BBVA

Creando Oportunidades

MANUAL COBOL 101

---- Nerdearla OCTUBRE - BBVA ----

BBVA
Creando Oportunidades

ÍNDICE

1. Introducción

- Mención a C.O.B.O.L
- Concepto de programa
- Por qué se sigue usando
- Concepto de Programación Estructurada

2. COBOL 100%

- Las 4 divisiones
- VARIABLES

3. Conceptos y sentencias más utilizadas

- Sentencias más comunes

4. Operadores lógicos

5. Condicionales

- If
- Evaluate

6. Operaciones aritméticas

- Suma
- Resta
- División
- Compute

7. Tablas y su recorrido.

BBVA
Creando Oportunidades

¿Qué es COBOL?

El lenguaje COBOL (acrónimo de **CO**mmon **B**usiness-**O**riented **L**anguage, Lenguaje Común Orientado a Negocios) fue creado en el año 1959 con el objetivo de crear un lenguaje de programación universal que pudiera ser usado en cualquier máquina, y que estuviera orientado principalmente a los negocios, para ejecutar tareas como cálculos, control de inventario o un registro de los débitos y los créditos.

¿POR QUÉ SE SIGUE USANDO?

- Auto documentación
- Buena gestión de archivos
- Buena gestión de tipo de datos para la época
- Corre en cualquier plataforma
- Mantenible, confiable, seguro y probado.

CONCEPTO DE PROGRAMA

Es un conjunto de instrucciones a través de las cuales nos comunicamos con la máquina, que se ejecutan en una determinada combinación para obtener un resultado buscado.

Los programas se clasifican en:

- **On-line:** interactúan activamente con las personas.
- **Batch:** no interactúan con las personas, generalmente se ejecutan durante la noche. Posee capacidad de procesamiento por lotes.

Cobol es un lenguaje compilado, ¿qué significa esto?

Quiere decir que a partir del código

fuente se obtiene un código objeto traducido a instrucciones para la computadora. Este módulo se guarda en una librería y se usa cada vez que se tiene que ejecutar el programa. Si se realizan cambios en el código, se debe volver a compilar.

PRINCIPIOS DE PROGRAMACIÓN ESTRUCTURADA

- **MODULAR:** Se divide en partes cada una de las cuales cumplen funciones concretas.
- **JERÁRQUICA:** Tiene un único módulo raíz con uno o más subordinados. Cada módulo tiene sólo uno superior y su superior puede tener uno o más inferiores.
- Las funciones de alto nivel son de **"Control"** y las de más bajo nivel son de **"Detalle."**
- **DESCENDENTE:** La metodología descendente consiste en efectuar una relación entre las sucesivas etapas de estructuración, de modo que se relacionen unas con otras mediante entradas y salidas de información.

BBVA
Creando Oportunidades

COBOL 100%

LAS 4 DIVISIONES DE UN PROGRAMA COBOL

1. IDENTIFICATION DIVISION:

Es el identificador del programa, lleva los datos de información: nombre de autor, fecha de compilación, etc.

```
*-----------------------------------------------------------------*
*          I D E N T I F I C A T I O N - D I V I S I O N          *
*-----------------------------------------------------------------*
IDENTIFICATION DIVISION.

PROGRAM-ID.    PROGEJEM.
AUTHOR.        AGUSTINA.
DATE-WRITTEN.  31/03/2020.
```

2. ENVIRONMENT DIVISION:

Indica los recursos de hardware donde se ejecuta el programa, así como la asignación de salida de información por medios de comunicación.

```
*-----------------------------------------------------------------*
* ENVIROMENT - DIVISION                                           *
*-----------------------------------------------------------------*
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    DECIMAL-POINT IS COMMA.
```

3. DATA DIVISION:

En esta división se establecen las variables que serán utilizadas por el sistema y la declaración de archivos.

```
*-----------------------------------------------------------------*
* DATA DIVISION                                                   *
*-----------------------------------------------------------------*
 DATA DIVISION.
*-----------------------------------------------------------------*
 FILE SECTION.
*-----------------------------------------------------------------*
* FICHEROS DE ENTRADA                                             *
*-----------------------------------------------------------------*

 FD  AGUS-FILE
     RECORDING MODE IS F
     LABEL RECORD ARE STANDARD
     BLOCK  CONTAINS  0 RECORDS
     DATA RECORD IS REG-AGUS.
     COPY COPY1234.
```

4. PROCEDURE DIVISION:

Como su nombre lo indica, se ejecutan las instrucciones codificadas.

(Acá hacemos la MAGIC)

```
*-----------------------------------------------------------------*
* PROCEDURE DIVISION                                              *
*-----------------------------------------------------------------*
 PROCEDURE DIVISION.

*-----------------------------------------------------------------*
* 000000-CONTROL                                                  *
* PARRAFO PRINCIPAL DEL PROGRAMA.                                 *
*-----------------------------------------------------------------*
 000000-CONTROL.

     PERFORM 100000-INICIO
     PERFORM 20000-PROCESO UNTIL WSS-AGUSFILE-EOF
     PERFORM 300000-FINAL.
```

- Los nombres de párrafos deben terminar con un punto.
- También se debe agregar un punto al final del código del párrafo.

```
Command ===>                                                  Scroll ===> CSR
=COLS> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7-
****** ******************************** Top of Data ********************************
000001         IDENTIFICATION DIVISION.
000002         PROGRAM-ID. PROGRAMA.
000003        * ESTE ES UN COMENTARIO                                          *
000004         ENVIRONMENT DIVISION.
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  6 Line(s) not Displayed
000011         DATA DIVISION.
000012         WORKING-STORAGE SECTION.
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  268 Line(s) not Displayed
000281         LINKAGE SECTION.
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  3 Line(s) not Displayed
000285        ***********************************************************************
000286         PROCEDURE DIVISION.
000287             PERFORM 100000-INICIO
000288             PERFORM 200000-PROCESO
```

VARIABLES EN COBOL

Se ubican en la WORKING-STORAGE section de DATA DIVISION, y se indican con el siguiente formato:

```
01 VARIABLE    PIC X(10)
               VALUE SPACES.
```

Donde 01 es el nivel, luego viene el nombre de dato, después el formato y por último el valor inicial, que es opcional.
La declaración de variables siempre finaliza con un punto.

Formato (PIC)
En COBOL debemos indicar explícitamente el formato (tipo de dato y longitud) que tendrán las variables.

Lo indicamos con la palabra PIC, el tipo de dato, y la cantidad.

Los formatos más utilizados son:
Numérico: `PIC 9(01)`
Alfanumérico: `PIC X(05)`

Nivel
Sirve para agrupar las variables en estructuras según su número, que puede estar entre 01 y 48.
Cuanto más bajo el número de

nivel, mayor será la jerarquía del campo.

Hay niveles especiales como el 88, que indican un valor lógico.

Tipos de estructuras

- Variables que no tienen campos subordinado: Se indica el formato con la palabra PIC, y puede indicarse su contenido con VALUE.
- Variables que tienen campos subordinados: Tienen un nivel inferior a sus subordinadas, y no informan formato. Son simplemente un contenedor.
- Variables lógicas: Se indican con el nivel 88, y representan todos los posibles valores que puede obtener dicho campo. No se indica PIC en los niveles 88 porque nos estamos refiriendo al contenido de su superior.

```
01 TIPO-TARJETA PIC X(02).
   88 DEBITO VALUE 'DB'.
   88 CREDITO VALUE 'CR'.
```

Constantes

En COBOL no existe una forma de declarar constantes explícitamente, por lo que se suelen designar con alguna convención en su nombre. Por ejemplo, que todos los nombres de las constantes empiecen con "C-".

SENTENCIAS MÁS COMUNES

PERFORM

Agrupamiento de sentencias COBOL según una lógica conveniente.

Razones para utilizarlo:

- Acciones que se repiten en el programa
- Prolijidad
- Reutilización
- Estructura
- Estándar
- Facilidad de mantenimiento futuro

Formato 1 - PERFORM Repetitivo controlado por condición:

PERFORM procedimiento-1 [THRU procedimiento-2] UNTIL condición

Formato 2 - PERFORM Repetitivo controlado por cantidad de ejecuciones fijas:

PERFORM procedimiento-1 [THRU procedimiento-2] n TIMES

Formato 3 - PERFORM Repetitivo controlado por cantidad de ejecuciones variables:
PERFORM procedimiento-1 [THRU procedimiento-2] VARYING identificador-1 FROM identificador-2 BY identificador-3 UNTIL condición

DISPLAY
Muestra mensajes o datos por consola.
Formato:
DISPLAY [nombre-dato-n/literal]

Ejemplo: *DISPLAY CT-LEIDOS*

ACCEPT
Acepta datos desde el proceso.

Formato:
ACCEPT [nombre-dato] FROM SYSIN
ACCEPT [nombre-dato] FROM DATE

ACCEPT WS-FECHA FROM DATE
MOVE CT TO WS-VALOR

MOVE
Transfiere datos a campos.

Formato:
MOVE [nombre-dato/literal] TO [nombre-dato-n]

INITIALIZE
Inicializar en ceros o espacios según el tipo de dato definido.

Formato:
INITIALIZE [nombre-dato-n]

Ejemplo:
```
01 CN-CONTADOR-LEIDOS PIC 9
(09).
01 WS-AUXILIAR PIC X(04).
INITIALIZE CN-CONTADOR-
LEIDOS
```
→ **Completa con CEROS**
```
INITIALIZE WS-AUXILIAR
```
→ **Completa con ESPACIOS**

Se puede inicializar una Tabla o Vector:
```
INITIALIZE TAB-NOTAS.
01 TABLA.
  05 TAB-NOTAS OCCURS 5.
  10 MAT PIC X(10).
  10 NOTA PIC 99.
```

CALL
Llama a rutinas.

Formato:
CALL identificador USING nombre-dato-1 [nombre-dato-2 ... nombre-dato-n] [ON OVERFLOW sentencia-1]

COPY
Utiliza estructuras definidas externamente

Formato:
COPY identificador

Donde identificador es:

- Una estructura de datos si la sentencia está en la **WORKING -STORAGE**
- Un conjunto de sentencias si la sentencia COPY está en la **PROCEDURE**

STOP RUN
Finaliza el programa.

Formato Batch: STOP RUN
Formato On-Line / Subprogramas: EXIT PROGRAM

CONDICIONALES COMUNES

OPERADORES LÓGICOS

Operador	Ejemplo
= , EQUAL	**A = B**
NOT = , NOT EQUAL	**A NOT EQUAL B**
> , GREATER	**A > B**
< , LESS	**A < B**
NOT > , NOT GREATER	**A NOT > B**
NOT < , NOT LESS	**A NOT < B**

CONECTORES LÓGICOS

AND
OR
NOT

IF
Determina la ejecución de una u otra instrucción (o conjunto de instrucciones), según una condición.

```
IF condición A
    PERFORM instrucciónB
ELSE
    PERFORM instrucciónB
END-IF.
```

Se puede poner más de una condición.
Se puede evaluar una variable, una constante, o si se está evaluando una variable de nivel 88, evaluar dicho valor con TRUE o FALSE.

Las instrucciones terminan con un punto. Tener en cuenta que si se agrega un punto para terminar otra sentencia dentro de la estructura IF, el compilador lo tomará como equivalente al END-IF, y nos puede dar un error de compilación.

EVALUATE
Permite evaluar una serie de valores para una variable. Es el equivalente a una instrucción SWITCH en otros lenguajes.

```
EVALUATE nombre-dato-1
  WHEN valor-1
       sentencia-1
  WHEN valor-2
       sentencia-2
       sentencia-2a
  WHEN OTHER
       sentencia-n
END-EVALUATE
```

- La instrucción WHEN OTHER es opcional, y se utiliza para ejecutar instrucciones cuando no se haya cumplido ninguna de las condiciones anteriores.
- Se puede indicar un valor específico (WHEN 'A') , o si se está evaluando una variable de nivel 88, utilizar sus valores (WHEN VALOR1).
- Este condicional es equivalente al SWITCH de otros lenguajes más modernos, donde cada CASE (switch) es igual a un WHEN (evaluate).

OPERACIONES ARITMÉTICAS

SUMA
Formato 1:

```
ADD nombre-dato-1 TO
nom-dato-2 [ROUNDED][ON
SIZE ERROR sentencia 1]
literal-numérico
```

Ejemplo:

```
ADD VALOR-A TO VALOR-C.
ADD VALOR-A - 100 TO VALOR-C.
```

Formato 2:

```
ADD nombre-dato-1 TO nombre
```

```
-dato-2 GIVING nombre-dato-3
[ROUNDED]  literal-numérico
[ON SIZE ERROR sentencia-1]
```

Ejemplo:

```
ADD VALOR-A + 300 GIVING
VALOR-C ROUNDED ON SIZE
ERROR DISPLAY 'ERROR  '.
ADD VALOR-A TO VALOR-B
GIVING VALOR-C ROUNDED ON
SIZE ERROR DISPLAY 'ERROR  '.
```

RESTA

Formato 1:

```
SUBTRACT nombre-dato-1 FROM
nombre-dato-2 [ROUNDED]
literal-numérico [ON SIZE
ERROR sentencia-1]
```

Ejemplo:

```
SUBTRACT VALOR-A FROM
VALOR-C.
SUBTRACT VALOR-A - 100 FROM
VALOR-C.
```

Formato 2:

```
SUBTRACT nombre-dato-1 FROM
nombre-dato-2 GIVING nombre
-dato-3  literal-numérico
[ROUNDED] [ON SIZE ERROR
sentencia 1]
```

Ejemplo:

```
SUBTRACT VAL-A FROM VAL-B
GIVING VAL-C ROUNDED ON SIZE
ERROR DISPLAY 'ERROR '
```

MULTIPLICACIÓN

Formato 1:

```
MULTIPLY nombre-dato-1 BY
nombre-dato-2[ROUNDED][ON
SIZE ERROR  literal-numéri-
co sentencia-1]
```

Ejemplo:

```
MULTIPLY VALOR-A BY VA-
LOR-B.
MULTIPLY VALOR-A BY 3 ROUN-
DED.
```

Formato 2:

```
MULTIPLY nombre-dato-1 BY
nombre-dato-2 GIVING nom-
bre-dato-3  literal-numéri-
co [ROUNDED] [ON SIZE ERROR
sentencia-1]
```

Ejemplo:

```
MULTIPLY VALOR-A BY -0,15
GIVING VALOR-C ROUNDED ON
SIZE ERROR DISPLAY
'ERROR'.
MULTIPLY VALOR-A BY -300
GIVING VALOR-C ROUNDED.
MULTIPLY -300 BY VALOR-A
GIVING VALOR-C ROUNDED.
```

DIVISIÓN

DIVIDE = palabra clave.

Formato 1:

```
DIVIDE nombre-dato-1 INTO
nombre-dato-2 [ROUNDED][ON
SIZE ERROR sentencia-1]
literal-numérico
```

Formato 2:

```
DIVIDE nombre-dato-1 BY
nombre-dato-2 GIVING nom-
bre-dato-3  [ROUNDED]
literal-numérico-1 lite-
ral-numérico-2 [REMAINDER]
nombre dato-4
ON SIZE ERROR sentencia-1]
```

Ejemplo:

```
DIVIDE VALOR-A BY VA-
LOR-BGIVING VALOR-C.
DIVIDE VALOR-A BY 3 GIVING
VALOR-C ROUNDED.
DIVIDE 6 BY 3 GIVING VA-
LOR-C ROUNDED ON SIZE ERROR
DISPLAY  'ERROR '.
DIVIDE VALOR-A BY 2 GIVING
VALOR-C ROUNDED ON SIZE
ERROR DISPLAY 'ERROR  '.
DIVIDE 1251 BY VALOR-A
GIVING VALOR-C ROUNDED RE-
MAINDER VALOR-D.
```

COMPUTE

Sirve para computar (mover un dato) de una variable a otra, donde en la primera parte de la operación se realiza una operación aritmética.

Beneficio: ahorrás en una línea varias operaciones.

Formato:

```
COMPUTE nombre-dato-1
[ROUNDED [nombre-dato-2
[ROUNDED]…] =  expre-
sión-aritmética [ON SIZE
ERROR sentencia-1]
```

TABLAS: CLÁUSULAS OCCURS

- Es el equivalente a un array o matriz. Al disponer las variables en estructuras jerárquicas con el uso de niveles, podemos utilizar la palabra reservada OCCURS para generar nuestras tablas. Podemos luego hacer referencia a la tabla completa o a valores individuales a través de subíndices.
- El subíndice puede ser tanto una variable como una constante, pero siempre que se trate de un número entero.
- Las tablas pueden ser uni, bi y tridimensionales.
- La cláusula OCCURS no puede

especificarse en un campo de nivel 01 ni 77.

Ejemplo:

Tabla de una dimensión

```
01 WS-TABLA.
   05 WS-A PIC A(10) OCCURS
   10 TIMES.
```

Tabla de dos dimensiones

```
01 WS-TABLE.
   05 WS-A OCCURS 10 TIMES.
      10 WS-B PIC A(10).
      10 WS-C OCCURS 5 TIMES.
         15 WS-D PIC X(6).
```

Para acceder a la instancia específica de la tabla se utiliza el subíndice:

```
WS-A(1)
WS-C(2,1)
```