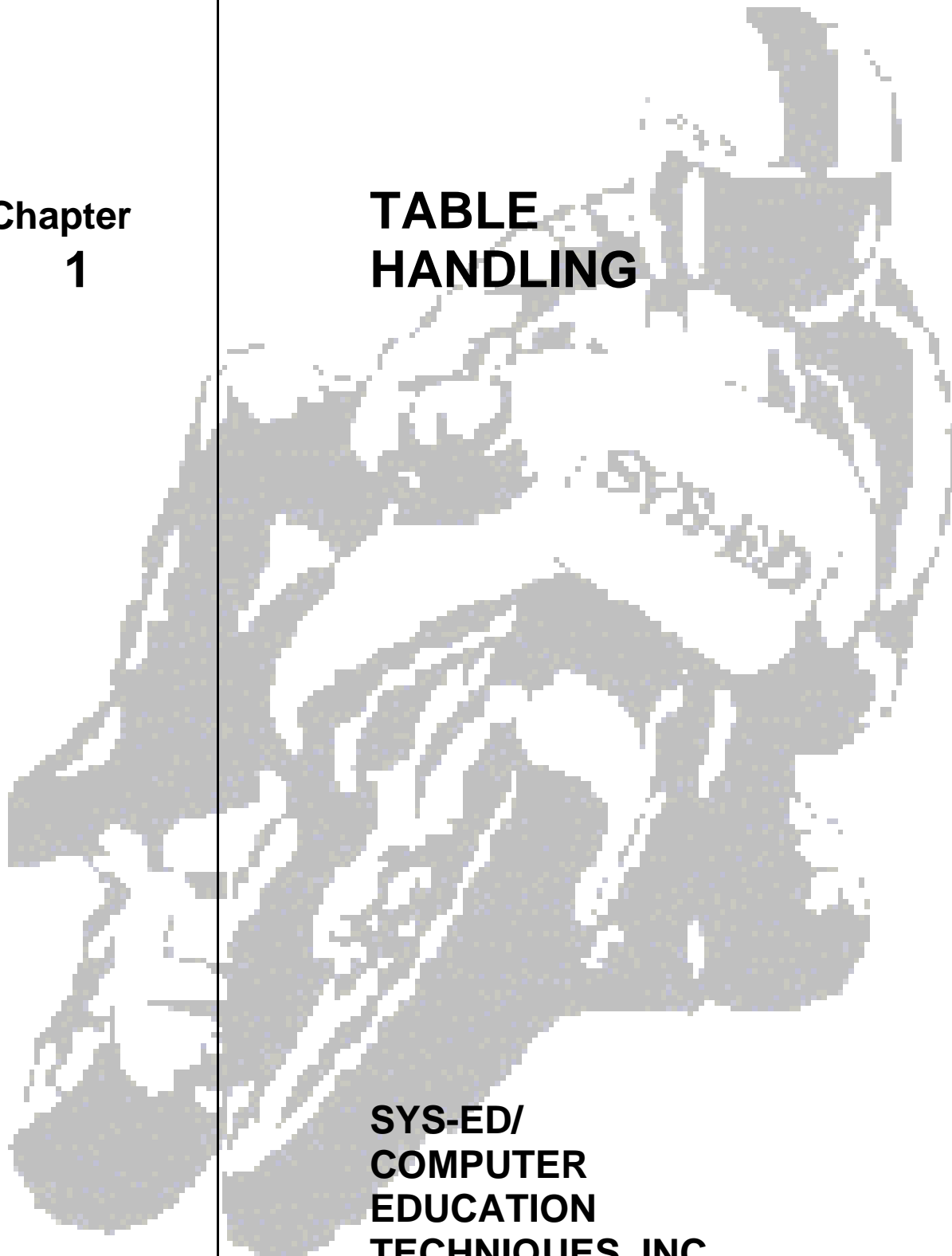


**Chapter  
1**

**TABLE  
HANDLING**



**SYS-ED/  
COMPUTER  
EDUCATION  
TECHNIQUES, INC.**

**Objectives**

You will learn:

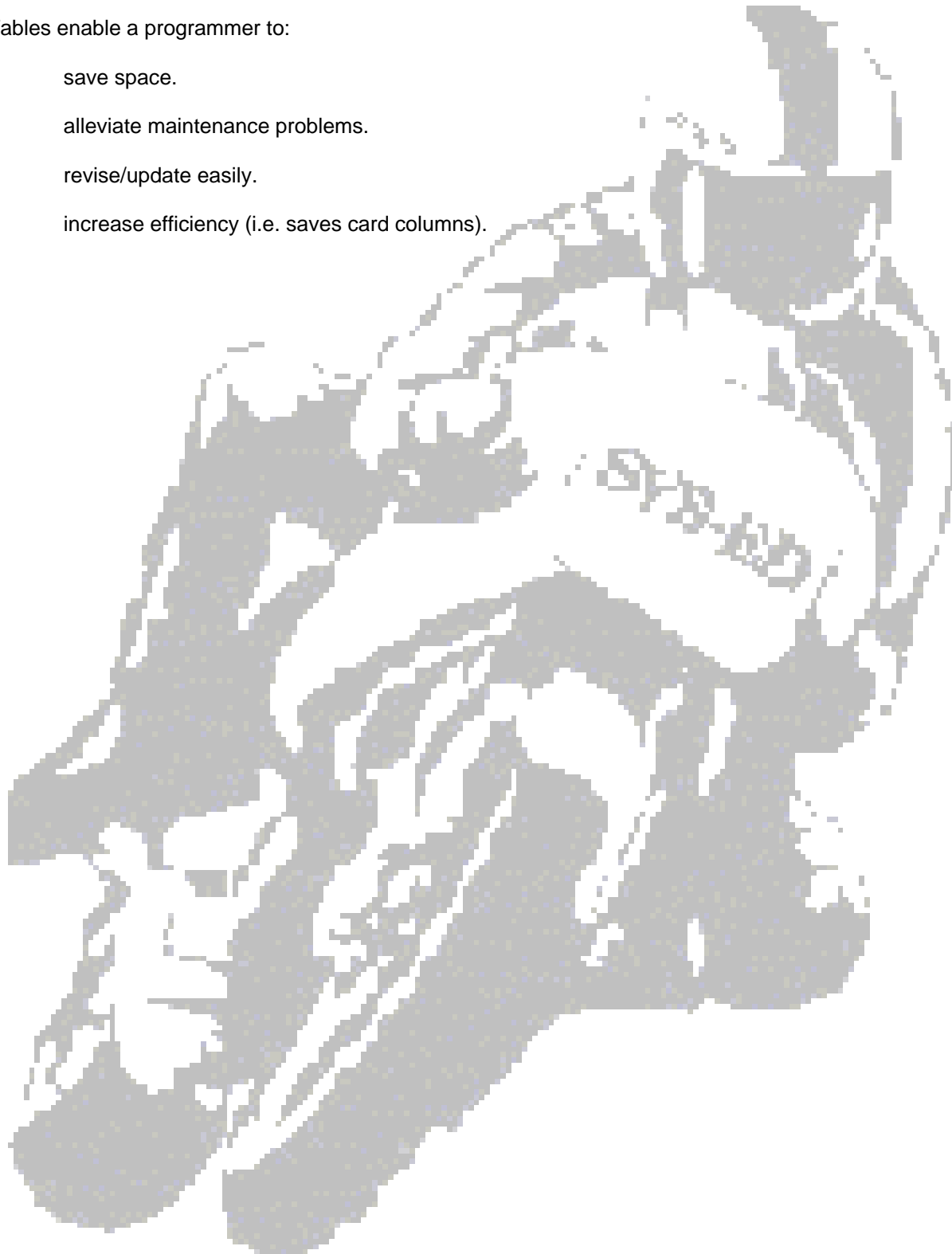
- C When to use a table.
- C How to allocate and initialize a table.
- C Differences between a subscripted and indexed table.
- C Sequential searching of a table.
- C How to perform a binary search of a table.
- C Table handling techniques.

---

**1 Why Use Tables?**

Tables enable a programmer to:

- C save space.
- C alleviate maintenance problems.
- C revise/update easily.
- C increase efficiency (i.e. saves card columns).



## 2 Table Handling

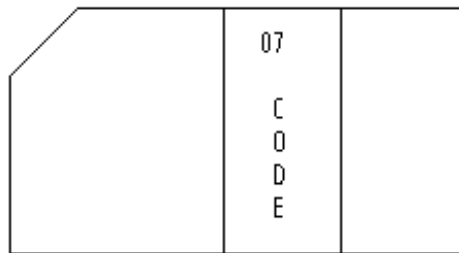
The table handling features of COBOL enables the programmer to conveniently process tables of data, or lists of repeating data items.

A COBOL table is an array consisting of up to seven dimensions.

Tables are constructed in the Data Division in the same manner as record descriptions and data descriptions, except that an OCCURS clause is used to indicate the number of times a data item is to be repeated.

Entries within a table are accessed by means of a subscript or an index.

### 2.1 Table Example



PAY RATE CODE

\$1.25	\$3.00	\$4.00			
0	0	0	0	\$8.25	\$9.00
				\$9.75	

Table Pay Rate

---

### 3 Examples

---

#### 3.1 Simple Table Program

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     LAB1.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT EMPF ASSIGN TO PAYDD.
DATA DIVISION.
FILE SECTION.
1000 FD      EMPF
1100        LABEL RECORD IS STANDARD.
1200 01      EMP-REC.
1300        05 EMP-ID      PIC X(9).
1400        05 EMP-NAME   PIC X(30).
1600        05 FILLER    PIC X(41).

WORKING-STORAGE SECTION.
01 EOF-FLAG      PIC X VALUE 'N'.
01 SUB           PIC S9(4) COMP VALUE ZERO.
01 TBL.
    05 TBL-ID      PIC X(9) OCCURS 10 TIMES.
    05 TBL-NAME   PIC X(30) OCCURS 10 TIMES.
PROCEDURE DIVISION.
MAINLINE SECTION.
    DISPLAY 'START OF TABLE PROGRAM'
    OPEN INPUT EMPF.
    READ EMPF
        AT END MOVE 'Y' TO EOF-FLAG
    END-READ
    PERFORM UNTIL EOF-FLAG = 'Y'
        ADD 1 TO SUB
        IF SUB > 10
            DISPLAY 'TO MANY RECORDS. SORRY CHARLIE'
            CLOSE EMPF
            GOBACK
        END-IF
        MOVE EMP-ID TO TBL-ID(SUB)
        MOVE EMP-NAME TO TBL-NAME (SUB)
        DISPLAY EMP-ID ' ' EMP-NAME
        READ EMPF
            AT END MOVE 'Y' TO EOF-FLAG
        END-READ
    END-PERFORM
    CLOSE EMPF
    GOBACK.
```

---

### 3.2 Two Dimensional Table

```
IDENTIFICATION DIVISION.
PROGRAM-ID.      LAB1.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT EMPF ASSIGN TO PAYDD.
DATA DIVISION.
FILE SECTION.
1000  FD  EMPF
1100      LABEL RECORD IS STANDARD.
1200  01  EMP-REC.
1300      05  EMP-ID      PIC X(9).
1400      05  EMP-NAME   PIC X(30).
1500      05  EMP-DEPT  PIC 99.
1600      05  FILLER    PIC X(39).

WORKING-STORAGE SECTION.
01  EOF-FLAG      PIC X VALUE 'N'.
01  SUB          PIC S9(4) COMP VALUE ZERO.
*TWO DIMENSIONAL TABLE
*EMP-DEPT IS A VALUE BETWEEN 1 AND 10
01  TBL.
    05  FILLER OCCURS 10 TIMES.
    10  FILLER OCCURS 100 TIMES.
    15  TBL-ID      PIC X(9).
    15  TBL-NAME   PIC X(30).

01  TBL-SUB.
    05  FILLER OCCURS 10 TIMES.
    10  MAX-SUB  PIC S9(4) COMP.

PROCEDURE DIVISION.
MAINLINE SECTION.
    DISPLAY 'START OF TABLE PROGRAM'
    OPEN INPUT EMPF.
    INITIALIZE TBL-SUB
    READ EMPF
        AT END MOVE 'Y' TO EOF-FLAG
    END-READ
```

```

PERFORM UNTIL EOF-FLAG = 'Y'
  ADD 1 TO MAX-SUB (EMP-DEPT)
  MOVE MAX-SUB (EMP-DEPT) TO SUB

  IF SUB > 10
    DISPLAY 'TO MANY RECORDS. SORRY CHARLIE'
    CLOSE EMPF
    GOBACK
  END-IF

  MOVE EMP-ID TO TBL-ID(EMP-DEPT , SUB)
  MOVE EMP-NAME TO TBL-NAME (EMP-DEPT , SUB)

  DISPLAY EMP-ID ' ' EMP-NAME

  READ EMPF
  AT END MOVE 'Y' TO EOF-FLAG
  END-READ
END-PERFORM
CLOSE EMPF
GOBACK.

```

### 3.3 Three Dimensional Table

```

IDENTIFICATION DIVISION.
PROGRAM-ID.      LAB1.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
  SELECT EMPF ASSIGN TO PAYDD.
DATA DIVISION.
FILE SECTION.
1000  FD  EMPF
1100    LABEL RECORD IS STANDARD.
1200  01  EMP-REC.
1300    05  EMP-ID      PIC X(9).
1400    05  EMP-NAME    PIC X(30).
        05  EMP-DEPT   PIC 99.
        05  EMP-STATUS PIC X.
1600    05  FILLER     PIC X(38).

```

```
WORKING-STORAGE SECTION.
01 EOF-FLAG          PIC X VALUE 'N'.
01 SUB               PIC S9(4) COMP VALUE ZERO.
01 SUB2              PIC S9(4) COMP VALUE ZERO.
*THREE DIMENSIONAL TABLE
*EMP-DEPT IS A VALUE BETWEEN 1 AND 10
01 TBL.
   05 FILLER OCCURS 10 TIMES.
   10 FILLER OCCURS 100 TIMES.
   15 FILLER OCCURS 2 TIMES.
       20 TBL-ID          PIC X(9).
       20 TBL-NAME       PIC X(30).

01 TBL-SUB.
   05 FILLER OCCURS 10 TIMES.
   10 MAX-SUB PIC S9(4) COMP.

PROCEDURE DIVISION.

MAINLINE SECTION.
  DISPLAY 'START OF TABLE PROGRAM'
  OPEN INPUT EMPF.
  INITIALIZE TBL-SUB
  READ EMPF
  AT END MOVE 'Y' TO EOF-FLAG
  END-READ
  PERFORM UNTIL EOF-FLAG = 'Y'
    ADD 1 TO MAX-SUB (EMP-DEPT)
    MOVE MAX-SUB (EMP-DEPT) TO SUB

    IF SUB > 10
      DISPLAY 'TO MANY RECORDS. SORRY CHARLIE'
      CLOSE EMPF
      GOBACK
    END-IF

    IF EMP-STATUS = 'P'
      MOVE 1 TO SUB2
    ELSE
      MOVE 2 TO SUB2
    END-IF
```

```
MOVE EMP-ID TO
    TBL-ID(EMP-DEPT , SUB, SUB2)
MOVE EMP-NAME TO
    TBL-NAME (EMP-DEPT , SUB, SUB2)

DISPLAY EMP-ID ' ' EMP-NAME

READ EMPF
    AT END MOVE 'Y' TO EOF-FLAG
END-READ
END-PERFORM
CLOSE EMPF
GOBACK.
```

---

## 4 Data Division Considerations for Tables

The OCCURS clause is used to indicate the number of times a series of items with identical format is repeated.

---

### 4.1 OCCURS Clause Format

Format 1:

OCCURS integer-2 times

```
[ < ASCENDING |  
  > KEY IS data-name-2 [data-name-3]... ]...  
[ DESCENDING |  
  ]...  
[INDEXED BY index-name-1 [index-name-2]...]...
```

Format 2:

OCCURS integer-1 TO integer-2 times  
[DEPENDING ON data-name-1]

```
[ < ASCENDING |  
  > KEY IS data-name-2 [data-name-3]... ]...  
[ DESCENDING |  
  ]...  
[INDEXED BY index-name-1 [index-name-2]...]...
```

An example of a fixed-length table without an index:

```

01  TABLE-1.
      05  TABLE-ENTRIES OCCURS 10 TIMES.
          10  DEPT-NO          PIC XX.
          10  DEPT-NAME       PIC X(20).
    
```

An example of a variable-length table without an index:

```

01  TABLE-2.
      05  DEPENDING-COUNTER   PIC S(4) COMP.
      05  TABLE-ENTRIES OCCURS 1 TO 10 TIMES
          DEPENDING ON DEPENDING-COUNTER.
          10  DEPT-NO          PIC XX.
          10  DEPT-NAME       PIC X(20).
    
```

An example of a fixed-length table with an index and without a key specified:

```

01  TABLE-3.
      05  TABLE-ENTRIES OCCURS 10 TIMES INDEXED BY INDEX-ONE.
          10  DEPT-NO          PIC X(2).
          10  DEPT-NAME       PIC X(20).
    
```

An example of a fixed-length table with an index and with a key specified:

```

01  TABLE-4.
      05  TABLE-ENTRIES OCCURS 10 TIMES ASCENDING KEY IS DEPT-NO
          INDEXED BY INDEX-ONE.
          10  DEPT-NO          PIC XX.
          10  DEPT-NAME       PIC X(20).
    
```

---

## 5 Subscripting

Subscripts may be used to refer to individual items within a table or a list of items which have been assigned individual data-names.

---

### 5.1 SUBSCRIPT Format

---

```
data-name (subscript1 [, subscript2] [,subscript3])
```

---

An example of subscripting:

```
MOVE DATA-ITEM (SUB1, SUB2) TO FIELD-5.
```

## Programming Example of Subscripting

The following example illustrates the concept of table handling using subscripting.

One Dimensional Table (Using Subscripting):

```

ID DIVISION.
.
DATA DIVISION.
.
WORKING-STORAGE SECTION.
.
77   FOUND-SWITCH                PIC X VALUE SPACE.
88   DEPT-FOUND                  VALUE 'Y'.

01   WS-TRANS-RECORD
05   WS-SS-NUMBER                PIC X(9).
05   WS-DEPT-NO                 PIC XX.
05   WS-HOURS                   PIC 99.
05   FILLER                     PIC X(63).
05   WS-TRANS-CODE              PIC XX.

01   DEPT-TABLE-ENTRIES.
05   FILLER                     PIC X(22) VALUE
`01DATA PROCESSING'.
05   FILLER                     PIC X(22) VALUE
`02ACCOUNTING'.
05   FILLER                     PIC X(22) VALUE
`03PAYROLL'.
05   FILLER                     PIC X(22) VALUE
`04ACCOUNTS PAYABLE'.
05   FILLER                     PIC X(22) VALUE
`05SALES'.
05   FILLER                     PIC X(22) VALUE
`06ACCOUNTS RECEIVABLE'.
05   FILLER                     PIC X(22) VALUE
`07SERVICE'.
05   FILLER                     PIC X(22) VALUE
`08TELEPHONE'.
05   FILLER                     PIC X(22) VALUE
`09HOUSEKEEPING'.
05   FILLER                     PIC X(22) VALUE
`99MISC'.

01   TABLE-1 REDEFINES DEPT-TABLE-ENTRIES.
05   TABLE-ENTRIES OCCURS 10 TIMES.
10   DEPT-NO                    PIC X(2).
10   DEPT-NAME                  PIC X(20).

01   SUB-ONE                    PIC S9(4) COMP.

```

```
PROCEDURE DIVISION.  
.  
PERFORM 100-TABLE-LOOK-UP VARYING SUB-ONE  
    FROM 1 BY 1  
    UNTIL SUB-ONE > 10  
    OR DEPT-FOUND.  
  
IF DEPT-FOUND  
    NEXT SENTENCE  
ELSE  
    .  
    .  
100-TABLE-LOOK-UP.  
    IF DEPT-NO (SUB-ONE) EQUAL WS-DEPT-NO  
        MOVE DEPT-NAME (SUB-ONE) TO _____  
        MOVE `Y' TO FOUND-SWITCH.
```

---

## 6 Indexing

An index can be used similarly to a subscript in the Procedure Division to reference individual items in a table.

However, an index is defined in the Working Storage Section by using the INDEXED BY option when defining a table or USAGE IS INDEXED, and when defining it as an elementary data item.

The SET Statement is used to alter an index, while the SEARCH Statement is used to locate an item in an INDEXed table.

---

### 6.1 Searching without the SEARCH Verb

Searching a table using an index is similar to searching a table using a subscript, except that an index must be initialized, incremented or decremented using the SET statement or a PERFORM-VARYING statement.

---

### 6.2 SET Statement Format

Format 1:

```

SET      | index-name-1 [index-name-2] . . . |
      <  | identifier-1 [identifier-2] . . . |
      >
      |
      | index-name-3 |
TO      < identifier-3 >
      | literal-1   |

```

Format 2:

```

SET index-name-4 [index-name-5] . . .
      | UP BY      | identifier-4 |
      <          > <          >
      | DOWN BY   | literal-2   |

```

Examples of the SET statement:

Example 1:

```
SET INDEX-1 TO 1.
```

Example 2:

```
SET DEPT-IND UP BY 3.
```

---

### 6.3 Indexed Table

```
IDENTIFICATION DIVISION.
PROGRAM-ID.      LAB1.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT EMPF ASSIGN TO PAYDD.
DATA DIVISION.
FILE SECTION.
1000 FD      EMPF
1100        LABEL RECORD IS STANDARD.
1200 01      EMP-REC.
1300        05  EMP-ID          PIC X(9).
1400        05  EMP-NAME       PIC X(30).
1600        05  FILLER         PIC X(41).

WORKING-STORAGE SECTION.
01  EOF-FLAG          PIC X VALUE 'N'.
01  SUB              PIC S9(4) COMP VALUE ZERO.
01  TBL.
    05 TBL-ID         PIC X(9) OCCURS 10 TIMES
                        INDEXED BY IND-1.
    05 TBL-NAME       PIC X(30) OCCURS 10 TIMES
                        INDEXED BY IND-2.

PROCEDURE DIVISION.

MAINLINE SECTION.
    DISPLAY 'START OF TABLE PROGRAM'

    set ind-1 to 1
    set ind-2 to 1

    OPEN INPUT EMPF.
    READ EMPF
        AT END MOVE 'Y' TO EOF-FLAG
    END-READ
    PERFORM UNTIL EOF-FLAG = 'Y'
        MOVE EMP-ID TO TBL-ID(IND-1)
        MOVE EMP-NAME TO TBL-NAME (IND-2)
        DISPLAY EMP-ID ' ' EMP-NAME
```

```

SET IND-1 UP BY 1
SET IND-2 UP BY 1
IF ind-1 > 10
    DISPLAY 'TO MANY RECORDS. SORRY CHARLIE'
    CLOSE EMPF
    GOBACK
END-IF

READ EMPF
    AT END MOVE 'Y' TO EOF-FLAG
END-READ
END-PERFORM
CLOSE EMPF
GOBACK.

```

One dimensional table using indexing and the PERFORM-VARYING statement:

```

ID DIVISION.
.
DATA DIVISION.
.
WORKING-STORAGE SECTION.
.
77  FOUND-SWITCH          PIC X VALUE SPACE.
   88  DEPT-FOUND         VALUE 'Y'.

01  WS-TRANS-RECORD.
   05  WS-SS-NUMBER       PIC X(9).
   05  WS-DEPT-NO        PIC XX.
   05  WS-HOURS          PIC 99.
   05  FILLER             PIC X(63).
   05  WS-TRANS-CODE     PIC XX.

01  DEPT-TABLE-ENTRIES.
   05  FILLER             PIC X(22) VALUE
      `01DATA PROCESSING'.
   05  FILLER             PIC X(22) VALUE
      `02ACCOUNTING'.
   05  FILLER             PIC X(22) VALUE
      `03PAYROLL'.
   05  FILLER             PIC X(22) VALUE
      `04ACCOUNTS'.
   05  FILLER             PIC X(22) VALUE
      `05SALES'.
   05  FILLER             PIC X(22) VALUE
      `06ACCOUNTS RECEIVABLE'.
   05  FILLER             PIC X(22) VALUE
      `07SERVICE'.
   05  FILLER             PIC X(22) VALUE
      `08TELEPHONE'.

```

```
05 FILLER PIC X(22) VALUE
   `09HOUSEKEEPING'.
05 FILLER PIC X(22) VALUE
   `99MISC'.
```

01 TABLE-3 REDEFINES DEPT-TABLE-ENTRIES.

```
05 TABLE-ENTRIES OCCURS 10 TIMES
   INDEXED BY INDEX-ONE.
10 DEPT-NO PIC XX.
10 DEPT-NAME PIC X(20).
```

PROCEDURE DIVISION.

```
.
.
MOVE `N' TO FOUND-SWITCH.
PERFORM 200-TABLE-LOOK-UP THRU 200-EXIT
VARYING INDEX-ONE FROM 1 BY 1
   UNTIL INDEX-ONE > 10
   OR DEPT-FOUND.
IF DEPT-FOUND
   MOVE DEPT-NAME (INDEX-ONE) TO _____
ELSE
.
.
.
200-TABLE-LOOK-UP.
   IF DEPT-NO (INDEX-ONE) EQUAL WS-DEPT-NO
   MOVE `Y' TO FOUND-SWITCH
200-EXIT.
EXIT.
```

**7 SEARCH Statement**

The SEARCH Statement is used to locate a specific table element.

There are two types of searches:

SEARCH	initiates a sequential search.
SEARCH ALL	is a binary search that can be used in tables in which the elements are in ascending or descending order by key field.

SEARCH or SEARCH ALL is coded with a WHEN condition which is evaluated and if true executes an imperative statement.

---

## 7.1 SEARCH Statement Format

---

Format 1 (used for a sequential search):

```

SEARCH identifier-1 [VARYING] < identifier-2 |
                                | index-name-1 | >
[AT END imperative-statement-1]
WHEN condition-1 < imperative-statement-2 |
                  | NEXT SENTENCE       | >
[WHEN condition-2 < imperative-statement-3 |
                  | NEXT SENTENCE       | > ] . . .

```

Format 2 (used for a binary search):

```

SEARCH ALL identifier-1
[AT END imperative-statement-1]
WHEN < relation-condition-2 |
      | condition-name-1   | >
AND < relation-condition-2 |
     | condition-name-2   | > ] . . .
      < imperative-statement-2 |
      | NEXT SENTENCE       | >

```

---

One dimensional sequential table search using the SEARCH verb:

```

ID DIVISION.
.
DATA DIVISION.
.
WORKING-STORAGE SECTION.
.
77   FOUND-SWITCH                               PIC X VALUE SPACES.
88   DEPT-FOUND                                VALUE `Y' .

01   WS-TRANS-RECORD.
05   WS-SS-NUMBER                             PIC X(9).
05   WS-DEPT-NO                               PIC XX.
05   WS-HOURS                                 PIC 99.
05   FILLER                                   PIC X(63).
05   WS-TRANS-CODE                            PIC XX.

01   DEPT-TABLE-ENTRIES
05   FILLER                                   PIC X(22) VALUE
`01DATA PROCESSING' .
05   FILLER                                   PIC X(22) VALUE
`02ACCOUNTING' .
05   FILLER                                   PIC X(22) VALUE
`03PAYROLL' .
05   FILLER                                   PIC X(22) VALUE
`04ACCOUNTS PAYABLE' .
05   FILLER                                   PIC X(22) VALUE
`05SALES' .
05   FILLER                                   PIC X(22) VALUE
`06ACCOUNTS RECEIVABLE' .
05   FILLER                                   PIC X(22) VALUE
`07SERVICE' .
05   FILLER                                   PIC X(22) VALUE
`08TELEPHONE' .
05   FILLER                                   PIC X(22) VALUE
`09HOUSEKEEPING' .
05   FILLER                                   PIC X(22) VALUE
`99MISC' .

01   TABLE-3 REDEFINES DEPT-TABLE-ENTRIES.
05   TABLE-ENTRIES OCCURS 10 TIMES INDEXED BY INDEX-ONE.
10   DEPT-NO                                  PIC XX.
10   DEPT-NAME                               PIC X(20).
    
```

```

PROCEDURE DIVISION.
.
.
PERFORM B200-TABLE-LOOK-UP.

IF DEPT-FOUND
    MOVE DEPT-NAME (INDEX-ONE) TO _____
ELSE
.
.
.
B200-TABLE-LOOK-UP.
    SET INDEX-ONE TO 1.

        SEARCH TABLE-ENTRIES
        AT END
            MOVE `N' TO FOUND-SWITCH
        WHEN DEPT-NO (INDEX-ONE) = WS-DEPT-NO
            MOVE `Y' TO FOUND-SWITCH.
    
```

One dimensional binary table search using the SEARCH Verb:

```

ID DIVISION.
.
DATA DIVISION.
.
WORKING-STORAGE SECTION.
.
77   FOUND-SWITCH              PIC X VALUE SPACES.
88   DEPT-FOUND                VALUE `Y'.

01   WS-TRANS-RECORD.
05   WS-SS-NUMBER              PIC X(9).
05   WS-DEPT-NO                PIC XX.
05   WS-HOURS                  PIC 99.
05   FILLER                    PIC X(63).
05   WS-TRANS-CODE             PIC XX.

01   DEPT-TABLE-ENTRIES.
05   FILLER                    PIC X(22) VALUE
    `01DATA PROCESSING'.
05   FILLER                    PIC X(22) VALUE
    `02ACCOUNTING'.
05   FILLER                    PIC X(22) VALUE
    `03PAYROLL'.
05   FILLER                    PIC X(22) VALUE
    `04ACCOUNTS PAYABLE'.
05   FILLER                    PIC X(22) VALUE
    `05SALES'.
    
```

```
05 FILLER PIC X(22) VALUE
`06ACCOUNTS RECEIVABLE'.
05 FILLER PIC X(22) VALUE
`07SERVICE'.
05 FILLER PIC X(22) VALUE
`08TELEPHONE'.
05 FILLER PIC X(22) VALUE
`09HOUSEKEEPING'.
05 FILLER PIC X(22) VALUE
`99MISC'.

01 TABLE-4 REDEFINES DEPT-TABLE-ENTRIES

05 TABLE-ENTRIES OCCURS 10 TIMES
ASCENDING KEY IS DEPT-NO INDEXED BY
INDEX-ONE.
10 DEPT-NO PIC XX.
10 DEPT-NAME PIC X(20).

PROCEDURE DIVISION.
.
PERFORM 2300-TABLE-LOOK-UP.
IF DEPT-FOUND
MOVE DEPT-NAME (INDEX-ONE) TO _____
ELSE
.
.
.
2300-TABLE-LOOK-UP.
SEARCH ALL TABLE-ENTRIES
AT END
MOVE `N' TO FOUND-SWITCH
WHEN DEPT-NO (INDEX-ONE) = WS-DEPT-NO
MOVE `Y' TO FOUND-SWITCH.
```