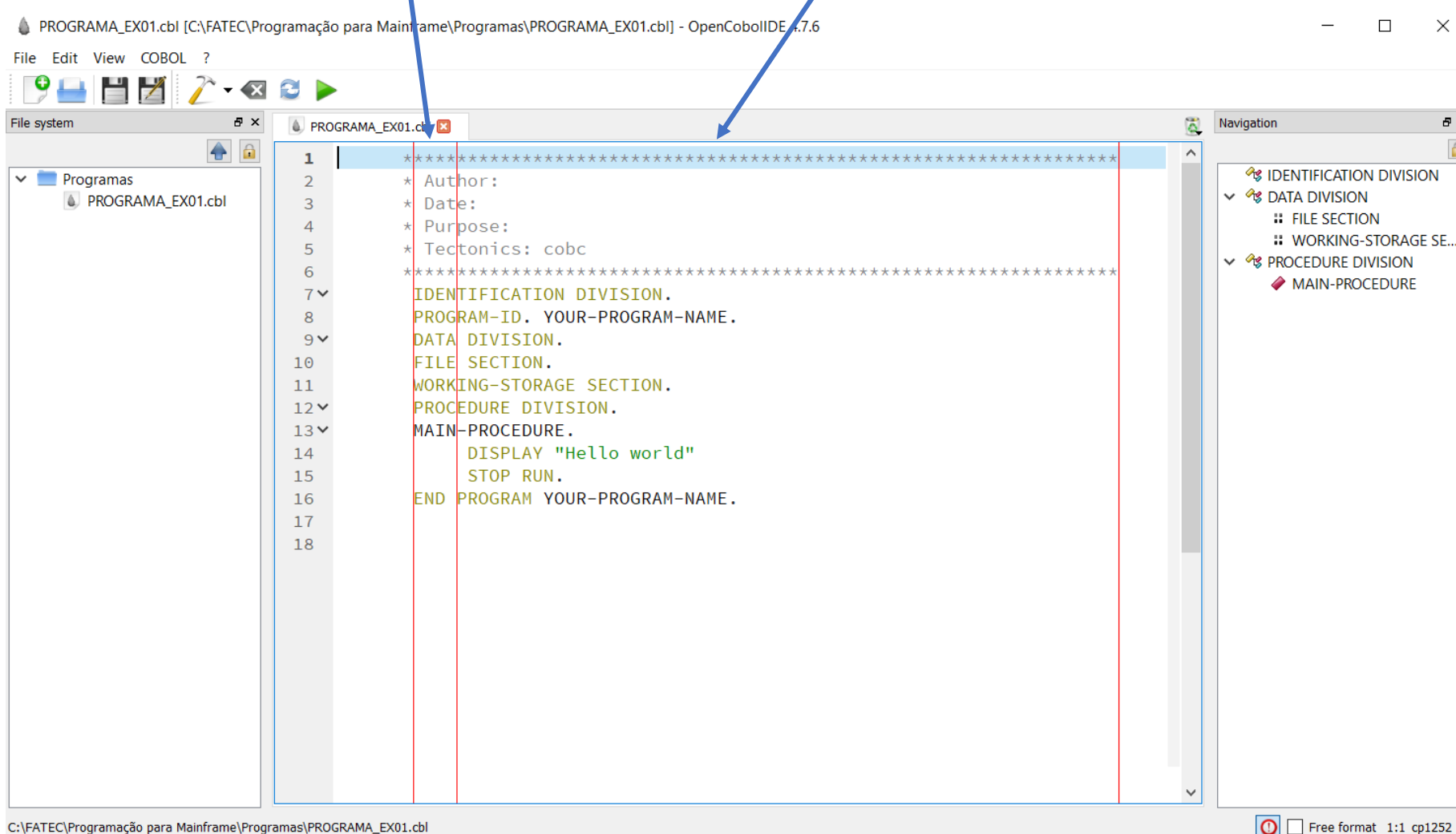
PROGRAMA_EX01.cbl

```
1 *****
2 * Author:
3 * Date:
4 * Purpose:
5 * Tectonics: cobc
6 *****
7 IDENTIFICATION DIVISION.
8 PROGRAM-ID. YOUR-PROGRAM-NAME.
9 DATA DIVISION.
10 FILE SECTION.
11 WORKING-STORAGE SECTION.
12 PROCEDURE DIVISION.
13 MAIN-PROCEDURE.
14     DISPLAY "Hello world"
15     STOP RUN.
16 END PROGRAM YOUR-PROGRAM-NAME.
17
18
```

Navigation

- IDENTIFICATION DIVISION
- DATA DIVISION
 - FILE SECTION
 - WORKING-STORAGE SE...
- PROCEDURE DIVISION
 - MAIN-PROCEDURE

Sequencia		área A	área B	não utilizado
1	6	7 8 11	12 72	73 80



PROGRAMA_EX01.cbl [C:\FATEC\Programação para Mainframe\Programas\PROGRAMA_EX01.cbl] - OpenCOBOLIDE v.7.6

File Edit View COBOL ?

File system

- Programas
 - PROGRAMA_EX01.cbl

```

1 *****
2 * Author:
3 * Date:
4 * Purpose:
5 * Tectonics: cobc
6 *****
7 IDENTIFICATION DIVISION.
8   PROGRAM-ID. YOUR-PROGRAM-NAME.
9 DATA DIVISION.
10  FILE SECTION.
11  WORKING-STORAGE SECTION.
12  PROCEDURE DIVISION.
13  MAIN-PROCEDURE.
14      DISPLAY "Hello world"
15      STOP RUN.
16  END PROGRAM YOUR-PROGRAM-NAME.
17
18

```

Navigation

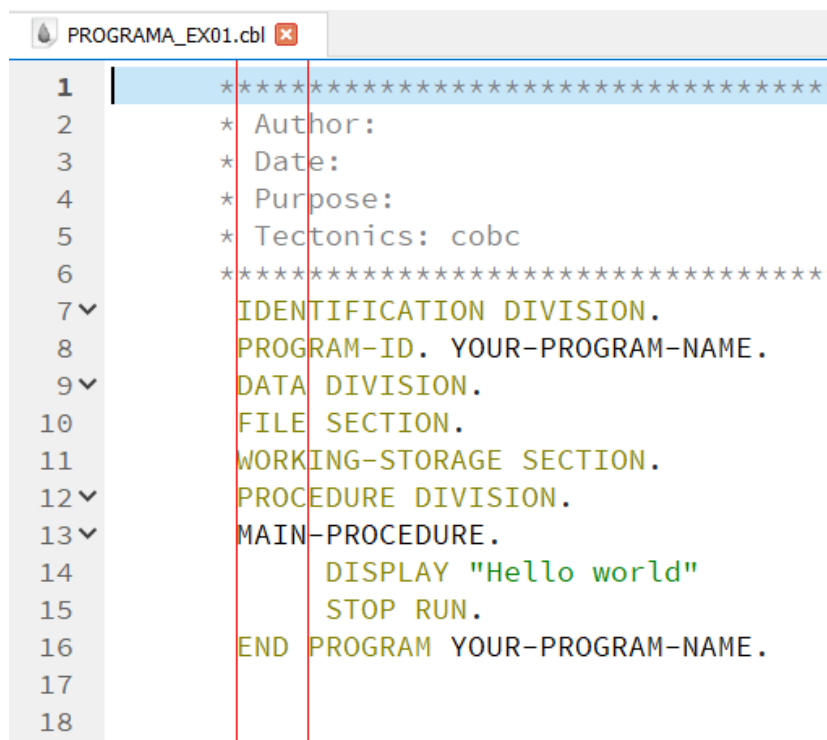
- IDENTIFICATION DIVISION
- DATA DIVISION
 - FILE SECTION
 - WORKING-STORAGE SE...
- PROCEDURE DIVISION
 - MAIN-PROCEDURE

C:\FATEC\Programação para Mainframe\Programas\PROGRAMA_EX01.cbl

Free format 1:1 cp1252

Introdução ao COBOL

O nosso OpenCobolIDE já traz as 4 divisões, porém ainda precisa de alguns ajustes ... Lembrando que em alguns casos você terá que digitar tudo a partir do 0.



```
PROGRAMA_EX01.cbl
1 *****
2 * Author:
3 * Date:
4 * Purpose:
5 * Tectonics: cobc
6 *****
7 IDENTIFICATION DIVISION.
8 PROGRAM-ID. YOUR-PROGRAM-NAME.
9 DATA DIVISION.
10 FILE SECTION.
11 WORKING-STORAGE SECTION.
12 PROCEDURE DIVISION.
13 MAIN-PROCEDURE.
14     DISPLAY "Hello world"
15     STOP RUN.
16 END PROGRAM YOUR-PROGRAM-NAME.
17
18
```

Introdução ao COBOL

Vamos fazer a identificação do nosso programa.

Segundo as boas práticas, o nome do programa deve ser o mesmo do arquivo, em algumas empresas isso fica estabelecido na “WorkSpace” que é comum a todos os membros da equipe de desenvolvimento.

Introdução ao COBOL

IDENTIFICATION DIVISION

Divisão de identificação do programa, seu nome IDENTIFICATION pode ser abreviado por ID.

Possui um único parágrafo obrigatório: **PROGRAM-ID**, que será seguido pelo nome do programa, formado por uma palavra com até 8 caracteres (letras ou números), começando por uma letra.

Introdução ao COBOL

IDENTIFICATION DIVISION

Sintaxe:

ID DIVISION.

PROGRAM-ID.

nomeprog.

Introdução ao COBOL

IDENTIFICATION DIVISION

Parâmetros obsoletos (foram descontinuados e, se usados, serão considerados comentários):

AUTHOR. comentário.

INSTALLATION. comentário.

DATE-WRITTEN. comentário.

DATE-COMPILED. comentário.

SECURITY. comentário.

Introdução ao COBOL

```
* PROGRAMA_EX01.cbl x
1 *****:
2 * Author: CLAUDIO BENOSSE
3 * Date: 03/09/2021
4 * Purpose: NOSSO PRIMEIRO PROGRAMA EM COBOL
5 * Tectonics: cobc
6 *****:
7 IDENTIFICATION DIVISION.
8 PROGRAM-ID. PROGRAMA_EX01.
9 DATA DIVISION.
10 FILE SECTION.
11 WORKING-STORAGE SECTION.
12 PROCEDURE DIVISION.
13 MAIN-PROCEDURE.
14     DISPLAY "Hello world"
15     STOP RUN.
16 END PROGRAM PROGRAMA_EX01.
17
```

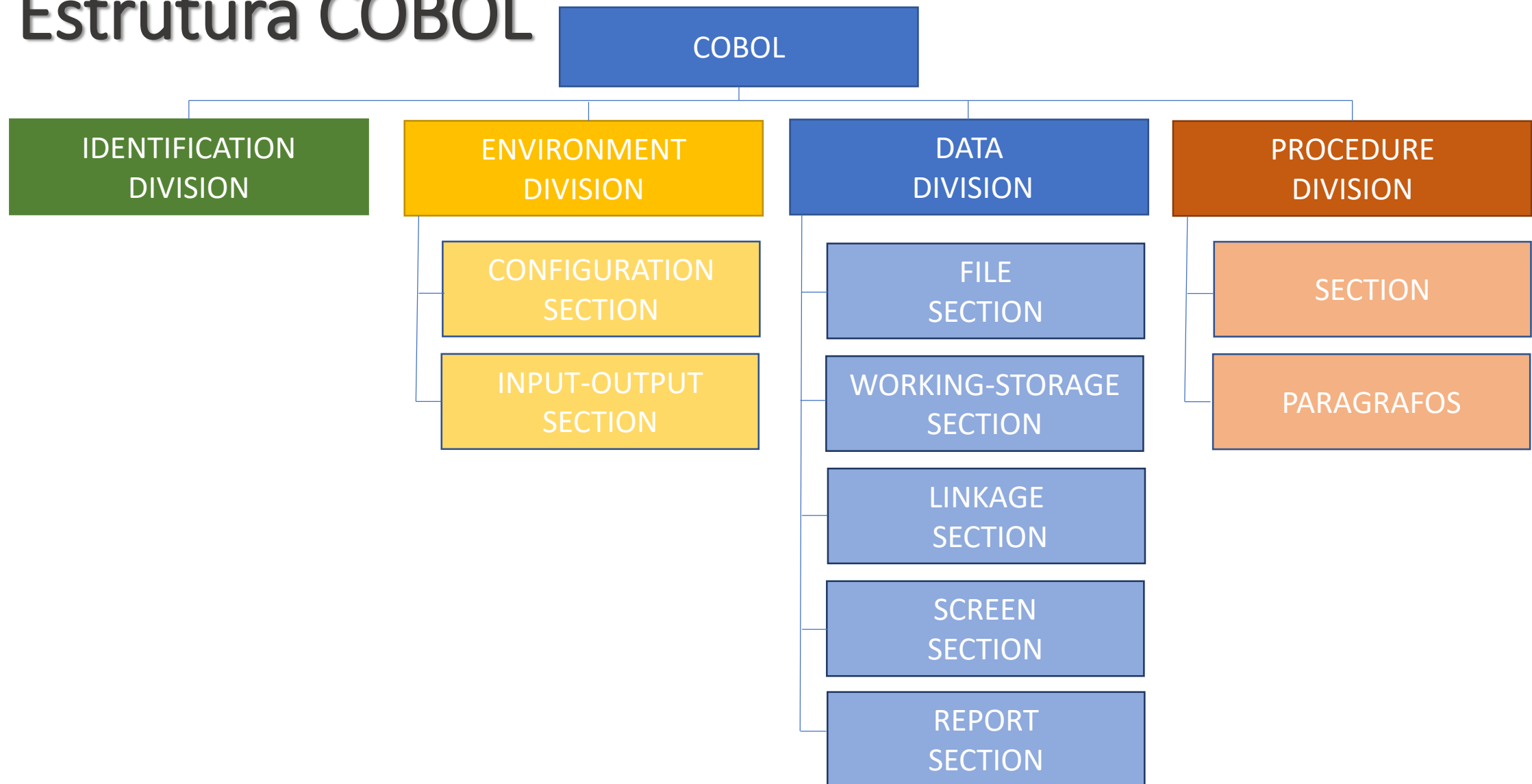
Introdução ao COBOL

Agora vamos definir as nossas seções.

Uma **SECTION** é constituída por um cabeçalho de seção opcionalmente seguido por um ou mais parágrafos.

A seção de cabeçalho é um nome da seção seguido de: a palavra-chave **SECTION**, um número de segmento opcional e um período de separação.

Estrutura COBOL



Estrutura COBOL

A **CONFIGURATION SECTION** é uma seção opcional para programas e classes e pode descrever o ambiente do computador no qual o programa ou classe é compilado e executado.

A seção de configuração pode ser especificada apenas na **ENVIRONMENT DIVISION** do programa mais externo de um programa de origem **COBOL**.

Estrutura COBOL

A **INPUT-OUTPUT SECTION** (seção de entrada e saída) pode ser especificada em uma **CONFIGURATION SECTION** (seção de configuração) do método.

As entradas se aplicam apenas ao método em que a **CONFIGURATION SECTION** (seção de configuração) está especificada. A seção de entrada e saída da divisão de ambiente contém dois parágrafos:

- Parágrafo FILE-CONTROL
- Parágrafo I-O-CONTROL

Estrutura COBOL

A **FILE SECTION** seção da **DATA DIVISION**, define todos os arquivos de entrada e saída, ela representa o nível mais alto de organização na seção Arquivo.

Fornece informações sobre a estrutura física e identificação de um arquivo, e dá o registro nome (s) associado com esse arquivo.

Estrutura COBOL

A **WORKING-STORAGE SECTION** seção da **DATA DIVISION** que descreve registros de dados que não fazem parte dos arquivos de dados, mas são desenvolvidos e processados por um programa ou método, ou seja, onde definimos nossas variáveis.

Também descreve itens de dados cujos valores são atribuídos no programa ou método de origem e não são alterados durante a execução do programa de objeto.

Estrutura COBOL

A **LINKAGE SECTION** seção da **DATA DIVISION** é usado para transmitir dados de um programa para outro ou para passar dados de um PROC para um programa.

É parte de um programa chamado que 'links' ou mapas para itens de dados que usaremos para enviar dados para outros programas.

Usado para uma interface externa, conversa com outros programas.



Estrutura COBOL

A **SCREEN SECTION** seção da **DATA DIVISION** é usado para descreve as telas a serem exibidas durante a execução do programa.

Estrutura COBOL

A **REPORT SECTION** seção da **DATA DIVISION** esta seção nos não vamos utilizar em nosso exemplo, até por boas práticas, essa seção está em desuso, praticamente não é utilizada, nesse caso de relatórios (Ler e Imprimir dados), hoje tratamos isso na **PROCEDURE DIVISION**.

Estrutura COBOL

A **SECTION** seção da **PROCEDURE DIVISION** é constituída por um cabeçalho de seção opcionalmente seguido por um ou mais parágrafos.

A seção de cabeçalho é um nome da seção seguido de: a palavra-chave **SECTION**, um número de segmento opcional e um período de separação.

A **SECTION-NAME** é uma palavra definida pelo usuário que identifica uma seção. Se referenciado, um **SECTION-NAME** deve ser exclusivo dentro do programa no qual ele é definido, porque ele não pode ser qualificado.

Estrutura COBOL

A **PARAGRAFO** seção da **PROCEDURE DIVISION** consiste em um nome de ponto seguido por um período de separação, opcionalmente seguido por uma ou mais sentenças.

Um **PARAGRAPH-NAME** é uma palavra definida pelo usuário que identifica um parágrafo.

Introdução ao COBOL

Vamos voltar ao nosso primeiro programa em Cobol.

```
* PROGRAMA_EX01.cbl x
1      *****:
2      * Author: CLAUDIO BENOSSE
3      * Date: 03/09/2021
4      * Purpose: NOSSO PRIMEIRO PROGRAMA EM COBOL
5      * Tectonics: cobc
6      *****:
7      IDENTIFICATION DIVISION.
8      PROGRAM-ID. PROGRAMA_EX01.
9      DATA DIVISION.
10     FILE SECTION.
11     WORKING-STORAGE SECTION.
12     PROCEDURE DIVISION.
13     MAIN-PROCEDURE.
14         DISPLAY "Hello world"
15         STOP RUN.
16     END PROGRAM PROGRAMA_EX01.
17
```

Introdução ao COBOL

Na primeira divisão a **IDENTIFICATION DIVISION** não será necessário criar nenhuma seção e somente na segunda divisão a **ENVIRONMENT DIVISION**, nesta divisão teremos duas seções a **CONFIGURATION SECTION** (seção de configuração) e a **INPUT-OUTPUT SECTION** (Seção de entrada e saída).

Introdução ao COBOL

Na **IDENTIFICATION DIVISION** são identificados e definidos recursos do ambiente que serão usados pelo programa.

O parágrafo **I-O-CONTROL** da **CONFIGURATION SECTION**, **opcional**, não é utilizado com frequência em programas aplicativos e serve para especificar controles para eventual reinício do programa.

Introdução ao COBOL

```
* PROGRAMA_EX01.cbl
1 *****
2 * Author: CLAUDIO BENOSSE
3 * Date: 03/09/2021
4 * Purpose: NOSSO PRIMEIRO PROGRAMA EM COBOL
5 * Tectonics: cobc
6 *****
7 IDENTIFICATION DIVISION.
8 PROGRAM-ID. PROGRAMA_EX01.
9 ENVIRONMENT DIVISION.
10 CONFIGURATION SECTION.
11 INPUT-OUTPUT SECTION.
12 DATA DIVISION.
13 FILE SECTION.
14 WORKING-STORAGE SECTION.
15 PROCEDURE DIVISION.
16 MAIN-PROCEDURE.
17     DISPLAY "Hello world"
18     STOP RUN.
19 END PROGRAM PROGRAMA_EX01.
```

Introdução ao COBOL

Na **DATA DIVISION** a seção **FILE SECTION** descreve os arquivos selecionados na **INPUT-OUTPUT SECTION** e a estrutura de seus registros.

Contém dois parágrafos: FD (**File Description**) e SD (**Sort Description**).

Introdução ao COBOL

Sintaxe das principais opções:

FD nomearquivo

LABEL RECORD STANDARD | OMMITED

RECORDING MODE F

BLOCK CONTAINS n RECORDS.

01 nomeregistro . . .

Introdução ao COBOL

Sintaxe das principais opções:

FD nomearquivo

LABEL RECORD STANDARD | OMMITED

RECORDING MODE F

BLOCK CONTAINS n RECORDS.

01 nomeregistro . . .

nomearquivo definido no SELECT

Introdução ao COBOL

Sintaxe das principais opções:

FD nomearquivo

LABEL RECORD STANDARD | OMMITED

RECORDING MODE F

BLOCK CONTAINS n RECORDS.

01 nomeregistro . . .

LABEL RECORD STANDARD | OMMITED tratado como
comentário

Introdução ao COBOL

Sintaxe das principais opções:

FD nomearquivo

LABEL RECORD STANDARD | OMITTED

RECORDING MODE F

BLOCK CONTAINS n RECORDS.

01 nomeregistro . . .

RECORDING MODE F arquivo contém registros de tamanho fixo.

Introdução ao COBOL

Sintaxe das principais opções:

FD nomearquivo

LABEL RECORD STANDARD | OMMITED

RECORDING MODE F

BLOCK CONTAINS n RECORDS.

01 nomeregistro . . .

BLOCK CONTAINS n RECORDS número de registros lógicos dentro do registro físico. Se n igual a 0 assume valor especificado no parâmetro **BLKSIZE** do **JCL** (arquivos novos ou catalogados).

Introdução ao COBOL

No nosso exemplo atual, nosso primeiro programa em cobol, não vamos usar a **FILE SECTION**.

Voltaremos a falar dela nos próximos exemplos.

Introdução ao COBOL

Agora vamos criar uma variável com o nome **WS-EXIBIR** na **DATA DIVISION** na seção **WORKING-STORAGE SECTION**.

Nessa seção serão definidas as variáveis e áreas de trabalho utilizadas no programa que serão referenciadas na lógica descrita na **PROCEDURE DIVISION**.

Lembrando que uma variáveis em um programa são abstrações de células de memória.

Introdução ao COBOL

Exemplos:

- 01 nome PIC **A**(010) value spaces.
 - É um exemplo de declaração de variável alfabética.
- 77 numero PIC **9**(006) value zeros.
 - É um exemplo de declaração de variável numérica.
- 01 nome PIC **X**(010) value spaces.
 - É um exemplo de declaração de variável alfanumérica.

Introdução ao COBOL

Exemplos:

- 77 WRK-CT-LIN PIC 9(003) VALUE 50.
- 77 WRK-AC-PAG PIC 9(003) VALUE ZERO.
- 77 WRK-CH-FIM PIC X(003) VALUE 'NAO'.

A decorative graphic in the top-left corner consisting of a cluster of hexagons in various colors including yellow, orange, red, and grey.

Introdução ao COBOL

Exemplos:

01 WRK-DATA.

05 WRK-ANO PIC 9(002).

05 WRK-MES PIC 9(002).

05 WRK-DIA PIC 9(002).

Introdução ao COBOL

Estruturando definições de dados

Informações recebidas, geradas ou áreas de trabalho como acumuladores, totalizadores e linhas de impressão devem estar definidas na **DATA DIVISION** do programa COBOL para que possam ser referenciadas nos comandos e instruções codificadas na **PROCEDURE DIVISION**.

Introdução ao COBOL

Estruturando definições de dados

A definição segue uma estrutura hierárquica na qual uma área de nível superior (**item de grupo**) é subdividida em áreas associadas a níveis inferiores (outros itens de grupos ou elementares).

Introdução ao COBOL

Estruturando definições de dados

Nível É um número variando de 01 a 49 que define a hierarquia dos dados dentro de uma estrutura.

O número 01 define o mais alto nível e os demais (02 a 49) definem sequências decrescentes onde uma área associada a um número maior é de mais baixo nível e está subordinada aos níveis mais altos (número menor).

Introdução ao COBOL

Estruturando definições de dados

Existem dois outros níveis: 77 e 88.

01 DATA-NASCIMENTO

02 DIA-NASCIMENTO

02 MES-NASCIMENTO

02 ANO-NASCIMENTO

Introdução ao COBOL

Estruturando definições de dados

Toda definição de área de dados deve começar na área A (entre as colunas 8 e 11) e usar o nível 01 ou 77.

Item de grupo – identifica uma área de dados que será subdividida. Deve estar associada aos níveis 01 até 48.

Item elementar – identifica uma área de dados que não será subdividida. Definir com níveis 01 a 49 ou 77.

Introdução ao COBOL

Estruturando definições de dados

Nível 77 – define um item independente (não pode ser subdividido e não pode estar subordinado a um item de grupo).

Nível 88 – atribui um nome a uma condição e, a usado, deve seguir a variável que poderá conter a condição indicada.

Introdução ao COBOL

Exemplo:

Definindo um item elementar:

nível nome-do-dado

01 SOMA-CREDITOS

formato valor-inicial

PIC 9(5)V99 VALUE ZEROS.

Introdução ao COBOL

Exemplo:

Definindo um item elementar:

nível nome-do-dado

01 SOMA-CREDITOS

formato valor-inicial

PIC 9(5)V99 VALUE ZEROS.

Nível – número de 01 a 49 e 77.

Introdução ao COBOL

Exemplo:

Definindo um item elementar:

nível nome-do-dado

01 **SOMA-CREDITOS**

formato

valor-inicial

PIC 9(5)V99 VALUE ZEROS.

Nome-do-dado – de 1 a 30 caracteres, incluindo letras, números e hífen, sendo que pelo menos um dos caracteres deve ser letra.

Introdução ao COBOL

Exemplo:

Definindo um item elementar:

nível nome-do-dado

01 SOMA-CREDITOS

formato

valor-inicial

PIC 9(5)V99 VALUE ZEROS.

Formato - especificado pela palavra reservada PICTURE, ou pela sua abreviação PIC. Pode ser numérico (9), alfabético (A) ou alfanumérico (X).

Introdução ao COBOL

Exemplo:

Definindo um item elementar:

nível nome-do-dado

01 SOMA-CREDITOS

formato

valor-inicial

PIC 9(5)V99 **VALUE ZEROS.**

Valor-inicial - cláusula opcional, usada para atribuir um valor inicial para a área de memória definida. Se omitida, o item correspondente terá valores imprevisíveis.

Usar exclusivamente na **WORKING-STORAGE SECTION.**

Introdução ao COBOL

O nível 77 é utilizado para declarar variáveis que não irão possuir sub-itens, este nível é utilizado da mesma forma que o nível 01.

```
* PROGRAMA_EX01.cbl
1 *****
2 * Author: CLAUDIO BENOSSE
3 * Date: 03/09/2021
4 * Purpose: NOSSO PRIMEIRO PROGRAMA EM COBOL
5 * Tectonics: cobc
6 *****
7 IDENTIFICATION DIVISION.
8 PROGRAM-ID. PROGRAMA_EX01.
9 ENVIRONMENT DIVISION.
10 CONFIGURATION SECTION.
11 INPUT-OUTPUT SECTION.
12 DATA DIVISION.
13 FILE SECTION.
14 WORKING-STORAGE SECTION.
15 77 WS-EXIBIR PIC X(20) VALUE SPACES.
16 PROCEDURE DIVISION.
17 MAIN-PROCEDURE.
18 DISPLAY "Hello world"
19 STOP RUN.
20 END PROGRAM PROGRAMA_EX01.
```

Introdução ao COBOL

Na **PROCEDURE DIVISION**, Descrever as ações para que o programa execute os comandos do algoritmo planejado pelo programador (lógica).

Os comandos (instruções) são formados por um único verbo da língua inglesa seguido dos parâmetros necessários.

As instruções do programa podem ser reunidas em parágrafos, e estes em seções definidas pelo programador, com a finalidade de tornar o programa mais fácil de ser entendido.

Introdução ao COBOL

Se o programador executou adequadamente as fases anteriores da tarefa de desenvolvimento, na **PROCEDURE DIVISION** ele simplesmente transcreverá seu diagrama de blocos ou pseudocódigo para o COBOL.

Na **PROCEDURE DIVISION** não existe seções ou parágrafos pré-definidos, mas é a única seção onde é possível criar seções e parágrafos.

Introdução ao COBOL

Os nomes de seções e parágrafos que serão adotados seguirão a seguinte regra: xxxx-nome, indica o início do parágrafo ou seção, onde xxxx é um número seqUencial iniciado em 1000 com incrementos de 1000.

xxxx-fim-nome, indica o final da seção ou parágrafo de mesmo número.

Exemplo de Procedure Division.

PROCEDURE DIVISION.

----- ROTINAS PRINCIPAIS DO PROGRAMA -----

PERFORM 1000-INICIALIZAR.

PERFORM 2000-PROCESSAR

UNTIL WRK-CHAV-FIM = 'SIM'.

PERFORM 3000-FINALIZAR.

STOP RUN.

-----PARAGRAFO INICIALIZAR -----

1000-INICIALIZAR.

...

1000-FIM-INICIALIZAR.

EXIT.

-----PARAGRAFO PROCESSAR -----

2000-PROCESSAR.

...

2000-FIM-PROCESSAR.

EXIT.

-----PARAGRAFO FINALIZAR -----

3000-FINALIZAR.

...

3000-FIM-FINALIZAR.

EXIT.

Introdução ao COBOL

Instrução **MOVE** da **PROCEDURE DIVISION**

Formato:

MOVE [ALL] {identificador-1 | literal-1} TO {identificador-2 ...}

alinhamento numérico - os campos, identificador1 e identificador2 são numéricos.

Introdução ao COBOL

Instrução **MOVE** da **PROCEDURE DIVISION**

Os dados são acomodados no campo receptor alinhando-se da direita para a esquerda.

Se o campo emissor for maior que o receptor os BYTES a esquerda do campo emissor, no campo receptor, serão truncados. alinhamento alfanumérico - o campo identificador2 é alfanumérico.

Introdução ao COBOL

Instrução **MOVE** da **PROCEDURE DIVISION**

Os dados são acomodados no campo receptor alinhando-se da esquerda para direita.

Se o campo emissor for maior que o receptor os BYTES a direita do campo emissor, no campo receptor, serão truncados.

Introdução ao COBOL

Instrução **MOVE** da **PROCEDURE DIVISION**

Formato:

MOVE [ALL] {identificador-1 | literal-1} TO {identificador-2 ...}

Os campos: identificador1 e identificador2 podem ser itens de grupo ou elementares.

Um item de grupo será tratado sempre como alfanumérico.

Introdução ao COBOL

Instrução **PERFORM** da **PROCEDURE DIVISION**

A instrução **PERFORM** permite que o controle passe temporariamente para um parágrafo diferente e depois retorne para o parágrafo original do qual a instrução **PERFORM** foi executada.

Introdução ao COBOL

Instrução **PERFORM** da **PROCEDURE DIVISION**

Há dois tipos de instrução **PERFORM**:

- **PERFORM out-line**: nome do parágrafo ou seção é especificado.
- **PERFORM in-line**: as instruções estão logo abaixo do comando **PERFORM**.

Introdução ao COBOL

Instrução **PERFORM** da **PROCEDURE DIVISION**

Deve ser delimitado pela frase END-PERFORM.

Há 4 formatos de PERFORM:

- PERFORM básico
- PERFORM com opção TIMES
- PERFORM com opção UNTIL
- PERFORM com opção VARYING

Introdução ao COBOL

PERFORM básico

PERFORM parágrafo [THRU] parágrafo-fim

A opção THRU é opcional e poderá ser utilizada nos demais formatos. Exemplo:

PERFORM INICIALIZAR.

PERFORM PROCESSAR THRU PROCESSAR-FIM.

Na opção THRU o parágrafo-fim indica o ultimo parágrafo que será executado pelo PERFORM.

Introdução ao COBOL

PERFORM TIMES

PERFORM parágrafo n TIMES

O parágrafo referido é executado **n** TIMES, onde **n** pode ser uma constante ou variável numérica.

Exemplo: PERFORM com opção TIMES

20-00-CALCULA-TOTAL.

MOVE ZEROS TO WRK-TOTAL

PERFORM 25-00-CALCULO 3 TIMES.

20-99-EXIT. EXIT.

Introdução ao COBOL

PERFORM UNTIL

PERFORM parágrafo **UNTIL** condição

O parágrafo referido é executado até que a condição especificada pela opção UNTIL seja verdadeira. Exemplo:

00-00-MAIN-LINE SECTION.

PERFORM INICIALIZAR

PERFORM 30-00-PROCESSAR

UNTIL WRK-FIM = "S"

PERFORM 50-00-FINALIZAR.

00-99-EXIT. EXIT.

Introdução ao COBOL

Exemplo - PERFORM com opção UNTIL

```
00-00-MAIN-LINE.  
    PERFORM INICIALIZAR.  
    PERFORM UNTIL WRK-FIM = "S"  
        IF WRK-LIN > 50  
            PERFORM 60-00-ROT-CABECALHO  
        END-IF  
        PERFORM 60-10-ROT-DETALHE  
        PERFORM 10-00-LER-ARQUIVO  
    END-PERFORM.  
    PERFORM FINALIZAR.  
00-99-EXIT. EXIT.
```


Introdução ao COBOL

PERFORM VARYING

PERFORM parágrafo **VARYING** campo
FROM n **BY** m **UNTIL** condição

Executa o parágrafo indicado, até que a condição especificada seja satisfeita. Antes de executar o bloco de instruções pela primeira vez, atribui o valor n a variável campo. Após cada execução do bloco, antes de voltar a executá-lo, incrementa m à variável campo. O teste é efetuado antes do desvio. O programa pode utilizar a variável campo no parágrafo chamado ou em outra rotina, normalmente.

Introdução ao COBOL

Exemplo - PERFORM com opção VARYING (OUT-LINE)

```
3000-00-PROCESSAR.  
    PERFORM 5000-00-ROT-CABECALHO.  
    PERFORM 5000-10-ROT-DETALHE  
    VARYING WRK-LIN FROM 1 BY 1  
    UNTIL WRK-LIN = 60  
  
    ...  
  
5000-10-ROT-DETALHE.  
    PERFORM 5000-15-IMPRIMIR-DETALHE.  
    PERFORM 2000-00-LER-ARQUIVO
```

Introdução ao COBOL

Exemplo - PERFORM com opção VARYING (IN-LINE)

```
0001-00-MAIN-LINE SECTION.  
    PERFORM 1000-00-INICIALIZAR  
    PERFORM 2000-00-LER-ARQUIVO  
  
    PERFORM UNTIL WRK-FLAG-FIM = 'S'  
        PERFORM 4000-00-ROT-CABECALHO  
        PERFORM VARYING WRK-AC-LIN  
            FROM 1 BY 1 UNTIL WRK-AC-LIN = 60  
            PERFORM 4000-10-IMPRIMIR-DETALHE  
            PERFORM 2000-00-LER-ARQUIVO  
        END-PERFORM  
    END-PERFORM.  
  
    PERFORM 9000-00-FINALIZAR.
```

Introdução ao COBOL

Agora vamos explorar o comando **MOVE** (*O comando **MOVE** transfere dados de uma área de armazenamento para outra*) e o comando **PERFORM** (*O comando **PERFORM** transfere o controle explicitamente para uma ou mais declarações e retorna implicitamente o controle à próxima instrução executável após a conclusão das instruções especificadas*) para movimentar o foco da nossa programação.

Introdução ao COBOL

Vamos criar nossa primeira seção com o nome de **MAIN-PROC** dentro da **PROCEDURE DIVISION** e incluir 3 paragrafos

```
PROGRAMA_EX01.cbl
16  ▼  PROCEDURE DIVISION.
17      ***** AQUI VAMOS INICIAR A NOSSA PROGRAMAÇÃO *****
18      ***** VAMOS CRIAR A NOSSA SEÇÃO COMO MAIN-PROC *****
19  ▼  MAIN-PROC SECTION.
20  ▼  ***** AGORA VOU CRIAR MEUS PARAGRAFOS *****
21  ▼  P001-PROC1.
22      MOVE 'P001-PROC1'    TO WS-EXIBIR
23      DISPLAY WS-EXIBIR
24      PERFORM SEC-PROC.
25  ▼  P001-PROC2.
26      MOVE 'P001-PROC2'    TO WS-EXIBIR
27      DISPLAY WS-EXIBIR
28      PERFORM SEC-PROC.
29  ▼  P001-PROC3.
30  ▼  IF WS-EXIBIR NOT EQUAL 'P002-PROC3' THEN
31      MOVE 'P001-PROC3'    TO WS-EXIBIR
32      DISPLAY WS-EXIBIR
33      PERFORM P002-PROC3
34  ▼  ELSE
35      PERFORM FIM-PROC
36      END-IF.
```

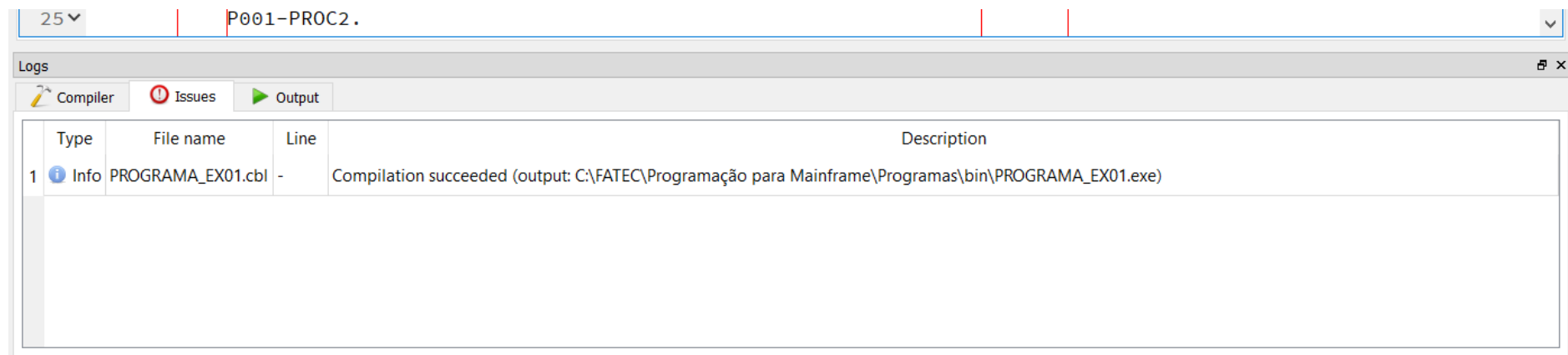
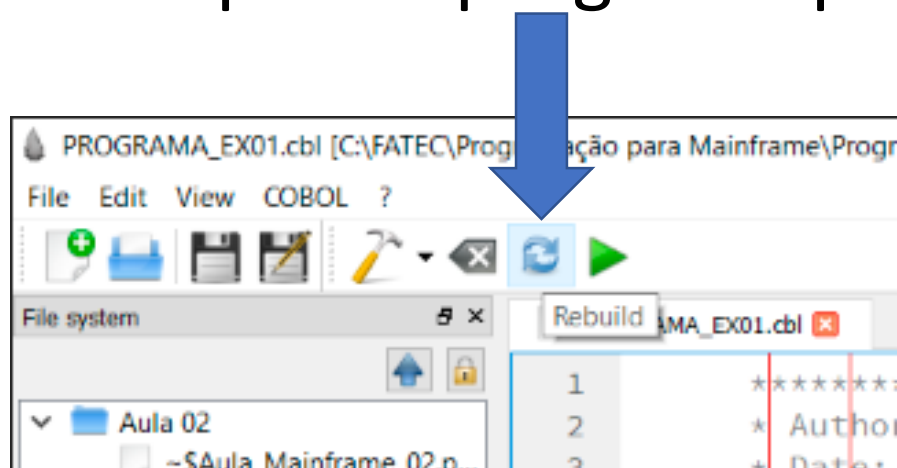
Introdução a

Agora a
segunda seção
SEC-PROC e
terceira seção
a **FIM-PROC**

```
PROGRAMA_EX01.cbl
36      END-IF.
37      ***** VAMOS CRIAR A NOSSA SEGUNDA SEÇÃO *****
38      SEC-PROC SECTION.
39      ***** AGORA VOU CRIAR MEUS PARAGRAFOS DA SEGUNDA SEÇÃO *****
40      P002-PROC1.
41      IF WS-EXIBIR NOT EQUAL 'P001-PROC2' THEN
42          MOVE 'P002-PROC1' TO WS-EXIBIR
43          DISPLAY WS-EXIBIR
44          PERFORM P001-PROC2
45      END-IF.
46      P002-PROC2.
47          MOVE 'P002.PROC2' TO WS-EXIBIR
48          DISPLAY WS-EXIBIR
49          PERFORM P001-PROC3.
50      P002-PROC3.
51          MOVE 'P002-PROC3' TO WS-EXIBIR
52          DISPLAY WS-EXIBIR
53          PERFORM P001-PROC3.
54      ***** VAMOS CRIAR A SEÇÃO DE FINALIZAÇÃO *****
55      FIM-PROC SECTION.
56          MOVE 'FIM-PROC' TO WS-EXIBIR
57          DISPLAY WS-EXIBIR
58          STOP RUN.
59      END PROGRAM PROGRAMA_EX01.
```

Introdução ao COBOL

Vamos Copilar o programa para verificar se existe algum erro.

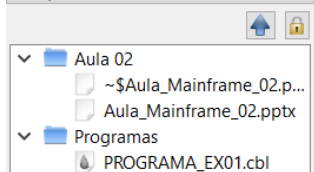


PROGRAMA_EX01.cbl [C:\FATEC\Programação para Mainframe\Programas\PROGRAMA_EX01.cbl] - OpenCobolIDE 4.7.6

File Edit View COBOL ?



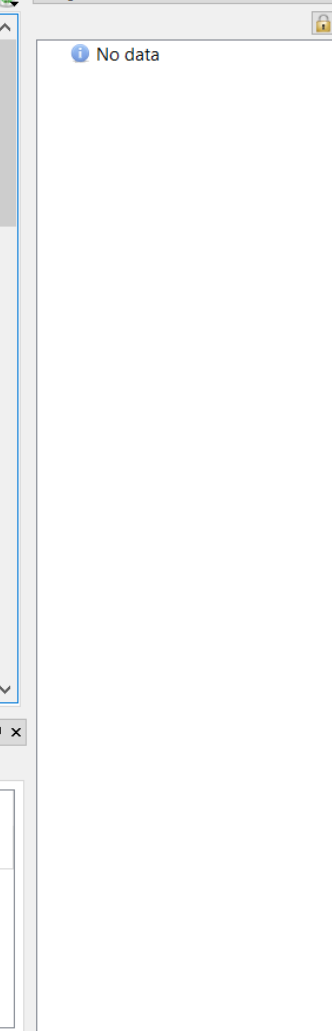
File system



```

1 *****
2 * Author: CLAUDIO BENOSSE
3 * Date: 03/09/2021
4 * Purpose: NOSSO PRIMEIRO PROGRAMA EM COBOL
5 * Tectonics: cobc
6 *****
7 IDENTIFICATION DIVISION.
8 PROGRAM-ID. PROGRAMA_EX01.
9 ENVIRONMENT DIVISION.
10 CONFIGURATION SECTION.
11 INPUT-OUTPUT SECTION.
12 DATA DIVISION.
13 FILE SECTION.
14 WORKING-STORAGE SECTION.
15     77 WS-EXIBIR    PIC X(20) VALUE SPACES.
16 PROCEDURE DIVISION.
17 ***** AQUI VAMOS INICIAR A NOSSA PROGRAMAÇÃO *****
18 ***** VAMOS CRIAR A NOSSA SEÇÃO COMO MAIN-PROC *****
19 MAIN-PROC SECTION.
20 ***** AGORA VOU CRIAR MEUS PARAGRAFOS *****
21 P001-PROC1.
22     MOVE 'P001-PROC1'    TO WS-EXIBIR
23     DISPLAY WS-EXIBIR
24     PERFORM SEC-PROC.
25 P001-PROC2.
  
```

Navigation



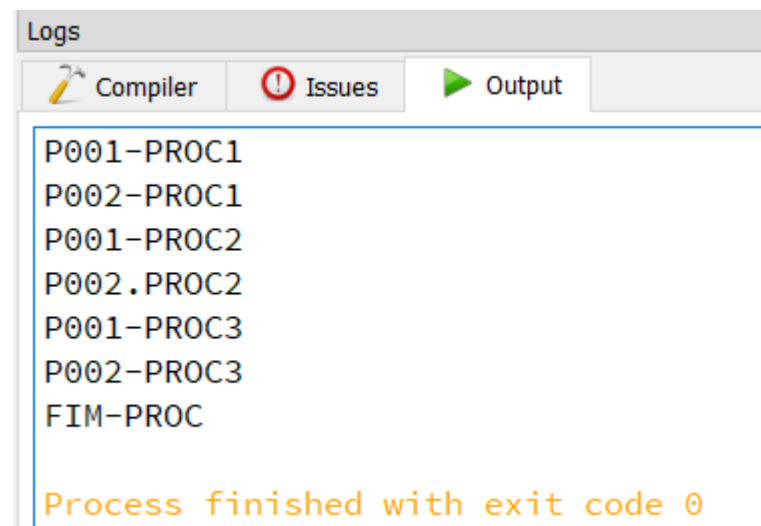
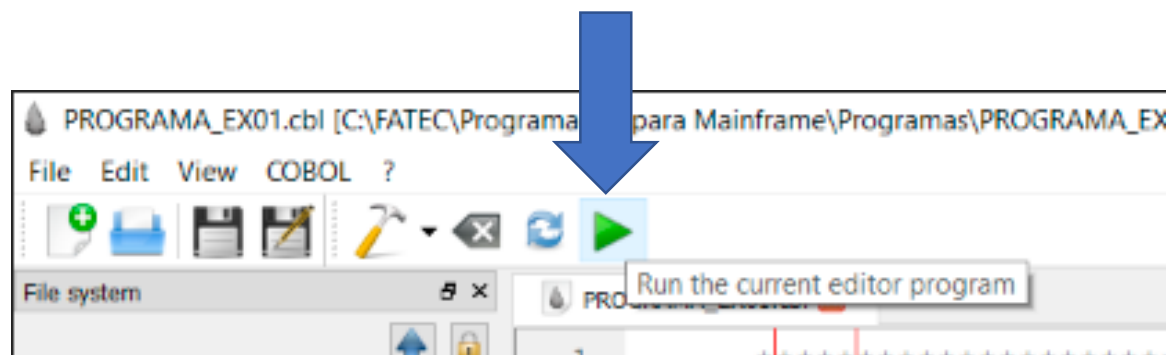
Logs

Compiler Issues Output

Type	File name	Line	Description
1 Info	PROGRAMA_EX01.cbl	-	Compilation succeeded (output: C:\FATEC\Programação para Mainframe\Programas\bin\PROGRAMA_EX01.exe)

Introdução ao COBOL

Agora vamos executar e verificar o resultado final.



PROGRAMA_EX01.cbl [C:\FATEC\Programação para Mainframe\Programas\PROGRAMA_EX01.cbl] - OpenCobolIDE 4.7.6

File Edit View COBOL ?



File system

- ▼ Aula 02
 - ~\$Aula_Mainframe_02.p...
 - Aula_Mainframe_02.pptx
- ▼ Programas
 - PROGRAMA_EX01.cbl

PROGRAMA_EX01.cbl

```

1 *****
2 * Author: CLAUDIO BENOSSE
3 * Date: 03/09/2021
4 * Purpose: NOSSO PRIMEIRO PROGRAMA EM COBOL
5 * Tectonics: cobc
6 *****
7 IDENTIFICATION DIVISION.
8 PROGRAM-ID. PROGRAMA_EX01.
9 ENVIRONMENT DIVISION.
10 CONFIGURATION SECTION.
11 INPUT-OUTPUT SECTION.
12 DATA DIVISION.
13 FILE SECTION.
14 WORKING-STORAGE SECTION.
15     77 WS-EXIBIR    PIC X(20) VALUE SPACES.
16 PROCEDURE DIVISION.
17 ***** AQUI VAMOS INICIAR A NOSSA PROGRAMAÇÃO *****
18 ***** VAMOS CRIAR A NOSSA SEÇÃO COMO MAIN-PROC *****
19 MAIN-PROC SECTION.
20 ***** AGORA VOU CRIAR MEUS PARAGRAFOS *****
21 P001-PROC1.
22     MOVE 'P001-PROC1'    TO WS-EXIBIR
23     DISPLAY WS-EXIBIR
24     PERFORM SEC-PROC.

```

Navigation

No data

Logs

Compiler Issues Output

```

C:\FATEC\Programação para Mainframe\Programas\bin\PROGRAMA_EX01.exe
P001-PROC1
P002-PROC1
P001-PROC2
P002-PROC2
P001-PROC3
P002-PROC3
FIM-PROC

Process finished with exit code 0

```

Introdução ao COBOL

Podemos desenvolver e testar nossos programas em Cobol de forma Online através do link:

<https://www.jdoodle.com/execute-cobol-online/>

Introdução ao COBOL – Atividades

Realizar pesquisa sobre:

1. Formato de dados das variáveis, incluindo a Clausula **USAGE**;
2. Comando **COMPUTE**;
3. Comando **ACCEPT**;
4. Comando **DISPLAY**;
5. Comando **GO TO**;
6. Instrução **STOP RUN**;
7. Comandos **NEXT SENTENCE** e **CONTINUE**;
8. Comando **EVALUATE**;

"Saber muito não lhe torna
inteligente.

A inteligência se traduz na
forma que você reconhece,
julga, maneja e, sobretudo,
onde e como aplica esta
informação"



Carl Sagan

Obrigado!

Se precisar ...

Prof. Claudio Benossi

claudio.benossi@fatec.spg.gov.br

