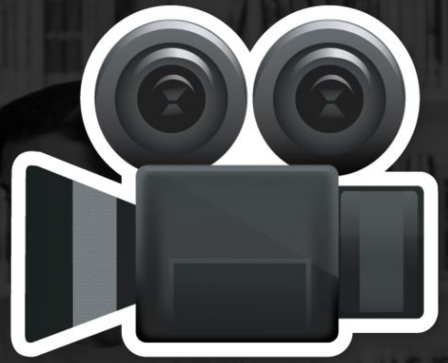

Programación COBOL
BATCH – Parte 4
Codeki™

Recuerda poner a grabar la clase.



... PROCEDURE DIVISION – OPERADORES

Carácter	Significado	Ejemplo
+	Suma	A+B
-	Resta	A-B
*	Multiplicación	A*B
/	División	A/B
**	Potenciación	A**B
=	Asignación	C=A+B
()	Dar Prioridad	D=(A+B)*C

... PROCEDURE DIVISION – OPERADORES DE RELACIÓN

Carácter	Significado	Ejemplo
=	Igual que	A=B
<	Menor que	A<B
>	Mayor que	A>B
<=	Menor igual	A<=B
>=	Mayor igual	A>=B

... PROCEDURE DIVISION – COMPUTE

COMPUTE identifier-1 ROUNDED = expresión-aritmética ON SIZE ERROR sentencia-1
EQUAL

Opcional
NOT ON SIZE ERROR sentencia-2

END-COMPUTE.

PROCEDURE DIVISION – COMPUTE

```
COMPUTE  AUMENTO-SALARIAL ROUNDED = SALARIO * 1.5

COMPUTE ÁREA = ALTURA * LONGITUD.

COMPUTE PROMEDIO = ( ITEM-1 + ITEM-2 + ITEM-3 ) / 3

COMPUTE PROBLEMA = ITEM-1 - ITEM-2 * ITEM-3
```

PROCEDURE DIVISION

- SENTENCIAS DE CONTROL-

... PROCEDURE DIVISION – IF

EJECUCIÓN CONDICIONAL. SENTENCIA IF.

La sentencia IF toma una decisión referente a la acción a ejecutar en un programa, basándose en el resultado, verdadero o falso, de una condición. Su formato es:

La condición viene dada por una expresión de BOOLE. Si en la condición intervienen operadores de diferentes tipos, los operadores aritméticos tienen mayor prioridad, después se ejecutan los operadores de relación y por último los operadores lógicos.

PROCEDURE DIVISION – IF

EJECUCIÓN CONDICIONAL. SENTENCIA IF.

```
IF A = B     ACCIÓN 1
ELSE
      ACCIÓN 2
END-IF.
```

La forma en que se ejecuta la sentencia IF es la siguiente:

1. Si el resultado de la condición es verdadero, se ejecutará lo indicado por la acción-1.
2. Si el resultado de la condición es falso, se ejecutará lo indicado por la acción-2.
3. Si el resultado de la condición es falso, y la cláusula ELSE se ha omitido, no se ejecutará lo indicado acción-1 y el programa continuará con la siguiente instrucción.

PROCEDURE DIVISION – SENTENCIAS DE CONTROL

En cualquier caso de los anteriormente expuestos, la ejecución continúa con la siguiente sentencia ejecutable. El ámbito de la sentencia IF finaliza de cualquiera de las formas siguientes:

1- Por un punto.

2- Por la cláusula END-IF.

Cuando la acción-1 o la acción-2 están formadas por varias sentencias, solamente la última finaliza con un punto, ya que este indica el final de una sentencia IF. Si se especifica la frase END-IF no se puede utilizar la frase NEXT SENTENCE.

También, hay que tener presente que el operador NOT puede preceder a una condición simple o una condición combinada.

PROCEDURE DIVISION – SENTENCIAS DE CONTROL

PRUEBAS DE RELACIÓN:

El formato general para formar un condición de relación es:

Condiciones combinadas.

Una condición combinada está formada por un conjunto de condiciones simples unidas por los operadores OR y AND. El formato es el siguiente:
IF A = (1 OR 5 OR 7) AND B = 4

También, hay que tener presente que el operador NOT puede preceder a una condición simple o a una condición combinada.

PROCEDURE DIVISION – SENTENCIAS DE CONTROL

ANIDAMIENTO DE SENTENCIAS IF.

La estructura presentada a continuación, aparece con bastante frecuencia y es por lo que la damos un tratamiento por separado. Esta estructura es consecuencia del anidamiento de sentencias IF.
Las sentencias IF...THEN pueden estar anidadas. Esto quiere decir que como acción-1 o acción-2, de acuerdo con el formato, puede escribirse otra sentencia IF.

```
IF ……..  ACCIÓN-1
  IF ….  ACCIÓN-2
  ELSE   ACCIÓN-3
  END-IF
END-IF.
```

PROCEDURE DIVISION – SENTENCIAS DE CONTROL

ESTRUCTURA IF.

Si se cumple la condición-1, se ejecutan las sentencias-1 y si no se cumplen se examinan secuencialmente las condiciones siguientes hasta ELSE, ejecutándose las sentencias correspondientes al primer ELSE IF, cuya expresión sea cierta. Si todas las expresiones son falsas, se ejecutan las sentencias-n correspondientes al último ELSE. En cualquier caso, se continúa con la siguiente sentencia en la estructura.

PROCEDURE DIVISION – SENTENCIAS DE CONTROL

ESTRUCTURA IF.

```
IF      condición-1   THEN  sentencia-1   ELSE   sentencia-2
                            NEXT SENTENCE            NEXT SENTENCE
END-IF.
```

```
        IF DATO-1 IS NOT  NUMERIC THEN . . . . . .
        END-IF.

        IF NUMERO-CLIENTE IS LESS THAN PREV-NUMERO-CLIENTE
           MOVE 'CLIENTE FUERA DE SECUENCIA' TO MESSAGE-TEXTO
           PERFORM WRITE-MESSAGE
                END-IF.
```

... PROCEDURE DIVISION – SENTENCIAS DE CONTROL

ESTRUCTURA IF.

```
01 DEDIC-EMPLEADO     PIC X.
        88     PARTTIME   VALUE 'P'.
        88     FULL-TIME  VALUE 'F'.

        IF PARTTIME
           PERFORM CALCULO-SALARIO-PARTTIME.

        IF DEDIC-EMPLEADO = 'F'
            PERFORM CALCULO-SALARIO.
```

... PROCEDURE DIVISION – SENTENCIAS DE CONTROL

Estructura IF. CON OPERADORES LÓGICOS

```
IF      FECHA-AA = 2020 AND FECHA-MM = 02 AND FECHA-DD = 29
        PERFORM CALCULO-CIERRE-BISIESTO
ELSE
        PERFORM CALCULO-CIERRE-NORMAL
END-IF.

IF      PARM-MM = 01
        OR PARM-MM = 03
        OR PARM-MM = 05
        OR PARM-MM = 07
        OR PARM-MM = 08
        OR PARM-MM = 10
        OR PARM-MM = 12 AND PARM-DD = 31 PERFORM CALCULO-FIN-MES
ELSE
        PERFORM ANALISIS-MES.
```

PROCEDURE DIVISION – SENTENCIAS DE CONTROL

Estructura IF. ANIDADOS

EVITAR SU USO.

```
IF      ESTADO-EMPLEADO = 'EXEMPL'
        PERFORM CALCULO-SALARIO-EXEMPL
ELSE
        IF ESTADO-EMPLEADO = 'DPEMPL'
                PERFORM CALCULO-SALARIO DPEMPL
        ELSE
                PERFORM CALCULO-SALARIO-COMUN
        END-IF
END-IF.
```

PROCEDURE DIVISION – SENTENCIAS DE CONTROL

Estructura IF. ANIDADOS

```
MOVE ‘abcdefgh’ TO DISPLAY-AREA.

IF DISPLAY-AREA IS ALPHABETIC-UPPER
THEN MOVE ‘MAYUSCULA’ TO MESSAGE-TEXT
ELSE
        IF DISPLAY-AREA IS ALPHABETIC-LOWER
                THEN MOVE  ‘MINUSCULA’ TO MESSAGE-TEXT
        END-IF
END-IF.
```

PROCEDURE DIVISION
- PERFORM -

PROCEDURE DIVISION – PERFORM

Llamada a procedimientos. Sentencia PERFORM.

La sentencia PERFORM es utilizada para transferir explícitamente el control a uno o más procedimientos y devolver el control implícitamente, cuando la ejecución del procedimiento especificado, finalice.

La sentencia PERFORM se puede utilizar para controlar la ejecución de una o más sentencias, las cuales están dentro del ámbito de la sentencia PERFORM.

PROCEDURE DIVISION – PERFORM

Formato 1:

PERFORM procedimiento-1 [THRU procedimiento-2]

Si no se especifica la sentencia THRU, la sentencia PERFORM ejecuta una vez, el conjunto de sentencias que forman el procedimiento-1. Si la opción THRU se especifica, entonces se ejecutan, una vez todos los procedimientos existentes en el programa entre procedimiento-1 y procedimiento-2, ambos inclusive.

Formato 2:

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

Si no se especifica la opción THRU, la sentencia PERFORM ejecuta el número de veces especificado por entero o por un nombre de datos, el conjunto de sentencias que forman procedimiento-1. Si la opción THRU se especifica, entonces se ejecutan el número de veces especificado, todos los procedimientos existentes en el programa entre procedimiento-1 y procedimiento-2, ambos inclusive.

PROCEDURE DIVISION – PERFORM

Formato 3:

PERFORM UNTIL …….

END-PERFORM.

Cuando se emplea este formato, la sentencia PERFORM ejecuta el número de veces especificado por entero o por nombre de datos, el conjunto de sentencias que hay entre PERFORM y END-PERFORM. Un punto, como final de alguna de las sentencias de esta estructura, daría lugar a un error, ya que se entendería como final de la sentencia PERFORM.

Formato 4:

PERFORM THRU……. VARYING I FROM 1 BY 1 UNTIL …….

Cuando se emplea este formato, la sentencia PERFORM se ejecuta el número de veces indicado en la condición de VARING

PROCEDURE DIVISION – PERFORM

ANIDAMIENTO DE SENTENCIAS PERFORM.

Dentro del ámbito de una sentencia PERFORM, puede especificarse otra sentencia PERFORM, aunque hay que tener presentes las siguientes reglas:

1. El procedimiento PERFORM ejecutado desde el ámbito de otro PERFORM debe ser totalmente exterior o totalmente interior a este.
2. Los ámbitos de dos PERFORM se pueden solapar cuando las sentencias de llamada para su ejecución están fuera de estos ámbitos.
3. Las sentencias PERFORM pueden ser anidadas libremente.
4. Un procedimiento PERFORM puede llamarse a sí mismo, esto es, la recursividad está permitida.

PROCEDURE DIVISION – PERFORM

PERFORM

PERFORM procedure-name-1 THRU procedure-name-2.

PERFORM a THRU m

a
h **PERFORM** f THRU j
m
f
j

P

PROCEDURE DIVISION – PERFORM

PERFORM In-line

- Este tipo de PERFORM se realiza sin invocar ningún párrafo, se realiza todo dentro de la sentencia PERFORM hasta el END-PERFORM

PERFORM invocando párrafo

```
PERFORM MOVEIT
        VARYING X FROM 1 BY 1 UNTIL X = 5.
.....
MOVEIT.
        MOVE DATA-FLD (X) TO PRINT(X).
```

PERFORM in-line

```
PERFORM VARYING X FROM 1 BY 1 UNTIL X= 5
        MOVE DATA-FLD (X) TO PRINT (X)
END-PERFORM.
```

PROCEDURE DIVISION – PERFORM

PERFORM EJEMPLOS

```
77  X          PIC 9(3) BINARY VALUE 1.
.....
           PERFORM PARA1   UNTIL 100 TIMES              Se ejecuta 100 veces
           END-PERFORM.
           PERFORM UNTIL X GREATER THAN 100             Se ejecuta 100 veces
                   ADD A(X) TO TOTAL
                   ADD  1 TO X
           END-PERFORM.
           PERFORM VARYING X FROM 1 BY 1 UNTIL X GREATER THAN 100 (Se ejecuta 100 veces)
                   ADD A(X) TO TOTAL
           END-PERFORM.
```

PROCEDURE DIVISION – PERFORM

Sentencia EXIT.

Esta sentencia se utiliza como complemento de la sentencia PERFORM, para proporcionar un punto final para uno o más procedimientos, a fin de permitir la salida desde cualquier punto.

Formato: EXIT

La sentencia EXIT forma por sí sola un párrafo identificado por un nombre.

PROCEDURE DIVISION – PERFORM

*** PERFORM...IN-LINE**

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
*
PROCEDURE DIVISION.


        PERFORM
            DISPLAY 'HOLA COBOL'
        END-PERFORM.
        GOBACK.
```

PROCEDURE DIVISION – PERFORM

```
*PERFORM...THRU.

IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
*
PROCEDURE DIVISION.

        PERFORM PARRAFO-1
        PERFORM PARRAFO-1 THRU PARRAFO-2
        STOP RUN.
PARRAFO-1.
        DISPLAY 'HOLA ALUMNOS'
                .
PARRAFO-2.
        DISPLAY 'HOLA PROFESORES'
```

PROCEDURE DIVISION – PERFORM

```
*PERFORM...TIMES.

IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
*
PROCEDURE DIVISION.

        PERFORM PARRAFO-1 2 TIMES
        PERFORM PARRAFO-1 THRU PARRAFO-2 3 TIMES
        STOP RUN.
PARRAFO-1.
        DISPLAY 'HOLA ALUMNOS'
                .
PARRAFO-2.
DISPLAY 'HOLA* PROFESORES'
```

... PROCEDURE DIVISION – PERFORM

***PERFORM...UNTIL.**

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
77 LIMITE PIC 99 VALUE 0.

PROCEDURE DIVISION.
        PERFORM PARRAFO-1 UNTIL LIMITE = 5
        STOP RUN.
PARRAFO-1.
        DISPLAY 'HOLA ALUMNOS ' LIMITE
        ADD 1 TO LIMITE
*
```

PROCEDURE DIVISION – PERFORM

```
*PERFORM...VARYING.

IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
77 LIMITE PIC 99 VALUE 0.

PROCEDURE DIVISION.

        PERFORM VARYING LIMITE FROM 1 BY 1 UNTIL LIMITE >5
                DISPLAY "Saludo Nro: ", LIMITE
        END-PERFORM.
        GOBACK.
```

PROCEDURE DIVISION
- GO TO -

PROCEDURE DIVISION – GO TO

Sentencia de bifurcación GO.

Esta sentencia permite transferir el control desde un punto de la división PROCEDURE a otro.

- Formato 1: GO TO [nombre procedimiento]

- Formato 2:
  GO TO nombre procedimiento-1 [nombre procedimiento-2]... DEPENDING ON nombre datos

PROCEDURE DIVISION
- CONDICIÓN EVALUATE-

PROCEDURE DIVISION – EVALUATE

EVALUATE

Se usa en reemplazo del IF condicional para una programación estructurada. Se recomienda el uso del EVALUATE en vez del IF.

```
EVALUATE DATA-NAME1
    WHEN VALUE1                    sentencia-1
    WHEN VALUE2  THRU VALUES       sentencia-2
    WHEN NOT VALUE9                sentencia-3
    ..........................
    WHEN OTHER                     sentencia-n
END-EVALUATE.
```

PROCEDURE DIVISION – EVALUATE

EVALUATE

```
EVALUATE     TRANS-ID
         WHEN ‘A001'          PERFORM ALTA
         WHEN ‘D001’          PERFORM  BAJA
         WHEN ‘U001”          PERFORM MODIFICA
         WHEN ‘C001’          PERFORM CONSULTA
         WHEN OTHER           PERFORM TRANS-ID-INVALIDA
END-EVALUATE.
```

PROCEDURE DIVISION – EVALUATE

EVALUATE

```
EVALUATE identificador-1 ALSO ANY
        literal-1                   literal-2
        expresión-1         expresión-2
        TRUE                TRUE
        FALSE                       FALSE
        WHEN ANY            ALSO VALUE-2
        ...............                 sentencia-1
        WHEN OTHER ......
END-EVALUATE.
```

PROCEDURE DIVISION – EVALUATE

EVALUATE EJEMPLOS TRUE / FALSE

```
EVALUATE    TRUE
            WHEN  HORARIO  IS  LESS  THAN  8  COMPUTE  HORARIO  =  HORARIO  *  0.65
      PERFORM MENSA1
            WHEN HORARIO IS GREATER THAN 8 COMPUTE HORARIO = HORARIO * 1.25 PERFORM
      MENSA2
          WHEN    OTHER PERFORM MENSA3
END-EVALUATE.
EVALUATE A IS LESS THAN B AND C IS GREATER THAN D
            WHEN TRUE PERFORM SEDIO
            WHEN FALSE PERFORM NO-SEDIO
            WHEN OTHER PERFORM ERROR    END-EVALUATE.
```

PROCEDURE DIVISION – EVALUATE

EVALUATE EJEMPLOS ALSO / ANY

```
EVALUATE     TRUE  ALSO TIPO-EMPLEADO
          WHEN HORAS IS GREATER THAN 40       ALSO “E” MOVE “NORMAL” TO MENSAJE
          WHEN HORAS IS LESS THAN 40   ALSO ANY MOVE “PARTTIME' TO MENSAJE
          WHEN HORAS IS GREATER THAN 40       ALSO ANY MOVE “EXTRAS” TO MENSAJE
          WHEN HORAS IS LESS THAN 40   ALSO “E” MOVE SPACES TO MENSAJE
          WHEN OTHER   .....      SENTENCIA-3
END-EVALUATE
```

PROCEDURE DIVISION
- ACCEPT / DISPLAY -

PROCEDURE DIVISION – ACCEPT

ACCEPT

- Permite el ingreso de datos desde SYSIN, el dato deberá estar definido en WORKING-STORAGE
- Permite obtener fecha y hora del sistema operativo.

```
            ACCEPT identificador-1 FROM SYSIN.
77  FECHA-EJECUCION              PIC X(10).
……..
//SYSIN DD *
19/10/2019
……..
            ACCEPT FECHA-EJECUCION FROM SYSIN.
            DISPLAY FECHA-EJECUCION.
…… SALIDA
19/10/2019
```

PROCEDURE DIVISION – ACCEPT

ACCEPT

```
        ACCEPT identificador1  FROM DATE YYYYMMDD
                                        DAY     YYYYDD
                                        DAY-OF-WEEK
                                        TIME

01 HOY.
        02 HOY-AA     PIC 9(4).
        02 HOY-MM     PIC 99.
        02 HOY-DD     PIC 99.
.....
                ACCEPT HOY FROM DATE YYYYMMDD.
```

PROCEDURE DIVISION – ACCEPT

ACCEPT DATE / TIME / DAY-OF-WEEK

IDENTIFICADOR-2	PICTURE	FORMATO
DATE	9(6)	YYMMDD
	9(8)	YYYYMMDD
DAY	9(5)	YYDDD
	9(7)	YYYYDDD
DAY-OF-WEEK	9	

Número del día de la semana: 1 lunes**; 2** martes**; 3** miércoles; **4** jueves; **5** viernes; **6** sábado **7** domingo

TIME	**9(8)**	**HHMMSS99**	**Décimas de segundo**

PROCEDURE DIVISION – DISPLAY

DISPLAY

- Esta sentencia transfiere el contenido de cada DATA-ITEM a una salida SYSOUT / CEEMSG
- DISPLAY identificador-1 UPON CONSOLE
  literal-1 identificador-2

```
77 CANT-EMPL                    PIC99     VALUE ZERO.
.....
        ADD  1  TO CANT-EMPL.
                DISPLAY  "CANTIDAD DE EMPLEADOS = "  CANT-EMPL.
                DISPLAY   "CANCELA  TAREA   JOBCTRACT  —  PARAMETRO  EJECUCION
INVALIDA"
                        UPON CONSOLE.
```

PROCEDURE DIVISION
- CALL -

PROCEDURE DIVISION – CALL

CALL

- La sentencia CALL transfiere el control, desde un programa objeto a otro, en una unidad de ejecución.
- El programa llamador es llamado programa MAIN, y el programa invocado es normalmente llamado subprograma o rutina.
- Ahora un subprograma o rutina no puede realizar CALL, si no tiene atributo de RECURSIVE.
- El CALL al llamar a un programa, lo detecta con el ENTRY POINT y este se encuentra en la dirección de memoria donde fue cargado el PROCEDURE DIVISION del programa llamado.
- El compilador COBOL, lleva la opción DYNAM / NODYNAM. Por defecto los compiladores tienen NODYNAM.
- Si el compilador llevará DYNAM, la rutina sea invocada con literal o con data-name siempre va a ser dinámica.

PROCEDURE DIVISION – CALL

CALL

- EL subprograma o rutina la PROCEDURE DIVISION debe usarse USING, por donde ingresarán la información pasado por el programa llamador o MAIN.
- Se puede manejar RETURN-CODE del subprograma usando RETURNING, donde el subprograma debe dejar el código de retorno. Y en el programa MAIN en WORKING-STORAGE, debe estar definido el campo de RETURN-CODE para que allí se impacte la información dada por RETURNING el subprograma.
- El programa MAIN puede realizar CALL con RETURNING al subprograma, entonces el subprograma debe especificar el RETURNING en la PROCEDURE DIVISION.
- Las rutinas o subprogramas pueden ser invocados en forma dinámica o estática.

PROCEDURE DIVISION – CALL

CALL ESTÁTICO

- CALL ESTÁTICO, lleva literales, para que en la LINKEDICIÓN del programa, previamente debe existir el objeto del CALL y debe estar en una biblioteca a nivel de SYSLIB del LINKED del programa.
- Las rutinas estáticas se cargan en memoria del programa MAIN. Por lo tanto ante el CALL de la rutina no hay LOAD de la misma, y esto es eficiente.
- Las rutinas elegidas para ser estáticas son aquellas QUE NO SUFREN MODIFICACIONES. Por ejemplo, rutinas de manejo de errores, de fecha.
- Si una rutina estática, sufre modificación se deben realizar la recompilación de todos los programas que la invocan, y esto es grave, por la envergadura del trabajo a realizar y con la posibilidad de errores.

```
CALL “RUTERROR” USING .....
        ON EXCEPTION      SET GOOD-CALL TO TRUE
        NOT ON EXCEPCION          SET BAD-CALL TO TRUE
END-CALL.
```

PROCEDURE DIVISION – CALL

CALL DINÁMICO

- CALL DINÁMICO, se invoca desde un DATA-NAME, definido en WORKING-STORAGE.
- En tiempo de ejecución cuando el programa MAIN ejecute un CALL a una rutina dinámica, la misma produce un LOAD en memoria, llevando el tiempo de LOAD en RESPTIME y CPUTIME.
- Las rutinas elegidas para ser dinámicas puede ser cualquiera, ya que puede sufrir o no modificaciones que siempre se va a tomar la última versión en un CALL. Es el tipo de rutina conveniente en una instalación.

```
77 RUT-PGM                PICX(8) VALUE “RUTERROR”.
CALL RUT-PGM USING …….
        ON EXCEPTION SET GOOD-CALL TO TRUE
        NOT ON EXCEPCION SET BAD-CALL TO TRUE
END-CALL.
```

PROCEDURE DIVISION – CALL

CALL Y LINKAGE SECTION

- La LINKAGE SECTION se define en la DATA DIVISION, y describe los datos a ser pasados por un programa MAIN.
- Esta memoria no está reservada por el programa MAIN como la WORKING-STORAGE.
- Los DATA-ITEM no pueden llevar la cláusula VALUE, salvo que el NIVEL sea 88.

```
DATA DIVISION.
WORKING-STORAGE SECTION.
……..
LINKAGE SECTION.
01 INFO-REGISTRO.
        02 INFO 1 ……
        02 INFO 2……..
```

PROCEDURE DIVISION – CALL Y LINKAGE

DATA DIVISION. — **Programa TOTOPGM**
WORKING-STORAGE SECTION.

```
01 DATOS-ERROR.
      02 COD-ERR                PIC9(3).
      02 PGM-MAIN-ERROR         PIC X(8).
      02 HORA-ERROR             PIC X(8).
      02 PROC-NAME-ERROR        PICX(40).
```

PROCEDURE DIVISION.
MOVE…… TO DATOS-ERROR
MOVE “RUTERROR” TO RUT-ERROR.
CALL RUT-ERROR USING **DATOS-ERROR**.

Rutina RUTERROR
LINKAGE SECTION.

```
01 DATOS-ERROR.
      02 COD-ERROR PIC9(3).
      02 PGM-MAIN-ERROR PIC X(8).
      02 HORA-ERROR PIC X(8).
      02 PROCEDURE-NAME-ERROR PIC X(40).
```

PROCEDURE DIVISION USING **DATOS-ERROR.**

PROCEDURE DIVISION – CALL Y LINKAGE

LINKAGE SECTION y JCL PARAMETERS

```
//STEP01 EXEC PGM=TOTOPGM,PARM='PARM1,PARM2'
Programa TOTOPGM
LINKAGE SECTION.
01 PARM-LIST.
        02 PARM-LENGTH              PIC 99 BINARY.
        02 PARM1                    PICX(8).
        02 PARM2                    PICX(6).
PROCEDURE DIVISION USING PARM-LIST.
        EVALUATE PARM-LENGTH
                WHEN 0              PERFORM NO-HAY PARAMETRO
                WHEN 8              MOVE PARM1 TO .....
                WHEN 14             MOVE PARM1 TO .....
                                    MOVE PARM2 TO.....
                WHEN OTHER          PERFORM PARAMETRO-INVALIDA
END-EVALUATE.
```

PROCEDURE DIVISION
- FUNCIONES INTRÍNSECAS -

PROCEDURE DIVISION – FUNCIONES INTRÍNSECAS

- Son utilizadas para ciertos valores obtenidos en ejecución.
- Por ejemplo de
  - CURRENT-DATE
  - DATE-OF-INTEGER
  - LENGTH
  - LOWER-CASE

FUNCTION es una palabra reservada
MOVE FUNCTION CURRENT-DATE TO D-STRING
IF FUNCTION DATE-OF-INTEGER(BASE-DATE).....
WHEN FUNCTION DAY-OF-INTEGER(BASE-DATE).....

PROCEDURE DIVISION – FUNCIONES INTRÍNSECAS

- ARGUMENTOS y VALORES
  - El número y formato del ARGUMENTO de la FUNCTION
  - El resultado del valor es un DATA-ITEM definido en WORKING-STORAGE
  - FUNCTIONS numéricos y enteros pueden ser solamente usados en expresiones aritméticas
- MOVE FUNCTION CURRENT-DATE(1:8) TO REPORT-DATE
- FORMATOS de FECHA
  - Rango ENERO 1, 2020 TO MARZO 20, 2020
  - Gregoriana YYYYMMDD
  - Enteros DATE -1 TO 3.067.671 NUMBER OF DAYS DECEMBER 31, 1600
  - JULIAN YYYYDDD

PROCEDURE DIVISION – FUNCIONES INTRÍNSECAS

- CURRENT-DATE retorna YYYYMMDDHHmmsshhShhmm es 21 caracteres
- Viene con fecha gregoriana (YYYYMMDD), la hora (HHmmsshh) y la diferencia desde GMT (Shhmm)

```
77 FECHA-HOY                PIC 9(8).
77 FECHA-ENT                PIC S9(9).
.....
        MOVE FUNCTION CURRENT-DATE(1:8) TO FECHA-HOY.
        COMPUTE FECHA-ENT = FUNCTION INTEGER-OF-DATE(FECHA-HOY).
        ADD 30 TO FECHA-ENT.
        COMPUTE FECHA-HOY = FUNCTION DATE-OF-INTEGER(FECHA-ENT)
```

Ejemplo si CURRENT-DATE=19970924, FECHA-HOY=19970924

COMPUTE FECHA-ENT = FUNCTION INTEGER-OF-DATE(19970924) FECHA-ENT = 144933

ADD 30 TO FECHA-ENT(144933) FECHA-ENT = 144963

COMPUTE FECHA-HOY = FUNCTION DATE-OF-INTEGER(144963) FECHA-HOY = 19971024

PROCEDURE DIVISION – FUNCIONES INTRÍNSECAS

- INTEGER-OF-DATE
  - Retorna un entero cuyo INPUT es una fecha gregoriana, dando un entero desde la creación de la fecha gregoriana 31 de diciembre 1600.
  - Por lo tanto si ingresamos la fecha gregoriana 19960317 entonces el cálculo de la función es 144337

```
02 FECHA-ENT            PIC 9(7).
.....
02 FECHA-HOY            PIC 9(8) VALUE 19970924.
.....
COMPUTE FECHA-ENT = FUNCTION INTEGER-OF-DATE(FECHA-HOY).
FECHA-ENT = 144933
```

PROCEDURE DIVISION – FUNCIONES INTRÍNSECAS

- INTEGER-OF-DATE
  - Retorna un entero cuyo INPUT es una fecha gregoriana, dando un entero desde la creación de la fecha gregoriana 31 de diciembre 1600.
  - Por lo tanto si ingresamos la fecha gregoriana 19960317 entonces el cálculo de la función es 144337

```
02 FECHA-ENT          PIC 9(7).
.....
02 FECHA-HOY          PIC 9(8) VALUE 19970924.
.....
COMPUTE FECHA-ENT = FUNCTION INTEGER-OF-DATE(FECHA-HOY).
FECHA-ENT = 144933
```

PROCEDURE DIVISION – FUNCIONES INTRÍNSECAS

```
INTEGER-OF-DAY
        02 FECHA.
                03                  PICXX VALUE “19”.
                03 FECHA-HOY        PIC X(5).
                77 FECHA-ENT        PIC 9(7).
……..……..
                ACCEPT FECHA-HOY FROM DAY. Nos trae YYDDD supongamos que nos trae
        97267.
                COMPUTE FECHA-ENT = FUNCTION INTEGER-OF-DAY(FECHA-HOY).
                FECHA-ENT = 144933
```

Lo mismo podríamos obtener con DAY-OF-INTEGER, en forma viceversa.

PROCEDURE DIVISION – FUNCIONES INTRÍNSECAS

- Nuevas FUNCIONES
  - DATE-TO-YYYYMMDD
  - DAY-TO-YYYY
  - YEAR-TO-YYYY
- FUNCIONES NO MÁS SOPORTADAS EN COBOL V5
  - DATEVAL
  - UNDATE
  - YEARWINDOW

PROCEDURE DIVISION – FUNCIONES INTRÍNSECAS

- FUNCTION LOWER-CASE y UPPER-CASE
  - MOVE FUNCTION UPPER-CASE(ANSWER) TO UPPER-ANSWER
    Si ANSWER contiene ‘y’, entonces UPPER-ANSWER contendrá ‘Y’
  - MOVE FUNCTION LOWER-CASE(UPPER-ANSWER) TO ANSWER
    Si UPPER-ANSWER contiene ‘Y’, entonces ANSWER contendrá ‘y’
- FUNCTION WHEN-COMPILED
  IF DESASTRE
    MOVE FUNCTION WHEN-COMPILED TO ERROR1
    Toma la fecha del objeto de la compilación, sirve para determinar cambios ante problemas

PROCEDURE DIVISION – FUNCIONES INTRÍNSECAS

- FUNCTION CHAR y ORD
  - Sirve para realizar comparaciones entre ASCII y EBCDIC

```
IF FUNCTION ORD('1') IS LESS THAN FUNCTION('A')
        PERFORM ASCII-COMPARE
ELSE
        PERFORM EBCDIC-COMPARE
END-IF.
```

PROCEDURE DIVISION – FUNCIONES INTRÍNSECAS

FUNCIONES INTRÍNSECAS ARITMÉTICAS PARA NEGOCIOS

- MAX toma el valor más alto de una lista de mismo tipo de alfabéticos, numéricos o alfanumérico
- MEDIAN toma el valor mediano de una lista de numéricos
- MIN toma el valor más bajo de una lista de mismo tipo de alfabéticos, numéricos o alfanuméricos

PROCEDURE DIVISION – FUNCIONES INTRÍNSECAS

FUNCIONES INTRÍNSECAS ARITMÉTICAS PARA NEGOCIOS

- FUNCTION MIN y MAX
  - COMPUTE MAX-HOLD = FUNCTION MAX(EMPISAL, EMP2SAL, EMP3SAL, EMP4SAL, EMP5SAL)
  - MOVE MAX-HOLD TO TOP-SAL.
- SUSCRIPTOR 'ALL' EN FUNCTION
  - COMPUTE TOTAL-IN = FUNCTION SUM(EMPL-SAL(ALL))

01 SALARIO.

02 EMPL-SAL —OCCURS 500 TIMES PIC S9(7)V99.

.....

COMPUTE TOTAL-IN = FUNCTION SUM(EMPL-SAL(ALL))

El subcriptor "ALL en FUNCTION MAX / MIN / SUM / MEDIAN

... CONSULTAS?

Codeki™