UNIT 5

TABLE HANDLING

Introduction : Table Handling

Consider a situation of accepting 100 numbers from the user , display all The numbers in sorted order.

Types of tables

- **One dimensional table**
- **Two dimensional table**
- **Multidimensional table**

Table is a list of logically similar items.

Figure 5-1 ACCESS MODE : DYNAMIC

Notes:

Obviously declaring 100 data items in W-S section and sorting them becomes practically impossible.

Tables or Arrays provide the solution to handle situations discussed above.

If volume of data to be processed is large and if they are not stored in files, then tables are used.

OCCURS Clause

Specifies number of occurrences or elements of the table.

```
WORKING-STORAGE SECTION
01 Marks.
   02 Mark – Table1      OCCURS 10 TIMES PIC 9(2).        Valid

   02 Mark – Table2      PIC 9(2) OCCURS 10 TIMES         Valid

   02 Mark – Table3      PIC 9(2) OCCURS 10TIMES VALUE    Invalid
```

Figure 5-2 OCCURS Clause

Notes:

OCCURS clause causes setting up of area for holding the table elements.

Following rules must be followed with the usage of 'OCCURS' clause.

1. The Integer must be positive.
2. Clause cannot be specified for an item whose level is 01, 66, 77, 88.
3. Value clause should not be specified with occurs clause.
4. OCCURS clause can be specified for file-section entries for both group items as well as elementary items.

Subscript

Indicates the position of an element in the table.

PROCEDURE DIVISION

Marks – Table (Subscript) **Parentheses required**

Marks Table (I)	**Valid provided I declared in data division.**
Marks – Table (5)	**Valid.**
Marks – Table (12)	**Invalid.**

Figure 5-3 Subscript

Notes:

- Subscript can be a COBOL variable or a literal. Value of subscript must not exceed the range of no. of occurrences specified by ‘OCCURS’ clause.
- If OCCURS clause is specified for a group items subscript should be specified for all elementary items of that group.
- Subscript should be specified for only data items defined with OCCURS clause, whenever used in procedure division.

INDEXING

- **An "index-name" is an identifier that becomes associated with a particular table. The value in an index is the displacement from the beginning of the table based upon the length of the table element.**

- **An "index-name" may appear on an OCCURS clause, e.g.**

```
01   TABLE-OF-MONTHS.
     02  MONTHS   OCCURS  12 TIMES.
         PIC  X(10)  INDEXED BY NDX.
```

- **The "index-name" is created by the compiler; it does not have to be defined elsewhere in the program.**

- **The contents of an index may be changed by the SET TO statement**

- **An index may not be used in a MOVE statement or an INITIALIZE statement.**

Figure 5-4 INDEXING

Notes:

Indexing allows such operations as table searching and manipulating specific items. To use indexing you associate one or more index-names with an item whose data description entry contains an OCCURS clause. An index associated with an index-name acts as a subscript, and its value corresponds to an occurrence number for the item to which the index-name is associated.

The INDEXED BY phrase, by which the index-name is identified and associated with its table, is an optional part of the OCCURS clause. There is no separate entry to describe the index associated with index-name. At run time, the contents of the index corresponds to an occurrence number for that specific dimension of the table with which the index is associated.

One Dimensional Tables

Specified by one subscript or index

Example:

```
01  ABC.
      02  XYZ      PIC    X(10) OCCURS 10 TIMES.
```

XYZ(1) WHERE 1 Specifies the first element of XYZ.

Figure 5-5 Two Dimension Tables

Notes:

Two Dimensional Tables

Specified by two subscripts or indexes.

Student (3 5) =5th Subject of 3rd Student.
Marks (3 5) =5th Marks of 3rd Student.

If it requires storing the Marks of N subject for M students then we require two 'OCCURS' clauses.

Figure 5-6 Two Dimension Tables

Notes:

Two dimension tables are used most frequently in applications. Consider for examples, 10 Students of a class appeared for 8 subjects in their annual exams and you need to code a program to store and retrieve the data.
Data includes names of all the students, marks and names of corresponding subjects.
To store the marks of 'n' subjects of one student, one dimension table serves the purpose. If number of students is more than one than for each student there OCCURS 'n' subjects and marks. Next foil shows the W-S declarations for this example.

Multidimensional Table

Each OCCURS clause adds a dimension in nested occurs.

Ex:

```
01    Multidimensional.
      02    First-dim       OCCURS 10 TIMES     PIC  X.
      02    Second-dim      OCCURS 5 TIMES.
            05    Second                        PIC  A.
            05    Third-dim  OCCURS 10 TIMES
            10    Third                         PIC  5.
```

Figure 5-7 Multidimensional Table

Notes:

COBOL supports multidimensional tables up to 7 levels.

Table-Sorting

Use Sorting techniques to sort a table.
e.g Bubble sort

```
PERFORM VARYING I FROM 1 BY 1 UNTIL I = N
      PERFORM  J FROM I BY 1 UNTIL J > N
            IF A[I] > A[J]
               MOVE A[I] TO TEMP
               MOVE A[J] TO A[I]
               MOVE  TEMP TO A[J]
            END-IF
      END-PERFORM
END-PERFORM.
```

I and J are used as subscripts for comparing elements of the table.

Figure 5-8 Table-Sorting

Notes:

Sorting is the process of arranging the elements of table in order. Searching for a particular element of the sorted table, requires less time when compared to searching from an unsorted table.

'SORT' verb available in COBOL is limited to File sorting.

SET Verb

SET verb initializes and / or changes the value of index.

E.g. :

SET K To 1	K is initialized to 1
SET K UP BY 2	Value increment by step of 2.
SET K DOWN BY 2	Decrements by step of 2

MOVE verb cannot be used for index.

E.g. : SET data-name-1, data-name-2 TO K
SET verb moves of value of K to data-name-1 and data-name-2.

Figure 5-9 SET

Notes:

Even though indexes assume the displacement values for table elements internally, programmer sets the value of an index by specifying the position of an element.

This means an index indicates the position of an element in the table similar to subscript, but internally it is processed in a different manner, but more efficient.

SEARCH Verb

Searches for a particular value in the table, which has an index.

SEARCH Table – name AT END statement
WHEN condition statement

Ex : SET K TO 1
SEARCH table-name AT END DISPLAY ‘not found’.
WHEN field-1 = element (K) DISPLAY element (K).

Figure 5-10 SET

Notes:

In the above example, field-1 contains the required value to be searched for in the table, More than one ‘condition’ can be checked, with more than one ‘WHEN’ clause. All valid arithmetic operators can be used.

This form of search statement is called serial search.

Binary Search

Searches the table previously sorted, by splitting the table.
Faster than serial search

Only one 'WHEN' clause is allowed.

SEARCH ALL Table-Name **other clauses**
……. **remain same.**
WHEN…….…

Figure 5-11 SET

Notes:

Before applying 'SEARCH ALL' clause the table must be sorted.

SEARCH ALL causes the table to split into two halves. Then it determines which half of the table contains the required value by comparing it to the last element of the first half and first element of the second half.

Again the selected half-table splits and continues and so on until the value is located.

ILLUSTRATES ONE-DIMENSIONAL ARRAYS

```
DATA DIVISION.

WORKING STORAGE SECTION.
77    CT    PIC  99    VALUE     0.
01    TAX-RATE.
      05    RATE       PIC   999   OCCURS    5 TIMES.
```

*** There should be a space before and after the braces.**

```
01    MONTH-TABLE.
      02    FILLER     PIC   X(9)  VALUE "January".
      02    FILLER     PIC   X(9)  VALUE "February".
      02    FILLER     PIC   X(9)  VALUE "March:"
      02    FILLER     PIC   X(9)  VALUE "April".
      02    FILLER     PIC   X(9)  VALUE "May".
      02    FILLER     PIC   X(9)  VALUE "June".
      02    FILLER     PIC   X(9)  VALUE "July".
      02    FILLER     PIC   X(9)  VALUE "August".
      02    FILLER     PIC   X(9)  VALUE "September".
      02    FILLER     PIC   X(9)  VALUE "October".
      02    FILLER     PIC   X(9)  VALUE "November".
      02    FILLER     PIC   X(9)  VALUE "December".
01    MONTH-NAME REDEFINES MONTH-TABLE.
      02    MONTH      PIC   X(9) OCCURS 12 TIMES.

PROCEDURE DIVISION.

100-MAIN-PARA.
      PERFORM 200-FILL-PARA VARYING CT FROM 1 BY 1.
            UNTIL CT>5.
      PERFORM 300-DISP-PARA VARYING CT FROM 1 BY 1.
            UNTIL CT>5.
      PERFORM 400-MNTH-PARA.
```

```
        STOP RUN.

200-FILL-PARA.
        COMPUTE RATE ( CT ) = CT*100.

300-DISP-PARA.
        DISPLAY RATE ( CT ).

400-MNTH-PARA.
        DISPLAY “Month as a number ?”
        ACCEPT CT.
        IF CT< 1 OR > 12 DISPLAY “Error in number”
                ELSE DISPLAY MONTH ( CT ).
```

ILLUSTRATES SEARCH VERB

```
DATA DIVISION.
01    MONTH-TABLE.
      02    FILLER    PIC  X(9)  VALUE "January".
      02    FILLER    PIC  X(9)  VALUE "February".
      02    FILLER    PIC  X(9)  VALUE "March:"
      02    FILLER    PIC  X(9)  VALUE "April".
      02    FILLER    PIC  X(9)  VALUE "May".
      02    FILLER    PIC  X(9)  VALUE "June".
      02    FILLER    PIC  X(9)  VALUE "July".
      02    FILLER    PIC  X(9)  VALUE "August".
      02    FILLER    PIC  X(9)  VALUE "September".
      02    FILLER    PIC  X(9)  VALUE "October".
      02    FILLER    PIC  X(9)  VALUE "November".
      02    FILLER    PIC  X(9)  VALUE "December".
01    MONTH-NAME REDEFINES MONTH-TABLE.
      02    MONTH     OCCURS   12 TIMES   INDEXED BY CT.
            05    FIRST-THREE PIC X (3).
            05    BALANCE-REST PIC X (6).
77    M-NAME    PIC X (9)    VALUE SPACES.

PROCEDURE DIVISION.

MAIN PARA.
      DISPLAY "Month's Name please".
      ACCEPT M-NAME.
      SET CT TO 1.
      SEARCH MONTH AT END DISPLAY "Not Found".
            WHEN M-NAME = MONTH ( CT ).
            DISPLAY FIRST-THREE ( CT ).
      STOP RUN.
```

ILLUSTRATE TWO-DIMENSIONAL ARRAYS

```
DATA DIVISION

WORKING-STORAGE SECTION.
01    WORK-AREA
      05    MORE DATA            PIC A VALUE "Y".
      88    NO-MORE-DATA         VALUE "N".
01    TEMPERATURE-ARRAY.
      05    DAY-IN-THE-WEEK    OCCURS    24 TIMES.
            10    HOURS-IN-THE-DAY   OCCURS    24 TIMES.
                  15    DEGREE-TEMP   PIC    S9(3).
77    DAY-OF-THE-WEEK    PIC 9.
77    TIME-OF-THE-DAY     PIC 99.
77    HOUR-COUNT          PIC 99.
77    DAY-COUNT           PIC 9.
77    TOT-TEMP            PIC S999.
77    AVERAGE-TEMP        PIC S999.

PROCEDURE DIVISION.

100-MAIN-PARA.
      PERFORM 200-DATA-ACCP-RTN UNTIL NO-MORE-DATA.
      PERFORM 300-DATA-DISP-RTN.
      STOP RUN.

200-DATA-ACCP-RTN.
      DISPLAY "Day of the Week : 1-Sunday….7-Saturday :"
      ACCEPT DAY-OF-THE-WEEK.
      DISPLAY "Time of the Day during Data Collection :"
      ACCEPT  TIME-OF-THE- WEEK.
      DISPLAY "Temperature"
      ACCEPT DEGREE-TEMP (DAY-OF-THE-WEEK, TIME-OF-THE-DAY).
      DISPLAY "Anymore (Y/N):"
      ACCEPT MORE-DATA.
```

```
300-DATA-DISP-RTN.
        PERFORM VARYING   DAY-COUNT FROM 1 BY 1
                UNTIL DAY-COUNT > 7.
        PERFORM VARYING   DAY-COUNT FROM 1 BY 1
                UNTIL HOUR-COUNT > 24.
        DISPLAY " DAY : ", DAY-COUNT, " HOUR: ",
                HOUR-COUNT, "TEMP : " ,
        DEGREE-TEMP ( DAY-COUNT, HOUR-COUNT )
                ADD DEGREE-TEMP ( DAY-COUNT, HOUR-COUNT )
                        TO TOT-TEMP.
        COMPUTE AVERAGE-TEMP = TOT-TEMP / 168.
        DISPLAY " Week's Average Temperature Is : ",
                AVERAGE-TEMP".
```

ILLUSTRATES MULTIPLE INDEXES AND 3D ARRAYS

```
DATA DIVISION.

WORKING-STORAGE SECTION.
01      ENROLL-TABLE.
        02  FACULTY OCCURS 3 TIMES INDEXED BY F1, F2.
          03  DEPARTMENT OCCURS 6 TIMES INDEXED BY D1, D2.
              04  YEAR OCCURS 5 TIMES INDEXED BY Y1, Y2.
                    05  FAC            PIC X (15).
                    05  DEPT           PIC X (10).
                    05  YY             PIC 9 (4).
77      ANYMORE  PIC A VALUE "Y".

PROCEDURE DIVISION.

100-MAIN-PARA.
        SET F1, D1, F2, D2, Y2 TO 1.
        PERFORM 200-ACC-PARA UNTIL SNYMORE = 'N'.
        DISPLAY "THE CONTENTS OF 3 DIMENSIONSAL ARRAY ARE :"
        PERFORM VARYING R2 FROM 1 BY 1 UNTIL F2 > F1
              AFTER       D2 FROM 1 BY 1 UNTIL D2 > D1.
              AFTER       Y2 FROM 1 BY 1 UNTIL Y2 > Y1
                    DISPLAY RAC (F2, D2, Y2)
                    DISPLAY DEPT (F2, D2, Y2)
                    DISPLAY YY (F2, D2, Y2).
        STOP RUN.


200-ACC-PARA.
        DISPLAY "ENTER FACULTY NAME".
        ACCEPT FAC (F1, D1, Y1).
        DISPLAY "ENTER DEPARTMENT NAME:.
        ACCEPT DEPT (F1, D1, Y1).
        DISPLAY "ENTER YEAR".
        ACCEPT YY (F1, D1, Y1).
```

```
 IF Y1 = 5
   IF D1 = 6
      IF F1 = 3
        MOVE “N” TO ANYMORE
      ELSE
        SET F1 UP BY 1
      END-IF
  ELSE
       SET D1 UP BY 1
  END-IF
 ELSE
   SET Y1 UP BY 1.
DISPLAY “ANYMORE”.
ACCEPT ANYMORE
```