

COBOL: O Java EE do século 19

Cobol mutila a mente.

Java EE mutila a alma.

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

COBOL

A história que me contaram...



- Criado por Grace Hopper
 - Almirante da marinha dos EUA
- Baseada em duas linguagens:
 - Flow-Matic (Criada por Grace)
 - COMTRAN (Criada pela IBM)
- Influenciou outras duas linguagens:
 - PL/I da IBM
 - ABAP da SAP, utilizada até hoje no ERP da empresa;

COBOL

A história que me contaram...



- Especificada nos anos 50;
- Padronizada nos anos 60 por causa dos diversos dialetos de empresas diferentes (similar ao que ocorreu com o SQL)
- ANS COBOL 1968, 1974, 1985
- COBOL 2002 e COBOL Orientado a Objetos
 - Antes da definição final do COBOL 2002, vários fabricantes implementaram OO baseado nos rascunhos de 1997 do comitê da ISO
 - Recursividade, funções do usuário, unicode, novos tipos de dados (float-point, bit, boolean), ponteiros, Orientação a objetos, interoperabilidade com .NET e Java
- COBOL 2014
 - Sobrecarga de métodos
 - Tabelas dinâmicas (que estava no rascunho de 2002)

COBOL

Como anda nos dias atuais?

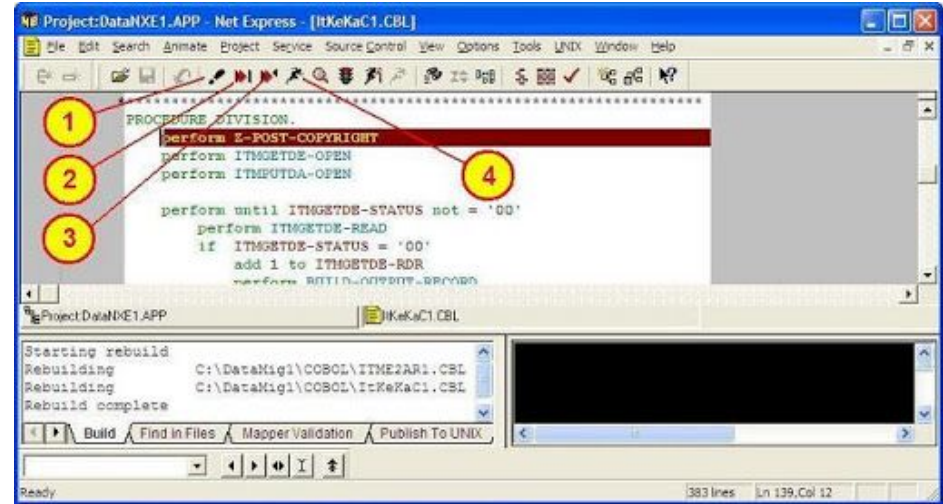


- Maior parte dos códigos roda em Mainframe (Sistema operacional IBM zOS)
- Em 2017, 43% dos sistemas bancários utilizavam COBOL, cerca 220 bilhões de linhas de código
- Como aprender COBOL nos dias atuais
 - IBM Master The Mainframe - competição de skills de mainframe. Acompanham pequenos tutoriais de, entre outras coisas, código COBOL
 - Learning COBOL Programming with VSCode - Curso com 16h de conteúdo

COBOL

Como programar nos dias atuais?

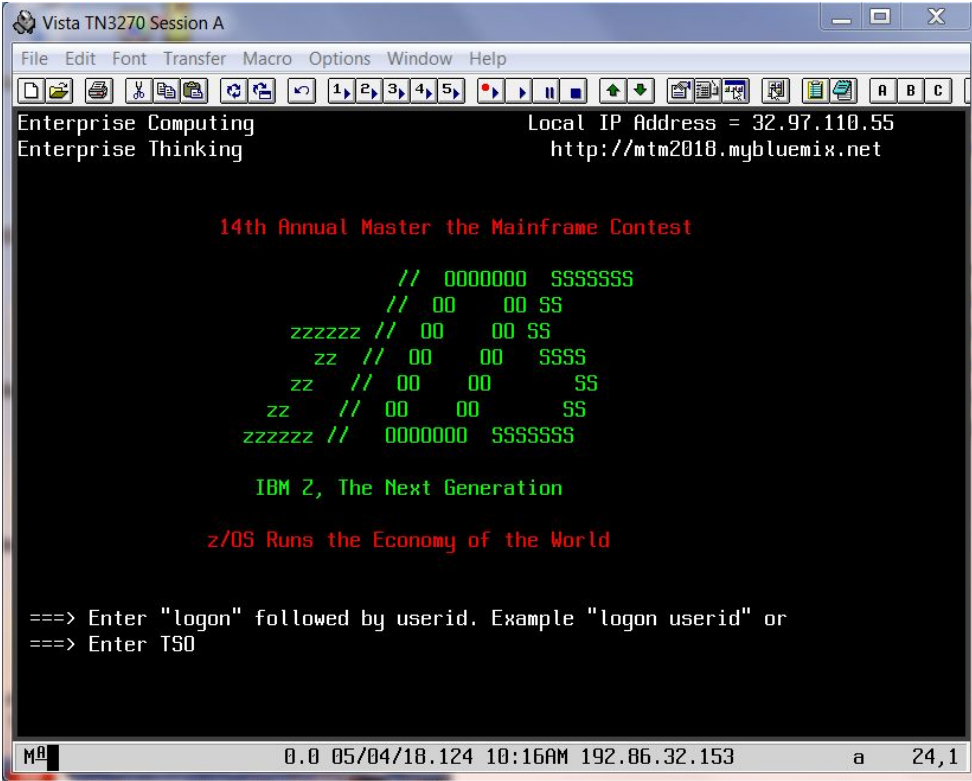
Utilizando IDEs proprietárias da Microfocus, Fujitsu e outras



COBOL

Como programar nos dias atuais?

Direto no mainframe, utilizando um terminal como o Vista TN3270



The screenshot shows a terminal window titled "Vista TN3270 Session A". The window has a menu bar with "File", "Edit", "Font", "Transfer", "Macro", "Options", "Window", and "Help". Below the menu bar is a toolbar with various icons. The main display area shows the following text:

```
Enterprise Computing
Enterprise Thinking

Local IP Address = 32.97.110.55
http://mtm2018.mybluemix.net

14th Annual Master the Mainframe Contest

      // 0000000 SSSSSSS
      // 00 00 SS
zzzzzz // 00 00 SS
      zz // 00 00 SSSS
      zz // 00 00 SS
      zz // 00 00 SS
zzzzzz // 0000000 SSSSSSS

      IBM Z, The Next Generation

z/OS Runs the Economy of the World

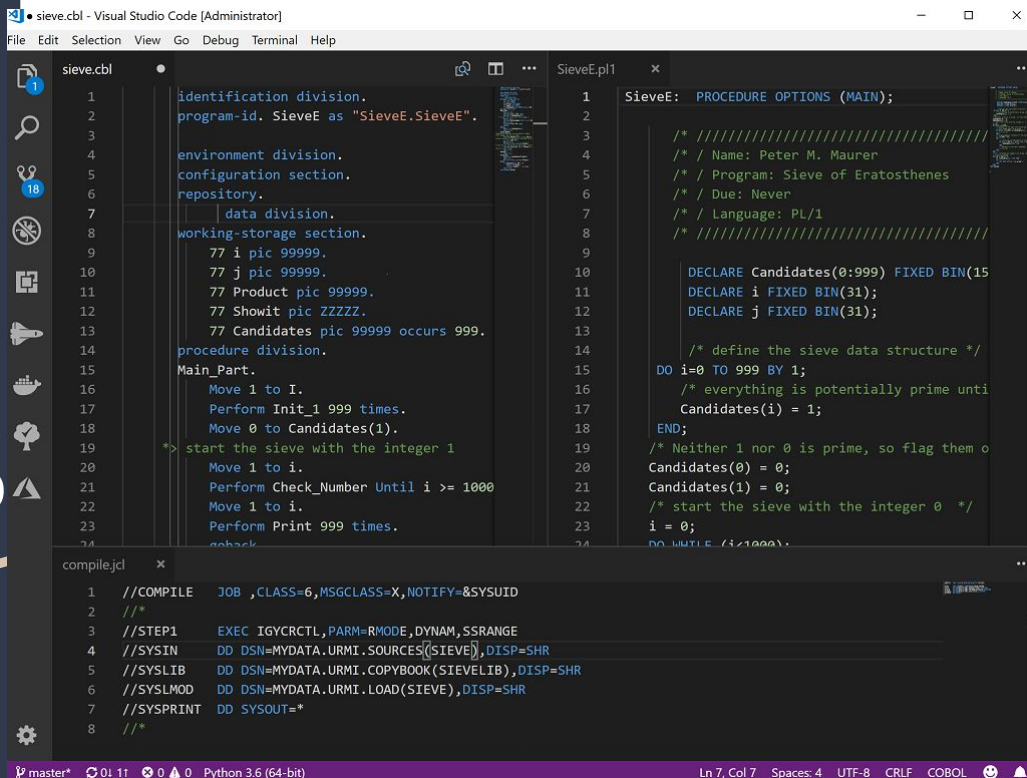
===> Enter "logon" followed by userid. Example "logon userid" or
===> Enter TSO
```

The status bar at the bottom of the window displays the following information: "M8", "0.0 05/04/18.124 10:16AM 192.86.32.153", "a", and "24,1".

COBOL

Como programar nos dias atuais?

No VSCode, utilizando plugins da IBM (como o Zowe) acessando mainframe para realização dos Jobs



The screenshot displays the Visual Studio Code interface with two open files. The left file, `sieve.cbl`, contains COBOL source code for a sieve program. The right file, `SieveE.pl1`, shows the same code with syntax highlighting. The bottom panel displays a JCL job named `compile.jcl` for compiling the COBOL program.

```
sieve.cbl
1  identification division.
2  program-id. SieveE as "SieveE.SieveE".
3
4  environment division.
5  configuration section.
6  repository.
7
8  data division.
9
10 working-storage section.
11   77 i pic 99999.
12   77 j pic 99999.
13   77 Product pic 99999.
14   77 Showit pic ZZZZZ.
15   77 Candidates pic 99999 occurs 999.
16
17 procedure division.
18 Main_Part.
19   Move 1 to i.
20   Perform Init 1 999 times.
21   Move 0 to Candidates(1).
22   *> start the sieve with the integer 1
23   Move 1 to i.
24   Perform Check_Number Until i >= 1000
25   Move 1 to i.
26   Perform Print 999 times.
27   goback.

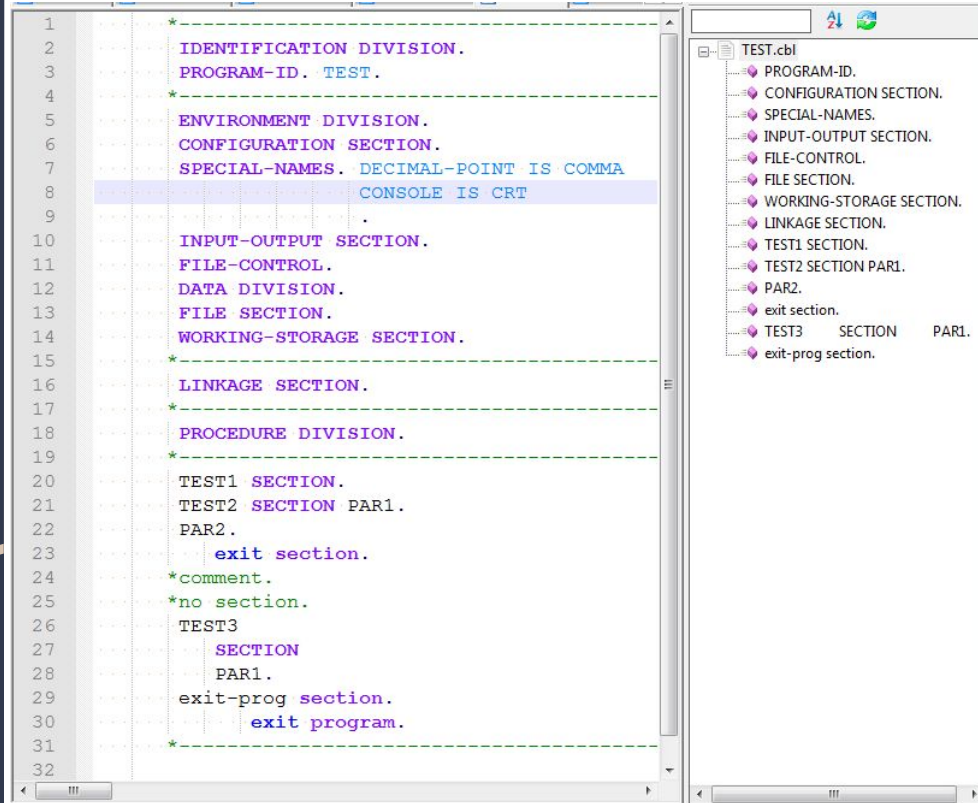
SieveE.pl1
1 SieveE: PROCEDURE OPTIONS (MAIN);
2
3 /* ////////////////////////////////////////////////////
4 /* Name: Peter M. Maurer
5 /* Program: Sieve of Eratosthenes
6 /* Due: Never
7 /* Language: PL/1
8 /* ////////////////////////////////////////////////////
9
10 DECLARE Candidates(0:999) FIXED BIN(15)
11 DECLARE i FIXED BIN(31);
12 DECLARE j FIXED BIN(31);
13
14 /* define the sieve data structure */
15 DO i=0 TO 999 BY 1;
16   /* everything is potentially prime until
17   Candidates(i) = 1;
18 END;
19 /* Neither 1 nor 0 is prime, so flag them 0
20 Candidates(0) = 0;
21 Candidates(1) = 0;
22 /* start the sieve with the integer 0 */
23 i = 0;
24 DO WHILE (i < 1000);
```

```
compile.jcl
1 //COMPILE JOB ,CLASS=6,MSGCLASS=X,NOTIFY=&SYSUID
2 /**
3 //STEP1 EXEC IGYCRCTL,PARM=RMODE,DYNAM,SSRANGE
4 //SYSIN DD DSN=MYDATA.URMI.SOURCES(SIEVE),DISP=SHR
5 //SYSLIB DD DSN=MYDATA.URMI.COPYBOOK(SIEVELIB),DISP=SHR
6 //SYSLMOD DD DSN=MYDATA.URMI.LOAD(SIEVE),DISP=SHR
7 //SYSPRINT DD SYSOUT=*
8 /**
```

COBOL

Como programar nos dias atuais?

Usando notepad++ e compilando com GnuCOBOL



The image shows a Notepad++ window with a COBOL program named 'TEST.cbl'. The program is structured as follows:

```
1 IDENTIFICATION DIVISION.  
2 PROGRAM-ID. TEST.  
3  
4  
5 ENVIRONMENT DIVISION.  
6 CONFIGURATION SECTION.  
7 SPECIAL-NAMES. DECIMAL-POINT IS COMMA  
8                  CONSOLE IS CRT  
9  
10 INPUT-OUTPUT SECTION.  
11 FILE-CONTROL.  
12 DATA DIVISION.  
13 FILE SECTION.  
14 WORKING-STORAGE SECTION.  
15  
16 LINKAGE SECTION.  
17  
18 PROCEDURE DIVISION.  
19  
20 TEST1 SECTION.  
21 TEST2 SECTION PAR1.  
22 PAR2.  
23     exit section.  
24 *comment.  
25 *no section.  
26 TEST3  
27     SECTION  
28     PAR1.  
29 exit-prog section.  
30     exit program.  
31  
32
```

On the right side, the Project Explorer shows the structure of the 'TEST.cbl' file:

- PROGRAM-ID.
- CONFIGURATION SECTION.
- SPECIAL-NAMES.
- INPUT-OUTPUT SECTION.
- FILE-CONTROL.
- FILE SECTION.
- WORKING-STORAGE SECTION.
- LINKAGE SECTION.
- TEST1 SECTION.
- TEST2 SECTION PAR1.
- PAR2.
- exit section.
- TEST3 SECTION PAR1.
- exit-prog section.

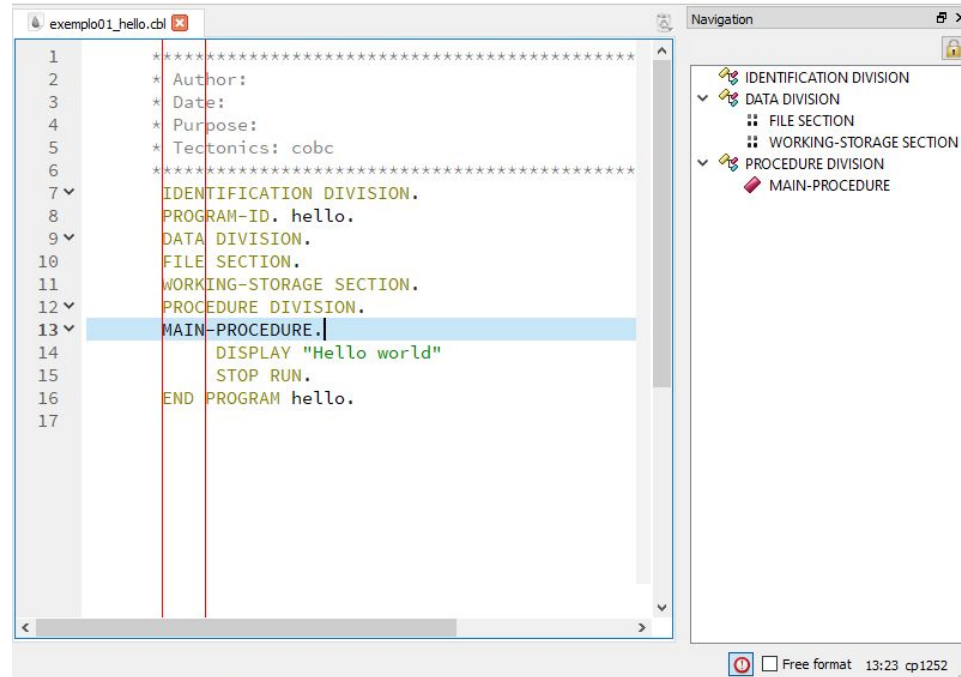
COBOL

Como programar
nos dias atuais?

Utilizando o
OpenCOBOL IDE

- Linux, Mac OS e Windows
- Suporte a SQL

\\exemplo01_hello.cbl] - OpenCobolIDE 4.7.6



```
1 *****
2 * Author:
3 * Date:
4 * Purpose:
5 * Tectonics: cobc
6 *****
7 IDENTIFICATION DIVISION.
8 PROGRAM-ID. hello.
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 hello.
17
```

Navigation

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

Free format 13:23 cp1252

COBOL

Bora pro código

CARAAAAI

SINTAXE BÁSICA

- **Comentários**
 - **asterisco (*) na coluna 7**
 - **asterisco + GT (*>) em qualquer coluna**
- **Código inicia na coluna 8**

COBOL
Bora pro código
CARAAAAI

ESTRUTURA BÁSICA DE UM PROGRAMA COBOL

- DIVISIONS
- SECTIONS
- PARÁGRAFOS

COBOL
Bora pro código
CARAAAAI



RELÔU O ORDI

IDENTIFICATION DIVISION.
PROGRAM-ID. hello.

PROCEDURE DIVISION.
MAIN-PROCEDURE.

DISPLAY "Hello world"
STOP RUN.

COBOL

Bora pro código

CARAAAAI

VARIÁVEIS

- **PICTURE CLAUSE**
 - **PIC X**
 - **PIC 9**
 - **PIC OTRAS COISO**
- **NÍVEIS DE VARIÁVEL**
 - **01**
 - **77**

COBOL
Bora pro código
CARAAAAI

ESTRUTURAS

- IF/ELSE
- SWITCH
- LOOPS
- GO TO (DE CU É ROLA)

COBOL
Bora pro código
CARAAAAI

ENTRADA DE DADOS

- **ACCEPT**
- **ATRAVÉS DE ARGUMENTOS (E TU NÃO DISSE QUE NÃO TINHA, MIZERA?)**

COBOL
Bora pro código
CARAAAAI

ARQUIVOS

- INDEXADOS
- SEQUENCIAIS
- TEXTO
- CSV
- OTRO MÓI DE COISA

FIM

