**FILE HANDLING IN COBOL**

**Different types of files:**

* Files are used to store the data in COBOL in form of DISK or TAPE.
* File is divided into records. Each record are divided into fields which contains the information about data.
* COBOL supports 3 types of files:
  + **Sequential File:**
    - Sequential file is also called as flat file (PS file).
    - ORGANISATION IS SEQUENTIAL.
    - The records are stored in sequential manner one after the other.
    - To access the Nth record, we have to read first (N-1) records also.
    - New records are always added at the end of the file.
    - It can have a fixed or variable length.
    - Recommended to use this type of file is simple file read and write is required and there is less frequent search of record is required
    - **Line Sequential file:**
      * It is a special type of sequential file where each record is separated by CR(X’0D’)/LF(X’0A’) at the end of last non-space character. Line sequential files always contain VB records.
      * It is also called text files. For report file we should define file as line sequential file.
  + **Indexed File:**
    - Indexed files are files which can be accessed faster.
    - Here access is done using Key values.
    - Indexed files use alphanumeric key as KEY.
    - We can access any record in any order using KEY.
    - ACCESS MODE can be SEQUENTIAL as well as RANDOM.
    - KEYS is INDEXED file must be unique.
    - ORGANISATION IS INDEXED.
    - We cannot update the key field in INDEXED file.
    - We can delete a particular record. The record cannot be physically deleted but that particular memory location for the records cannot be accessed.
  + **Relative organization file:**
    - They are also called as Relative Record Data Set (RRDS) file. The records in relative file can be accessed by using RRN (Relative Record Number).
    - ACCESS MODE can be SEQUENTIAL as well as RANDOM.
    - We can access the record in any order by declaring a ‘RECORD KEY’. Here the key should be a numeric key only.

**Sequential File:**

* **SELECT logical-fl ASSIGNED TO physical-fl**

**ORGANISATION IS SEQUENTIAL**

* **SELECT logical-fl ASSIGNED TO physical-fl**

**ORGANISATION IS LINE SEQUENTIAL**

* This must be coded in the FILE-CONTROL SECTION under INPUT-OUTPUT SECTION which is defined in ENVIRONMENT DIVISION.
* **FD Clause:**
  + FD stands for **File Descriptor.**
  + The layout and details of the file needs to be declared here.
* Syntax:

**ENVIRONMENT DIVISION.**

**INPUT-OUTPUT SECTION.**

**FILE-CONTROL.**

**SELECT** INFL **ASSIGN TO** DDNAME

**ORGANIZATION IS SEQUENTIAL**

**FILE STATUS** FS-INFL**.**

**DATA DIVISION.**

**FILE SECTION.**

**FD INFL.**

**01 INFL-REC.**

**05 STORE-ID PIC 9(05).**

**05 FILLER PIC X(01).**

**05 ITEM-ID PIC X(10).**

**05 FILLER PIC X(64).**

**Various options involved in File Handling – OPEN and READ:**

* A file must be opened (OPEN) first. After the successful open of the program, it can perform READ/WRITE/REWRITE/DELETE functionality.
* At the end, the file should be closed (CLOSE). After a successful execution of the program, all the files are automatically closed but it is always a good practice to close the file to avoid any issue.
* **OPEN:**
  + All the files must be opened in PROCEDURE DIVISION before any other operations are performed in it.
  + Syntax:
    - **OPEN INPUT file-name** : Opened for reading
    - **OPEN OUTPUT file-name** : Opened for writing
    - **OPEN I-O file-name** : Opened for read, write and re-write
    - **OPEN EXTEND file-name** : Opened for appending, used for Relative and Indexed
* **READ:**
  + All the input files must be READ in PROCEDURE DIVISION before the records in the files are used further.
  + At a time only one record can be read from the file, to read the next record we have to keep on reading the files till the end of the file reached.
  + Syntax: Format-1
    - **READ file-name [NEXT/PREVIOUS] RECORD INTO identifier1**

**AT END {imperative statement}**

**NOT AT END {imperative statement}**

**END-READ.**

* + Syntax: Format-2

**READ file-name RECORD INTO identifier1**

**KEY IS key-1**

**INVALID KEY {imperative statement}**

**NOT INVALID KEY {imperative statement}**

**AT END {imperative statement}**

**NOT AT END {imperative statement}**

**END-READ.**

* **START:**
  + If we want to position the pointer at a specific position in INDEXED or RELATIVE ORGANIZATION.
  + This can only be used, if file is opened in I-O mode or input mode.
  + Access mode can only be Sequential or Dynamic.
  + START will not get the value of any record but is just sets the pointer to the specific record for reading.
  + Syntax:

**START file-name**

**[KEY IS {=/>/</NOT LESS THAN/NOT > THAN ..} key1]**

**[INVALID KEY {imperative statement}]**

**[NOT INVALID KEY {imperative statement}]**

**END-START.**

**START idx-file-name**

**KEY IS EQUAL TO key1**

**INVALID KEY DISPLAY ‘INVALID KEY’**

**NOT INVALUD KEY DISPLAY ‘VALID KEY’**

**END-START.**

* **WRITE:**
  + It is used to write the content to the output file.
  + Syntax:

**WRITE file-record FROM identifier1/literal1**

**BEFORE/AFTER ADVANCING {identifier2/number2/PAGE} LINE/LINES**

**KEY IS key1**

**INVALID KEY {imperative statement}**

**NOT INVALID KEY {imperative statement}**

**AT END-OF-PAGE/EOP {imperative statement}**

**NOT AT END-OF-PAGE/EOP {imperative statement}**

**END-WRITE.**

**WRITE file-record | WRITE file-record FROM identifier1/literal1**

* + Syntax:

**FROM**-optional

**BEFORE ADVANCING** option – is used only for reports where the line must be printed before next page is advanced.

**AFTER ADVANCING** option is used only for Reports where the line must be printed after next page is advanced.

* **REWRITE:**
  + This is used to update record in a file
  + The file must be opened in I-O mode
  + Syntax:

**REWRITE file-records FROM identifier1/literal1**

**INVALID KEY {imperative statement}**

**NOT INVALID KEY {imperative statement}**

**END-REWRITE.**

* **DELETE:**
  + This is used to delete a record in a file
  + The file must be opened in I-O mode
  + Syntax:

**DELETE file-name RECORD FROM identifier1/liter1**

**INVALID KEY {imperative statement}**

**NOT INVALID KEY {imperative statement}**

**END-DELETE.**

* + DELETE operation is only used for indexed or relative files only
  + If the file is opened in random access mode, then provide the record key to perform the DELETE operation.
  + This cannot be done for sequential files
* **CLOSE:**
  + This is used to close file explicitly.
  + **CLOSE file-name.**

**Various access mode off file:**

* **Sequential Access:** The records are accessed in sequential order.
* **Random Access:** To access a record randomly. Suppose, we know the key value for access, we can use Random access.
* **Dynamic Access:** It is the combination of sequential access and random access.

**Program to read sequential file and output in SPOOL:**

**IDENTIFICATION DIVISION.**

**PROGRAM-ID. READFL.**

**ENVIRONMENT DIVISION.**

**INPUT-OUTPUT SECTION.**

**FILE CONTROL.**

**SELECT INFL ASSIGN TO INFLDD**

**ORGANIZATION IS SEQUENTIAL**

**FILE STATUS FS-INFL**

**DATA DIVISION.**

**FILE SECTION.**

**FD INFL.**

**01 INFL-REC.**

**05 STORE-ID PIC 9(05).**

**05 FILLER PIC X(01).**

**05 ITEM-ID PIC X(10).**

**05 FILLER PIC X(64).**

**WORKNG-STORAGE SECTION.**

**01 FS-INFL PIC X(02) VALUE SPACES.**

**88 FS-INFL-OK VALUE ‘00’.**

**88 FS-INFL-EOF VALUE ‘10’.**

**01 COUNTERS.**

**05 READ-COUNT PIC 9(02).**

**05 WRITE-COUNT PIC 9(02).**

**PROCEDURE DIVISION.**

**PERFORM OPEN-PARA THRU OPEN-EXIT-PARA.**

**PERFOMR PROCESS-PARA THRU PROCESS-EXIT-PARA.**

**PERFORM CLOSE-PARA THRU CLOSE-EXIT-PARA.**

**STOP RUN.**

**OPEN-PARA.**

**INITIALIZE FS-INFL READ-COUNT WRITE-COUNT.**

**OPEN INPUT INFL**

**IF FS-INFL-OK**

**CONTINUE**

**ELSE**

**DISPLAY ‘FILE OPEN FAILED: ‘ FS-INFL**

**GO TO EXIT-PARA**

**END-IF.**

**OPEN-EXIT-PARA.**

**EXIT.**

**PROCESS-PARA.**

**PERFORM UNTIL FS-INFL-EOF**

**READ INFL**

**AT END**

**IF READ-COUNT<1**

**DISPLAY ‘NO RECORD PRESENT’**

**GO TO EXIT-PARA.**

**END-IF**

**NOT AT END**

**PERFORM WRITE-PARA THRU WRITE-EXIT-PARA**

**END-READ**

**END-PERFORM.**

**PROCESS-EXIT-PARA.**

**EXIT.**

**WRITE-PARA. /\*PRINT OUTPUT IN SPOOL\*/**

**ADD 1 TO READ-COUNT.**

**IF STORE-ID > 12346**

**DISPLAY ‘STORE ID: ‘ STORE-ID**

**DISPLAY ‘ITEM ID: ‘ ITEM-ID**

**END-IF.**

**WRITE-EXIT-PARA.**

**EXIT.**

**CLOSE-PARA.**

**CLOSE INFL.**

**CLOSE-EXIT-PARA.**

**EXIT.**

**EXIT-PARA.**

**EXIT PROGRAM.**

**Program to read sequential file and output in another file:**

**IDENTIFICATION DIVISION.**

**PROGRAM-ID. WRITEFL.**

**ENVIRONMENT DIVISION.**

**INPUT-OUTPUT SECTION.**

**FILE CONTROL.**

**SELECT INFL ASSIGN TO INFLDD**

**ORGANIZATION IS SEQUENTIAL**

**FILE STATUS FS-INFL**

**SELECT OUTFL ASSIGN TO OUTFLDD**

**ORGANIZATION IS SEQUENTIAL**

**FILE STATUS FS-OUTFL**

**DATA DIVISION.**

**FILE SECTION.**

**FD INFL.**

**01 INFL-REC.**

**05 STORE-ID PIC 9(05).**

**05 FILLER PIC X(01).**

**05 ITEM-ID PIC X(10).**

**05 FILLER PIC X(64).**

**FD OUTFL.**

**01 OUTFL-REC.**

**05 O-STORE-ID PIC 9(05).**

**05 DELIMIT PIC X(01).**

**05 O-ITEM-ID PIC X(10).**

**05 O-FILLER PIC X(64).**

**WORKNG-STORAGE SECTION.**

**01 FS-INFL PIC X(02) VALUE SPACES.**

**88 FS-INFL-OK VALUE ‘00’.**

**88 FS-INFL-EOF VALUE ‘10’.**

**01 FS-OUTFL PIC X(02) VALUE SPACES.**

**88 FS-OUTFL-OK VALUE ‘00’.**

**88 FS-OUTFL-EOF VALUE ‘10’.**

**01 COUNTERS.**

**05 READ-COUNT PIC 9(02).**

**05 WRITE-COUNT PIC 9(02).**

**PROCEDURE DIVISION.**

**PERFORM OPEN-PARA THRU OPEN-EXIT-PARA.**

**PERFOMR PROCESS-PARA THRU PROCESS-EXIT-PARA.**

**PERFORM CLOSE-PARA THRU CLOSE-EXIT-PARA.**

**STOP RUN.**

**OPEN-PARA.**

**MOVE 0 TO READ-COUNT WRITE-COUNT.**

**OPEN INPUT INFL /\*OPEN INPUT FILE\*/**

**IF FS-INFL-OK**

**CONTINUE**

**ELSE**

**DISPLAY ‘FILE OPEN FAILED: ‘ FS-INFL**

**GO TO EXIT-PARA**

**END-IF.**

**OPEN OUTPUT OUTFL /\*OPEN OUTPUT FILE\*/**

**IF FS-OUTFL-OK**

**CONTINUE**

**ELSE**

**DISPLAY ‘FILE OPEN FAILED: ‘ FS-OUTFL**

**GO TO EXIT-PARA**

**END-IF.**

**OPEN-EXIT-PARA.**

**EXIT.**

**PROCESS-PARA.**

**PERFORM UNTIL FS-INFL-EOF**

**READ INFL**

**AT END**

**IF READ-COUNT<1**

**DISPLAY ‘NO RECORD PRESENT’**

**GO TO EXIT-PARA.**

**END-IF**

**NOT AT END**

**PERFORM WRITE-PARA THRU WRITE-EXIT-PARA**

**END-READ**

**END-PERFORM.**

**PROCESS-EXIT-PARA.**

**EXIT.**

**WRITE-PARA. /\*PRINT OUTPUT IN SPOOL\*/**

**ADD 1 TO READ-COUNT.**

**IF STORE-ID > 12346**

**MOVE ‘|’ TO DELIMIT**

**MOVE STORE-ID TO O-STORE-ID**

**MOVE ITEM-ID TO O-ITEM-ID**

**WRITE OUTFL-REC /\*WRITE RECORD TO OUTFILE\*/**

**END-IF.**

**WRITE-EXIT-PARA.**

**EXIT.**

**CLOSE-PARA.**

**CLOSE INFL OUTFL.**

**CLOSE-EXIT-PARA.**

**EXIT.**

**EXIT-PARA.**

**EXIT PROGRAM.**

**Syntax of INDEXED file:**

* **SELECT logical-fl ASSIGNED TO physical-fl**

**ORGANIZATION IS INDEXED**

**ACCESS MODE IS SEQUENTIAL**

**RECORD KEY IS i-key**

**ALTERNATE KEY IS ia-key.**

* **SELECT logical-fl ASSIGNED TO physical-fl**

**ORGANIZATION IS INDEXED**

**ACCESS MODE IS DYNAMIC**

**RECORD KEY IS i-key**

**ALTERNATE KEY IS ia-key.**

* **SELECT logical-fl ASSIGNED TO physical-fl**

**ORGANIZATION IS INDEXED**

**ACCESS MODE IS RANDOM**

**RECORD KEY IS r-key. /\* r-key must be defined in FD clause in the FILE SECTION.\*/**

**Syntax of RELATIVE file:**

* **SELECT logical-fl ASSIGNED TO physical-fl**

**ORGANIZATION IS RELATIVE**

**ACCESS MODE IS SEQUENTIAL**

**RECORD KEY IS r-key.**

* **SELECT logical-fl ASSIGNED TO physical-fl**

**ORGANIZATION IS INDEXED**

**ACCESS MODE IS RANDOM**

**RECORD KEY IS r-key. /\* r-key must be defined in FD clause in the FILE SECTION.\*/**

**Program to Read data from Indexed File and Display to Spool:**

**IDENTIFICATION DIVISION.**

**PROGRAM-ID. READFL.**

**ENVIRONMENT DIVISION.**

**INPUT-OUTPUT SECTION.**

**FILE CONTROL.**

**SELECT INFL ASSIGN TO INFLDD**

**ORGANIZATION IS INDEXED**

**ACCESS MODE IS RANDOM**

**RECORD KEY IS EMP-ID**

**FILE STATUS FS-INFL**

**DATA DIVISION.**

**FILE SECTION.**

**FD INFL.**

**01 INFL-REC.**

**05 EMP-ID PIC 9(05).**

**05 FILLER PIC X(01).**

**05 EMP-NAME PIC X(19).**

**05 EMP-BYTE PIC X(55).**

**WORKNG-STORAGE SECTION.**

**01 FS-INFL PIC X(02) VALUE SPACES.**

**88 FS-INFL-OK VALUE ‘00’.**

**88 FS-INFL-DUP-KEY VALUE ‘02’.**

**88 FS-INFL-EOF VALUE ‘10’.**

**01 FS-OUTFL PIC X(02) VALUE SPACES.**

**88 FS-OUTFL-OK VALUE ‘00’.**

**88 FS-OUTFL-EOF VALUE ‘10’.**

**PROCEDURE DIVISION.**

**PERFORM OPEN-PARA THRU OPEN-EXIT-PARA.**

**PERFORM PROCESS-PARA THRU PROCESS-EXIT-PARA.**

**PERFORM CLOSE-PARA THRU CLOSE-EXIT-PARA.**

**STOP RUN.**

**OPEN-PARA.**

**INITIALIZE FS-INFL INFL-REC.**

**OPEN INPUT INFL /\*OPEN INPUT FILE\*/**

**IF FS-INFL-OK**

**CONTINUE**

**ELSE**

**DISPLAY ‘FILE OPEN FAILED: ‘ FS-INFL**

**GO TO EXIT-PARA**

**END-IF.**

**PROCESS-PARA.**

**MOVE ‘08972’ TO EMP-ID.**

**READ INFL**

**INVALID KEY**

**DISPLAY ‘INVALID KEY’**

**NOT INVALID KEY**

**DISPLAY ‘EMP-NAME: ’ EMP-NAME**

**END-READ.**

**OPEN-EXIT-PARA.**

**EXIT.**

**PROCESS-EXIT-PARA.**

**EXIT.**

**CLOSE-PARA.**

**CLOSE INFL OUTFL.**

**CLOSE-EXIT-PARA.**

**EXIT.**

**EXIT-PARA.**

**EXIT PROGRAM.**

**Program to Rewrite data from Indexed File: (File needs to be open in I/O Mode)**

**IDENTIFICATION DIVISION.**

**PROGRAM-ID. WRITEFL.**

**ENVIRONMENT DIVISION.**

**INPUT-OUTPUT SECTION.**

**FILE CONTROL.**

**SELECT IOFL ASSIGN TO IOFLDD**

**ORGANIZATION IS INDEXED**

**ACCESS MODE IS RANDOM**

**RECORD KEY IS EMP-ID**

**FILE STATUS FS-IOFL**

**DATA DIVISION.**

**FILE SECTION.**

**FD IOFL.**

**01 IOFL-REC.**

**05 EMP-ID PIC 9(05).**

**05 FILLER PIC X(01).**

**05 EMP-NAME PIC X(19).**

**05 EMP-BYTE PIC X(55).**

**WORKNG-STORAGE SECTION.**

**01 FS-IOFL PIC X(02) VALUE SPACES.**

**88 FS-IOFL-OK VALUE ‘00’.**

**88 FS-IOFL-DUP-KEY VALUE ‘02’.**

**88 FS-IOFL-EOF VALUE ‘10’.**

**PROCEDURE DIVISION.**

**PERFORM OPEN-PARA THRU OPEN-EXIT-PARA.**

**PERFORM PROCESS-PARA THRU PROCESS-EXIT-PARA.**

**PERFORM CLOSE-PARA THRU CLOSE-EXIT-PARA.**

**STOP RUN.**

**OPEN-PARA.**

**INITIALIZE FS-IOFL IOFL-REC.**

**OPEN I-O IOFL /\*OPEN INPUT FILE\*/**

**IF FS-INOFL-OK**

**CONTINUE**

**ELSE**

**DISPLAY ‘FILE OPEN FAILED: ‘ FS-IOFL**

**GO TO EXIT-PARA**

**END-IF.**

**OPEN-EXIT-PARA.**

**EXIT.**

**PROCESS-PARA.**

**MOVE ‘08814’ TO EMP-ID.**

**READ IOFL**

**KEY IS EMP-ID**

**INVALID KEY**

**DISPLAY ‘INVALID KEY’**

**NOT INVALID KEY**

**DISPLAY ‘EMP-NAME: ‘ EMP-NAME**

**END-READ**

**MOVE ‘ROGER FEDERER’ TO EMP-NAME**

**REWRITE IOFL-REC**

**END-REWRITE.**

**PROCESS-EXIT-PARA.**

**EXIT.**

**CLOSE-PARA.**

**CLOSE INFL OUTFL.**

**CLOSE-EXIT-PARA.**

**EXIT.**

**EXIT-PARA.**

**EXIT PROGRAM.**

**Program to Delete data from Indexed File: (File needs to be open in I/O Mode)**

**IDENTIFICATION DIVISION.**

**PROGRAM-ID. WRITEFL.**

**ENVIRONMENT DIVISION.**

**INPUT-OUTPUT SECTION.**

**FILE CONTROL.**

**SELECT IOFL ASSIGN TO IOFLDD**

**ORGANIZATION IS INDEXED**

**ACCESS MODE IS RANDOM**

**RECORD KEY IS EMP-ID**

**FILE STATUS FS-IOFL**

**DATA DIVISION.**

**FILE SECTION.**

**FD IOFL.**

**01 IOFL-REC.**

**05 EMP-ID PIC 9(05).**

**05 FILLER PIC X(01).**

**05 EMP-NAME PIC X(19).**

**05 EMP-BYTE PIC X(55).**

**WORKNG-STORAGE SECTION.**

**01 FS-IOFL PIC X(02) VALUE SPACES.**

**88 FS-IOFL-OK VALUE ‘00’.**

**88 FS-IOFL-DUP-KEY VALUE ‘02’.**

**88 FS-IOFL-EOF VALUE ‘10’.**

**PROCEDURE DIVISION.**

**PERFORM OPEN-PARA THRU OPEN-EXIT-PARA.**

**PERFORM PROCESS-PARA THRU PROCESS-EXIT-PARA.**

**PERFORM CLOSE-PARA THRU CLOSE-EXIT-PARA.**

**STOP RUN.**

**OPEN-PARA.**

**INITIALIZE FS-IOFL IOFL-REC.**

**OPEN I-O IOFL /\*OPEN INPUT FILE\*/**

**IF FS-INOFL-OK**

**CONTINUE**

**ELSE**

**DISPLAY ‘FILE OPEN FAILED: ‘ FS-IOFL**

**GO TO EXIT-PARA**

**END-IF.**

**OPEN OUTPUT OUTFL /\*OPEN OUTPUT FILE\*/**

**PROCESS-PARA.**

**MOVE ‘08814’ TO EMP-ID.**

**READ IOFL**

**KEY IS EMP-ID**

**INVALID KEY**

**DISPLAY ‘INVALID KEY’**

**NOT INVALID KEY**

**DISPLAY ‘EMP-NAME: ‘ EMP-NAME**

**END-READ**

**DELETE IOFL RECORD**

**INVALID KEY**

**DISPLAY ‘INVALID KEY’**

**NOT INVALID KEY**

**DISPLAY ‘EMP DELETED: ‘ EMP-NAME**

**END-DELETE.**

**OPEN-EXIT-PARA.**

**EXIT.**

**PROCESS-EXIT-PARA.**

**EXIT.**

**CLOSE-PARA.**

**CLOSE INFL OUTFL.**

**CLOSE-EXIT-PARA.**

**EXIT.**

**EXIT-PARA.**

**EXIT PROGRAM.**