

COBOL for the 21st Century

11th edition

John Wiley & Sons, Inc.



Chapter 4

Coding Complete COBOL Programs: The PROCEDURE DIVISION

Chapter Objectives

To familiarize you with methods used to

1. Access input and output files
2. Read data from an input file
3. Perform simple move operations
4. Write information to an output file
5. Accomplish end-of-job operations
6. Execute paragraphs from main module, return control to that main module

Review of First Three Divisions

- IDENTIFICATION DIVISION
 - Identifies program name
- ENVIRONMENT DIVISION
 - Defines files and equipment used by batch programs

Review of First Three Divisions

- DATA DIVISION
 - FILE SECTION
 - Detailed description of input/output records
 - WORKING-STORAGE SECTION
 - Defines keyed input, displayed output
 - Defines fields needed for processing but not part of input/output records

PROCEDURE DIVISION

Contains instructions to

- Read data
- Process data
- Produce output
- Perform end-of-job operations

PROCEDURE DIVISION

- Interactive processing instructions
 - Accept input from keyboard
 - Display output on screen
- Batch processing instructions
 - Access files and read them
 - Write output to files

Paragraphs

- PROCEDURE DIVISION divided into paragraphs
- Each is independent module or routine
- Made up of series of instructions to perform specific set of operations

Rules for Paragraph-Names

- Coded in Area A, followed by period
- Follow rules for forming data-names except may be all digits
 - 1010, 1020, 1030, etc. are valid paragraph names
- Must be unique

Procedure Division Statements

- All statements coded in Area B
- Statement begins with verb (READ, MOVE)
- Last statement in paragraph ends with period
- Sentence - series of statements ending with period

Batch Program Instructions

- **OPEN** - to open files to be processed
- **PERFORM UNTIL ... END-PERFORM**
 - Loop to continually **READ** and process input records and **WRITE** results to output file until there are no more records
- **CLOSE** - to close files when done processing
- **STOP RUN** - to end program

OPEN Statement

- Accesses and makes files available for processing
- Identifies whether files will be used for input or output

OPEN Statement

FORMAT

OPEN { INPUT file-name-1 ...
OUTPUT file-name-2 ... }

- File-names used must appear in SELECT statement
- File must be accessed with OPEN before reading from or writing to it

OPEN statement example

- May use a single OPEN statement

Open Input Payroll-File

Output PayChecks, Err-List

- May use multiple OPEN statements

Open Input Payroll-File

Open Output PayChecks

Open Output Err-List

PERFORM UNTIL ... END-PERFORM

FORMAT

```
PERFORM UNTIL condition-1  
    statement-1 ...  
[END-PERFORM]
```

- Repeatedly executes statement(s) between PERFORM UNTIL ... END-PERFORM until condition specified in UNTIL clause is met

PERFORM UNTIL ... END-PERFORM

- In typical batch program, instructions repeated until no more records in file
- When condition met, program continues with statement following END-PERFORM

PERFORM UNTIL ... END-PERFORM

EXAMPLE

Perform Until WS-More-Data = 'NO'

 Read Payroll-File

 At End

 Move 'NO' To WS-More-Data

 Not At End

 Perform 200-Process-Record

 End-Read

End-Perform

PERFORM UNTIL ... END-PERFORM

- Assume WS-More-Data initially = 'YES'
 - Condition not met, so statements in PERFORM loop will be executed
- WS-More-Data set to 'NO' when MOVE statement executed
 - When condition checked again loop ends since condition has been met

READ Statement

- Reads record from file opened for input
- Transfers record to input storage area
- Makes one record available at a time, not entire file

READ Statement

FORMAT

```
READ file-name-1  
    AT END statement-1 ...  
    [NOT AT END statement-2 ...]  
[END-READ]
```

- File-name appears in SELECT statement, FD entry and OPEN
- AT END tests if there are more records

READ Statement

- If no more records
 - Executes statement(s) after AT END
 - Typically statement(s) to cause loop containing READ to end
- If more records
 - Reads in next record
 - Executes statement(s) after NOT AT END
 - Typically statement(s) to process record just read

Simple PERFORM

FORMAT

PERFORM paragraph-name

- To execute instructions in separate paragraph or module one time
- Transfers control to named paragraph
- Executes all instructions in paragraph
- Control returns to statement following PERFORM

Out-Of-Line PERFORM

FORMAT

PERFORM paragraph-name
UNTIL condition

- Repeats paragraph until condition met
- Control transfers to named paragraph
- Executes instructions in paragraph and returns

In-Line PERFORM

FORMAT

PERFORM UNTIL condition
statement(s)
END-PERFORM

- Repeats statement(s) until condition met
- Statement(s) to be executed are in-line, not in separate paragraph

End-Of-Job Processing

- Steps performed after all records processed
- Release all files
- Terminate processing

CLOSE statement

FORMAT

CLOSE file-name-1 ...

- Close all files opened
- Indicates files no longer needed for processing
- Releases files and deactivates devices

Closing multiple files

- May use a single CLOSE statement
Close Payroll-File
PayChecks
Err-List
- May use multiple CLOSE statements
Close Payroll-File
Close PayChecks
Close Err-List

STOP RUN

- Terminates the program
- Usually last instruction in main module
- Execution continues with next paragraph if STOP RUN is omitted

Interactive Program Statements

- DISPLAY to prompt for input
- ACCEPT to store input in WORKING-STORAGE areas
- Various statements to process input
- DISPLAY to show output
- DISPLAY to ask if there is more input
- ACCEPT to get response

Interactive Program Statements

- Statements coded in an in-line PERFORM loop
- Repeated until user responds that there is no more input
- Like batch programs, loop may include simple PERFORM to execute instructions for processing input that are in a separate paragraph

MOVE statement

FORMAT

MOVE identifier-1 TO identifier-2

- Copies contents of identifier-1 to identifier-2

WRITE statement

FORMAT

WRITE record-name-1

- Transmits data from output record area to associated file on device specified
- Record-name is 01 level name following FD for a file opened for output
- READ file-name, WRITE record-name

Comments in COBOL

- Start with asterisk (*) in column 7
- Use as reminders and explanations of processing performed by program
- Use to describe program in IDENTIFICATION DIVISION
- Use to describe each module in PROCEDURE DIVISION

Batch Programs Review

- Programs that process large quantity of data stored in files
- Require ENVIRONMENT DIVISION
- Require FILE SECTION in DATA DIVISION
- Use OPEN, READ, WRITE, CLOSE verbs to process files

Interactive Programs Review

- Programs that process individual items rather than large group of items
- Provide quick solution to ACCEPT data, process it and DISPLAY results

Chapter Summary

- Batch Program Structure
 - Main module's in-line PERFORM UNTIL repeatedly executes READ statement
 - READ gets and processes next record until there are no more
 - Files are opened before PERFORM loop and closed after loop ends

Chapter Summary

- Interactive Program Structure
 - Main module's in-line PERFORM UNTIL repeatedly
 - Prompts user for input and accepts it
 - PERFORMs other paragraphs as needed to process input
 - Asks user if there is more input
 - Loop repeats until user indicates there is no more data

Chapter Summary

- Paragraph-names coded in Area A and end with period
- All other PROCEDURE DIVISION statements coded in Area B
- Statements executed in order unless
 - PERFORM UNTIL loop executes
 - PERFORM transfers control to another paragraph