MICRO FOCUS

Welcome to:

Introduction to COBOL Programming

2.2.20 Compiler JCL

- Refer to text for samples……..

2.3 Workshop

DO ALL

- 2.3 Review Questions

Extra Fun

- Debug Chap2bug.cbl

2.3 Review Questions

1

A DIVISION A SECTION

A Level 01 entry B Level 05 entry

B SELECT A FD

B BLOCK CONTAINS B PICTURE

2

```
05 LAST-NAME       PIC X(30).
05 FIRST-NAME      PIC X(20).
05 STREET-ADDRESS  PIC X(30).
05 CITY            PIC X(20).
05 STATE           PIC A(2).
05 ZIP-CODE        PIC 9(5).
05 AMOUNT-PAID     PIC 9(7)V99.
```

2.3 Review Questions

3

A ‘HSRP’	D ZERO
B 29.95	E LOW VALUES
B -2036330359	C FILLER
D SPACES	D HIGH-VALUES
B +898.6	A/E ‘999V99’
E -1/2	A ‘DECEMBER 7, 1941’

4

E SELECT	A ASSIGN
B DATA DIVISION	C PROGRAM-ID
D PICTURE	F FIGURATIVE CONSTANT

2.3 Review Questions

5

| E7 | E8 | E9 | 40 |

| F3 | F8 | D4 |

| 95 | 30 | 0F |

| 00 | 03 | 7C |

| 00 | 64 |

| C3 | D6 | C2 | D6 | D3 | 40 | 40 | 40 |

2.3 Review Questions

```
PROGRAM-ID. PROGRAM1.
AUTHOR. PETER MOLCHAN.
INSTALLATION. CLASSROOM.
DATE-COMPILED.
SECURITY. UNCLASSIFIED.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  IBM-370.
OBJECT-COMPUTER.  IBM-370.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT SALES-FILE-IN ASSIGN TO UT-S-SALESIN.
DATA DIVISION.
FILE SECTION.
FD  SALES-FILE-IN
   LABEL RECORDS ARE STANDARD
   RECORDING MODE IS F
   RECORD CONTAINS 80 CHARACTERS
   BLOCK CONTAINS 0 RECORDS
   DATA RECORD IS SALES-RECORD.
01 SALES-RECORD.
   05 ITEM-SOLD              PIC 9(1).
   05 LAST-NAME              PIC X(20).
   05 FIRST-NAME             PIC X(10).
   05 STREET-ADDRESS                 PIC X(20).
   05 CITY                   PIC X(10).
   05 STATE                  PIC A(2).
   05 ZIP-CODE               PIC 9(5).
   05 AMOUNT-PAID            PIC 9(5)V99.
   05 SALESPERSON-CODE               PIC 9(3).
   05 FILLER                 PIC X(2).
WORKING-STORAGE SECTION.
77 END-OF-FILE-SWITCH               PICTURE X VALUE 'N'.
01 PRINT-CONTROL.
   05 LINE-COUNTR            PICTURE 9(2) VALUE 99.
   05 PAGE-COUNTR            PICTURE 9(4) VALUE 0.
   05 LINES-PER-PAGE         PICTURE 9(2) VALUE 60.
```

2.3 Debug Chap2bug.cbl

Problems

```
FILE-SECTION.                          Should be FILE SECTION.
FD  SALES-FILE-IN
    LABEL RECORDS ARE STANDARD
    RECORDING MODE IS F
    RECORD CONTAINS 80 CHARACTERS
    BLOCK CONTAINS 0 RECORDS
    DATA RECORD IS SALES-RECORD.
01 SALES-RECORD                PIC X(80).      No picture clause on a group item
    05 ITEM-SOLD               PIC 9(1).
    05 LAST-NAME               PIC X(20).
    05 FIRST-NAME              PIC X(10).
    05 STREET-ADDRESS          PIC X(20).
    05 CITY                 PIC X(10).
    05 STATE                PIC A(2).
    05 ZIP CODE             PIC 9(5).          Invalid data-name
    05 AMOUNT-PAID             PIC '9(5)V99'.      Invalid picture clause
```

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***

2.1 Objectives

After completing this chapter, you will be able to code basic COBOL statements in the Procedure Division. Specifically, you will be able to:

- Code file I/O statements (OPEN,CLOSE,READ,WRITE)
- Code special I/O statements (ACCEPT,DISPLAY)
- Perform basic data transfer (MOVE)
- Detect when an end of file condition is reached
- Create a simple COBOL program using Mainframe Express
- End the program as needed (GOBACK, STOP RUN)
- Compile, link, and test a simple COBOL program
- Understand the function of an optimizer

3.2 Topics to be covered:

- Procedure Division
- Paragraphs
- I/O Statements
- MOVE statements
- Allowable moves
- GOBACK and STOP RUN
- Compiling and Linking
- Code Optimization

3.2.1 Procedure Division

- Statements
  - Combination of Words & Symbols causing Action

```
MOVE INPUT-RECORD TO WORK-RECORD
```

  - Sentences

```
ADD 1 TO TOTAL-COUNTERS.
IF MONTH = 'JANUARY'
    THEN
      PERFORM JANUARY-ROUTING
ELSE
      PERFORM OTHER-ROUTINE.
```

3.2.1 Paragraphs

- One or more logically related statements
- Begins with Paragraph Name
- Ends with next Paragraph Name

```
    TOP-LEVEL.
PERFORM INIT-ROUTINE.
        PERFORM PROCESS-EACH-RECORD UNTIL END-OF-DATA.
        PERFORM WRAP-UP.
STOP RUN.

    PROCESS-EACH-RECORD.
```

3.2.2 Input/Output Statements

- OPEN
- CLOSE
- READ
- WRITE
- ACCEPT
- DISPLAY

3.2.3 OPEN Statement

- Prepares File for processing
- Must be executed for all I/O
- Designate file as Input or Output
- Example:

```
OPEN INPUT IN-EMP-FILE.
OPEN OUTPUT OUT-FILE.
```

3.2.4 CLOSE Statement

- Terminates processing of files
- Should be executed for all files
- Residue data in file are can be written
- Example:

```
CLOSE EMP-FILE.
CLOSE OUT-FILE.
```

```
CLOSE EMP-FILE
      OUT-FILE.
```

3.2.5 READ Statement

- Retrieves next record from file
- Allows detection of End of File
- Can Transfer external file data to internal area (INTO)
- File must be opened before READ

3.2.5 READ Statement

- Examples

```
READ IN-EMP-FILE

READ IN-EMP-FILE
   AT END MOVE 'Y' TO SW-END-OF-DATA.

READ IN-EMP-FILE  INTO WS-EMP-FILE
   AT END MOVE 'Y' TO SW-END-OF-DATA.
```

3.2.6 WRITE Statement

- Sends record to file
- Requires Record Name
- File must be open
- Can transfer data from other part of program

3.2.6 WRITE Statement

- Examples

```
WRITE NEW-MASTER-RECORD.

WRITE NEW-MASTER-RECORD FROM WORK-MASTER-RECORD

WRITE REPORT-RECORD AFTER ADVANCING 2 LINES
```

3.2.7 ACCEPT Statement

- Retrieves special low-volume data from external source
- DATE, DAY, TIME
- System Input Device (SYSIN)
- Example

```
ACCEPT RUN-DATE FROM DATE
```

3.2.8 DISPLAY Statement

- Sends special low volume data to external source
- Good for Debugging purposes
- Sent to SYSOUT or CONCOLE
- Display Elementary or group items and constants and literals

```
DISPLAY ‘TOTAL RECORDS = ‘ WS-TOTAL-RECORDS
```

3.2.9 MOVE Statement

- Copies contents of input are to output area
- Literal may be specified
- Data conversion is done, if necessary, to meet description of output area
- Truncation and padding may occur
- Can move to more than one output area

3.2.9 MOVE Statement Examples

```
01 INPUT-FIRST-NAME          PIC X(9).
01 OUTPUT-FIRST-NAME         PIC X(15)
     MOVE INPUT-FIRST-NAME  TO  OUTPUT-FIRST-NAME
```

3.2.9 MOVE Statement Examples

```
01 INPUT-FIRST-NAME          PIC X(9).
01 OUTPUT-FIRST-NAME         PIC X(5)
     MOVE INPUT-FIRST-NAME  TO  OUTPUT-FIRST-NAME
```

| A | R | I | S | T | O | T | L | E |

| A | R | I | S | T |

3.2.9 MOVE Statement Examples

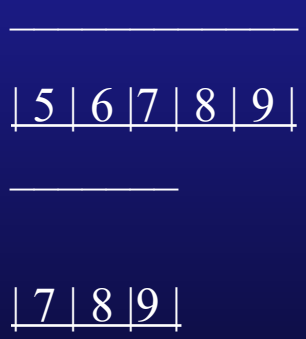
01 INPUT-FIRST-NAME PIC X(4).

01 OUTPUT-FIRST-NAME PIC X(8)

MOVE INPUT-FIRST-NAME TO OUTPUT-FIRST-NAME
JUSTIFIED RIGHT

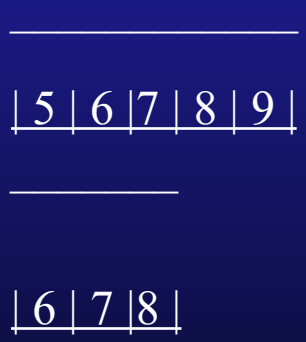
3.2.10 Numeric MOVE Statement Examples

```
01 MONTHLY-CHARGE          PIC 9(3).
01 AMOUNT-OWED              PIC 9(5)
     MOVE MONTHLY-CHARGE  TO AMOUNT-OWED
```



3.2.10 Numeric MOVE Statement Examples

```
01 MONTHLY-CHARGE          PIC 9(3)V99.
01 AMOUNT-OWED             PIC 9(2)V9.
     MOVE MONTHLY-CHARGE  TO AMOUNT-OWED
```



3.2.10 Numeric MOVE Statement Examples

```
01 MONTHLY-CHARGE              PIC 9(3)V99.
01 AMOUNT-OWED                 PIC 9(2)V99.
01  SALARY-AMOUNT              PIC 9(3)V99 COMP-3.
      MOVE ZERO  TO MONTHLY-CHARGE
                    AMOUNT-OWED
                    SALARY-AMOUNT.
```

| 0 | 0 | 0 | 0 | 0 |

| 0 | 0 | 0 |

| 00 | 0 0 | 00 | 0 C |

3.2.12 GOBACK Statement

- Terminate Execution of program
- No further statements executed
- Files should be closed
- Control returns to calling program

3.2.13 STOP RUN Statement

- Terminate Execution of program
- No further statements executed
- Files should be closed
- Control does not return to calling program

3.3 Workshop

- DO Pages 25 AND 26
- Review page 27
- Using the Micro Focus Workbench: (Page 28)
  - Edit the program PROGRAM1.CBL.
  - Your SELECT ASSIGN must be coded as follows:

```
SELECT SALES-FILE-IN ASSIGN TO UT-S-SYSUT1
    ORGANIZATION IS LINE SEQUENTIAL.
```

  - Code a simple Procedure Division that will:
    - read the first record in the SALES file.
    - Print the record to the screen using the DISPLAY verb
    - Close the SALES file.
  - Check and Animate the program
    - be sure to use the ASSIGN 'EXTERNAL' compiler directive
- On-line Quiz……
  - Edit Prog02.cbl
  - Fill in the required blanks - Indicated at Exercise # points
  - Get a clean check (compile) If you wish, Animate
- Change PROGRAM1.CBL so it reads/writes the entire file

3.3 Workshop

1

```
OPEN INPUT CUSTOMER-ORDER-FILE.
CLOSE CUSTOMER-ORDER-FILE.

OPEN OUTPUT CUSTOMER-ORDER-REPORT.
CLOSE CUSTOMER-ORDER-REPORT.

OPEN OUTPUT CUSTOMER ERROR REPORT.
CLOSE CUSTOMER ERROR REPORT.
```

3.3 Workshop

2

```
PROCEDURE DIVISION.
OPEN INPUT SALES-FILE-IN.
      READ SALES-FILE-IN.
    OPEN OUTPUT SALES-FILE-OUT.
      WRITE SALES-REPORT.
```

3

| C6 | C9 | C5 | D3 | C4 | 40 | 40 | 40 | 40 | 40 | 40 |

4

| F0 | F2 | F3 | F8 | F7 |

3.3 Workshop

```
PROGRAM-ID. PROGRAM1.
AUTHOR. PETER MOLCHAN.
INSTALLATION. CLASSROOM.
DATE-COMPILED.
SECURITY. UNCLASSIFIED.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  IBM-370.
OBJECT-COMPUTER.  IBM-370.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT SALES-FILE-IN ASSIGN TO UT-S-SYSUT1
  ORGANIZATION IS LINE SEQUENTIAL.
DATA DIVISION.
FILE SECTION.
FD  SALES-FILE-IN
    LABEL RECORDS ARE STANDARD
    RECORDING MODE IS F
    RECORD CONTAINS 78 CHARACTERS
    BLOCK CONTAINS 0 RECORDS
    DATA RECORD IS SALES-RECORD.
01  SALES-RECORD    PIC X(78).
```

```
WORKING-STORAGE SECTION.
77 END-OF-FILE-SWITCH          PICTURE X VALUE 'N'.
01  DATA-RECORD.
    05  DR-ITEM       PIC 9.
    05  DR-LASTNAME    PIC X(20).
    05  DR-FIRSTNAME   PIC X(10).
    05  DR-STREET         PIC X(20).
    05  DR-CITY           PIC X(10).
    05  DR-STATE          PIC A(2).
    05  DR-ZIP            PIC 9(5).
    05  DR-AMOUNT       PIC 9(5)V99.
    05  DR-SALESCODE  PIC 9(3).
PROCEDURE DIVISION.
MAIN.
    OPEN INPUT SALES-FILE-IN.
    READ SALES-FILE-IN INTO DATA-RECORD.
    DISPLAY DATA-RECORD.
    CLOSE SALES-FILE-IN.
```

3.3 Workshop

```
MAIN-ROUTINE.
    OPEN INPUT SALES-FILE-IN.
    READ SALES-FILE-IN INTO DATA-RECORD.
    PERFORM PROCESS-RECORD THRU PROCESS-RECORD-EXIT
    UNTIL END-OF-FILE-SWITCH = 'Y' .
    CLOSE SALES-FILE-IN.
    GOBACK.


PROCESS-RECORD.
    DISPLAY DATA-RECORD.
    READ SALES-FILE-IN INTO DATA-RECORD
     AT END
      MOVE 'Y' TO END-OF-FILE-SWITCH.
PROCESS-RECORD-EXIT.
```

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***

4.1 Objectives

After completing this chapter, you be able to code basic editing and branching statements in the Procedure Division. Specifically, you will be able to:

- Flowcharting Overview
- Test to determine proper action
- Unconditionally branch to another part of the Procedure Division
- Execute sequence, selection, and iteration in a COBOL program.
- Validate data for numeric contents
- Test logical conditions using AND, OR, and NOT
- Use condition names to clarify and reduce coding

4.2 Topics to be Covered

- Flowcharting Overview
- GO TO
- PERFORM
- EXIT
- Condition names (88)
- COBOL Logic (IF-THEN-ELSE)
- Allowable comparisons
- Truth tables

4.2.0 Flowcharting

- Flowcharts map program logic
- Stand symbols to represent programming functions

Process Decision Document Connector

4.2.1 GO TO

- Transfers control from one part of the program to another
- Paragraph name follows GO TO statement
- Minimal use recommended
- Example

```
GO TO READ-RTN.
```

4.2.3 PERFORM

- Transfers control from one part of the program to another
- Paragraph name follows PERFORM statement
- Returns to statement following PERFORM when finished
- TYPES
  - THROUGH / THRU
  - until
- Use of PERFORM over GO TO recommended

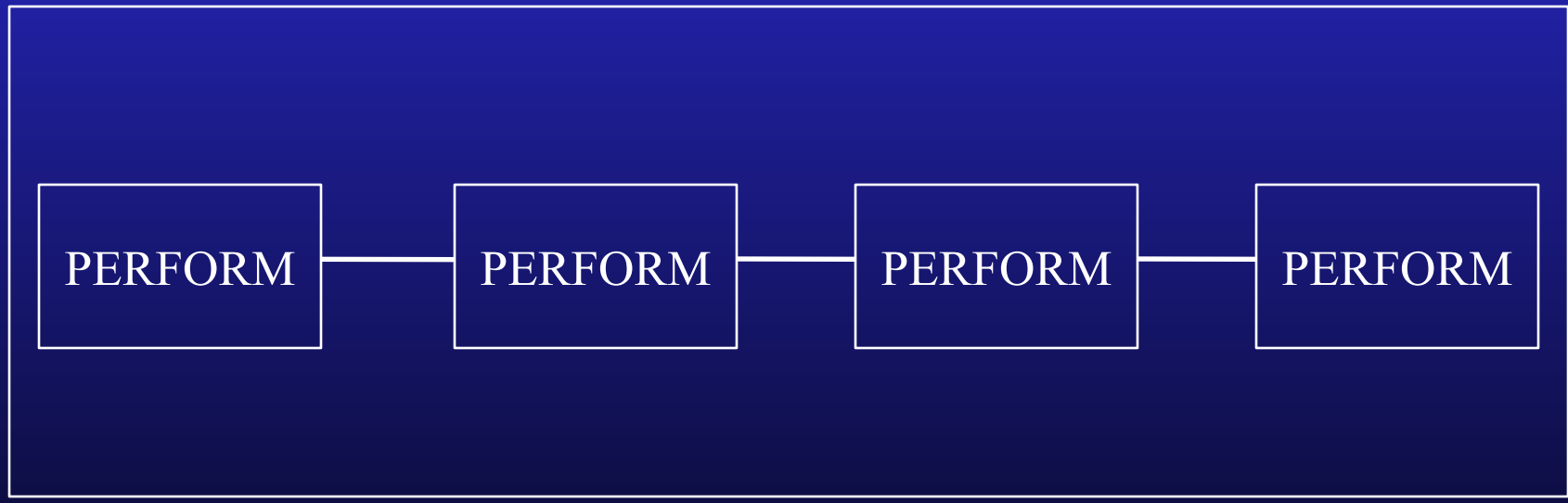
4.2.3 PERFORM

- Example

```
TOP-LEVEL.
    PERFORM 100-HOUSEKEEPING.
    PERFORM 200-MAIN-RTN.
     PERFORM 300-TERMINATION.
100-HOUSE-KEEPING
     OPEN INPUT       SYSUT1
                OUTPUT    SYSUT2.
200-TERMINATION.
     CLOSE SYSUT1
                  SYSUT2.
```

4.2.3 PERFORM

- Sequence Structure
  - TOP-LEVEL paragraph is an example



4.2.4 PERFORM times

- Performs paragraph repetitively
- Number specified must be integer
- Example

```
PERFORM 100-COUNT-RTN 17 TIMES.

PERFORM 200-TOTAL-RTN TOTAL-CTR TIMES
```

4.2.5 PERFORM Thru

- May use THROUGH or THRU
- Executes a series of paragraphs before returning
- Example

```
TOP-LEVEL.
    PERFORM 200-READ THROUGH 300-WRITE.
    PERFORM 400-TERMINATING.
200-READ.
    READ INPUT-FILE.
300-WRITE.
    ADD 1 TO COUNTER-1
    WRITE OUTPUT-RECORD
```

4.2.6 Exit.

- Coded in B Margin
- Provides end point for paragraph
- Only word in paragraph
- Commonly used with Perform Thru
- Example

```
TOP-LEVEL.
        PERFORM 200-READ THROUGH 200-READ-EXIT
200-READ.
        READ INPUT-FILE.
200-READ-EXIT.
        EXIT.
```

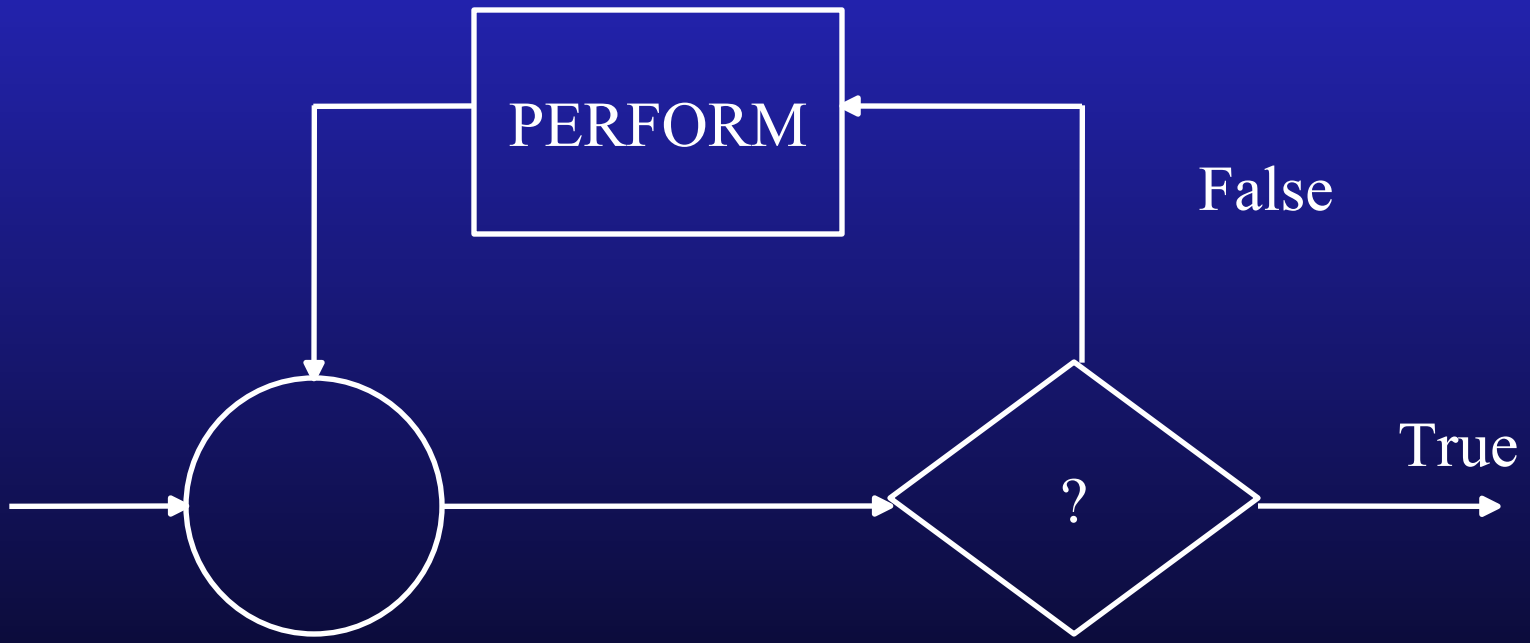
4.2.7 PERFORM Until

- Executes paragraph until a specified condition is true
- Commonly used with THRU option
- Example

```
PERFORM 200-PROCESS-RECORDES THRU
        200-PROCESS-RECORDS-EXIT
       UNTIL END-OF-DATA
200-PROCESS-RECORDS
     READ INPUT-FILE
    AT END MOVE 'Y' TO SW-END-OF-DATA
200-PROCESS-RECORDS-EXIT
     EXIT.
```

4.2.7 PERFORM Until

- Example of Iteration Structure



4.2.8 Condition Names

- Name of the VALUE of a field, not the field itself
- English-like
- Must be unique in the program
- Must be an 88 level
- May be more than one value
- Does not have a PICTURE Clause

4.2.8 Condition Names

- Example

```
01 SW-END-OF FILE        PIC X    VALUE 'N'.
    88 END-OF-DATA                VALUE 'Y'


    PERFORM 200-PROCESS-RECORDES THRU
                    200-PROCESS-RECORDS-EXIT
        UNTIL END-OF-DATA
200-PROCESS-RECORDS
    READ INPUT-FILE
    AT END MOVE 'Y' TO SW-END-OF-DATA
200-PROCESS-RECORDS-EXIT
    EXIT.
```

4.2.8 Condition Names

- Example

```
01 INPUT-INTEGER        PIC 9.
    88 EVEN-INTEGER      VALUE '0,2,4,6,8'
    88 ODD-INTEGER       VALUE '1,3,5,7,9'


IF EVEN-INTEGER
  PERFORM EVEN-ROUTINE.
IF ODD-INTEGER
   PERFORM ODD-ROUTINE.
```

4.2.9 IF-THEN-ELSE

- Causes evaluation to occur
- Action taken depends on result being TRUE or FALSE
  - If TRUE statements immediately following are executed
  - If FALSE statements following ELSE are executed
- Nesting is allowed

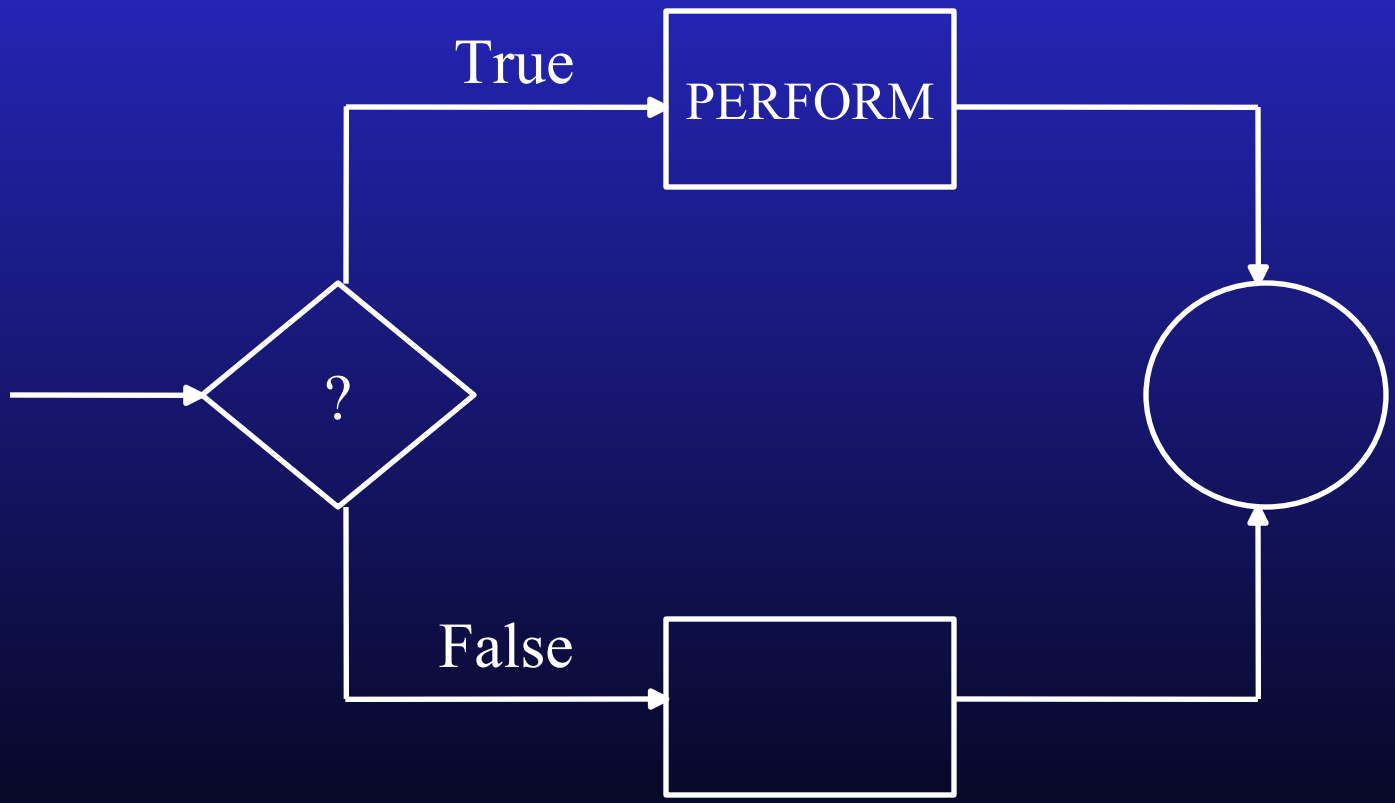
4.2.9 IF-THEN-ELSE

- Syntax

IF field condition comparative
{THEN} {statements} {NEXT SENTENCE}
{ELSE} {statements} {NEXT SENTENCE}

4.2.10 IF-THEN-ELSE

- Selection Structure



4.2.10 Class Condition

- IF field {IS} {NOT} {NUMERIC}
  {ALPHABETIC}
- Example

```
IF INPUT-TOTAL NOT NUMERIC
   THEN
      PERFORM NON-NUMERIC-TOTAL-RTN
         THRU NON-NUMERIC-TOTAL-EXIT.
```

4.2.11 Sign Condition

- IF field {IS} {NOT} {POSITIVE}
  {NEGATIVE}
  {ZERO}
- Example

```
IF BOTTOM-LINE NOT POSITIVE NOT NUMERIC
   THEN
      PERFORM FILE-CHAPTER-11-RTN
         THRU FILE-CHAPTER-11-EXIT.
IF TOTAL-VIOLATIONS  IS ZERO
   THEN
      PERFORM BEST-CUSTOMER-RTN
         THRU BEST-CUSTOMER-EXIT.
```

4.2.12 Relation condition

- IF field1 {IS} {NOT} {LESS THAN} {EQUAL TO} {GREATER THAN} field2
- Example

```
IF GROSS-INCOME GREATER THAN GROSS-EXPENSES
   THEN
      PERFORM NET-PROFIT-ROUTINE
         THRU NET-PROFIT-EXIT.
IF TOTAL-PAID  IS EQUAL TO TOTAL-BILLED
   THEN
      PERFORM BEST-CUSTOMER-RTN
         THRU BEST-CUSTOMER-EXIT.
```

4.2.13 Condition-name condition

- IF {NOT} condition
- Example

```
01 INPUT-INTEGER        PIC 9.
    88  EVEN-INTEGER     VALUE ‘0,2,4,6,8’.
    88 ODD-INTEGER       VALUE ‘1,3,5,7,9’.
```

```
IF EVEN-INTEGER
    THEN
 PERFORM EVEN-ROUTINE.
IF ODD-INTEGER
      PERFORM ODD-ROUTINE .
```

4.2.15 Compound and negated IF-THEN-ELSE

- AND
  - Conjunction
  - All must be true
- OR
  - Inclusive
  - At least 1 must be true
- NOT
  - Negation
  - Condition Not true
- Parentheses

4.2.15 Compound and negated IF-THEN-ELSE

- Examples

```
IF US-CITIZEN AND AGE > 34
    THEN
        MOVE 'Y' TO NEXT-PRESIDENT.

IF STATE-CODE = 'CT' OR 'RI' OR 'MA' OR 'VT'
    THEN
        MOVE 'Y' TO TOP NEW-ENGLAND-STATE

IF NOT CURRENT-CUSTOMER
    THEN
        PERFORM ADD-TO-DATABASE.
```

4.2.15 Compound and negated IF-THEN-ELSE

- Example

```
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.
```

4.2.16 Truth tables

A	B	A and B	A or B	Not A
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

4.3 Workshop

Do written exercises on page 4-26 thru 4-28

Do ***not*** do page 4-29

The following replaces page 4-29

1. Make a copy of your existing program1.cbl and call it program2.cbl
2. Expand the Procedure Division to test each salesperson code to be sure it is numeric. It should only print (display) if it is numeric. Compile and test the program - one record has a non-numeric salescode.
3. Add an error counter and add 1 to the counter in the Procedure Division for each record with a non-numeric sales code. Display this counter value (should be 1) at the end of processing. Compile and test.
4. Restructure your process record routine to only print records that contain an amount sold greater than 0. Use an 88 level to test this condition. (Note: this is not an error condition). Compile and test. 1 record has a 0 value in its amount.
5. Now, lets expand processing to also write our data to an output file. Here's what you need to do…..

- Add a select statement for the new file….
  - SELECT SALES-FILE-OUT ASSIGN TO PRNTFILE.
- Add an FD for the new file…...
  - FD SALES-FILE-OUT
  - LABEL RECORDS ARE STANDARD
  - RECORDING MODE IS F
  - RECORD CONTAINS 133 CHARACTERS
  - BLOCK CONTAINS 0 RECORDS
  - DATA RECORD IS REPORT-RECORD.
  - 01 REPORT-RECORD PIC X(133).
- Add a record description in Working-Storage for your output record
  - 01 SALES-REPORT.
  - 05 SR-LASTNAME PIC X(20).
  - 05 SR-FIRSTNAME PIC X(10).
  - 05 FILLER PIC X(2).
  - 05 SR-SALESCODE PIC X(3).
  - 05 FILLER PIC X(2).
  - 05 SR-AMOUNT PIC $$$$9.99.
- Add the following code to your program just before or just after you Display the record
  - MOVE DR-LASTNAME TO SR-LASTNAME
  - MOVE DR-FIRSTNAME TO SR-FIRSTNAME
  - MOVE DR-SALESCODE TO SR-SALESCODE
  - MOVE DR-AMOUNT TO SR-AMOUNT
  - WRITE REPORT-RECORD FROM SALES-REPORT
  - DISPLAY DATA-RECORD
- Compile and test. Your Display to the Screen should be the same. Verify that the records have been written to your output file by editing the file REPORT.DAT.

4.3 Workshop

1. c.
2. a. c. d. e. only b. is bad
3. IF COUNTER-3 EQUAL 5
   THEN WRITE OUTPUT-RECORD
   ELSE DISPLAY COUNTER-3.
4. IF CURRENT-SALES GREATER THAN 5000.00
   THEN PERFORM DOUBLE-AGENT-COMMISSN.
5. IF CUST-AGE GREATER THAN 62
   0R (CITY EQUAL ‘TALLAHASSEE’ AND STATE EQUAL ‘FL’)
   PERFORM CALC-RTN.
6. IF NOT MANAGER
   THEN PERFORM BONUS-RTN.
7. 05 INPUT STATE PIC X(2).
   88 MASSACHUSETTS VALUE ‘MA’.
   88 NEWYORK VALUE ‘NY’.
8. c. sequence b. selection a. iteration

4.3 Workshop - Program2.cbl

```
PROGRAM-ID. PROGRAM2.
AUTHOR. PETER MOLCHAN.
INSTALLATION. CLASSROOM.
DATE-COMPILED.
SECURITY. UNCLASSIFIED.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  IBM-370.
OBJECT-COMPUTER.  IBM-370.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT SALES-FILE-IN ASSIGN TO UT-S-SYSUT1
     ORGANIZATION IS LINE SEQUENTIAL.
   SELECT SALES-FILE-OUT ASSIGN TO PRNTFILE.
DATA DIVISION.
FILE SECTION.
FD  SALES-FILE-IN
   LABEL RECORDS ARE STANDARD
   RECORDING MODE IS F
   RECORD CONTAINS 78 CHARACTERS
   BLOCK CONTAINS 0 RECORDS
   DATA RECORD IS SALES-RECORD.
01  SALES-RECORD    PIC X(78).
FD  SALES-FILE-OUT
   LABEL RECORDS ARE STANDARD
   RECORDING MODE IS F
   RECORD CONTAINS 133 CHARACTERS
   BLOCK CONTAINS 0 RECORDS
   DATA RECORD IS REPORT-RECORD.
01  REPORT-RECORD   PIC X(133).
WORKING-STORAGE SECTION.
77 END-OF-FILE-SWITCH          PICTURE X VALUE 'N'.
77 ERROR-COUNTER               PICTURE 9(2) VALUE 0.
01  DATA-RECORD.
   05  DR-ITEM        PIC 9.
   05  DR-LASTNAME    PIC X(20).
   05  DR-FIRSTNAME   PIC X(10).
   05  DR-STREET      PIC X(20).
   05  DR-CITY        PIC X(10).
   05  DR-STATE       PIC A(2).
   05  DR-ZIP         PIC 9(5).
   05  DR-AMOUNT      PIC 9(5)V99.
    88 ZERO-AMOUNT VALUE ZERO.
   05  DR-SALESCODE   PIC 9(3).

01  SALES-REPORT.
   05  SR-LASTNAME    PIC X(20).
   05  SR-FIRSTNAME   PIC X(10).
   05  FILLER         PIC X(2).
   05  SR-SALESCODE   PIC X(3).
   05  FILLER         PIC X(2).
   05  SR-AMOUNT      PIC $$$$9.99.


PROCEDURE DIVISION.
MAIN-ROUTINE.
   OPEN INPUT SALES-FILE-IN
        OUTPUT SALES-FILE-OUT.
   READ SALES-FILE-IN INTO DATA-RECORD.
   PERFORM PROCESS-RECORD THRU PROCESS-RECORD-EXIT
   UNTIL END-OF-FILE-SWITCH = 'Y'.
   DISPLAY 'FILE ERRORS ' ERROR-COUNTER.
   CLOSE SALES-FILE-IN
            SALES-FILE-OUT.
   GOBACK.


PROCESS-RECORD.
   IF NOT ZERO-AMOUNT
      IF DR-SALESCODE NUMERIC
        MOVE DR-LASTNAME TO SR-LASTNAME
        MOVE DR-FIRSTNAME TO SR-FIRSTNAME
        MOVE DR-SALESCODE TO SR-SALESCODE
        MOVE DR-AMOUNT TO SR-AMOUNT
        WRITE REPORT-RECORD FROM SALES-REPORT
        DISPLAY DATA-RECORD
      ELSE
        ADD 1 TO ERROR-COUNTER.
   READ SALES-FILE-IN INTO DATA-RECORD
```

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***