Welcome to:

Introduction to COBOL Programming

5.1 Objectives

After completing this chapter, you be able to test and debug COBOL programs you have written. Specifically, you will be able to:

- Describe three strategies for testing new or revised code
- List nine debugging tools to solve programming problems
- Describe common errors and their solutions
- Describe common abnormal and (ABEND) codes

5.2 Topics to be Covered

- Testing strategies
- Testing and debugging tools
- Common ABEND codes

5.2.1 Testing Strategies

Unit Testing

- Compile and link program with no diagnostic errors
- Prepare test data which will test every line of code
- Develop Test Plan
- Review Output
- Iteratively correct errors as they occur and retest
- Be sure program documentation is in place
- Place program in a place for system testing

System (Integration) Testing

- How inputs and outputs mesh with other components

Regression Testing

- Are Inputs and Outputs alike? Changed?

5.2.2 Testing and Debugging Tools

- JOB Output
  - First page in output may tell a story
- SYSDBOUT
  - Compile with STATE and FLOW options
  - If program ABENDS, paragraph and Program Status Word Listed
  - Program Status Word
    - Address - error location
    - Offset - position within module
    - Contents directory - program list
    - Load modules - yours is usually first
    - General Purpose Registers

- SYSUDUMP
  - A dump of your program in hexadecimal
- SYSABEND
  - A dump of system nucleus in addition to your program
  - Only if requested by system programmer
- SYSOUT
  - Use DISPLAY verb liberally to help test
  - Use READY TRACE to track paragraph names
- TEST or TESTCOB
  - Online facility for testing COBOL programs

- COBTEST
  - Online and batch facility for testing VS COBOL II
- CA/Easy Test
  - Third Party vendor product

5.3 Workshop

Do review questions on pages 5-8 and 5-9. Do not do page 5-10

Diagnose the following program problems. Determine the mainframe equivalent of the problem conditions you find.

- Open the program BOMB03.CBL for animation. Compile the program, and then, use ***Step*** to move through the logic. Do ***not*** use RUN!!!!!
- Open the program BOMB06.CBL for animation. Compile the program. You may use Step or Run for this diagnostic.
- Open the program BOMB06.CBL for animation. Compile the program You may use Step or Run for this diagnostic.
- Open the program BOMB09.CBL for animation. Compile the program You may use Step or Run for this diagnostic.
- Clean-up all syntax errors in BOMB00.CBL. Compile and Run

5.3 Workshop

1. Unit Testing, System (Integration) Testing Regression Testing
2. READY TRACE
3. COBTEST
4. 0C7
5. 001
6. c.- SYSOUT
7. B37

BOMB03.CBL - Infinite Loop - B37
BOMB06.CBL - Illegal Character in Numeric Field - OC7
BOMB09.CBL - Subscript Out of Range - OC4

Review……

At this point we should be able to:

- Describe the steps of the Programming Life Cycle
- Describe the function of the four COBOL divisions
- List the advantages and disadvantages of COBOL
- Describe the purpose of the COBOL compiler
- Understand the column structure of COBOL
- Use the Micro Focus Workbench to Edit, Syntax Check and Animate a program
- Code an identification division
- Code an environment division
- Code a data division
- Tell whether statements belong in the A-margin or B-margin
- Write a record description for a file
- Process literals and figurative constants
- Describe the mainframe COBOL compiler
- Code file 1/0 statements (OPEN, CLOSE, READ, WRITE)
- Code special 1/0 statements (ACCEPT, DISPLAY)
- Perform basic data transfer (MOVE)
- Detect when an end-of-file condition is reached
- Create a simple COBOL program using TSO/ISPF, Micro Focus
- End the program as needed (GOBACK, STOP RUN)
- Compile, link, and test a simple COBOL program
- Understand the function of an optimizer
- Test data to determine proper action
- Perform unconditional branches
- Execute sequence, selection and iteration
- Perform valid comparisons of data
- Validate data for numeric contents
- Test logical conditions using AND, OR, or NOT
- Use conditional names to clarify and reduce coding
- Use switches in a program
- ***Describe testing strategies***
- ***Describe testing and debugging tools***
- ***Recognize common abend codes***

6.1 Objectives

After completing this chapter, you be able to use compound statements and artihmetic in your COBOL programs. Specifically, you will be able to:

- Use counters in a program
- Use COBOL statements to
  - ADD
  - SUBTRACT
  - MUTIPLY
  - DIVED
  - COMPUTE
- Round routines ON SIZE ERROR
- Use RETURN-CODE register

6.2 Topics to be Covered

- COBOL arithmetic and options
- ADD
- SUBTRACT
- MULTIPLY
- DIVIDE
- COMPUTE
- Return-Code

6.2.1 COBOL Arithmetic

- 18 digits
- Five Verbs
  - ADD
  - SUBTRACT
  - MULTIPLY
  - DIVIDE
  - COMPUTE
- Return-Code

6.2.2 COBOL Arithmetic Options

- ROUNDED
  - Excess Digit Dropped
  - If first digit dropped is 5 or larger, rounding occurs

Computation Result	PICTURE CLAUSE	VALUE STORED
25.24	S99V9	25.2
25.25	S99V9	25.3
- 25.24	S99V9	-25.2
-25.25	S99V9	-25.3

6.2.2 COBOL Arithmetic Options

- ON SIZE ERROR
  - Division by zero always causes a size errot
    - Value is not stored
    - Instruction(s) performed
  - Example

```
ADD A TO B ON SIZE ERROR
        PERFORM NUMBER-RTN.
```

6.2.3 ADD

```
ADD (CORRESPONDING } identifier-1 TO identifier-2   [ROUNDED]
  [ON SIZE ERROR imperative statement-1]

    [NOT ON SIZE ERROR imperative statement-2]

    [END-ADD]
```

Examples:

```
ADD TOT-1-COUNTER TO TOT-2-COUNTER
ADD TOT-1-COUNTER, TOT-2-COUNTER GIVING TOT-3-COUNTER
ADD FICA, INCOME-TAX, HEALTH-INSURANCE
            GIVING TOTAL-DEDUCTIONS
ADD 57 TOT--1-COUNTER TO TOT-2-COUNTER
ADD 97 TO AMOUNT-DUE
ADD MONTHLY SALES TO QUARTLY-SALES ROUNDED
ADD COMMISSION TO INCOME
      ON SIZE ERROR
            PERFORM 800-CALL-IRS
                  THRU 800-CALL-IRS-EXIT
```

6.2.4 SUBTRACT

```
SUBTRACT identifier-1 FROM identifier-2 [ROUNDED]
   [GIVING identifier-3]
   [ON SIZE ERROR imperative-statement-1]
   [NOT ON SIZE ERROR imperative-statement-2]

   [END-SUBTRACT]
```

Examples:

```
SUBTRACT  TOT-1-COUNTER TOT-2-COUNTER FROM TOT-3-COUNTER
SUBTRACT FICA, INCOME-TAX, HEALTH-INSURANCE FROM
            GROSS-PAY GIVING NET-PAY
SUBTRACT 57  FROM TOT--1-COUNTER GIVING TOT-2-COUNTER
SUBTRACT  97.5  FROM AMOUNT-DUE ROUNDED
SUBTRACT COSTS FROM INCOME GIVING PROFITS
           ON SIZE ERROR
                     PERFORM 600-FILE-CHAPTER-11
```

6.2.5 MULTIPLY

```
MULTIPLY identifier-1  BY  identifier-2  GIVING {identifier-3 [ROUNDED]}...

   [ON SIZE ERROR imperative-statement-1]

   [NOT ON SIZE ERROR imperative-statement-2]

   [END-MULTIPLY]
```

Examples:

```
MULTIPLY PERCENT-1 BY SALE GIVING COMMISSION
MULTIPLY 1.08 BY DEPOSIT GIVING CURRENT-BALANCE ROUNDED
MULTIPLY NATIONAL-DEBT BY INFLATION-RATE
            GIVING NEW-DEBT
            ONSIZE ERROR
            PERFORM 900-GRAMM-RUDMAN-HOLLINGS
```

6.2.6 DIVIDE

```
DIVIDE identifier-1  [INTO] identifier-2    GIVING identifier-3 [ROUNDED]…
                     [BY]
   [REMAINDER  identifier-4]
   [ON SIZE ERROR imperative-statement-1]
   [NOT ON SIZE ERROR imperative-statement-2]
   [END-DIVIDE]
```

Examples:

```
DIVIDE COMMISSION INTO TOTAL-AMOUNT
DIVIDE 1.08 INTO CURRENT-AMOUNT ROUNDED
DIVIDE TOTAL-CHARGE BY 2 GIVING BIG-DISCOUNT
DIVIDE 5 BY PARTS-IN-STOCK GIVING CALC-PERCENTAGE
DIVIDE PARTS-IN-STOCK INTO 50 GIVING CALC-PERCENTAGE
```

6.2.7 COMPUTE

```
COMPUTE {identifier-1 [ROUNDED]}...

    [EQUAL]
    [ =   ] arithmetic-expression

    [ON SIZE ERROR imperative-statement-1]

    [NOT ON SIZE ERROR imperative-statement-2]

    [END-COMPUTE]
```

Examples:

```
COMPUTE COUNTER = 9
COMPUTE TOTAL = A-CTR + B-CTR + C-CTR
COMPUTE COST = COST * 1.10
        ON SIZE ERROR
              PERFORM 900-COST-ERROR-RTN
```

6.2.8 RETURN-CODE

- Special Register
- MOVE value (0-4095) immediately GOBACK
- Example

```
CLOSE SALES-FILE
      REPORT-FILE.
IF ERROR-COUNTER = 0
    THEN
         MOVE 0 TO RETURN-CODE
    ELSE
         DISPLAY '*** THERE WERE ' ERROR-COUNTER
                      'ERRORS IN THE PROGRAM  ***'
         MOVE 8 TO RETURN-CODE.
GOBACK.
```

6.3 Workshop

Do exercises on pages 6-11 and 6-12. Do not do the exercise on 6-13.

Add an accumulator to Program2.cbl to total a gross sales figure for the SALES.DAT file. At the end of processing, Display this total to the screen. Compile and test.

Add a line of code in Program2.cbl to Display a RETURN-CODE of 8 if the SALESCODE is not numeric. Compile and test.

6.3 Workshop

1.

A. ADD SUM-1 TO SUM-2 GIVING SUM-3.

B. ADD SALESPNL, OFFICERS, WORKERS GIVING TOTAL-EMPLYEES

C. SUBTRACT INV-SOLD FROM INVENTRY-IN-STOCK GIVING INVENTRY-IN-STOCK

D. MULTIPLY HOURS-WORKED BY PAY-RATE GIVING GROSS-PAY

E. DIVIDE TOTAL-PAYROLL BY TOTAL-EMPLYEES GIVING AVERAGE-PAY ROUNDED

2.

```
IF CURRENT-SALES IS GREATER THAN  5000
   MULTIPLY AGENT-COMMISSN BY 2 GIVING AGENT-COMMISSN
```

3.

A. COMPUTE HOURLY-RATE = ANNUAL-SALARY / 2080

B. COMPUTE FICA-WITHHELD = GROSS-PAY * FICA-RATE

```
COMPUTE GROSS-PAY = ((RATE * STD-HRS) + ((ACTUAL-HRS - STD-HRS) * OT-FACTOR))
      ROUNDED
```

4.

```
IF ACCOUNT-AMT NOT NUMERIC
      THEN MOVE 1234 TO RETURN-CODE
        DISPLAY '*** ACCOUNT AMOUNT IS NOT NUMERIC'
```

- Describe the steps of the Programming Life Cycle
- Describe the function of the four COBOL divisions
- List the advantages and disadvantages of COBOL
- Describe the purpose of the COBOL compiler
- Understand the column structure of COBOL
- Use the Micro Focus Workbench to Edit, Syntax Check and Animate a program
- Code an identification division
- Code an environment division
- Code a data division
- Tell whether statements belong in the A-margin or B-margin
- Write a record description for a file
- Process literals and figurative constants
- Describe the mainframe COBOL compiler
- Code file 1/0 statements (OPEN, CLOSE, READ, WRITE)
- Code special 1/0 statements (ACCEPT, DISPLAY)
- Perform basic data transfer (MOVE)
- Detect when an end-of-file condition is reached
- Create a simple COBOL program using TSO/ISPF, Micro Focus
- End the program as needed (GOBACK, STOP RUN)
- Compile, link, and test a simple COBOL program
- Understand the function of an optimizer
- Test data to determine proper action
- Perform unconditional branches
- Execute sequence, selection and iteration
- Perform valid comparisons of data
- Validate data for numeric contents
- Test logical conditions using AND, OR, or NOT
- Use conditional names to clarify and reduce coding
- Use switches in a program
- Describe testing and debugging tools
- Describe testing strategies
- Recognize common abend codes

Review……

At this point we should be able to:

- ***Use counters in a program***
- ***Perform calculations***
- ***Round arithmetic results***
- ***Perform an on size error***
- ***Use the RETURN-CODE Register***

7.1 Objectives

After completing this chapter, you utilize structured programming constructs in your COBOL programs. Specifically, you will be able to:

- Code sequence, selection, and iteration structures in COBOL
- Remove GO TO statements from your programs
- Create COBOL code which is readable and maintainable
- Recognize and correct unstructured code
- Call a separate COBOL module using CALL, USING and LINKAGE
- VS COBOL II Specifics

7.2 Topics to be Covered

- Advantages of structured COBOL
- Elements of structured COBOL
- Sequence, selection, iteration
- Readability
- Modularity
- CALL Statement
- Linking a subprogram

7.2.1 Advantages of Structured COBOL

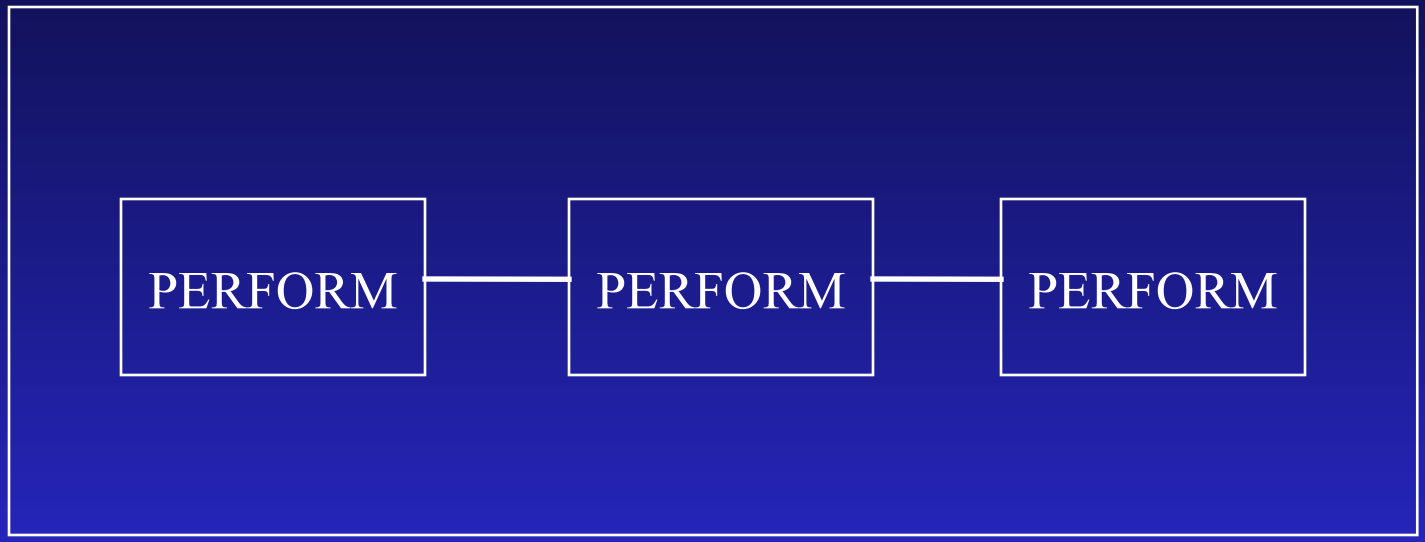
- Encourages developmental discipline
- Program is more readable
- Program logic is easier to follow
- Program is more easily maintained
- Program is better documented
- Studies show productivity increase of 70 - 300 percent
- Testing is easier
  - Each part can be tested separately

7.2.2 Elements of Structured COBOL

- Three control logic structures
  - Sequence
  - Selection
  - Iteration
- One entry and one exit
  - GO TO only used to branch to EXIT statement
- Modularity (programs, subprograms, paragraphs

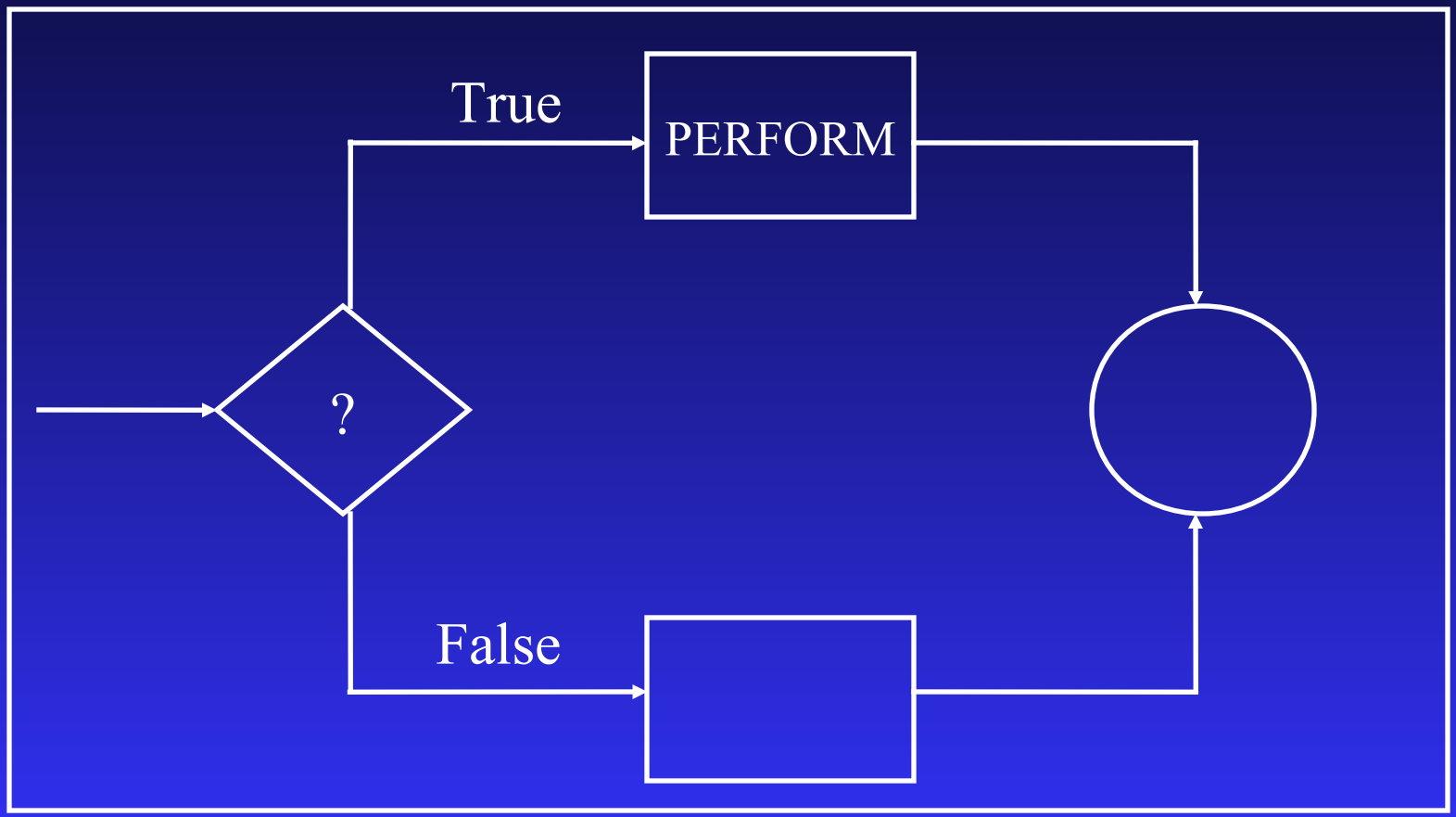
7.2.3 Sequence

- Program statements are executed in sequence



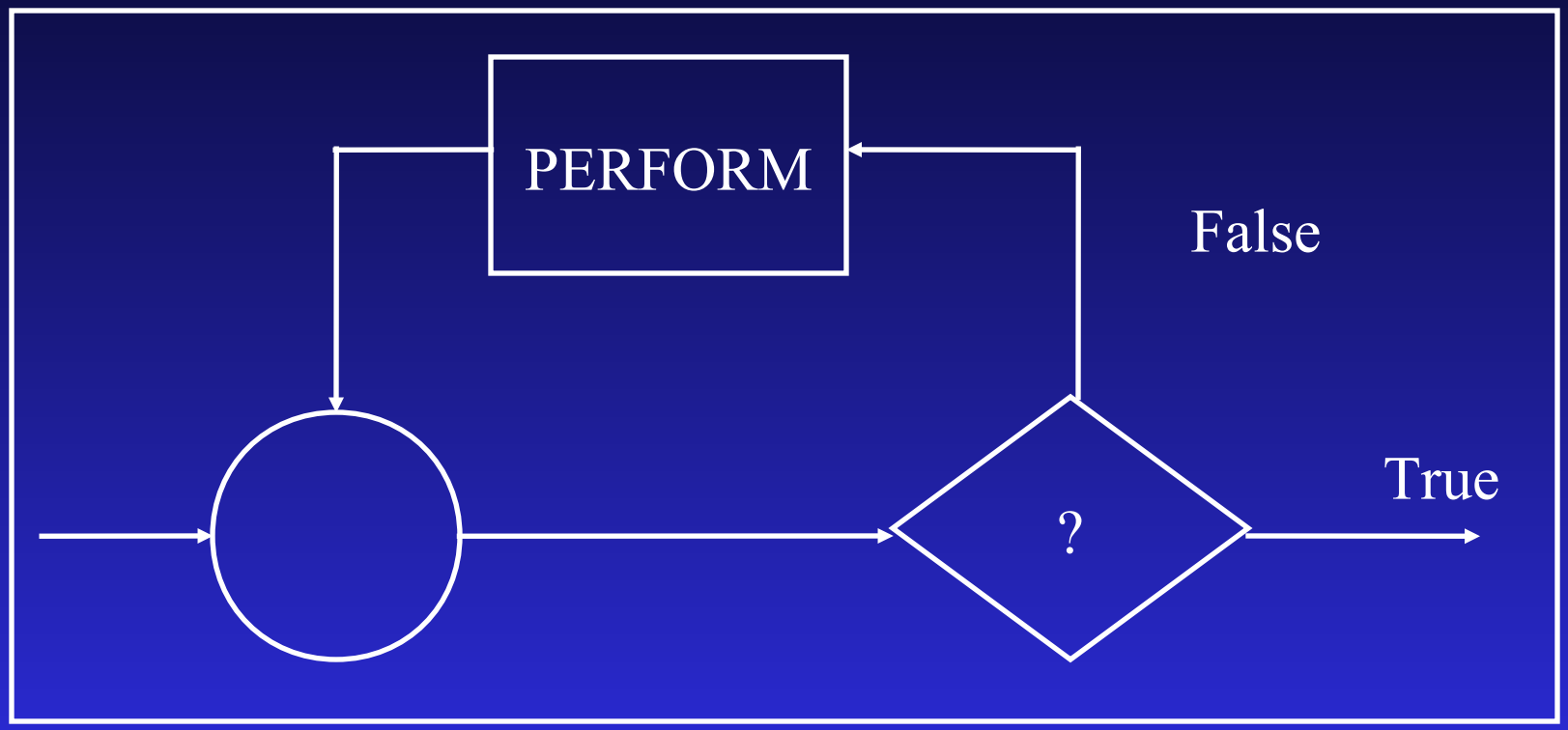
7.2.4 Selection

- Choice between two (and only two) actions, based on a condition



7.2.5 Iteration

- PERFORM UNTIL



7.2.6 Elements of Readability

- Indentation
- Descriptive data-names and paragraph-names
- Data-names with similar function (like switches and counters) are grouped together
- Align picture clause
- Limit one data-name per line in procedure division
- Place THE and ELSE on separate lines
- Avoid use of NOT
- Liberally sprinkle comments
- Blank lines for separation

7.2.7 Examples of Readability

- POOR

```
IF MALE AND EMPLOYEE ADD 1 TO MALE-EMPLOYEE-CTR, TOTAL-CTR
ELSE IF MALE AND CONTRACTOR ADD 1 TO MALE-CONTRACTOR-CTR, TOTAL-CTR
ELSE IF FEMALE AND EMPLOYEE
        ADD 1 TO FEMALE-EMPLOYEE-CTR, TOTAL-CTR
         ELSE IF FEMALE AND CONTRACTOR
                 ADD 1 TO FEMALE-CONTRACTOR-CTR, TOTAL-CTR
                 ELSE IF NOT CONTRACTOR AND NOT EMPLOYEE
                         ADD 1 TO OTHER-CTR, TOTAL-CTR.
```

7.2.7 Examples of Readability

- GOOD

```
IF MALE AND EMPLOYEE ADD 1 TO MALE-EMPLOYEE-CTR,
                                        TOTAL-CTR.

IF MALE AND CONTRACTOR ADD 1 TO MALE-CONTRACTOR-CTR,
                                        TOTAL-CTR.

IF FEMALE AND EMPLOYEE ADD 1 TO FEMALE-EMPLOYEE-CTR,
                                        TOTAL-CTR.

IF FEMALE AND CONTRACTOR ADD 1 TO FEMALE-CONTRACTOR-CTR,
                                        TOTAL-CTR.

IF NOT CONTRACTOR AND NOT EMPLOYEE ADD 1 TO OTHER-CTR,
                                        TOTAL-CTR.
```

7.2.8 Examples of Readability

- BETTER

```
IF MALE AND EMPLOYEE
   THEN
      ADD 1 TO MALE-EMPLOYEE-CTR TOTAL-CTR
   ELSE
       IF IF MALE AND CONTRACTOR
          THEN
             ADD 1 TO MAILE-CONBTRACTOR-CTR  TOTAL-CTR
          ELSE
              IF FEMALE AND EMPLOYEE
                 THEN
                     ADD 1 TO FEMALE-EMPLOYEE-CTR  TOTAL-CTR
                  ELSE
                      IF FEMALE AND CONTRACTOR
                         THEN
                        ADD 1 TO FEMALE-CONTRACTOR-CTR  TOTAL-CTR
                          ELSE
                       IF NOT CONTRACTOR AND NOT EMPLOYEE
                         THEN
                             ADD 1 TO OTHER-CTR  TOTAL-CTR.
```

7.2.9 Examples of Readability

↘ BEST

```
IF MALE AND EMPLOYEE
    THEN
        ADD 1 TO MALE-EMPLOYEE-CTR
                                TOTAL-CTR
        GO TO 400-COUNT-WORKERS-EXIT.

IF MALE AND CONTRACTOR
   THEN
       ADD 1 TO MAILE-CONBTRACTOR-CTR
                        TOTAL-CTR
       GO TO 400-COUNT-WORKERS-EXIT.

 IF FEMALE AND EMPLOYEE
    THEN
       ADD 1 TO FEMALE-EMPLOYEE-CTR
                            TOTAL-CTR
       GO TO 400-COUNT-WORKERS-EXIT.

IF FEMALE AND CONTRACTOR
   THEN
        ADD 1 TO FEMALE-CONTRACTOR-CTR
                                    TOTAL-CTR
        GO TO 400-COUNT-WORKERS-EXIT.

IF NOT CONTRACTOR AND NOT EMPLOYEE
   THEN
        ADD 1 TO OTHER-CTR  TOTAL-CTR.
```

7.3.1 COBOL modules

- Static
  - Compiled with NODYNAM
  - Linked with calling program in main storage
  - Preferred method
  - Executes faster than dynamic call
- Dynamic
  - Compiled with DYNAM
  - Loaded when needed (called)
  - When subprogram called infrequently
  - When subprogram called infrequently and very large
  - When subprogram changes often
  - Execution slower than static

7.3.2 CALL Statement

- CALL ‘subprogram-name {USING data-name(s)}
- Execute a subprogram
- Control given to subprogram
- Parameters can be passed to subprogram
- Upon completion, next statement in calling program run
- Can refer to program-id or an ENTRY Statement

7.3.2 CALL Statement

Calling Program
PROCEDURE DIVISION **CALL 'PROGRAM2'.** **COMPUTE A = B + C**

Called Program
IDENTIFICATION DIVISION **PROGRAM-ID. PROGRAM2.** **PROCEDURE DIVISION** **……..** **GOBACK.**

Calling Program
PROCEDURE DIVISION **CALL 'PROGRAM3'.** **COMPUTE Z =X + Y**

Called Program
PROCEDURE DIVISION **……..** **ENTRY 'PROGRAM3'** **……..** **GOBACK.**

7.3.3 CALL Statement with Parameters

Calling Program

```
WORKING-STORAGE SECTION.

01  PARM-LIST
      05 AMOUNT-PAID  PIC 9(5)V99.
      05  SALES-CODE     PIC 9(3).

PROCEDURE DIVISION.

CALL 'PROGRAM2'
          USING PARM-LIST
```

Called Program

```
LINKAGE SECTION.

01  PARMS-USED.
      05 DOLLARS          PIC 9(5).
      05 CENTS            PIC V99.
      05 CODE             PIC 9(3).

PROCEDURE DIVISION
                USING PARMES-USED.
     ……….
   GOBACK.
```

7.3.4 Linking a subprogram

```
//STEP01  EXEC COBUCL,PARM.COM='SXREF,DMAP'
//COB.SYSIN  DD  DSN=PROGRAM1,DISP=SHR
//LKED.SYSLMOD  DD  DSN=LOADLIB(PROGRAM1),DISP=SHR
//*
// STEP02  EXEC COBUCL,PARM.COM='SXREF,DMAP'
//COB.SYSIN  DD  DSN=PROGRAM2,DISP=SHR
//LKED.SYSLMOD  DD  DSN=LOADLIB(PROGRAM2),DISP=SHR
//LKED.SUBPGM  DD  DSN=LOADLIB,DISP=SHR
//LKED.SYSIN  DD *
  INCLUDE SUBPGM(PROGRAM1)
```

7.2.13 VS COBOL II - Specifics

- Improves readability
  - SET TO TRUE
  - Scope terminators
  - PERFORM UNTIL with test after
  - PERFORM UNTIL with test before
  - INLINE PERFORM
  - EVALUATE
  - CASE Structure

7.2.14 SET TO TRUE

- Used with condition names (88-levels)

```
05  END-OF-FILE-SWITCH     PIC  X    VALUE 'N'.
     88   END-OF-FILE                 VALUE 'Y'.

      READ INPUT-FILE
         AT END
          SET END-OF-FILE TRUE.

05  SALES-CODE                    PIC  X(3).
     88  JONES                    VALUE '001'.
     88  SMITH                    VALUE '002'.
     88  BOND                     VALUE '003'.

        MOVE '002'  TO SALES-CODE

        SET SMITH TO TRUE
```

7.2.15 Scope Terminators

- Facilitate Structure
- Conform more closely to flowchart
- Explicit Terminator
  - Alternative to Implicit terminator
    - A period at the end of a sentence
    - An ELSE in a conditional statement
  - Delimit scope of certain statements

7.2.15 Scope Terminators

END-ADD	**END-READ**
END-CALL	**END-RETURN**
END-COMPUTE	**END-REWRITE**
END-DELETE	**END-SEARCH**
END-DIVIDE	**END-START**
END-EVALUATE	**END-STRING**
END-IF	**END-SUBTRACT**
END-MULTIPLY	**END-UNSTRING**
END-PERFORM	**END-WRITE**

7.2.16 Scope Terminators

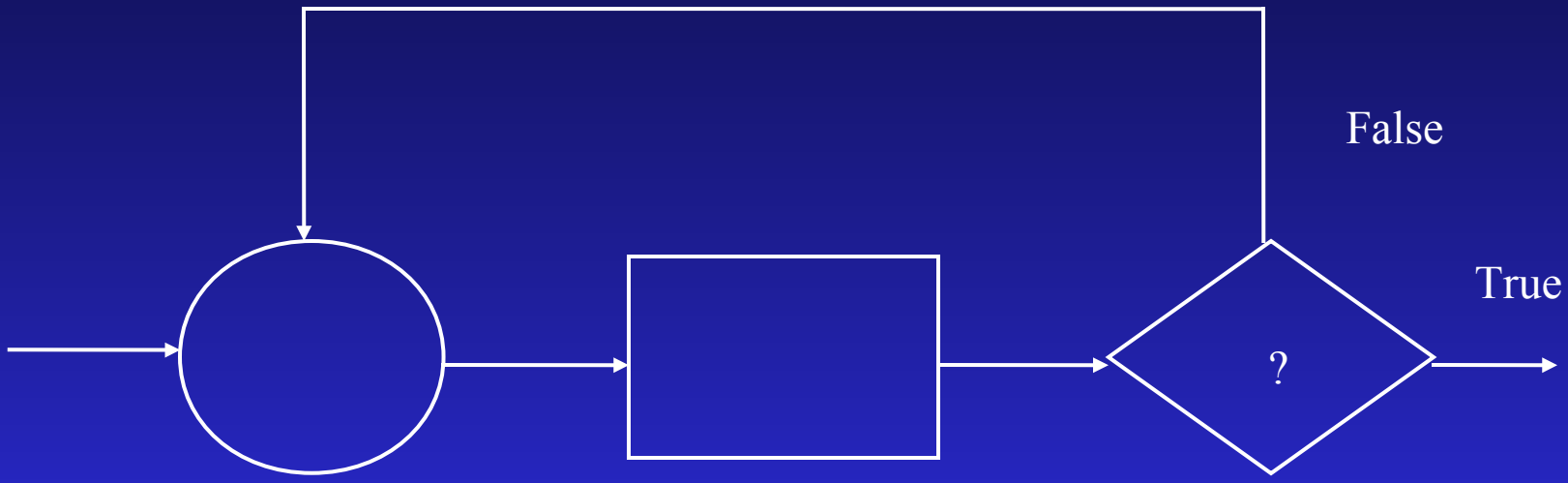
```
IF MALE AND EMPLOYEE
        ADD 1 TO MALE-EMPLOYEE-CTR
                                TOTAL-CTR
        END-ADD
END-IF.
IF MALE AND CONTRACTOR
        ADD 1 TO MAILE-CONBTRACTOR-CTR
                        TOTAL-CTR
        END-ADD
 END-IF.
 IF FEMALE AND EMPLOYEE
        ADD 1 TO FEMALE-EMPLOYEE-CTR
                            TOTAL-CTR
        END-ADD
 END-IF.
IF FEMALE AND CONTRACTOR
        ADD 1 TO FEMALE-CONTRACTOR-CTR
                                    TOTAL-CTR
        END-ADD
END-IF.
IF NOT CONTRACTOR AND NOT EMPLOYEE
        ADD 1 TO OTHER-CTR  TOTAL-CTR.
        END-ADD
END-IF.
```

7.2.17 Value of Scope Terminators

```
IF MALE AND EMPLOYEE
        ADD 1 TO MALE-EMPLOYEE-CTR
                                TOTAL-CTR
        END-ADD
END-IF.
IF MALE AND CONTRACTOR
        ADD 1 TO MAILE-CONBTRACTOR-CTR
                        TOTAL-CTR
        END-ADD
 END-IF.
 IF FEMALE AND EMPLOYEE
        ADD 1 TO FEMALE-EMPLOYEE-CTR
                            TOTAL-CTR
        END-ADD
 END-IF.
IF FEMALE AND CONTRACTOR
        ADD 1 TO FEMALE-CONTRACTOR-CTR
                                    TOTAL-CTR
        END-ADD
END-IF.
IF NOT CONTRACTOR AND NOT EMPLOYEE
        ADD 1 TO OTHER-CTR  TOTAL-CTR.
        END-ADD
END-IF.
```

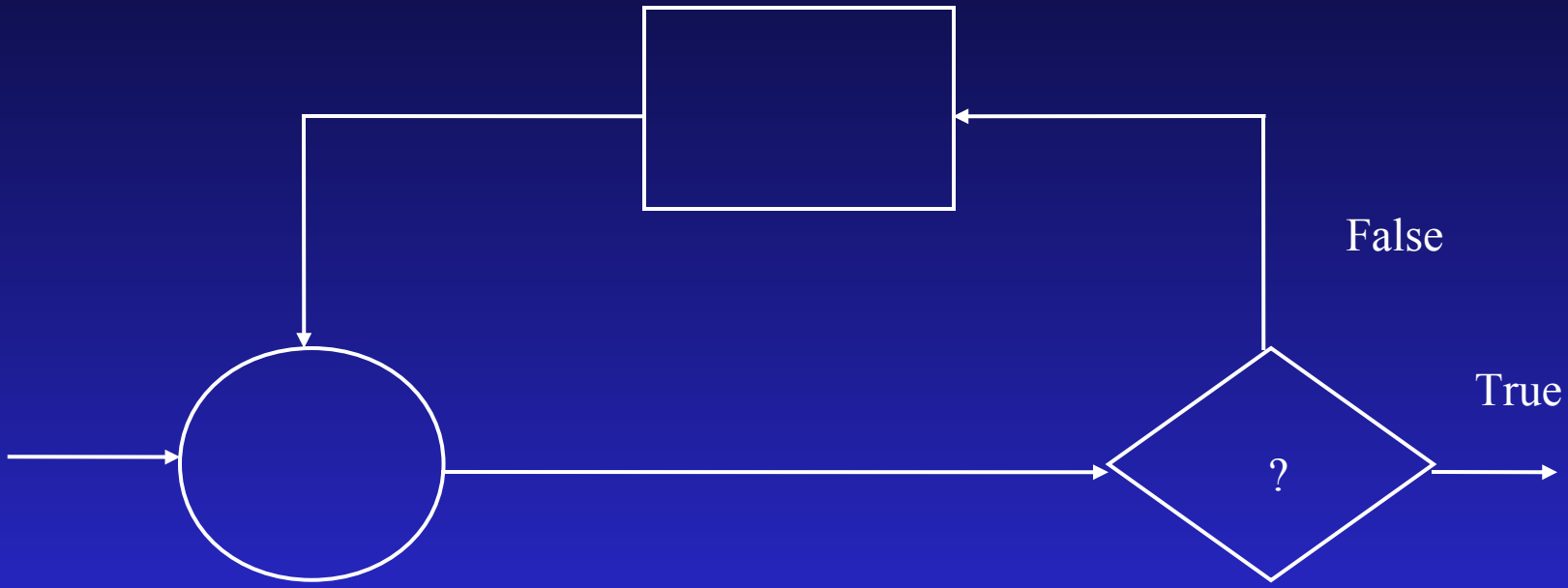
7.2.18 PERFORM UNTIL with test after

PERFORM ABC-RT WITH TEST AFTER
UNTIL END-OF-DATA



7.2.19 PERFORM UNTIL with test before

PERFORM ABC-RT WITH TEST BEFORE
UNTIL END-OF-DATA
or
PERFORM ABC-TRN UNTIL END-OF-DATA



7.2.20 PERFORM UNTIL with test before

- No need for separate paragraph

```
PERFORM UNTIL END-OF-FILE
        IF COUNTER-1 IS GREATER THAN 60
            PERFORM PRINT-A-PAGE
        END-IF
        ADD 1 TO COUNTER-1
        ADD 1 TO SUB-3
        MOVE IP-RECORD TO TBL-3(SUB-3)
        READ INPUT-FILE
                AT END SET END-OF-FILE TO TRUE
END-PERFORM.
```

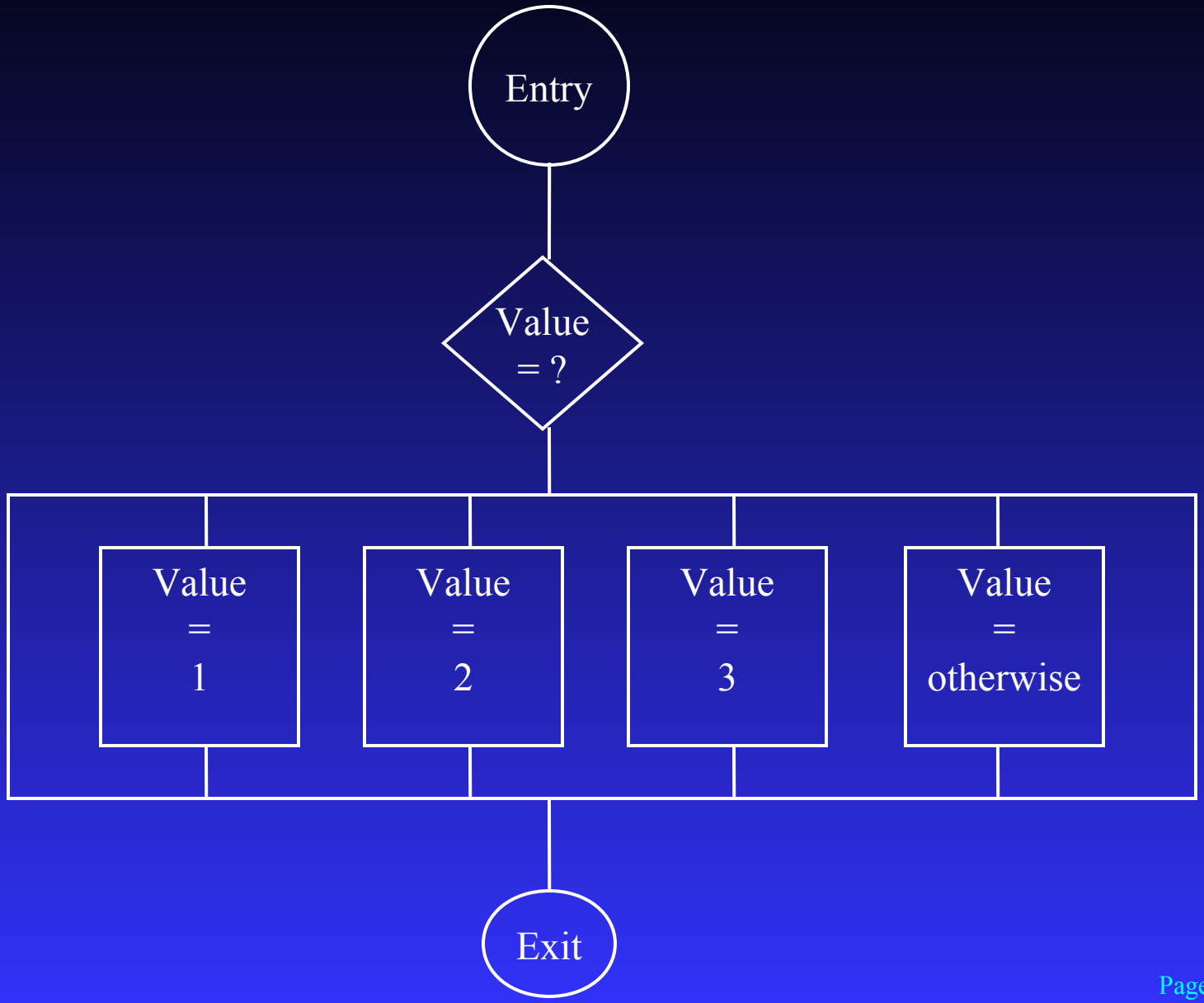
7.2.21 EVALUATE

- CASE STRUCTURE
- Used with 88-levels

```
EVALUATE SALES CODE
        WHEN JONES   PERFORM JONES-RTN
        WHEN SMITH  PERFORM SMITH-RTN
      WHEN BOND    PERFORM BOND-RTN
        WHEN '005'      PERFORM FIVE-RTN
        WHEN OTHER PERFORM OTHER-RTN
END-EVALUATE.

EVALUATE    TRUE      ALSO     TRUE
       WHEN   MALE     ALSO     EMPLOYEE
              ADD 1 TO MALE-EMPLOYEE-CTR
       WHEN   MALE     ALSO     CONTRACTOR
              ADD 1 TO MALE-CONTRACTOR-CTR
       WHEN   FEMALE  ALSO     EMPLOYEE
              ADD 1 TO FEMALE-EMPLOYEE-CTR
       WHEN   FEMALE  ALSO     CONTRACTOR
              ADD 1 TO FEMALE-CONTRACTOR-CTR
       WHEN  OTHER
              ADD   1   TO   OTHER-CTR
END-EVALUATE.
```

7.2.22 CASE Structure



7.3 Workshop

Do exercises on pages 7-16 and 7-17. Do not do the exercise on 7-18.

Create a subprogram structure to print a salesperson phone number on each record in program2.cbl.

- Changes required for program2.
  - add a PARM-LIST to Working-Storage with a data-item for the Salescode and Phone
  - add a CALL….USING statement in your Procedure Division calling PROGRAM3.
  - add a 10 byte numeric field in your Working-Storage SALES-REPORT record for the phone number
  - be sure to move the phone number from your PARM-LIST to your output record
  - if it doesn't already, have your Display statement Display your REPORT RECORD, not your DETAIL RECORD.
- Coding required for program3.
  - you may clone an existing program like program2.
  - this program accesses the personnl file only, so replace all SELECT statements with:

```
SELECT PERSONNL-FILE-IN ASSIGN TO UT-S-SYSUT2
        ORGANIZATION IS LINE SEQUENTIAL.
```

  - replace FD's with:

```
FD  PERSONNL-FILE-IN
   LABEL RECORDS ARE STANDARD
   RECORDING MODE IS F
   RECORD CONTAINS 43 CHARACTERS
   BLOCK CONTAINS 0 RECORDS
   DATA RECORD IS PERSONNL-RECORD.

01  PERSONNL-RECORD PIC X(43).
```

  - your record description will appear as follows:

```
01  PERSONNEL-RECORD.
   05  SALESCODE        PIC 9(3) VALUE ZERO.
   05  LNAME      PIC X(20).
   05  FNAME      PIC X(10).
   05  PHONE.
      10  ARCODE  PIC 9(3).
      10  PREFIX  PIC 9(3).

      10  XCHANGE PIC 9(4).
```

  - code your LINKAGE SECTION and PROCEDURE DIVISION USING statements
  - code a very simple PROCEDURE DIVISION to read through the PERSONNL file until it finds a match between the LK-SALESCODE and the SALESCODE in the PERSONNL file. You will then have a phone number to pass back through linkage.
  - be sure to code a GOBACK when the record has been found.

The programs LISTFIX.CBL and SORTFIX.CBL have a combination of good and poor elements of structured programming. You should fix:

- Program-Id's
- Align PIC clauses
- Properly space and indent the Procedure Divisions
- Informational after checking LISTFIX.CBL, note how it pulled in a Copybook for a needed record description.

7.3 Workshop

1. Sequence, Selection and Iteration
2. Iteration
3. Selection
4. Entry and Exit
5. Elements of Readability page 7-8
6.
   - development discipline
   - readability
   - understandable logic flow
   - easy maintenance
   - better documentation
   - increased productivity
   - easier testing
7.

```
 IF NAME IS EQUAL TO 'DAVID'
    ADD 1 TO D-CTR
    MOVE IN-REC TO WORK-REC
 ELSE
   ADD 1 TO A-CTR
   CLOSE OUTFILE.

 IF NAME IS EQUAL TO DAVID
          AND (D-CTR = 60 OR A-CTR = 1)
    MOVE SPACES TO WORK-REC
                        OUTREC
    ADD 1 TO A-CTR.
```

```
      OPEN INPUT INFILE
    LOGFILE
              MASTER
OUTPUT  NEWMAST
    RPTFILE.
```

8. CALL……..USING

- Describe the steps of the Programming Life Cycle
- Describe the function of the four COBOL divisions
- List the advantages and disadvantages of COBOL
- Describe the purpose of the COBOL compiler
- Understand the column structure of COBOL
- Use the Micro Focus Workbench to Edit, Syntax Check and Animate a program
- Code an identification division
- Code an environment division
- Code a data division
- Tell whether statements belong in the A-margin or B-margin
- Write a record description for a file
- Process literals and figurative constants
- Describe the mainframe COBOL compiler
- Code file 1/0 statements (OPEN, CLOSE, READ, WRITE)
- Code special 1/0 statements (ACCEPT, DISPLAY)
- Perform basic data transfer (MOVE)
- Detect when an end-of-file condition is reached
- Create a simple COBOL program using TSO/ISPF, Micro Focus
- End the program as needed (GOBACK, STOP RUN)
- Compile, link, and test a simple COBOL program
- Understand the function of an optimizer
- Test data to determine proper action
- Perform unconditional branches
- Execute sequence, selection and iteration
- Perform valid comparisons of data
- Validate data for numeric contents
- Test logical conditions using AND, OR, or NOT
- Use conditional names to clarify and reduce coding
- Use switches in a program
- Describe testing and debugging tools
- Describe testing strategies
- Recognize common abend codes

Review……

At this point we should be able to:

- Use counters in a program
- Perform calculations
- Round arithmetic results
- Perform an on size error
- Use the RETURN-CODE Register
- ***Understand and apply elements of structured programming***
- ***Describe sequence, selection and iteration***
- ***Apply concepts of readability and modularity***
- ***CALL and LINK a subprogram***

8-1 Objectives

After completing this chapter, you use COBOL to produce reports. Specifically, you will be able to:

- REDEFINE data areas in the COBOL Program
- INSPECT and alter data
- Use switches to trigger control breaks
- Use line counters and page counters in a program
- Update a master file using input data
- Create an error report

8.2 Topics to be Covered

- Report definition, components, and preparation
- Switches and counters
- Data editing
- INSPECT
- REDEFINES
- Paper positioning
- Report Writer

8.2.1 Report Definition

- Pictorial representation of data
- Columns
- Rows (or detail lines)
- Categories (or groups or levels)
- Control breaks

8.2.2 Report Components

- Report Heading
- Page Heading with page number
- Control headings (detail line headings)
- Detail Lines
- Control footings
- Page footings
- Report footings

8.2.3 COBOL report preparation

- Switches
- Counters
- Data editing
- INSPECT
- REDEFINES
- Paper positioning

8.2.4 Switches

- True/False
- 0 or 1
- ‘T’ or ‘F’
- 88-LEVELS

```
01  program-counters
    05  SW-END-OF-DATA                 PIC X      VALUE  ‘N’.
        88  END-OF-FILE                           VALUE  ‘Y’.
    05  SW-ERRORS-IN-PROCESSING        PIC 9      VALUE 0.
        88  ERRORS-IN-PROCESSING                  VALUE 1.
    05  SW-CONTROL-BREAK               PIC X      VALUE ‘N’.
        88  CONTROL-BREAK                         VALUE ‘Y’.
```

8.2.5 Switches

- Page counter
  - Usually PIC 9(4)
- Line counter
  - Usually PIC 9(2)
  - 88 level of 45 or 60 (lines per page)

8.2.6 Data Editing

- Additional PICTURE clauses
  - B , . $ - + CR DB Z *
- Occurs with MOVE
- Two kinds of data editing
  - Insertion
  - B , . $ - + CR DB Z *
  - Suppression and replacement
  - Z * $ + -

8.2.7 Data Editing Characters

- B
  - Space character will appear
- ,
  - Comma will appear
- .
  - Period will appear

8.2.7 Data Editing Characters

```
ITEM-1     PIC   9(7)          VALUE   6330359.
ITEM-2     PIC   999B9999
ITEM-3     PIC   999,9999
ITEM-4     PIC   999.9999

MOVE  ITEM-1   TO   ITEM-2, ITEM-3, ITEM-4
```

633 0359 ITEM-2

633,0359 ITEM-3

633.0359 ITEM-4

8.2.8 Data Editing Characters

- $
  - Dollar sign will appear
- -
  - When a value is positive, space will appear
  - When a value is negative, negative sign will appear
- +
  - When a value is positive, a positive sign will appear
  - When a value is negative, negative sign will appear

8.2.8 Data Editing Characters

```
ITEM-1    PIC   9(7)          VALUE   +4230660.
ITEM-2    PIC   $9(7)
ITEM-3    PIC   9(7)+
ITEM-4    PIC   -9(7)

MOVE  ITEM-1   TO   ITEM-2, ITEM-3, ITEM-4
```

$4230660	ITEM-2
4230660+	ITEM-3
4230660	ITEM-4

8.2.9 Data Editing Characters

- Credit(s) CR
  - When value is positive, two spaces appear
  - When value is negative, CR appears
- Debit(s) DB
  - When a value is positive, two spaces appear
  - When a value is negative, DB appears

8.2.9 Data Editing Characters

```
ITEM-1     PIC   9(3)          VALUE   -123.
ITEM-2     PIC   $9(5)CR
ITEM-3     PIC   $9(5)DB
ITEM-4     PIC   +$9(3).99

MOVE  ITEM-1   TO   ITEM-2, ITEM-3, ITEM-4
```

$00123CR	ITEM-2
$00123db	ITEM-3
-$123.00	ITEM-4

8.2.10 Data Editing Characters

- Suppressing - Z
  - Digit-suppressing character
- Protecting - *
  - Protects leading digits when printing checks

8.2.10 Data Editing Characters

```
ITEM-1     PIC   9(2)V99          VALUE   ZEROS.
ITEM-2     PIC   ZZZZ.ZZ
ITEM-3     PIC   ZZZZ.99
ITEM-4     PIC   $ZZ,ZZ9+
ITEM-5     PIC    S**,*99

MOVE  ITEM-1   TO   ITEM-2, ITEM-3, ITEM-4, ITEM-5
```

bbbbbbb	ITEM-2
bbbb.00	ITEM-3
$bbbbo+	ITEM-4
$***00	ITEM-5

8.2.11 Data Editing Characters

- $
  - Floating dollar sign
- +
  - Floating plus sign
- -
  - Floating minus sign

8.2.11 Data Editing Characters

```
ITEM-1     PIC   9(3)          VALUE   -123.
ITEM-2     PIC   $$$$.99+
ITEM-3     PIC   $$$$$DB
ITEM-4     PIC   +++++.99

MOVE  ITEM-1   TO   ITEM-2, ITEM-3, ITEM-4
```

$123.00-	ITEM-2
b$123DB	ITEM-3
b-123.00	ITEM-4

8.2.12 INSPECT

```
INSPECT identifier-1      REPLACING      [ALL]
                                         [LEADING]
                                         [FIRST]
                                                    [UNTIL FIRST]
      LITERAL-1      BY    LITERAL-2
```

- Field examined left to right 1 character at a time
- Characters are replaced

8.2.12 INSPECT

INSPECT DATE-FIELD REPLACING LEADING SPACE BY 0

BEFORE	AFTER
b10190	010190

INSPECT REPORT-DATE REPLACING ALL '-' BY '/'

BEFORE	AFTER
12-25-90	12/25/90

8.2.13 REDEFINES

- Gives another name and description to data
- In File Section, use DATA RECORDS ARE and an additional 01
- Immediately follow data-name with same level number and REDEFINES clause
- Same storage area used

8.2.13 REDEFINES

```
01   DOLLARS                PIC  9(4)V99.
01    DRULAS                PIC   9(5)V9.
                    REDEFINES DOLLARS.

05  PHONE-NUMBER       PIC X(12).
05  PHONE-DETAIL   REDEFINES  PHONE-NUMBER.
      10  OPEN-PAREN       PIC X.
      10  AREA-CODE        PIC 9(3).
      10  CLOSE-PAREN      PIC X.
      10  EXCHANGE         PIC 9(3).
      10  NUMBERS          PIC 9(4).
```

8.2.14 Paper positioning

- Single space
- Double space
- Triple space
- Overstriking
  - Works on impact printers - laser printers unpredictable
- Top of form

8.2.15 Spacing Paper

- WRITE record-name AFTER ADVANCING

```
WRITE   OUTPUT-RECORD  AFTER ADVANCING   1 LINE.
WRITE   OUTPUT-RECORD  AFTER ADVANCING   2 LINES.
WRITE   OUTPUT-RECORD  AFTER ADVANCING   3 LINES.
WRITE   OUTPUT-RECORD  AFTER ADVANCING   0 LINE.
```

8.2.16 Positioning Paper to Top of Form

- Special Names
- CO1

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
OBJECT-COMPUTER.     IBM-370.
OBJECT-COMPUTER.     IBM-370.
SPECIAL-NAMES.
  CO1  IS  TOP-OF-PAGE.
INPUT-OUT SECTION.
FILE-CONTROL.
  SELECT SALES-FILE-IN    ASSIGN TO UT-S-SALESIN
PROCEDURE DIVISION.
WRITE OUTPUT-RECORD
```

8.3 Workshop

Do exercises on pages 8-24 and 8-25. Do not do the exercise on 8-26.

Formalize the reporting in program2.cbl so it appears exactly as shown below!!!.

- Formal underlined headers
- Formatted data phone numbers and amounts - 9 records per page.
- Footer line with page number
- End details line
- Formatted display of total sales and the error report

```
LAST NAME       FIRST NAME       PHONE        CODE  AMOUNT
---------       ----------       -----        ----  ------
O'SHEA            RICK       (312)569-3287  005   $235.91
CASHMAN           BARBARA    (800)844-1111  006   $534.22
KNOLL             DONALD     (212)555-1212  007   $935.20
SKIRTZ            MINNIE     (800)844-1111  006   $444.75
OATES             OPHELIA    (212)555-1212  004   $735.94
GONNAWAY          HERMAN     (212)555-1212  004   $439.97
MONEY             XAVIER     (212)555-1212  004   $336.98
EGGO              SANDY      (312)569-3287  005   $235.91
TERRAINE          ROCKY      (800)844-1111  006   $534.22
------------------------------------------------------ PAGE  6
LAST NAME       FIRST NAME       PHONE        CODE  AMOUNT
---------       ----------       -----        ----  ------
CHAVEZ            RAFAEL     (212)555-1212  007   $935.20
QUINTANA          VINCENT    (800)844-1111  006   $444.75
LIGHT             TRUDY      (212)555-1212  004   $735.94
THOMPSON          CHARLES    (212)555-1212  004   $439.97
-------------------------------------------------------------

TOTAL SALES    - $30587.44
REPORT ERRORS -  1
------------------------------------------------------ PAGE  7
```

8.3 Workshop

1. Report Heading, Page Heading, Control Headings, Detail Lines, Control Footings, Page Footings, Report Footings.

2. 88 levels

3. 710-NEW-PAGE THRU 710-NEW-PAGE-EXIT
   ELSE
   NEXT SENTENCE

4. Report Writer

5. Working-Storage

6. INSPECT LOCATION REPLACING ALL “DISTRICT OF COLUMBIA” BY “D.C.”

item2 | 2 | 7 | 7 | 7 | 1 | 0 | 8 | + |
item3 | | $ | 2 | , | 7 | 7 | 7 | , | 1 | 0 | 8 |
item4 | * | 2 |7 | , | 7 | 7 | 1 | . | 0 | 8 |

8.3 Workshop

Which is more readable…….

This……..

```
PROCEDURE DIVISION.
MAIN-ROUTINE.
    OPEN INPUT SALES-FILE-IN
         OUTPUT SALES-FILE-OUT.
    READ SALES-FILE-IN INTO DATA-RECORD.
    WRITE REPORT-RECORD FROM HEADING-1.
    DISPLAY REPORT-RECORD.
    WRITE REPORT-RECORD FROM HEADING-2.
    DISPLAY REPORT-RECORD.
    MOVE 2 TO LINE-COUNTR.
    MOVE 1 TO PAGE-COUNTR.
    PERFORM PROCESS-RECORD THRU PROCESS-RECORD-EXIT
    UNTIL END-OF-FILE-SWITCH = 'Y'.
    WRITE REPORT-RECORD FROM FOOTER-2.
    DISPLAY REPORT-RECORD.
    MOVE TOTAL-AMOUNT-ACCUMULATOR TO DL2-AMOUNT.
    WRITE REPORT-RECORD FROM DETAIL-LINE2.
    DISPLAY REPORT-RECORD.
    MOVE ERROR-COUNTER TO DL3-ERROR.
    WRITE REPORT-RECORD FROM DETAIL-LINE3.
    DISPLAY REPORT-RECORD.
    MOVE PAGE-COUNTR TO FT-PAGE-NUMBER
    WRITE REPORT-RECORD FROM FOOTER-1.
    DISPLAY REPORT-RECORD.
    CLOSE SALES-FILE-IN
          SALES-FILE-OUT.
    GOBACK.
```

```
PROCESS-RECORD.
    IF NOT ZERO-AMOUNT
       IF DR-SALESCODE NUMERIC
          ADD DR-AMOUNT TO TOTAL-AMOUNT-ACCUMULATOR
          MOVE DR-SALESCODE TO LK-SALESCODE
          CALL 'PROGRAM3'
            USING PARM-LIST
          MOVE DR-LASTNAME TO DL-LASTNAME
          MOVE DR-FIRSTNAME TO DL-FIRSTNAME
          MOVE LK-PHONE TO SR-PHONE
          MOVE SR-ARCODE TO DL-ARCODE
          MOVE SR-PREFIX TO DL-PREFIX
          MOVE SR-XCHANGE TO DL-XCHANGE
          MOVE DR-SALESCODE TO DL-SALESCODE
          MOVE DR-AMOUNT TO DL-AMOUNT
          PERFORM HEADER-ROUTINE
          WRITE REPORT-RECORD FROM DETAIL-LINE
          ADD 1 TO LINE-COUNTR
          DISPLAY REPORT-RECORD
       ELSE
          ADD 1 TO ERROR-COUNTER
          MOVE 8 TO RETURN-CODE.
    READ SALES-FILE-IN INTO DATA-RECORD
     AT END
      MOVE 'Y' TO END-OF-FILE-SWITCH.
PROCESS-RECORD-EXIT.

HEADER-ROUTINE.
    IF LINE-COUNTR IS GREATER THAN 10
      MOVE PAGE-COUNTR TO FT-PAGE-NUMBER
      WRITE REPORT-RECORD FROM FOOTER-1
       DISPLAY REPORT-RECORD
      WRITE REPORT-RECORD FROM HEADING-1
      DISPLAY REPORT-RECORD
      WRITE REPORT-RECORD FROM HEADING-2
      DISPLAY REPORT-RECORD
      MOVE 2 TO LINE-COUNTR
      ADD 1 TO PAGE-COUNTR.
HEADER-ROUTINE-EXIT.
```

8.3 Workshop

or this?????

```
PROCEDURE DIVISION.
000-MAIN-ROUTINE.
    OPEN INPUT SALES-FILE-IN
         OUTPUT SALES-FILE-OUT.
    PERFORM 900-HOUSEKEEPING-ROUTINE.
    PERFORM 100-PROCESS-RECORD-ROUTINE
        UNTIL END-OF-FILE-SWITCH = 'Y'.
    PERFORM 950-FINISH-REPORT-ROUTINE.
    CLOSE SALES-FILE-IN
          SALES-FILE-OUT.
    GOBACK.

100-PROCESS-RECORD-ROUTINE.
    IF NOT ZERO-AMOUNT
      IF DR-SALESCODE NUMERIC
        ADD DR-AMOUNT TO TOTAL-AMOUNT-ACCUMULATOR
        MOVE DR-SALESCODE TO LK-SALESCODE
        CALL 'PROGRAM3'
          USING PARM-LIST
        PERFORM 200-MOVE-FIELD-ROUTINE
        PERFORM 400-HEADER-ROUTINE
        WRITE REPORT-RECORD FROM DETAIL-LINE
        ADD 1 TO LINE-COUNTR
        DISPLAY REPORT-RECORD
      ELSE
        ADD 1 TO ERROR-COUNTER
        MOVE 8 TO RETURN-CODE.
    PERFORM 999-READ-ROUTINE.

200-MOVE-FIELD-ROUTINE.
    MOVE DR-LASTNAME TO DL-LASTNAME.
    MOVE DR-FIRSTNAME TO DL-FIRSTNAME.
    MOVE LK-PHONE TO SR-PHONE.
    MOVE SR-ARCODE TO DL-ARCODE.
    MOVE SR-PREFIX TO DL-PREFIX.
    MOVE SR-XCHANGE TO DL-XCHANGE.
    MOVE DR-SALESCODE TO DL-SALESCODE.
    MOVE DR-AMOUNT TO DL-AMOUNT.

400-HEADER-ROUTINE.
    IF LINE-COUNTR IS GREATER THAN 10
      MOVE PAGE-COUNTR TO FT-PAGE-NUMBER
      WRITE REPORT-RECORD FROM FOOTER-1
      DISPLAY REPORT-RECORD
      WRITE REPORT-RECORD FROM HEADING-1
      DISPLAY REPORT-RECORD
      WRITE REPORT-RECORD FROM HEADING-2
      DISPLAY REPORT-RECORD
      MOVE 2 TO LINE-COUNTR
      ADD 1 TO PAGE-COUNTR.

900-HOUSEKEEPING-ROUTINE.
    READ SALES-FILE-IN INTO DATA-RECORD.
    WRITE REPORT-RECORD FROM HEADING-1.
    DISPLAY REPORT-RECORD.
    WRITE REPORT-RECORD FROM HEADING-2.
    DISPLAY REPORT-RECORD.
    MOVE 2 TO LINE-COUNTR.
    MOVE 1 TO PAGE-COUNTR.

950-FINISH-REPORT-ROUTINE.
    WRITE REPORT-RECORD FROM FOOTER-2.
    DISPLAY REPORT-RECORD.
    MOVE TOTAL-AMOUNT-ACCUMULATOR TO DL2-AMOUNT.
    WRITE REPORT-RECORD FROM DETAIL-LINE2.
    DISPLAY REPORT-RECORD.
    MOVE ERROR-COUNTER TO DL3-ERROR.
    WRITE REPORT-RECORD FROM DETAIL-LINE3.
    DISPLAY REPORT-RECORD.
    MOVE PAGE-COUNTR TO FT-PAGE-NUMBER
    WRITE REPORT-RECORD FROM FOOTER-1.
    DISPLAY REPORT-RECORD.

999-READ-ROUTINE.
    READ SALES-FILE-IN INTO DATA-RECORD
      AT END
        MOVE 'Y' TO END-OF-FILE-SWITCH.
```

- Describe the steps of the Programming Life Cycle
- Describe the function of the four COBOL divisions
- List the advantages and disadvantages of COBOL
- Describe the purpose of the COBOL compiler
- Understand the column structure of COBOL
- Use the Micro Focus Workbench to Edit, Syntax Check and Animate a program
- Code an identification division
- Code an environment division
- Code a data division
- Tell whether statements belong in the A-margin or B-margin
- Write a record description for a file
- Process literals and figurative constants
- Describe the mainframe COBOL compiler
- Code file 1/0 statements (OPEN, CLOSE, READ, WRITE)
- Code special 1/0 statements (ACCEPT, DISPLAY)
- Perform basic data transfer (MOVE)
- Detect when an end-of-file condition is reached
- Create a simple COBOL program using TSO/ISPF, Micro Focus
- End the program as needed (GOBACK, STOP RUN)
- Compile, link, and test a simple COBOL program
- Understand the function of an optimizer
- Test data to determine proper action
- Perform unconditional branches
- Execute sequence, selection and iteration
- Perform valid comparisons of data
- Validate data for numeric contents
- Test logical conditions using AND, OR, or NOT
- Use conditional names to clarify and reduce coding
- Use switches in a program
- Describe testing and debugging tools
- Describe testing strategies
- Recognize common abend codes

Review……

At this point we should be able to:

- Use counters in a program
- Perform calculations
- Round arithmetic results
- Perform an on size error
- Use the RETURN-CODE Register
- Understand and apply elements of structured programming
- Describe sequence, selection and iteration
- Apply concepts of readability and modularity
- CALL and LINK a subprogram
- ***COBOL Report writing***
  - ***REDEFINE data storage***
  - ***INSPECT and alter***
  - ***Trigger control breaks***
  - ***Use line and page counters***
  - ***Display summary information***

9-1 Objectives

After completing this chapter, you will be able to build, search and process data within tables. Specifically, you will be able to:

- Describe basic table terminology
- Use “hard-coded” and “externally populated” tables within a program
- Use SEARCH and PERFORM VARYING statements to process tables
- Use subscripts and indexes within a table

9.2 Topics to be Covered

- Table handling / Terminology
- Table coding with REDEFINES
- Building a table (Data and Procedure Divisions
- Table Lookup
- PERFORM VARYING
- Indexes
- SET statement
- SEARCH [ALL] statement

9.2.1 Table Handling

```
01  SALES-TAX-TABLE.
     05  VA-TAX                  PIC  V999   VALUE .045.
     05  MD-TAX                  PIC  V999   VALUE .050.
     05  CT-TAX                  PIC  V999   VALUE .080.
     05  CA-TAX                  PIC  V999   VALUE .055.

ADD-SALES-TAX.
      MOVE ZERO TO SALES-TAX
      IF STATE = 'CT'
       MULTIPLY  COST BY CT-TAX  GIVING SALES-TAX.
      IF STATE = 'MD'
       MULTIPLY  COST BY MD-TAX  GIVING SALES-TAX.
      IF STATE = 'VA'
       MULTIPLY  COST BY VA-TAX  GIVING SALES-TAX.
      IF STATE = 'CA'
       MULTIPLY  COST BY CA-TAX  GIVING SALES-TAX.
```

9.2.2 Terminology

- Table
  - List of fields stored side by side with same format
- Elements
  - Fields within a table
- Subscript
  - Positive non-zero integer pointing to element(s) in table

9.2.2 Terminology

- Index (COBOL-generated data-item)
  - Positive non-zero integer pointing to element(s) in table
- Table Search
  - Method of finding element(s) within a table
- OCCURS
  - Identifies the number of elements
  - Cannot be used as 01 level
  - VALUE clause cannot be coded with OCCURS clause (OSVS)

9.2.3 Table coding with REDEFINES

```
01  SALES-TAX-TABLE.
     05  SALES-TAX-LIST.
         10  FILLER                  PIC  X(5)    VALUE .VA045.
         10  FILLER                PIC  X(5)    VALUE .MD050.
         10  FILLER                  PIC  X(5)    VALUE .CT080.
         10  FILLER                  PIC  X(5)    VALUE .CA055.
01  TAX-TABLE  REDEFINES SALES-TAX-TABL.
     05  TAXES          OCCURS  5 TIMES.
         10  STATE                   PIC X(2).
         10  TAX                     PIC V999.

ADD-SALES-TAX.
    MOVE ZERO TO SALES-TAX
    IF STATE = 'CT'
      MULTIPLY  COST BY TAX(3)   GIVING SALES-TAX.
    IF STATE = 'MD'
      MULTIPLY  COST BY TAX(2)  GIVING SALES-TAX.
    IF STATE = 'VA'
      MULTIPLY  COST BY TAX(1)  GIVING SALES-TAX.
    IF STATE = 'CA'
      MULTIPLY  COST BY TAX(4)  GIVING SALES-TAX.
```

9.2.4 Building a Table (Data Division)

```
FD  SALES-FILE
01   SALES-TAX-RECORD

WORKING-STORAGE SECTION
01  TAX-TABLE-SW                  PIC X        VALUE 'N'.
     88  TAX-TABLE-IS-FULL                     VALUE 'Y'.

01  TAX-TABLE-SW                  PIC X        VALUE 'N'.
     88  TAX-TABLE-IS-FULL                     VALUE 'Y'.

01  TAX-TABLE.
     05  TAXES          OCCURS  4 TIMES.
         10  STATE                             PIC X(2).
         10  TAX                               PIC V999.

01  TABLE-SUB                                  PIC S94) VALUE 1  COMP.

01  STATE-CODE-SW                 PIC X        VALUE 'N'.
     88  STATE-CODE FOUND                      VALUE 'Y'.
```

VA045
MD050
CT080
CA055

9.2.4 Building a Table (Procedure Division)

```
PERFORM P100-READ-SALES-TAX
        THRU P100-READ-SALES-EXIT

PERFORM P200-LOAD-TABLE
       THRU  P-200-LOAD-TABLE-EXIT
          UNTIL TAX-TABLE-IS-FULL
             OR  END-OF-SALE-FILE.

P100-READ-SALES-TAX.
       READ SALES-TAX-FILE
                AT END MOVE 'Y' TO SALES-TAX-FILE-SWITCH.
P100-READ-SALES-TAX-EXIT.
        EXIT.
P200-LOAD-TABLE.
      MOVE SALES-TAX-RECORD   TO TAXES(TABLE-SUB).
      ADD  1  TO  TABLE-SUB
      IF TABLE-SUB > 4
           THEN
              MOVE 'Y' TO  TAX-TABLE-SWITCH.
      PERFORM  P100-READ-SALES-TAX
            THRU  P100-READ-SALES-TAX-EXIT.
```

VA045
MD050
CT080
CA055

9.2.6 Table Lookup

- Table search until match is found
  - Endless loop prevented if not match
    - UNTIL TABLE-SUB > 4
- As table grows in size
  - Increase OCCURS to new size
  - May need to enlarge size of table subscript
  - May need to change termination value for perform
- No other program changes may be necessary

9.2.6 Table Lookup

```
01  TAX-TABLE.
     05  TAXES        OCCURS  5 TIMES.
        10  STATE                    PIC X(2).
        10  TAX                      PIC V999.

01  TABLE-SUB                        PIC S94) VALUE 1  COMP.

MOVE  'N'  TO  STATE-CODE-SWITCH
MOVE  1    TO  TABLE-SUB
PERFORM  P400-SEARCH-TABLE
      THRU  P400-SEARCH-TABLE-EXIT
       UNTIL  STATE-CODE-FOUND
          OR   TABLE-SUB  >  4.

P400-SEARCH-TABLE.
      IF SALES-STATE  =  STATE(TABLE-SUB)
         THEN
            MOVE  'Y'   TO  STATE-CODE-SWITCH
            MOVE TAX(TABLE-SUB)  TO  SALES-TAX.
      ADD  1  TO  TABLE-SUB
P400-SEARCH-TABLE-EXIT.
      EXIT.
```

9.2.7 PERFORM VARYING

```
PERFORM Paragraph-name [THRU paragraph-name-2]
      [ VARYING  Field-1  {num-literal}   BY   {num-literal}
                                 {Field-2}             {Field-3}
        [UNTIL  condition]
```

```
MOVE  'N'       TO   STATE-CODE-SWITCH
PERFORM  P400-SEARCH-TABLE
     THRU   P400-SEARCH-TABLE-EXIT
     VARYING   TABLE-SUB   FROM  1  BY 1
        UNTIL   STATE-CODE-FOUND
           OR    TABLE-SUB  >  4.
```

9.2.8 Indexes

- Define unique index name
  - Follows INDESED BY in OCCURS clause
  - No Working Storage Definition
- Used instead of subscripts

```
01  TAX-TABLE
     05  TAXES     OCCURS  4 TIMES
                    INDEXED BY TAX-INDEX.
          10  STATE          PIC X(2).
          10  TAX            PIC V999.
```

9.2.9 SET Statement

- Used to initialize INDEX values
- Used to change INDEX values

```
SET  TAX-INDEX   TO    1.
SET  TAX-INDEX   TO    FIELS-5.
SET  TAX-INDEX   UP      BY   5.
SET  TAX-INDEX   DOWN  BY   5.
SET  TAX-INDEX   DOWN  BY   FIELD-5.
```

9.2.10 SEARCH Statement

- Searches table sequentially until
  - WHEN condition satisfied
  - AT END (end of table) is reached
- INDEX is automatically incremented
- INDEX is required
- Must SET table index before SEARCH
- AT END option processes no match
- TABLE-NAME specifies data-item with OCCURS

9.2.10 SEARCH Statement

```
SEARCH   Table-Name  [VARYING Index-Name]
         [AT END……]
         [WHEN   condition …]
```

```
SET  TAX-INDEX    TO  1.
SEARCH TAXES-TABLE
    AT END  PERFORM  P600-NO-MATCH
                     THRU  P600-NO-MATCH-EXIT
    WHEN    SALES-STATE   =  STATE(TAX-INDEX)
             MOVE TAX(TAX-INDEX)  TO  SALES-TAX.
```

9.2.11 SEARCH ALL Statement

- Searches table using binary searc
  - Start at midpoint of table
  - Is WHEN condition satisfied
  - If yes, process WHEN True path
  - Is WHEN data > current value
  - If Yes, start at 1/4 table point
  - If No, start at 3/4 point
- Table must be stored in ascending or descending order
- INDEX is required
- Do not have to SET table index before SEARCH

9.2.11 SEARCH ALL Statement

```
SEARCH   ALL Table-Name  [VARYING Index-Name]
         [AT END……]
         [WHEN   condition …]
```

```
01  TAX-TABLE
    05   TAXES   OCCURS  4 TIMES
                 ASCENDING  KEY  IS STATE
                 INDEXED BY TAX-INDEX.
         10  STATE          PIC X(2).
         10  TAX            PIC V999.

SEARCH  ALL TAXES-TABLE
   AT END  PERFORM  P600-NO-MATCH
                    THRU  P600-NO-MATCH-EXIT
   WHEN    SALES-STATE   =  STATE(TAX-INDEX)
                MOVE TAX(TAX-INDEX)  TO  SALES-TAX.
```

9.2.12 OCCURS DEPENDING ON

```
OCCURS   integer-1   TO   integer-2
         DEPENDING ON  data-item
```

```
01  EMPLOYEE-REC
    05  YEAR-TO-DATE-PAYROLL.
        10  YEAR           PIC 9(2).
        10  MONTH          PIC 9(2).
              . . . . . . . . . . . . .
    05  PAYROLL-TABLE   OCCURS  1 TO 12  TIMES
              DEPENDING  ON  MONTH
              INDEXED  BY  MONTH-INDEX.
        10  HOURS          PIC 9(2)V99.
        10  OVERTIME       PIC V999.
```

9.2.13 Two-dimensional Tables

- OCCURS within an OCCURS
- Allows multiple levels of data storage
  - Accounting cross-tabulation
  - Calendar and date cycle processing
- Reference specific occurrences in table with field(idx-1,idx-2)
- Utilize PERFORM within PERFORM
  - Load Table
  - Process table occurances

9.2.13 Two-dimensional Tables (Data)

```
01  EMPLOYEE-PAYROLL-90-94
    05  YEAR-DATA    OCCURS  5  TIMES
                INDEXED  BY YEAR-IDX.
        10  YEAR                                   PIC  9(02).
        10  MONTH-DATA     OCCURS  12  TIMES
                INDEXED  BY  MONTH-IDX.
            15  MONTH                              PIC  X(03).
            15  MONTH-HOURS-TOTAL                  PIC 9(07)V9.
            15  MONTH-OT-TOTAL                     PIC 9(07)V9.
            15  MONTHLY-SALARIES-TOTAL             PIC 9(09)V99.
```

9.2.13 Two-dimensional Tables (Procedure)

```
000-MAIN-LINE.
  . . .
    PERFORM  100-LOAD-YEAR
          THRU  100-LOAD-YEAR-EXIT
           VARYING YEAR-IDX. . .
            UNTIL. . .
100-LOAD-YEAR
    MOVE    INPUT-YEAR     TO  YEAR(YEAR-IDX)
     . . .
    PERFORM  200-LOAD-MONTH
          THRU  200-LOAD-MONTH-EXIT
          VARYING MONTH-IDX . . .
          UNTIL . . .

200-LOAD-MONTH-TABLE
    MOVE  INPUT-MONTH TO MONTH(YEAR-IDX,MONTH-IDX).
    ADD INPUT-HOURS TO
                MONTH-HOURS-TOTAL(YEAR-IDX,MONTH-IDX).
```