

COBOL· Common business oriented language

1958 codasyl - pentagon

Dr· Grace hopper

Application programming language

English like language·

Add a to b giving c·

Divide a by b giving d remainder e·

Create a pds

HLQ·URNAME·REVAT·COBOL·PDS

LRECL=80,RECFM=FB,,DSORG=PDS

PQ=10,SQ=10,DB=10,

Create a member → COMPJCL → type the job mentioned in the chat·

Create another → RUNJCL → type the next job mentioned in the chat·

CREATE A LIBRARY

HLQ·URNAME·REVAT·COBOL·LOADLIB

LRECL=80,RECFM=U,DSORG=LIBRARY

PQ=10,SQ=10,DB=10,

1· Write the cobol pgm

2· Submit a compile job on behalf of the cobol pgm·

a· Step1 → check syntax errors in cobol stmnts

b· Step2 → convert the cobol codes in loda modules/object code

3. Submit another the job that will run the load module.

PROGRAM STRUCTURE:

4 DIVISIONS.

1. IDENTIFICATION DIVISION. (mandatory)

Note: identifies the program.

PROGRAM-ID. Entry. mandatory

[AUTHOR. Entry].

[INSTALLATION Entry].

[DATE-WRITTEN. Entry].

[DATA-COMPILED. Entry].

[SECURITY. Entry.

[REMARKS. Entry.]

2. ENVIRONMENT DIVISION. (optional)

CONFIGURATION SECTION.

SOURCE-COMPUTER. ZOS.

OBJECT COMPUTER. 390.

INPUT-OUTPUT SECTION.

FILE CONTROL. → where the files are linked.

Note: Files and other external resources are linked here.

logical name for the file is given here.

Accessing mode

Organization

3. DATA DIVISION. (optional)

Note: all the variables are declared

FILE SECTION.

FD - FILE'S FIELD DISCRIPTION (FILE LAYOUT/FILE VARIABLES)

WORKING-STORAGE SECTION.

ALL THE COMMONLY USED VARIABLES

LOCAL-STORAGE SECTION.

REPORTS

LINKAGE SECTION.

SUB PROGRAMS AND to which the values can be received from the user dynamically

PROCEDURE DIVISION.(Mandatory)

Has no predefined sections or paragraphs.

Executable statements are written/verbs

Must be ended by 'stop run'.

COLUMN DISCRIPTION OF COBOL PROGRAM

1-6 - SEQNUM

7 -INDICATOR. SPACE → executable sentence

'*' → comment

'-' → continuation of string from previous sentence.

8-11 AREA/MARGIN A

DIVISIONS, SECTIONS, PARAGRAPHS, FD, SD 01, 77 ENTERIES.

12-72 -AREA/MARGIN B

02-49, 66, 88 LEVEL NUMBERS. ALL EXECUTABLE SENTANCE IN PROCEDURE DIVISION(VERBS).

73-80 - IDENTIFIER - USERS DISCRETION.

Data types:

1. Char → A

2. Numeric → 9

3. Alphanumeric → X

DECLARE A VARIABLE

SYNTAX:

LEVEL-NUMBER SPACE NAME-VAR SPACES PICTURE-CLAUSE DATATYPE(SIZE) VALUE-CLAUSE

LEVEL NUMBER → 01, 77, 02-49, 66, 88

VAR-NAME (MAX 36). ALPHANUMERIC, -, FIRST CHAR MUST BE ALPHABET.

(let the name tell the story)

PIC MANDATORY

DATATYPE(SIZE)

A(05) → left justified with auto spaces on the suffix

9(05) → right justified with auto zeros prefixed

X(10) → left justified with auto spaces on the suffix

VALUE(OPTIONAL) → the user can assign the variable with some initial values.

Date: 29:09-2023

Topics to be covered:

Singed variables

Figurative constants

Decimal variables

Move statement

Reference Modification

Group items

Arithmetic operations

Conditions

Conditional statements

Loops.

Task of the day:

Max digits. Max chars → alphabetic, max chars → alphanumeric

Singed variables. Numeric.

Pic 9(03) → no sign

01 WS01-A PIC S9(03) VALUE +123. A23.

Note: Signed variable, by default is sign leading. The sign will be combined with the leading digit with the below convention.

01 WS01-A PIC S9(03) sign leading VALUE +123. A23

The sign will be combined with the leading digit with the below convention.

01 WS01-B PIC S9(03) sign trailing VALUE +123. 12C.

Convention → +1 → A, +2 → B... +9 → I, -1 → J, -2 → K...

-9 → R. +0 → {, -0 → }

01 WS01-STS PIC S9(03) sign trailing Separate VALUE +123.

→ 123+

01 WS01-SLS PIC 9(03) sign leading Separate VALUE +123.

→ +123

Figurative constants

Space, spaces, 0, zero,zeros,zeroes

Decimal variables.

Virtual → numeric → arithmetic operation. The decimal point will not be displayed. 123.456 → 123456

01 ws01-vd PIC 9(03)V(02) value 123.45

But when you display → 12345

- Physical (alphanumeric) → no arithmetic operation allowed on this Variable.
- no value clause is allowed.
- Move statement is allowed

01 WS01-PD PIC 9(03).9(02)

MOVE

Allocation of a value to a variable can be done 2 ways.

1. Value clause- data division.

2. Move statement in Procedure division.

Syntax:

Move source to dest-variables

Move 10 to ws01-var1

Move 'ca7' to ws05-name

Move ws01-a to ws05-b (provided their data type are same)

Move 10 to ws01-a, ws05-b,ws05-c

Move ws05-d to ws01-a, ws05-b,ws05-c

Move corresponding grp1 to grp2

It will move the values from source members to the dest members, **ONLY IF THEIR NAME MATCHES**, irrespective of their sequence.

Move grp1 to grp2

→ It moves values in the sequence.

Source → single Value/literal or variables

Destination → single or multiple variables.

Reference Modification.

A technique by which parts of a variable can be handled individually.

Syntax : var(stpos:length)

01 ws01-a pic a(05) value 12345.

01 ws01-b pic 9(05) value 00000.

1	2	3	4	5
---	---	---	---	---

Move ws01-a(2:3) to ws01-b(3:3)

0	0	2	3	4
---	---	---	---	---

Group items.

01 - level → 1page-4kb

01 ws01-A pic x(03)

77 level number - elementary data item.

The number bytes allocated is the number bytes mentioned in the size clause.

Suggest: Declare one variable at 01 level number and declare all the other variables in the same page/under that 01 level number variable.

A group item is a variable which has member variables defined.

01 ws01-vars.

02 ws02-var1 pic 9(03).

02 ws02-var2 pic X(04).

02 ws02-var3 pic A(15).

Note: level numbers 02 - 49 must be mentioned in margin b

Rules of group item.

- It cannot have picture clause.*
- It CAN HAVE value clause. Value are considered to alphanumeric.*
- A group item is identified as a group item, if it has variables with higher level number.*
- The member items can have their own different pic clauses and sizes.*
- The size of a group item is calculated by adding the sizes of its members.*

Rules of a member item.

- Must have a level number higher than its group item.*
- Any higher level number is the member of its immediate previous lower level number.*
- A member item can be a SUB GROUP item.*

01 ws01-vars VALUE "TOMMY007USANJK991111".

05 WS05-NAME PIC A(05).

05 WS05-ID PIC X(03).

05 WS05-ADDRESS.

10 WS10-COUNRTY PIC A(3).

10 WS10-STATE PIC A(3).

10 WS10-PCODE PIC 9(02).

05 WS05-PHONE PIC 9(4).

PD

DISPLAY → WS01-VARS.

Note: group move. Move corresponding. See video 02-10-2023.

Arithmetic operations:

Numeric data type.

1. ADD.

ADD 10 TO WS05-A \rightarrow $WS05-A = WS05-A + 10$

ADD WS05-A TO WS05-B \rightarrow $WS05-B = WS05-A + WS05-B$

ADD WS05-A TO WS05-B WS05-C. \rightarrow

$WS05-B = WS05-B + WS05-A$

$WS05-C = WS05-C + WS05-A$

ADD 10 TO WS05-B GIVING WS05-C

$WS05-C = WS05-B + 10.$

2. SUBTRACT.

SUBTRACT 10 FROM WS05-A \rightarrow $WS05-A = WS05-A - 10$

SUBTRACT WS05-A FROM WS05-B \rightarrow $WS05-B = WS05-B - WS05-A$

SUBTRACT WS05-A FROM WS05-B GIVING WS05-C

3. MULTIPLY.

MULTIPLY WS05-A BY WS05-B

MULTIPLY 10 BY WS05-B

MULTIPLY WS05-A BY WS05-B GIVING WS05-C

4. DIVIDE

DIVIDE WS05-A BY WS05-B \rightarrow $WS05-B(QUO) = WS05-A / WS05-B$

DIVIDE WS05-A BY WS05-B GIVING WS05-C REMAINDER WS05-D

5. COMPUTE. (NATURAL ARITHMATIC OPERATORS.)

*$C = A * B / F (- 23 + 2) **4$*

BODMAS RULE IS FOLLOWED

*COMPUTE $C = A * B / F (- 23 + 2) **4$*

*COMPUTE C ROUNDED = $(A * B) / 2 * (- 23 + 2) **4$*

Note: a Space must be given before and after the operator.

Conditions

- *Relation Condition*
 $<$, LESS THAN, $>$, GREATER THAN, EQUALS TO, $=$, $<=$, $>=$
- *Sign Condition* - POSITIVE NEGATIVE, ZERO
- *Class Condition* - ALPHABETIC, NUMERIC, ALPHABETIC-LOWER, ALPHABETIC-UPPER
- *Condition-Name* 88 level number declaration.
- *Negated Condition* - NOT
- *Combined Condition* - LOGICAL OPERATORS. AND & OR

Conditional statements

```

IF ( CONDITION) THEN
    IMP STMNT
ELSE IF(CONDITION) THEN
    IMP STMNTS
ELSE
    IMP STMNTS
END-IF
END-IF.

```

Note: The number of end-if must be equal to the number of IF in the structure.

There MUST NOT BE A PERIOD ANYWHERE INBETWEEN IF AND END-IF

Evaluate

```

EVALUATE TRUE/FALSE/VARIABLE
WHEN CONDITION

```

```
    IMP
WHEN CONDITON
    IMP
WHEN OTHER
    IMP
END-EVALUATE.
```

EXAMPLE:

```
IF ( WS05-A > WS05-B AND WS05-A > WS05-C ) THEN
    DISPLAY 'A IS THE GREATEST'
ELSE IF ( WS05-B > WS05-A AND WS05-B > WS05-C ) THEN
    DISPLAY ' B IS THE GREATEST'
ELSE
    DISPLAY ' C IS THE GREATEST'
END-IF
END-IF.
```

```
EVALUATE TRUE
WHEN (WS05-A > WS05-B AND WS05-A > WS05-C )
    DISPLAY ' A IS THE GREATEST'
WHEN ( WS05-B > WS05-A AND WS05-B > WS05-C )
    DISPLAY ' B IS THE GREATEST'
WHEN OTHER
    DISPLAY ' C IS THE GFREATEST'
END-EVALUATE.
```

LOOPS. ITERATION STATEMENTS.

PERFORM

1. CONDITIONAL

Until a condition is satisfied the loop run.

a. Inline perform

When the group of statements to be iterated lies between "perform" and 'end-perform'.

WS05-A PIC 9(02) VALUE 5.
PERFORM UNTIL WS05-A > 10
 DISPLAY 'HI'
 Add 1 TO ws05-a
END-PERFORM.

Ans: 6 times.

WS05-A PIC 9(02) VALUE 5.
PERFORM UNTIL WS05-A > 10
 DISPLAY 'HI'
END-PERFORM.

Ans: infinite

a. Infinity 722abend
Ws05-a pic 9(02) value 5.

PERFORM VARYING WS05-A FROM 1 BY 1 UNTIL WS05-A > 10
 DISPLAY 'HI'
 DISPLAY WS05-A
END-PERFORM.
ANS : 10

b. Out-of-the-line

When a perform statement performs/calls a paragraph that is/not in the sequence of the statements.

The perform statement jumps to the paragraph and executes the statements in the paragraph and after doing so it comes back to the next from where it went.

PROCEDURE DIVISION.

PERFORM PARA-A UNTIL WS05-A > 5

PERFORM PARA-B VARYING WS05-A FROM 1 BY 1 UNTIL WS05-A >10

PERFORM TERM-PARA.

TERMP-PARA.

STOP RUN.

PARA-A.

DISPLAY 'HI'.

ADD 1 TO WS05-A

PARA-B.

DISPLAY 'HELLO'.

2. UNCONDITIONAL

a. Inline perform

When the group of sentence that are to be iterated are written between perform and end-perform stmnts.

PERFORM 5 TIMES

DISPLAY ' HI'

END-PERFORM.

PERFORM

DISPLAY ' HI'

END-PERFORM.

PERFORM WS05-A TIMES

DISPLAY 'HI'

END-PERFORM

ANS:ws05-a times

PERFORM WS05-A TIMES

DISPLAY 'HI'

ADD 1 TO WS05-A

DISPLAY WS05-A

END-PERFORM

b. Out-of-the-line

Doesn't check any condition to be satisfied.

When a paragraph is performed.

The program can be executed out of sequence.

PROCEDURE DIVISION.

PERFORM PARA-A

PERFORM PARA-B 3 TIMES

PERFORM TERM-PARA.

TERMP-PARA.

STOP RUN.

PARA-A.

DISPLAY 'HI'.

PARA-B.

DISPLAY 'HELLO'.

CONTROL STATEMENTS.

CONTINUE

Transfers the control to the next COBOL statement which comes next in the program flow.

NEXT SENTENCE

Transfers control to the next COBOL statement, which is immediately after the sentence ending with period.

Copy:

A group reusable statements can be written in any pds member.

They can be COPIed into any division of your program.

*Note: we need to mention the PDS in which the copy member is kept,
in the compile jcl → compile.syslib.*

The member is called COPY BOOK

The PDS is called copy library.

Syntax: copy memname

FILE HANDLING

FILES → TRANSACTION, MASTER, BATCH

Processing data from a dataset and sending it to another datasets.

1. Link the file with program.

ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT T1001-PS ASSIGN TO DDNAME

ORGANIZATION IS SEQUENTIAL

ACCESS IS SEQUENTIAL

FILE STATUS IS WS05-FST-T1001.

ORGANIZATION → SEQUENTIAL PS,ESDS

INDEXED - KSDS

RELATIVE - RRDS

LINEAR - LDS

ACCESS → SEQUENTIAL

RANDOM

DYNAMIC

2. File variables. File layout.

Declare variables for the fields in the file.

6090 TOMMY INDIA 03800.89

DATA DIVISION.

FILE SECTION.

FD T1001-PS.

01 T1001-PS-REC.

05 T1001-ID	PIC 9(04).
05 FILLER	PIC X(01).
05 T1001-NAME	PIC A(05).
05 FILLER	PIC X(01).
05 T1001-LOC	PIC A(09).
05 FILLER	PIC X(01).
05 T1001-SAL	PIC 9(05).9(02).
05 FILLER	PIC X(51).

3. Open appropriate mode

PROCEDURE DIVISION.

SYNTAX: OPEN MODE LOGICAL-FILENAME

INPUT → reading from it

OUTPUT → Writing into it

I-O → reading, rewriting from and into it

EXTEND → Writing records by appending.

4. Read, write, rewrite, delete, start, read next

PROCEDURE DIVISION

READ FILENAME → one record from the dataset is copied into the file layout.

READ FILENAME

AT END

IMPERATIVE

NOT AT END

IMPERATIVE

END-READ.

WRITE RECORD-NAME → The values in the layout are moved into the file.

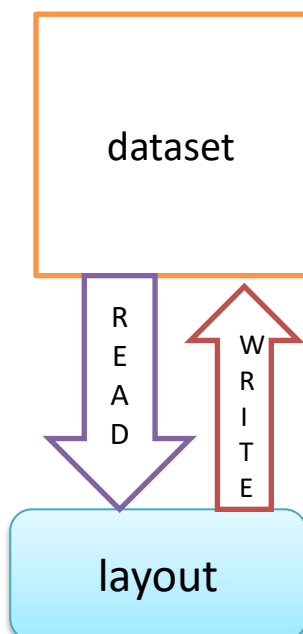
DELETE FILE-NAME → the matching record would be removed from the file.

REWRITE RECORD-NAME → MODIFIES THE RECORD
START, READ-NEXT DYNAMIC ACCESS.

DATASETS AND ITS ALLOWED OPERATIONS AND MODES.

<i>FILENAME</i>	<i>ORGANIZATION</i>	<i>ACCESS MODES</i>	<i>OPERATIONS</i>	<i>DDNAME IN CBL</i>
<i>PS</i>	<i>SEQUENTIAL</i>	<i>SEQUENTIAL</i>	<i>READ, WRITE</i>	<i>DDNAME</i>
<i>ESDS</i>	<i>SEQUENTIAL</i>	<i>SEQUENTIAL</i>	<i>READ,WRITE</i>	<i>AS-DDNAME</i>
<i>KSDS</i>	<i>INDEXED</i>	<i>SEQUENTIAL, RANDOM, DYNAMIC</i>	<i>READ, WRITE, DELETE, REWRITE, START, READ-NEXT</i>	<i>DDNAME</i>
<i>RRDS</i>	<i>RELATIVE</i>	<i>SEQUENTIAL, RANDOM DYNAMIC</i>	<i>READ,WRITE</i>	<i>DDNAME</i>

Movement of data from and to dataset and File layout



5. Close.

PROCEDURE DIVISION.

CLOSE FILENAME.

6. File operation status.

STATUS CODE: 00 → OPERATION SUCCESS.

10 → END OF FILE IS REACHED

SCENARIO.

Sweep / read all the records from the ps and validate the records with the below validations.

The id *MUST ONLY HAVE NUMBERS*

The salary must number before and after decimal point.

The decimal point must be a period.

Display all the valid records.

Don't display the invalid records.

DATA FLOW:

OPEN THE FILE → FAILURE → KILL THE PROGRAM

→ READ UNTIL EOF → VALIDATION → DISPLAY

INVALID → SKIP

CLOSE

PROCEDURE DIVISION

0000-MAIN-PARA.

PERFORM 1000-INIT-PARA

PERFORM 3000-PROC-PARA

THRU 3000-PROC-PARA-EXIT

PERFORM 9000-TERM-PARA

.

1000-INIT-PARA.

EXIT

.

3000-PROC-PARA.

PERFORM 3100-OPEN-PARA
THRU 3100-OPEN-PAR-EXIT
PERFORM 3200-READ-PARA
THRU 3200-READ-PARA-EXIT
UNTIL WS05-FST-T1001 = 10
PERFORM 3300-CLOSE-PARA
THRU 3300-CLOSE-PARA-EXIT

.

3000-PROC-PARA-EXIT.

EXIT

.

9000-TERM-PARA.

STOP RUN

.

3100-OPEN-PARA.

OPEN INPUT T1001-PS
EVALUATE TRUE
WHEN WS05-FS-T1001 = 00
DISPLAY ' T1001 OPEN SUCCESS '
WHEN OTHER
DISPLAY ' T1001 OPEN FAILED : ' WS05-FST-T1001
PERFORM 9000-TERM-PARA
END-EVALUATE

.

3100-OPEN-PARA-EXIT.

EXIT

.

3200-READ-PARA.

READ T1001-PS
EVALUATE TRUE
WHEN WS05-FS-T1001 = 00
ADD 1 TO WS05-REC-COUNT
PERFORM 3210-VALID-PARA

```

        THRU 3210-VALID-PARA-EXIT
    WHEN WS05-FST-T1001 = 10
        IF WS05-REC-COUNT = 0
            DISPLAY 'EMPTY INPUT FILE'
        ELSE
            DISPLAY ' ALL RECORDS PROCESSED'
        END-IF
    WHEN OTHER
        DISPLAY ' T1001- READ FAILED ' WS05-FST-T1001
    END-EVALAUTE

```

3200-READ-PARA-EXIT.

```

    EXIT

```

3210-VALID-PARA.

```

    EVALAUTE TRUE
    WHEN T1001-ID      IS NUMERIC    AND
        T1001SAL(1:5) IS NUMERIC    AND
        T1001-SAL(6:1) = '.'        AND
        T1001-SAL(7:2) IS NUMERIC
        PERFORM 3211-CAL-PARA
        THRU 3211-CAL-PARA-EXIT
    WHEN OTHER
        DISPLAY ' INVALID RECORD : ' T1001-ID
    END-EVALUATE

```

3210-VALID-PARA-EXIT.

```

    EXIT

```

3300-CLOSE-PARA.

```

    CLOSE T1001-PS

```

3211-CAL-PARA.

```
DISPLAY ' THE RECORD IS A VALID RECORD'
DISPLAY ' THE ID      : ' T1001-ID
DISPLAY ' THE NAME    : ' T1001-NAME
DISPLAY ' THE LOC     : ' T1001-LOC
DISPLAY ' THE OLD SAL : ' T1001-SAL
MOVE T1001-SAL TO WS05-SAL
IF (T1001-LOC = 'CHENNAI' OR
    T1001-LOC = 'CHINA' )
    COMPUTE WS05-SAL = WS05-SAL * 1.15
ELSE IF ( T1001-LOC = 'DETROIT' OR
         T1001-LOC = 'MICHIGEN' )
    COMPUTE WS05-SAL = WS05-SAL * 1.20
ELSE
    COMPUTE WS05-SAL = WS05-SAL * 1.30
END-IF
END-IF
MOVE WS05-SAL TO T1001-SAL
DISPLAY ' THE NEW SAL : ' T1001-SAL
.
```

3211-CAL-PARA-EXIT.

```
EXIT
.
```

3300-CLOSE-PARA-EXIT.

```
EXIT
.
```

1. EMPTY INPUT FILE HANDLING.

Define a counter and add 1 to it when the read is successful.

In the read para, when EOF is checked, use if condition to check the rec counter. If the counter is 0 → empty file

Else → end file of reached.

2. Read → Invalid → display 'invalid record'.
validated → display

id :
name :
loca:
old sal:10000
new sal: 11500

br →

if the location is in asia → 15 %
else if location US → 20 %
else → 30%

SCENARIO. READ ALL RECORDS FROM THE INPUT DATSET AND VALIDATE THEM.

if the record is invalid → display invalid record

if the record is valid

display the fields of the record as well as new field.

VALIDATIONS CONDITIONS

New field logic is given

if the location is in asia → 15 %
else if location US → 20 %
else → 30%

Thought process.

How do I link? Organization, access

What is the layout? → physical decimal point, and is there any requirement that expects me to perform arithmetic operation on that variable?

What mode do I open? input

Read until eof

- Move spaces to the layout
- Read a filename
- Check the file status variable
 - If file status is 00
 - Add 1 to rec-counter
 - Perform - valid-para
 - Else if the file status = 10
 - ***** check for empty file
 - Else
 - ** read failed

Close

Valid - fails → display 'invalid

Passes → calculation → display the record

Data flow

Open-para

Read-para until eof

Close-para

Read-para → valid-para → calc-para → display

Failed →

Data flow

Scenario 3:

Read records from ps,

Validate the records.

Calculate the new salary

Write the valid records into another ps.

The output ps layout is as below.

Id	name	loc	old sal	new sal
Pic9(04)	pic a(05)	pic a(09)	pic 9(05).9(02)	pic 9(05).9(02)

With one filler between fields.

1. Link.

ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT T1001-PS ASSIGN TO DD1

ORGANIZATION IS SEQUENTIAL

ACCESS IS SEQUENTIAL

FILE STATUS IS WS05-FST-T1001.

SELECT T0001-PS ASSIGN TO DD2

ORGANIZATION IS SEQUENTIAL

ACCESS IS SEQUENTIAL

FILE STATUS IS WS05-FST-T0001.

2. LAYOUT.

DATA DIVISION.

FILE SECTION.

FD T1001-PS.

01 T1001-PS-REC.

05 T1001-ID PIC 9(04).

05 FILLER PIC X(01).

05 T1001-NAME PIC A(05).

05 FILLER PIC X(01).

05 T1001-LOC PIC A(09).

05 FILLER	PIC X(01).
05 T1001-SAL	PIC 9(05).9(02).
05 FILLER	PIC X(51).

FD T0001-PS.

01 T0001-PS-REC.

05 T0001-ID	PIC 9(04).
05 FILLER	PIC X(01).
05 T0001-NAME	PIC A(05).
05 FILLER	PIC X(01).
05 T0001-LOC	PIC A(09).
05 FILLER	PIC X(01).
05 T0001-SAL	PIC 9(05).9(02).
05 FILLER	PIC X(01).
05 T0001-NEW-SAL	PIC 9(05).9(02)
05 FILLER	PIC X(41).

3. OPEN

OPEN INPUT T1001-PS

OPEN OUTPUT T0001-PS

4. READ → VALID → CAL → WRITE

WRITE → PLACE THE VALUES IN THE OUTPUT LAYOUT, THAT YOU ARE PLANNING TO WRITE.(MOVE STATEMENTS)

WRITE T0001-PS-REC.

PROCEDURE DIVISION

0000-MAIN-PARA.

PERFORM 1000-INIT-PARA

PERFORM 3000-PROC-PARA

THRU 3000-PROC-PARA-EXIT

PERFORM 9000-TERM-PARA

.

1000-INIT-PARA.

EXIT

.

3000-PROC-PARA.

PERFORM 3100-OPEN-PARA
THRU 3100-OPEN-PAR-EXIT
PERFORM 3200-READ-PARA
THRU 3200-READ-PARA-EXIT
UNTIL WS05-FST-T1001 = 10
PERFORM 3300-CLOSE-PARA
THRU 3300-CLOSE-PARA-EXIT

.

3000-PROC-PARA-EXIT.

EXIT

.

9000-TERM-PARA.

STOP RUN

.

3100-OPEN-PARA.

OPEN INPUT T1001-PS
EVALUATE TRUE
WHEN WS05-FS-T1001 = 00
DISPLAY ' T1001 OPEN SUCCESS '
WHEN OTHER
DISPLAY ' T1001 OPEN FAILED :' WS05-FST-T1001
PERFORM 9000-TERM-PARA
END-EVALUATE
OPEN OUTPUT T0001-PS
EVALUATE TRUE
WHEN WS05-FS-T0001 = 00
DISPLAY ' T0001 OPEN SUCCESS '
WHEN OTHER
DISPLAY ' T0001 OPEN FAILED :' WS05-FST-T0001
PERFORM 9000-TERM-PARA
END-EVALUATE

.
3100-OPEN-PARA-EXIT.

EXIT

.
3200-READ-PARA.

MOVE SPACES TO T1001-PS-REC TO001-PS-REC

MOVE 0 WS05-NEW-SAL

READ T1001-PS

EVALUATE TRUE

WHEN WS05-FS-T1001 = 00

ADD 1 TO WS05-REC-COUNT

PERFORM 3210-VALID-PARA

THRU 3210-VALID-PARA-EXIT

WHEN WS05-FST-T1001 = 10

IF WS05-REC-COUNT = 0

DISPLAY 'EMPTY INPUT FILE'

ELSE

DISPLAY ' ALL RECORDS PROCESSED'

END-IF

WHEN OTHER

DISPLAY ' T1001- READ FAILED ' WS05-FST-T1001

END-EVALAUTE

.
3200-READ-PARA-EXIT.

EXIT

.
3210-VALID-PARA.

EVALAUTE TRUE

WHEN T1001-ID IS NUMERIC AND

T1001SAL(1:5) IS NUMERIC AND

T1001-SAL(6:1) = '.' AND

T1001-SAL(7:2) IS NUMERIC

PERFORM 3211-CAL-PARA

THRU 3211-CAL-PARA-EXIT

WHEN OTHER

DISPLAY ' INVALID RECORD : ' T1001-ID
END-EVALUATE

.

3210-VALID-PARA-EXIT.

EXIT

.

3211-CAL-PARA.

MOVE T1001-ID TO T0001-ID
MOVE T1001-NAME TO T0001-NAME
MOVE T1001-LOC TO T0001-LOC
MOVE T1001-SAL TO T0001-SAL
MOVE T1001-SAL TO WS05-SAL
IF (T1001-LOC = 'CHENNAI' OR
T1001-LOC = 'CHINA')
COMPUTE WS05-SAL = WS05-SAL * 1.15
ELSE IF (T1001-LOC = 'DETROIT' OR
T1001-LOC = 'MICHIGEN')
COMPUTE WS05-SAL = WS05-SAL * 1.20
ELSE
COMPUTE WS05-SAL = WS05-SAL * 1.30
END-IF
END-IF
MOVE WS05-SAL TO T0001-SAL
WRITE T0001-PS-REC
EVALUATE TRUE
WHEN WS05-FST-T0001 = 00
DISPLAY ' WRITE SUCCESS'
WHEN OTHER
DISPLAY ' T0001 WRITR FAILED ' WS05-FST-T0001
END-EVALUATE

.

3211-CAL-PARA-EXIT.

EXIT

.

3300-CLOSE-PARA.

CLOSE T1001-PS T0001-PS

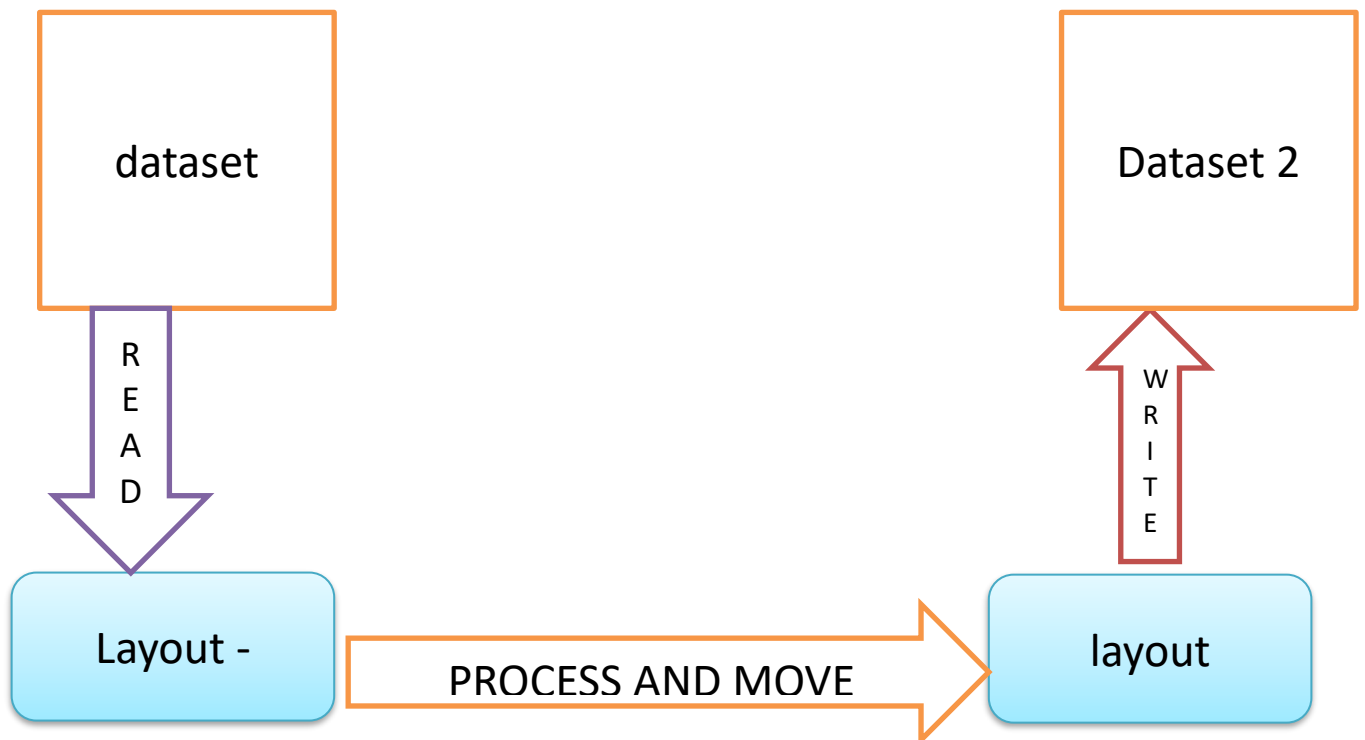
.

3300-CLOSE-PARA-EXIT.

EXIT

.

MOVEMENT OF DATA ACROSS FILES.



Scenario 4:

Read records from a ps and validate the same. For the valid records do the calculation and write in KSDS.

ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT T1001-PS ASSIGN TO DD1

ORGANIZATION IS SEQUENTIAL

ACCESS IS SEQUENTIAL

FILE STATUS	IS WS05-FST-T1001.
SELECT T0001-PS ASSIGN TO DD2	
ORGANIZATION	IS INDEXED
ACCESS	IS RANDOM
RECORD KEY	IS T0001-ID
FILE STATUS	IS WS05-FST-T0001.

Handling ksds in cobol.

1. Writing → access can be sequential → the inputs are in sorted order based on the key.
2. Writing → access can be random → never mind the sequence of the records written.
3. Reading → all records → access can be sequential.
4. Reading/deleting → by giving the key of the record → access must be RANDOM

04-10-2023

Agenda.

Delete, modify, read matching records from ksds.
 Sub program
 Character manipulation.

Delete:

Keep the key value in the key variable
 Eg: 9909. Move T1001-ID TO DB01-ID
 DELETE FILENAME.
 EVALUATE TRUE
 WHEN WS05-FST-T1002 = 00
 DISPLAY ' DELETED'
 WHEN OTHER
 DISPLAY ' DELETE FAILED ' T1002-ID
 DISPLAY WS05-FS-T1002
 END-EVALUATE

READ:

Keep the key value in the key variable

Eg: 9909. Move T1001-ID TO DB01-ID

READ FILENAME.

EVALUATE TRUE

WHEN WS05-FST-T1002 = 00

DISPLAY ' RECORD FOUND IN KSDS'

DISPLAY T1002-KSDS-REC

WHEN OTHER

DISPLAY ' MATCHING RECORD READ FAILED ' T1002-ID

DISPLAY WS05-FS-T1002

END-EVALUATE

MODIFY:

Keep the key value in the key variable

Eg: 9909. Move T1001-ID TO DB01-ID

*Note: READ → SUCCESSFUL → DO THE CHANGES TO THE
COLUMNS THAT YOU WANT TO MODIFY → REWRITE
RECORDNAME.*

READ FILENAME.

EVALUATE TRUE

WHEN WS05-FST-T1002 = 00

DISPLAY ' RECORD FOUND IN KSDS'

PERFORM 3210-REWRITE-PARA

THRU 3210-REWRITE-PARA-EXIT

WHEN OTHER

DISPLAY ' MATCHING RECORD READ FAILED ' T1002-ID

DISPLAY WS05-FS-T1002

END-EVALUATE

3210-REWRITE-PARA.

*** CHANGE THE VALUES IN THE COLUMNS EXCEPT ID CLOUMN.

REWRITE T1002-KSDS-REC

EVALUATE TRUE

WHEN WS05-FST-T1002 = 00

DISPLAY ' RECORD modified'

WHEN OTHER

DISPLAY ' rewrite failed ' T1002-ID

DISPLAY WS05-FS-T1002

END-EVALUATE

.

3210-REWRITE-PARA-EXIT.

EXIT

.

Logical file name →

H1q.q1.q2.q3.q4

ENVIRONMENT DIVISION.

INPUT-OTPUT SECTION.

FILE-CONTROL.

SELECT **MI01-PS** ASSIGN DDNAME

Record name.

DATA DIVISION.

FILE SECTION.

FD MI01-PS.

01 MI01-PS-REC.

05 MI01-ID

SCEANRIO 5:

DATASSETS:

HLQ·ALWYN·REVAT·VSAM·KSDS → DATABASE

ID (PIC 9(04), NAME PIC A(05) LOC PIC A(09) SAL PIC 9(05)·9(02)·

1 filler between fields·

TRANSACTION FILE:

HLQ·ALWYN·OPRTN·PS·

Id pic 9(04) f optrn pic A(01) fillers

1002 D

1234 R

3456 D

7987 M

5676 Z

8987 M

BR;

Sweep all the records from PS and perform the operations on KSDS·

Evaluate true

when the operation is D

Delete the matching records from ksd

when the operation is M

read → do the changes to the columns → rewrite

Modify the matching record

Change the location → new jersy

Change the salary → 10%

when the operation is R

Read the matching record and display the same in the spool

When other

Display ' ionvalid operation'

End-evaluate

Analysis:

PS select T1001-PS ASSSIGN TO DD1

Access is sequential

Organization is sequential

File status is ws05-fst-ti001.

SELECT DB01-KSDS ASSIGN TO DD2

ACCESS IS RANDOM

ORGANIZATION IS INDEXED

RECORD KEY IS DB01-ID

FILE STATUS IS WS05-FST-DB01.

OPEN INPUT T1001-PS

OPEN I-O DB01-KSDS

DATA FLOW.

PD

3000-PROC-PARA.

OPEN-PARA

READ-PS-PARA UNTIL EOF

CLOSE-PARA

3000-PROC-PARA-EXIT.

EXIT

.

READ → BRACNHC-PARA → D → DELETE PARA

R → READ-PARA

M → MODIFY PARA

O → DISPLAY INVALID OPERATION

BRNCH-PARA.

EVALUATE TRUE

WHEN T1001-OPRTN = 'D'

PERFORM 3211-DELETE-PARA

THRU 3211-DELETE-PARA-EXIT

WHEN T1001-OPRTN = 'R'

```

    PERFORM 3211-READ-PARA
        THRU 3211-READ-PARA-EXIT
    WHEN T1001-OPRTN = 'M'
        PERFORM 3211-MODIFY-PARA
            THRU 3211-MODIFY-PARA-EXIT
    WHEN OTHER
        DISPLAY 'INVALID OPERATION' T1001-OPRTN
    END-EVALUATE

```

SUB-PROGRAMs.

A reusable program that can be called from any program.

A sub program MUST NOT HAVE STOP RUN rather it can have GOBACK, END PROGRAM.

If values are to be received and sent from and to the sub program, LINKAGE SECTION must be defined in the subprogram.

The values from the main program WILL BE AUTOMATICALLY MAPPED ONLY TO THE LINKAGE SECTION variables in the subprogram.

Linkage section variables...

- Are not readily available to the procedure division.*
- Because, the linkage section variables live somewhere in the common buffer area between the main program and the sub program.*
- To use the linkage variables in the procedure division, write the USING CLAUSE in the procedure division along with list of variables in the exact sequence.*

SYNTAX of calling:

```
CALL 'SUBPGM' USING <LIST OF VARIABLES SEPERATAED BY COMMA>
```

The list of variables mentioned while calling will be mapped to the same sequence of variables in the linkage section of the subprogram. Name doesn't matter, but the data type matters.

Types:

Catalogued.

Subprogram is written in a different member.

- 1. Sub program must have GOBACK instead of end program.*
- 2. Compile the sub program first.*
- 3. Compile the main program with a new ddname
LKED.SYSLIB DD DSN=LOADLIB(SUBPGM),DISP=SHR.*
- 4. RUN THE MAINPROGRAM.*

Instream.

The subprogram is written after the last statement of the main program in the same member.

2 WAYS OF CALLING THE SUBPROGRAM.

1. CALL BY REFERENCE

By default.

The calling program's variables and the called program's variables(linkage section) share the same memory. Changing its values by sub program is reflected in the main program.

2. CALL BY CONTENT

The initial values are copied from the main program. But, the those values are changed in the sub program, it WILL NOT REFLECT IN THE MAIN PORGRAM.

The call by content variable have a different memory. Hence, changes in the sub program will not be reflected in the main.

Syntax:

*CALL 'SUBPGM' USING BY CONTENT WS05-A WS05-B
BY REFERENCE WS05-C.*