

Formação : COBOL Mainframe

Professor: Vagner Bellacosa

Disciplina: COBOL & Além

COBOL



1) Vagner Bellacosa seu facilitador

Analista programador desde 1989, onde comecei como auxiliar e tecnólogo em processamento de dados desbravando os primórdios da computação brasileira, um eterno aprendiz em processos Mainframe..

Desde então trabalhei em centenas de projetos, em 4 países e dezenas de instituições financeiras, ora como funcionário, consultor externo e freelancer.

A mais memorável foi o Banco REAL, uma verdadeira escola, onde aprendi muito e tive a oportunidade de participar no aliciante Projeto Y2K, o temível bug do milênio.

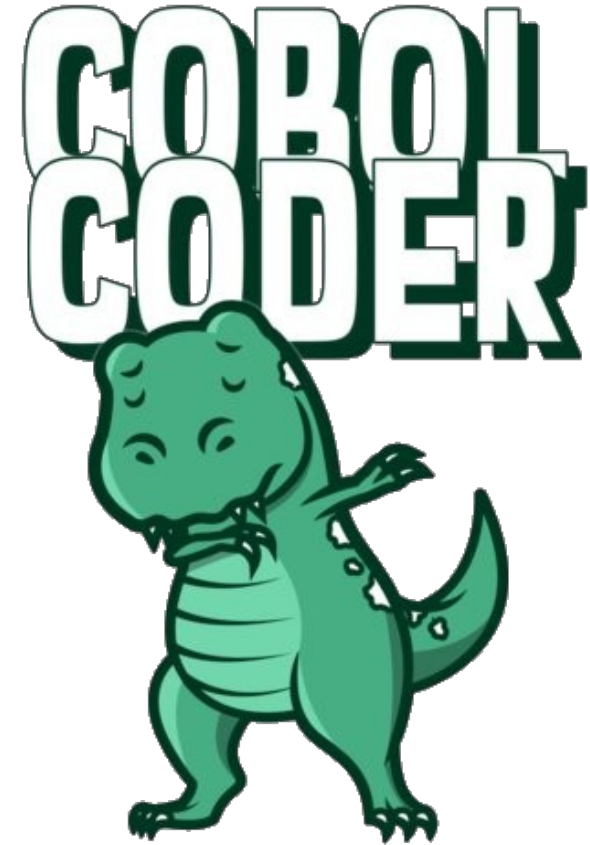
Participei em projetos na CESP, Fundação CESP, Transbrasil, ABSA, Real Seguros, BPN, Skandia, DGITA, BES, CGD, BPI, Barclay, Skandia, IBM Italia, Sistemi Informativi, Unicredit, Zurich Assicurazione, Banco Safra e Banco Itaú.

Atualmente trabalho na Spread no Projeto BRB em Brasília, via remoto.

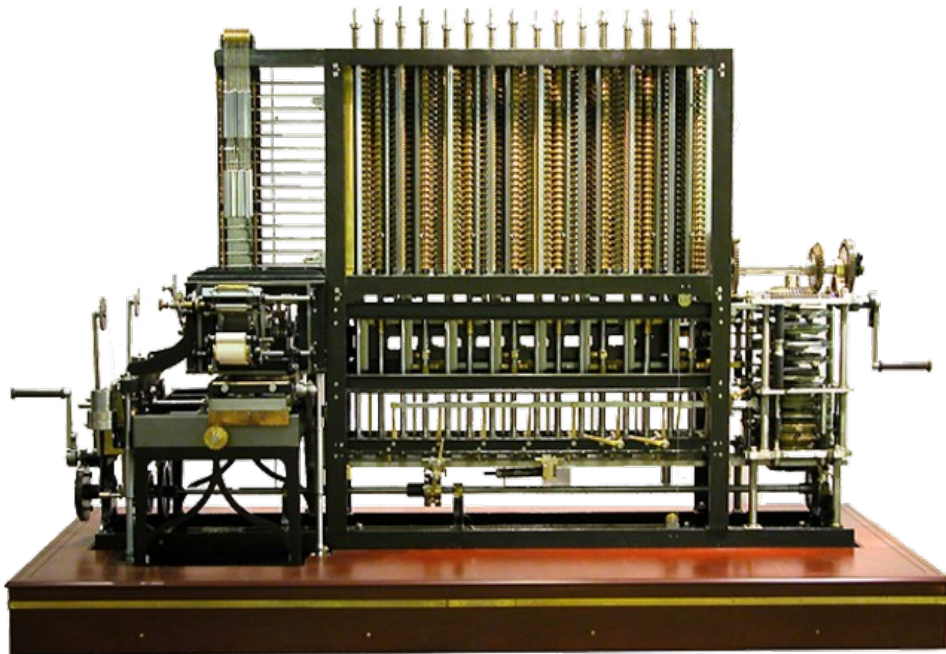
Logica de Programação

Modulo 3.02 – Linguagem de Programação

- | | |
|----------------------------------|--------------------------------|
| 01) Hermann Hollerith | 14) Comandos de Decisão |
| 02) Tear automático | 15) Comandos de Repetição |
| 03) Computador Humano | 16) String |
| 04) Evolução constante | 17) Manipulação de String |
| 05) FORTRAN | 18) COBOL |
| 06) Linguagens de Programação | 19) Ferramenta visual |
| 07) Refinamentos sucessivos | 20) WORKFLOW |
| 08) Componentes Básicos | 21) Diagrama de Fluxo de Dados |
| 09) Formato de um programa | 22) Dúvidas |
| 10) Interface de Desenvolvimento | |
| 11) Comando básicos | |
| 12) Boas praticas | |
| 13) Atribuição, Entrada e Saída | |



Logica de Programação



Modulo 3.02 – Linguagem de Programação

Estamos sempre em evolução, tudo muda, nada é para sempre, uma ideia do passado, volta numa roupagem nossa atendendo novas necessidades.

O primeiro programa surgiu em 1843, fruto do trabalho da jovem Ada Lovelace e seu algoritmo para calcular o numero de Bernoulli.

Logica de Programação



Hermann Hollerith

Adaptou os cartões perfurados utilizado em tecelagem para calcular e agilizar a entrega do Censo de 1890.

Sua empresa anos mais tarde se transformou na IBM e seus mainframes dominaram o mundo.

Logica de Programação

Tear automático



Em 1800 para baratear o custo de confecção dos tecidos e eliminar o trabalho repetitivo aborrecido, Joseph Marie Jacquard inventou um tear automático, que usava cartões perfurados para controlar o padrão e a troca de novelos.

Logica de Programação

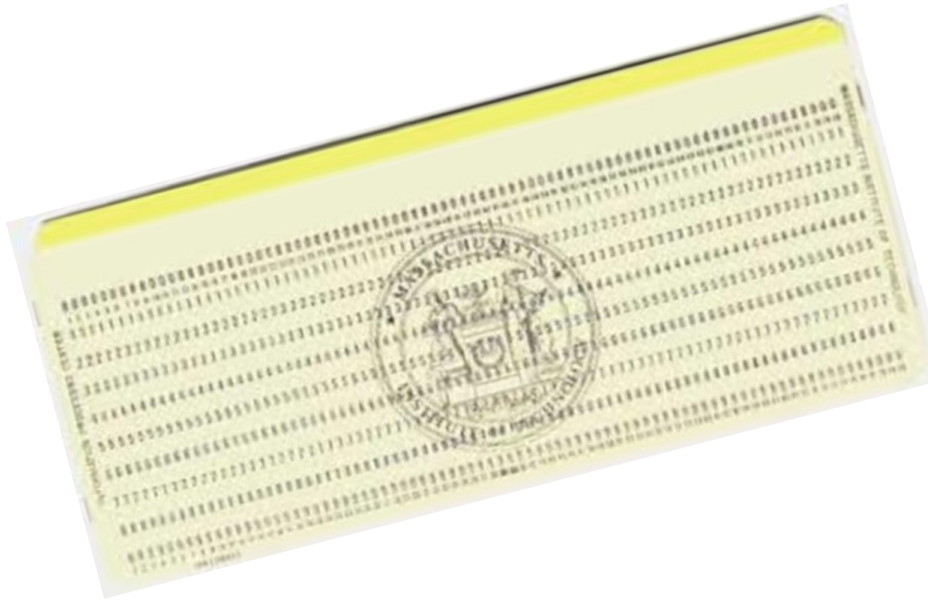
Computador Humano



Antes do adventos dos computadores, existiam os computadores humanos, pessoas especializadas em cálculos matemáticos Durante a guerra milhares de pessoas trabalharam calculando tabuas balísticas

Logica de Programação

Evolução constante.



Juntando as diversas áreas de conhecimento, impulsionada por uma necessidade sem precedentes na historia da humanidade.

O esforço humano para vencer e terminar com a Segunda Guerra Mundial culminaram com os computadores e as linguagens de programação.

Logica de Programação

FORTRAN



Primeira linguagem de programação comercial criada em 1957 pela IBM.

Utilizada no mainframe IBM 704 e seu nome original é: IBM Mathematical FORMula TRANslation System.

Logica de Programação

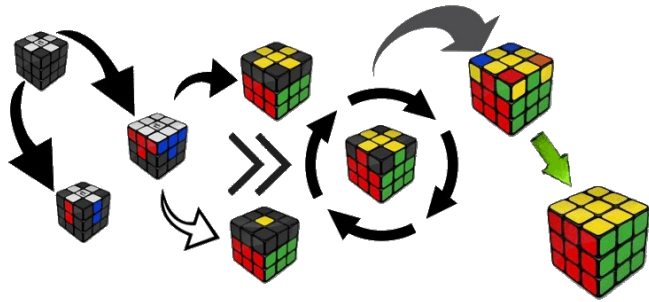
Linguagens de Programação

Para todos os gostos...



Logica de Programação

Refinamentos sucessivos



Um programa de computador é o resultado de refinamentos sucessivos, até o ponto onde todas as instruções são entendíveis e resultam no resultado esperado e desejado.

Logica de Programação

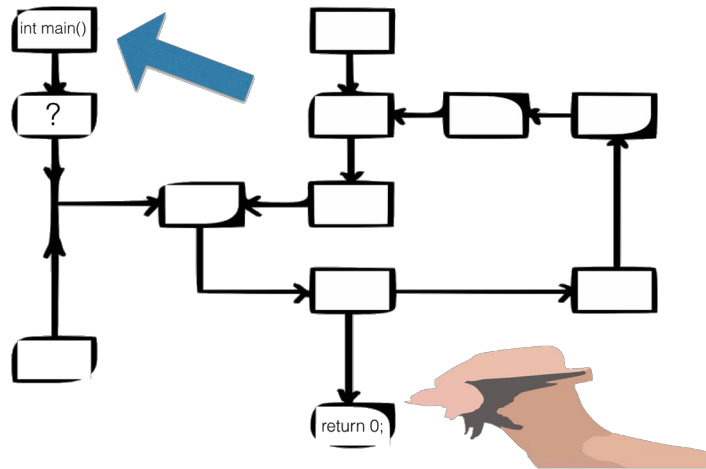
Componentes Básicos



Uma linguagem de programação é uma ciência humana, um experimento de comunicação criada por humanos para dialogar com maquinas, uma linguagem artificial, por isso tem uma serie de elementos obrigatórios.

Logica de Programação

Formato de um programa.

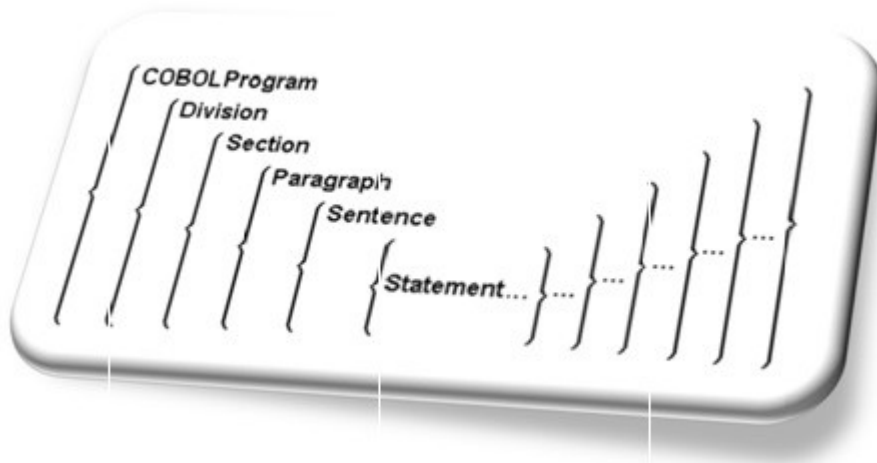


Um programa necessita de distintas partes, que no conjunto resultam no código-fonte.

Lembre-se sempre dos comandos, das variáveis, dos blocos de rotinas e funções.

Logica de Programação

Formato de um programa.



Na gravura vemos o formato padrão de um programa COBOL.

Sendo composto por Division, Section, Paragraph, Sentence e Statement.

No Statement temos os diversos comandos que executaram nossas ordens e entregarão o resultado esperado.

Logica de Programação

Interface de Desenvolvimento

```
Enterprise Computing      Local IP Address = 195.212.29.164
Enterprise Thinking      http://mtm2019.mybluemix.net

December 31, 2019 was last day of Master the Mainframe contest

      // 0000000 SSSSSS
      // 00 00 SS
zzzzzz // 00 00 SS
      zz // 00 00 SSSS
      zz // 00 00 SS
      zz // 00 00 SS
zzzzzz // 0000000 SSSSSS

      IBM Z, The Next Generation

      z/OS Runs the Economy of the World

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

M6 a 24/001
```

No desenvolvimento de software, a IDE ou Integrated Development Environment, Interface de Desenvolvimento Integrado é o editor onde iremos construir nosso código

No mainframe existem inúmeras possibilidades, as mais conhecidas são o TSO, o Roscoe e o Zowe.

Logica de Programação

Interface de Desenvolvimento



Na IDE encontramos inúmeras ferramentas de produtividade que irão nos auxiliar no processo de codificação.

No ambiente mainframe como exemplo temo o Copia e Cola, a Classificação, conjunto de cores para melhor identificação dos comandos, arquivos de bibliotecas e pesquisa de comandos.

Logica de Programação

Comando básicos

```

EDIT      MTH.COBOL.SRCLIB(EVALALSO) - 01.00
Command ==>
=COLS>  ----+-----1-----+-----2-----+-----3-----+-----4-----+-----
*****  ***** Top of Data *****
000100      IDENTIFICATION DIVISION.
000200      PROGRAM-ID. EVALALSO.
000300      ENVIRONMENT DIVISION.
000400      DATA DIVISION.
000500      WORKING-STORAGE SECTION.
000600          01 AGE                                PIC 9(03).
000700          01 GENDER                             PIC X(01).
000800      PROCEDURE DIVISION.
000900          ACCEPT AGE.
001000          ACCEPT GENDER.
001100          EVALUATE TRUE ALSO TRUE
001200              WHEN AGE > 018 ALSO GENDER = 'M'
    
```

Lembrando a linguagem de programação necessita poder tratar as variáveis de acordo com seu tipo.

Operadores matemáticos

Operadores de comparação

Operadores Lógicos

Estrutura condicionais

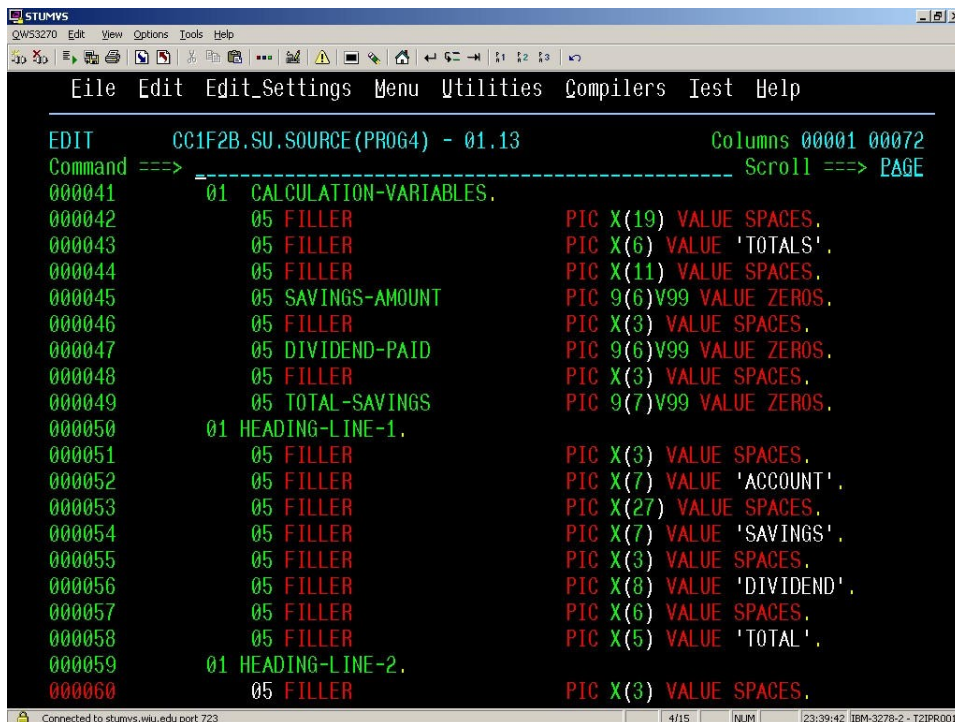
Estruturas de repetição

Funções

Procedimentos

Logica de Programação

Boas praticas



```

EDIT      CC1F2B.SU.SOURCE (PROG4) - 01.13      Columns 00001 00072
Command ===> ----- Scroll ===> PAGE
000041      01  CALCULATION-VARIABLES.
000042          05  FILLER                      PIC X(19) VALUE SPACES.
000043          05  FILLER                      PIC X(6) VALUE 'TOTALS'.
000044          05  FILLER                      PIC X(11) VALUE SPACES.
000045          05  SAVINGS-AMOUNT              PIC 9(6)V99 VALUE ZEROS.
000046          05  FILLER                      PIC X(3) VALUE SPACES.
000047          05  DIVIDEND-PAID               PIC 9(6)V99 VALUE ZEROS.
000048          05  FILLER                      PIC X(3) VALUE SPACES.
000049          05  TOTAL-SAVINGS               PIC 9(7)V99 VALUE ZEROS.
000050      01  HEADING-LINE-1.
000051          05  FILLER                      PIC X(3) VALUE SPACES.
000052          05  FILLER                      PIC X(7) VALUE 'ACCOUNT'.
000053          05  FILLER                      PIC X(27) VALUE SPACES.
000054          05  FILLER                      PIC X(7) VALUE 'SAVINGS'.
000055          05  FILLER                      PIC X(3) VALUE SPACES.
000056          05  FILLER                      PIC X(8) VALUE 'DIVIDEND'.
000057          05  FILLER                      PIC X(6) VALUE SPACES.
000058          05  FILLER                      PIC X(5) VALUE 'TOTAL'.
000059      01  HEADING-LINE-2.
000060          05  FILLER                      PIC X(3) VALUE SPACES.
  
```

Legibilidade: Os códigos devem ser legíveis para qualquer desenvolvedor entender.

Código limpo: Devemos criar códigos objetivos e limpos. **Nomes descritivos:** Os nomes das nossas variáveis, constantes e funções devem ser claros e descritivos.

Atenção ao criar comentários: Não crie comentários desnecessários nos códigos.

Indentação: Os códigos devem ser indentados. Dessa forma, conseguimos identificar os blocos e escopos.

Logica de Programação

```

EDIT      SYSADM.DEMO.SRCLIB(PROG05) - 01.01      Columns 00001 00060
Command ==>                                     Scroll ==> CSR
***** ***** Top of Data *****
=COLS> -----1-----2-----3-----4-----5-----6-----+
000100 000100 IDENTIFICATION DIVISION.
000200 000200 PROGRAM-ID. PROG05.
000300 000300*-----*
000400 000400* PROGRAM      : PROG05          BILL PROCESSING *
000500 000500*-----*
000600 000600* SYSTEM       : ZOS DEMO PKG      *
000700 000700* DESCRIPTION : READS THE CALL TRANSACTIONS, AND MONTHLY *
000800 000800* BILL. *
000900 000900*-----*
001000 001000* FILE ACCESS *
001100 001100*-----*
001200 001200* FILENAME    INP  OUT  UPD  LRECL *
001300 001300*-----*
001400 001400* CALLTRNS    X   ---  ---   80 *
001500 001500* BILLOUT     X   ---  ---   80 *
001600 001600*-----*
001900 001900* REVISION TRAIL *
002000 002000*-----*
002100 002100* NAME        DATE    TAG    DESCRIPTION *
002200 002200*-----*
002300 002300* Q CHUNAWALA 10/02/10 QSC001 CODE WRITTEN TO *
002400 002400* AND GENERATE BILL. *
002500 002500*-----*
002600 002600*

```

Atribuição, Entrada e Saída

A Linguagem deve prever comandos para atribuição de variáveis, garantir as operações de Input e Output.

No COBOL temos a Environment Division e o Data Division tem como função justamente cuidar dos IO's e variáveis

Logica de Programação

Comandos de Decisão

```

File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT SCHOOL.COBOL(PAISES) - 01.06 SNAP macro error
Command ==> Scroll ==> CSR
002300 *****
002400 IDENTIFICATION DIVISION.
002500 PROGRAM-ID.    PAISES.
002600 *----- ENVIRONMENT DIVISION *****
002700 ENVIRONMENT DIVISION.
002800 * CONFIGURATION SECTION *****
002900 CONFIGURATION SECTION.
003000 SPECIAL-NAMES.
003100     DECIMAL-POINT IS COMMA.
003200 * INPUT-OUTPUT SECTION *****
003300 INPUT-OUTPUT SECTION.
003400 FILE-CONTROL.
003500     SELECT IN-FILE ASSIGN TO ENTRADA
003600     ORGANIZATION IS SEQUENTIAL
003700     ACCESS MODE IS SEQUENTIAL
003800     FILE STATUS IS WS-FS-IN.
003900     SELECT OUT-FILE ASSIGN TO SALIDA
004000     ORGANIZATION IS SEQUENTIAL
004100     ACCESS MODE IS SEQUENTIAL
004200     FILE STATUS IS WS-FS-OUT.
004300 *----- DATA DIVISION *****

```

São aqueles comandos que controlam o fluxo do programa, direcionando e adequando o processamento.

Em COBOL temos dois comandos principais o IF e o EVALUATE.

Logica de Programação

Comandos de Repetição

```

EDIT      MATEPK.IKJEFT01.JCLLIB(DB2RUN) - 01.00      Columns 00001 00072
Command ==>      Scroll ==> CSR
=COLS> -----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----
***** Top of Data *****
000010 //MTHUSRI JOB (123), 'MTHUSR', CLASS=A, MSGCLASS=A, MSGLEVEL=(1,1),
000020 //          NOTIFY=&SYSUID
000030 //*****
000031 //* COBOL DB2 RUN JCL
000050 //*****
000060 //BIND EXEC PGM=IKJEFT01, DYNAMNBR=20, REGION=4096K
000070 //STEPLIB DD DSN=XXXXXX.DB2.SDSNEXIT, DISP=SHR
000080 //          DD DSN=XXXXXX.DB2.SDSNLOAD, DISP=SHR
001100 //SYSPRINT DD SYSOUT=*
001200 //SYSTSPRT DD SYSOUT=*
001300 //SYSUDUMP DD SYSOUT=*
001400 //SYSTSIN DD *
001500 DSN SYSTEM (DB01 )
001600 RUN PROGRAM (COBOLDB) -
001700 PLAN (DBPLAN ) -
001800 LIBRARY ('MTHUSR.LOADLIB')
002500 END
002600 /*
002700 //SYSOUT DD SYSOUT=*
***** Bottom of Data *****

```

Em muitas situações nosso programa precisará repetir determinando sequencia de comandos, executando-os até que uma condição seja atingida.

No COBOL temos o polivalente comando **PERFORM**, que nós auxiliar em repetições **FOR** e **WHILE**.

Logica de Programação

STRING

```

92 read in-file into in-record
93 at end move "Y" to eof-switch
94 not at end compute rce-counter = rce-counter + 1;
95 end-read.
96
97 unstring in-record delimited by spaces into temp-string.
98 (unstring temp-s
99 geoid in rceord-
100 sumlev in rceord-table(rce-counter),
101 state in rceord-table(rce-counter).
102 * move 23513 to state.
103 * Going to display the values in each structure then read in new ones
104 * it seems like the above is overwriting the same table location
  
```

Name	Value
IN-FILE	Open Input Last Status: 0/0
REC-COUNTER	1
IN-RECORD	{Length = 42}: "4001942600,140,4"
GEOID	{Length = 3}
SUMLEV	{Length = 3}
STATE	{Length = 3}

Algo que sera de uso obrigatório em quase todos os programas, será o tratamento de blocos de texto, ou STRINGS.

No COBOL temos varias maneiras de trabalhar com bloco de texto, desde o REDEFINE, UNSTRING até a função CONCATENATE.

Logica de Programação

Manipulação de String

```

File Edit Edit_Settings Menu Utilities Compilers Test Help
EDIT KC02289.MGMT3310.COBOL(YOUTUBE) - 01.02 Columns 00001 00072
***** Top of Data *****
000001 IDENTIFICATION DIVISION.
000002 PROGRAM-ID. YOUTUBE.
000003 *****
000004 *
000005 *****
000006 ENVIRONMENT DIVISION.
000007
000008 DATA DIVISION.
000009 WORKING-STORAGE SECTION.
000010
000011 PROCEDURE DIVISION.
000012
000013 DISPLAY "WELCOME YOUTUBE TO THE IBM MAINFRAME!".
000014 GOBACK.
000015
***** Bottom of Data *****
Command ==> HI_COBOL Scroll ==> CSR
F1=Help F2=Split F3=Exit F5=Rfind F6=Rchange F7=Up
F8=Down F9=Swap F10=Left F11=Right F12=Cancel
MA a 22/023

```

Toda linguagem de programação necessita de comandos para tratar cadeias de caracteres, obtendo partes do texto, agrupando-o ou mesmo substituindo caracteres.

Logica de Programação



COBOL 1959
COBOL 1968
COBOL 1974
COBOL 1985
COBOL 2002
COBOL 2014

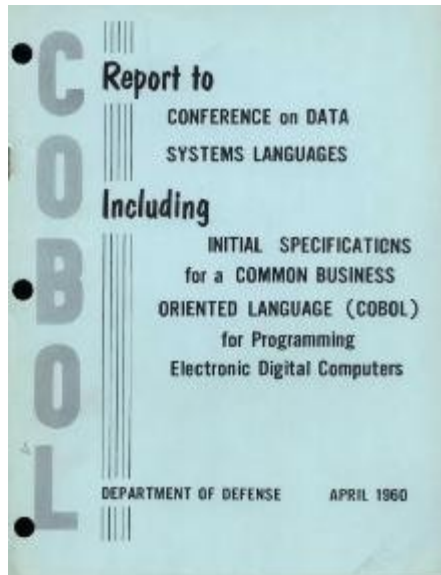
COBOL 01-02-1959

64 anos sendo codificada, gerando riquezas e processando dados de milhões de usuários

Deixando analistas e programadores de cabelos brancos ou sem eles...

Logica de Programação

COBOL

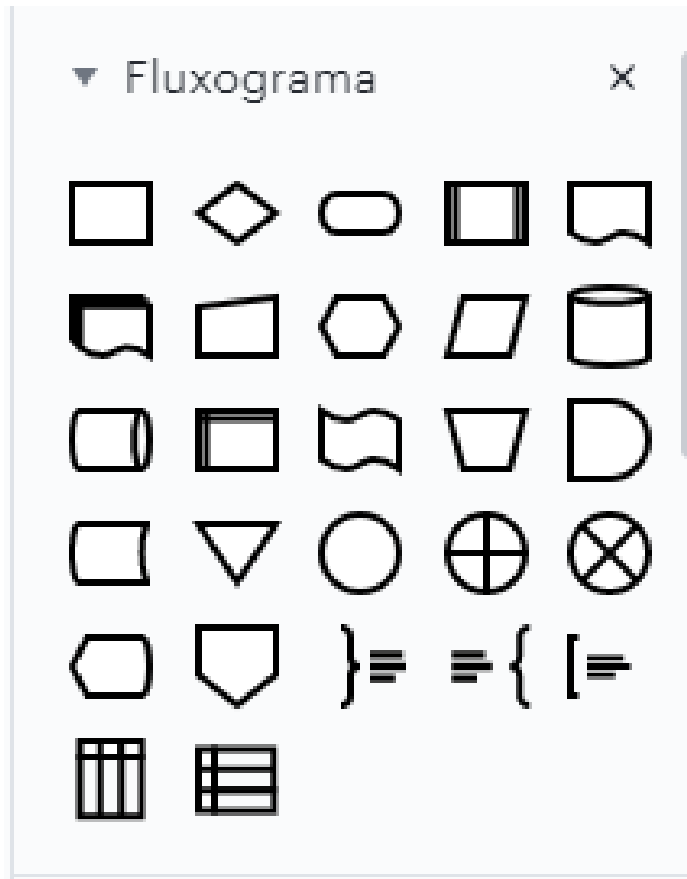


A Linguagem de Programação COBOL como foi visto no slide anterior é uma entidade viva, que continuamente recebe atualizações e adequando as melhores praticas de desenvolvimento de software.

Sendo que a sua principal vantagem é manter total compatibilidade com programa legados, bastando apenas uma nova compilação.

Logica de Programação

Ferramenta visual



O Fluxograma originalmente foi criado para a área de Engenharia.

Nos anos 70 a IBM criou uma versão de Fluxograma vocacionada para informática, que até atende nossas necessidades até os dias de Hoje.

Em CPDs poderão encontrar UMLs apesar que com a metodologia ágil tenha tirado um pouco do protagonismo destas ferramentas.

Logica de Programação

WORKFLOW



Como analistas programadores muitas vezes seremos expostos a problemas que necessitam de um fluxo de trabalho.

Passando por vários estágios até a sua conclusão, lembra sobre dividir um problema grande em pequenos problemas de fácil solução?

O Workflow auxilia enormemente nesse caso, com uma tarefa e suas mudanças de estado.

Logica de Programação

Diagrama de Fluxo de Dados

Uma outra ferramenta muito utilizada é o DFD, que registra de forma visual os dados utilizados no Sistema.

Suas origens e destinos, bem como transformações no decorrer do processamento.

Esta é a parte logica do negocio, onde os dados são o produto final.



Logica de Programação

Duvidas?

Espaço aberto para duvidas e complementos.

