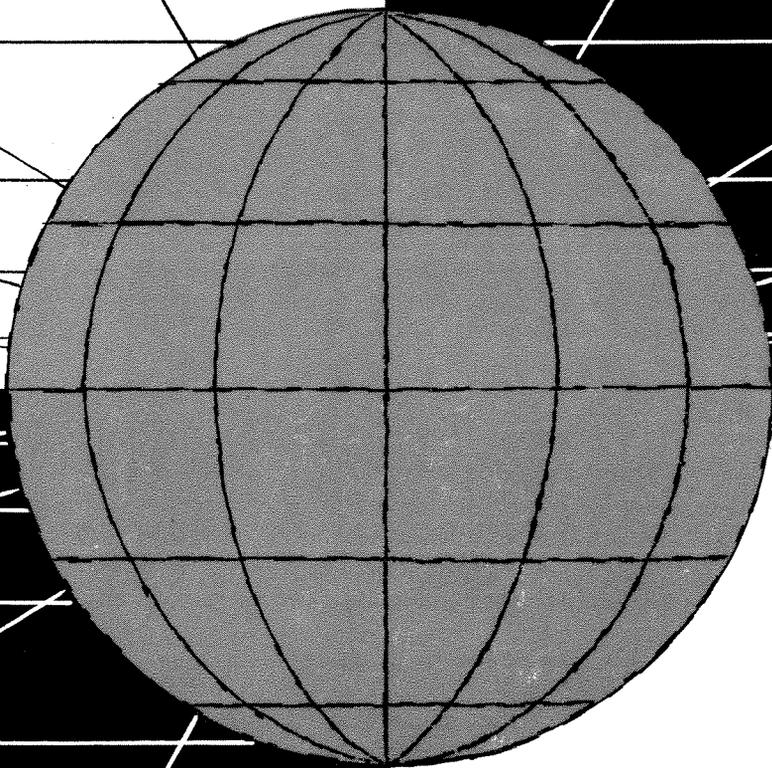


TRS-80 COBOL



Robert T. Grauer

**Volume One
Class Notes**

Cat. No. 26-2723

COBOL for the TRS-80® Microcomputer Volume One Class Notes

by Robert T. Grauer, Ph.D.

Radio Shack®



A DIVISION OF TANDY CORPORATION
FORT WORTH, TEXAS 76102

First Edition.

*"COBOL for the TRS-80® Microcomputer
Volume One Class Notes":*

*Copyright 1983, Tandy Corporation
All Rights Reserved.*

This manual was written by Robert T. Grauer, Ph.D.

Reproduction or use, without express written permission from Tandy Corporation, of any portion of this manual is prohibited, except as provided below. While reasonable efforts have been taken in the preparation of this manual to assure its accuracy, Tandy Corporation assumes no liability resulting from any errors or omissions in this manual, or from the use of the information obtained herein.

Pages in this manual may be duplicated as needed for making overhead transparencies for teaching, to be used by one teacher or with one class.

TABLE OF CONTENTS

Introduction	1
Chapter One: Introduction to COBOL.....	3
Chapter Two: File Processing	19
Chapter Three: COBOL on the TRS-80 Microcomputer.....	35
Chapter Four: The COBOL Language	55
Chapter Five: Debugging	85
Chapter Six: Advanced Features.....	113
Chapter Seven: Programming Style	147
Appendix: Multiple-Choice Review Questions.....	169

INTRODUCTION

This book of class notes for COBOL for the TRS-80® Microcomputer, Volume One, was designed for use with either of the following instructional software packages published by Radio Shack®.

- COBOL for the TRS-80 Models II, 12 and 16: Volume One (Catalog Number 26-2706),
or
- COBOL for the TRS-80 Model 4 and Model III, Volume One (Catalog Number 26-2702).

The above packages contain a COBOL textbook, plus hands-on exercises on diskette, and are designed for use in a classroom, or for self-teaching. These Class Notes are designed as an aid to instructors who wish to teach Volume One COBOL in a classroom setting, and can also be used as a study aid for self-teaching.

Chapters One through Seven of this manual present the instructional material from the Volume One COBOL course in skeleton form. Numerous examples of COBOL statements and portions of COBOL programs help illustrate the concepts covered. These notes can serve as a lecture aid for instructors and/or can be duplicated for use on overhead transparencies. A "True/False" quiz at the end of each chapter helps reinforce learning.

The Appendix contains multiple-choice review questions based on the material covered in the Volume One course. At the teacher's discretion, these questions can be used as quiz material or as an open-book review for the students.

CHAPTER ONE: INTRODUCTION TO COBOL

COBOL

THE FOUR DIVISIONS

THE FUNCTIONS OF ANY PROGRAM

INPUT

PROCESSING

OUTPUT

FUNDAMENTAL PROCEDURE DIVISION VERBS

THE FIRST PROGRAM

COBOL ELEMENTS

RESERVED WORDS

PROGRAMMER-SUPPLIED NAMES

LITERALS

SYMBOLS

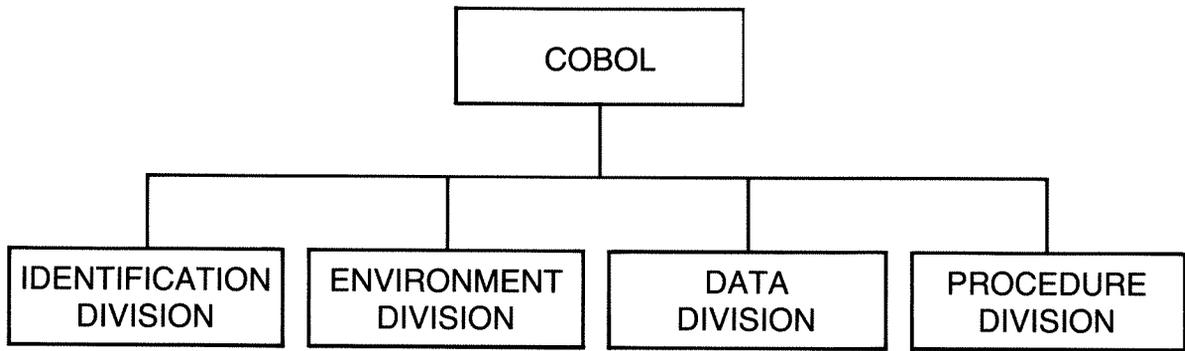
LEVEL NUMBERS

PICTURE CLAUSES

COBOL — COMMON BUSINESS-ORIENTED LANGUAGE

- HAS EXISTED FOR OVER TWENTY YEARS AS THE MOST WIDELY USED BUSINESS PROGRAMMING LANGUAGE
- IS AVAILABLE ON MANY COMPUTERS, FROM MICROS SUCH AS THE TRS-80 MICROCOMPUTER TO VERY LARGE MACHINES
- IS AN INDUSTRY-WIDE LANGUAGE, AND *NOT* THE PROPERTY OF ANY ONE VENDOR OR GROUP
- HAS ACHIEVED A DEGREE OF STANDARDIZATION THROUGH THE AMERICAN NATIONAL STANDARDS INSTITUTE; FOR EXAMPLE, *ANS 74 COBOL*.

A COBOL PROGRAM CONTAINS FOUR DIVISIONS, IN THIS ORDER: IDENTIFICATION, ENVIRONMENT, DATA, AND PROCEDURE.



IDENTIFICATION DIVISION — SPECIFIES PROGRAM AND AUTHOR'S NAME. MAY ALSO CONTAIN OTHER IDENTIFYING INFORMATION SUCH AS DATE WRITTEN, INSTALLATION, AND SO ON.

ENVIRONMENT DIVISION — MENTIONS THE COMPUTER BEING USED AND IDENTIFIES EXTERNAL FILES.

DATA DIVISION — FULLY DESCRIBES ALL DATA, BOTH INPUT AND OUTPUT, WHICH IS USED OR PRODUCED BY THE PROGRAM.

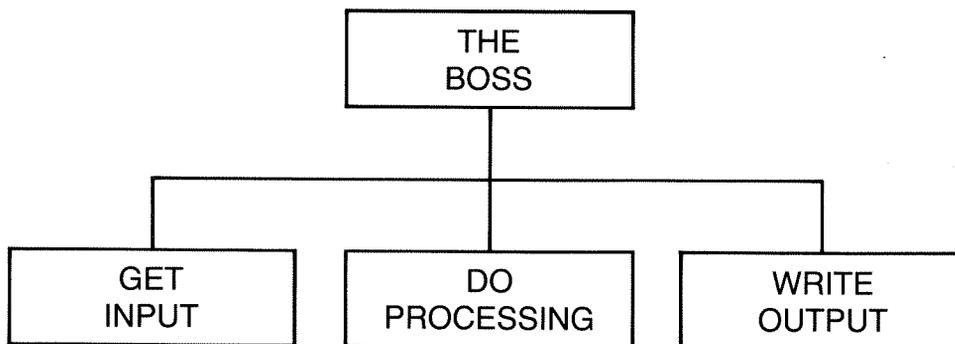
PROCEDURE DIVISION — CONTAINS THE PROGRAM'S LOGIC.

THE FIRST PROGRAM

WRITE A COBOL PROGRAM THAT WILL ACCEPT AS INPUT THREE TEST GRADES, COMPUTE THE AVERAGE, AND DISPLAY THE ANSWER AS OUTPUT.

THE PROCEDURE DIVISION IS “DRIVEN” BY A BOSS PARAGRAPH WHICH CALLS SUBORDINATE PARAGRAPHS.

EVERY PROGRAM MUST PERFORM THREE FUNCTIONS: INPUT, PROCESSING, AND OUTPUT.



FUNDAMENTAL PROCEDURE DIVISION VERBS:

- ACCEPT** — OBTAINS INPUT FROM THE
KEYBOARD
- DISPLAY** — WRITES OUTPUT TO THE MONITOR
(CRT)
- IF** — ENABLES DECISION MAKING
- PERFORM** — TRANSFERS CONTROL TO A
SUBORDINATE MODULE
- ADD AND DIVIDE** — CALCULATE
- STOP RUN** — TERMINATES EXECUTION

THE FIRST COBOL PROGRAM

```
000100 IDENTIFICATION DIVISION.  
000110 PROGRAM-ID. AVERAGE.  
000120 AUTHOR. ROBERT GRAUER.  
000130  
000140 ENVIRONMENT DIVISION.  
000150 CONFIGURATION SECTION.  
000160 SOURCE-COMPUTER. TRS-80.  
000170 OBJECT-COMPUTER. TRS-80.  
000180  
000190 DATA DIVISION.  
000200 WORKING-STORAGE SECTION.  
000210 77 TEST-1 PIC 999.  
000220 77 TEST-2 PIC 999.  
000230 77 TEST-3 PIC 999.  
000240 77 TOTAL-Score PIC 999.  
000250 77 AVERAGE PIC 999.  
000260  
000270 PROCEDURE DIVISION.  
000280 THE-BOSS.  
000290 PERFORM GET-INPUT.  
000300 PERFORM DO-PROCESSING.  
000310 PERFORM WRITE-OUTPUT.  
000320 STOP RUN.  
000330  
000340 GET-INPUT.  
000350 DISPLAY "THE COMPUTER WILL COMPUTE YOUR AVERAGE"  
000360 POSITION 10 LINE 1 ERASE.  
000370  
000380 DISPLAY "ENTER GRADE ON TEST 1" POSITION 6 LINE 4.  
000390 ACCEPT TEST-1.  
000400  
000410 DISPLAY "ENTER GRADE ON TEST 2" POSITION 6 LINE 8.  
000420 ACCEPT TEST-2.  
000430  
000440 DISPLAY "ENTER GRADE ON TEST 3" POSITION 6 LINE 12.  
000450 ACCEPT TEST-3.  
000460  
000470 DO-PROCESSING.  
000480 ADD TEST-1 TEST-2 TEST-3 GIVING TOTAL-Score.  
000490 DIVIDE TOTAL-Score BY 3 GIVING AVERAGE.  
000500  
000510 WRITE-OUTPUT.  
000520 DISPLAY "YOUR AVERAGE GRADE ON 3 TESTS = " LINE 16 AVERAGE.  
000530  
000540 IF AVERAGE > 89  
000550 DISPLAY "CONGRATULATIONS - YOU ARE AN A STUDENT" REVERSE.  
000560  
000570 IF AVERAGE < 70  
000580 DISPLAY "YOU SHOULD STUDY HARDER" REVERSE.
```

Identification Division

Environment Division

Data Division

Paragraph names

COBOL sequence numbers

COBOL ELEMENTS

RESERVED WORDS

PROGRAMMER-SUPPLIED NAMES

LITERALS

SYMBOLS

LEVEL NUMBERS

PICTURE CLAUSES

RESERVED WORDS

- A PREDEFINED LIST OF SOME 300 RESERVED WORDS WHICH MUST BE USED IN A SPECIFIED WAY.
- THE COMPLETE LIST IS SHOWN IN AN APPENDIX IN THE STUDENT TEXT.
- EVERY COBOL STATEMENT CONTAINS AT LEAST ONE RESERVED WORD TO GIVE THE STATEMENT ITS MEANING.
- MUST BE SPELLED CORRECTLY OR ELSE THEY WILL NOT BE RECOGNIZED: *ENVIRONMENT*, NOT *ENVIROMENT*.

PROGRAMMER-SUPPLIED NAMES

- MAY CONTAIN THE LETTERS A THROUGH Z, THE DIGITS ZERO THROUGH 9, AND HYPHENS.
- DATA NAMES MUST CONTAIN AT LEAST ONE LETTER; PARAGRAPH NAMES MAY BE ALL NUMERIC.
- LIMITED TO THIRTY CHARACTERS OR FEWER.
- CANNOT BEGIN OR END WITH A HYPHEN.
- RESERVED WORDS CANNOT BE USED AS DATA NAMES.

EXAMPLES OF *VALID* NAMES:

GROSS-PAY COUNTER-1 START-THE-PROGRAM

EXAMPLES OF *INVALID* NAMES:

GROSS-PAY-IN-\$ COUNTER ONE START

LITERALS

- *NUMERIC* LITERALS MAY HAVE A LEADING SIGN AND A DECIMAL POINT, BUT CANNOT END ON A DECIMAL POINT.
- *NON-NUMERIC* LITERALS ARE ENCLOSED IN QUOTES AND MAY CONTAIN ANYTHING AT ALL, INCLUDING BLANKS, RESERVED WORDS, AND SO ON.

EXAMPLES OF *VALID* LITERALS:

123 - 123.5 "123 +" "HELLO AND GOODBYE"

EXAMPLES OF *INVALID* LITERALS:

123. 123+ HELLO AND GOODBYE

SYMBOLS

- PUNCTUATION

- . ENDS A COBOL ENTRY
- , DELINEATES CLAUSES
- ; DELINEATES CLAUSES
- “ ” SETS OFF NON-NUMERIC LITERALS
- () ENCLOSSES EXPRESSIONS

- ARITHMETIC

- + ADDITION
- SUBTRACTION
- * MULTIPLICATION
- / DIVISION

- CONDITIONAL

- > GREATER THAN
- < LESS THAN
- = EQUAL TO

LEVEL NUMBERS

- USED IN DEFINING DATA NAMES.
- 77 IS USED TO INDICATE AN INDEPENDENT DATA NAME (THAT IS, ONE WITH NO RELATIONSHIP TO OTHER DATA NAMES IN THE PROGRAM).
- OTHER LEVEL NUMBERS MAY GO FROM 01 to 49 INCLUSIVE AND DEFINE THE HIERARCHY AMONG DATA NAMES; FOR EXAMPLE:

```
05  EMPLOYEE-NAME
    10  LAST-NAME
    10  FIRST-NAME
    10  MIDDLE-INITIAL
05  EMPLOYEE-ADDRESS
    10  STREET-NUMBER
    10  CITY
    10  STATE
```

- LEVEL NUMBERS ARE FURTHER EXPLAINED IN CHAPTER 2.

PICTURE CLAUSES

- DESCRIBE THE LENGTH OF A FIELD AND THE KIND OF DATA IT CONTAINS.
- PICTURES OF **A**, **9**, AND **X** DENOTE *ALPHABETIC*, *NUMERIC*, AND *ALPHANUMERIC*, RESPECTIVELY.
- **PIC 999** OR **PIC 9(3)** BOTH DENOTE A THREE-POSITION NUMERIC FIELD.
- **PIC X(4)** AND **PIC AA** DENOTE A FOUR-POSITION ALPHANUMERIC FIELD, AND A TWO-POSITION ALPHABETIC FIELD, RESPECTIVELY.
- EVERY DATA NAME USED IN A PROGRAM MUST BE DEFINED IN THE DATA DIVISION PRIOR TO ITS USE IN THE PROCEDURE DIVISION.

TRUE/FALSE REVIEW

- 1) A COBOL PROGRAM CAN RUN ON A VARIETY OF COMPUTERS.
- 2) THE DIVISIONS IN A COBOL PROGRAM MAY APPEAR IN ANY ORDER.
- 3) NON-NUMERIC LITERALS MAY NOT CONTAIN NUMBERS.
- 4) NUMERIC LITERALS MAY CONTAIN LETTERS.
- 5) PICTURE CLAUSES CAN APPEAR IN COLUMN 12 OR AFTER.
- 6) THE **ACCEPT** STATEMENT WAITS FOR A USER RESPONSE.
- 7) THE **DISPLAY** STATEMENT PRINTS A MESSAGE ON THE CRT.
- 8) A DATA NAME CANNOT CONTAIN ANY CHARACTERS OTHER THAN A LETTER OR A NUMBER.
- 9) THE PERIOD HAS NO EFFECT ON AN **IF** STATEMENT.
- 10) RESERVED WORDS MAY BE USED AS DATA NAMES.

CHAPTER TWO: FILE PROCESSING

VOCABULARY

EMPLOYEE SELECTION PROBLEM

TEST DATA

RESULTING OUTPUT

PSEUDOCODE

FLOWCHARTS

COBOL STATEMENTS REQUIRED FOR FILE PROCESSING

SELECT

FD

OPEN

CLOSE

READ

WRITE

COBOL IMPLEMENTATION OF A LOOP

VOCABULARY

- FIELD — A SINGLE FACT ABOUT A LOGICAL ENTITY (FOR EXAMPLE: NAME, SOCIAL SECURITY NUMBER, ETC.).
- RECORD — A SET OF FACTS (OR FIELDS) ABOUT A LOGICAL ENTITY (FOR EXAMPLE: NAME, ADDRESS, SALARY, AND TITLE MAY COMPRISE AN EMPLOYEE RECORD).
- FILE — A SET OF RECORDS (FOR EXAMPLE: A COMPANY WITH 500 EMPLOYEES WOULD REQUIRE 500 EMPLOYEE RECORDS, IN ONE EMPLOYEE FILE).

EMPLOYEE SELECTION PROBLEM

PROCESS A FILE OF EMPLOYEE RECORDS, SELECTING ANY INDIVIDUAL WHO IS BOTH A PROGRAMMER AND UNDER 30 YEARS OLD.

USE THE FOLLOWING TEST DATA:

NAME	TITLE	AGE	SALARY
JOHN DOE	ANALYST	35	23000
PEGGY WILCOX	PROGRAMMER	31	19000
JOHN SMITH	PROGRAMMER	24	15000
SHEILA LEVINE	PROGRAMMER	29	19000
MARSHAL CRAWFORD	MANAGER	33	28000
STANLEY STEAMER	PROGRAMMER	22	39000
BENJAMIN LEE	PROGRAMMER	26	12000
DICK PERSNICKETY	PROGRAMER	28	19000
MARION MILGROM	JR. PROG	24	10000

THE RESULTING OUTPUT IS:

SALRY REPORT FOR PROGRAMMERS UNDER 30		
JOHN SMITH	24	15000
SHEILA LEVINE	29	19000
STANLEY STEAMER	22	39000
BENJAMIN LEE	26	12000

- WHY WERE THE FOLLOWING RECORDS NOT INCLUDED IN THE REPORT: JOHN DOE, PEGGY WILCOX, DICK PERSNICKETY?
- WHY IS THE WORD "SALARY" MISSPELLED?

THE COMPUTER DOES EXACTLY WHAT IT HAS BEEN PROGRAMMED TO DO, WITHOUT REGARD FOR THE CORRECTNESS OF THE LOGIC OR DATA ON WHICH IT OPERATES.

PSEUDOCODE

HOUSEKEEPING

INITIAL READ

DO WHILE DATA REMAINS

IF EMPLOYEE IS A PROGRAMMER AND UNDER 30

WRITE NAME ON REPORT

ELSE

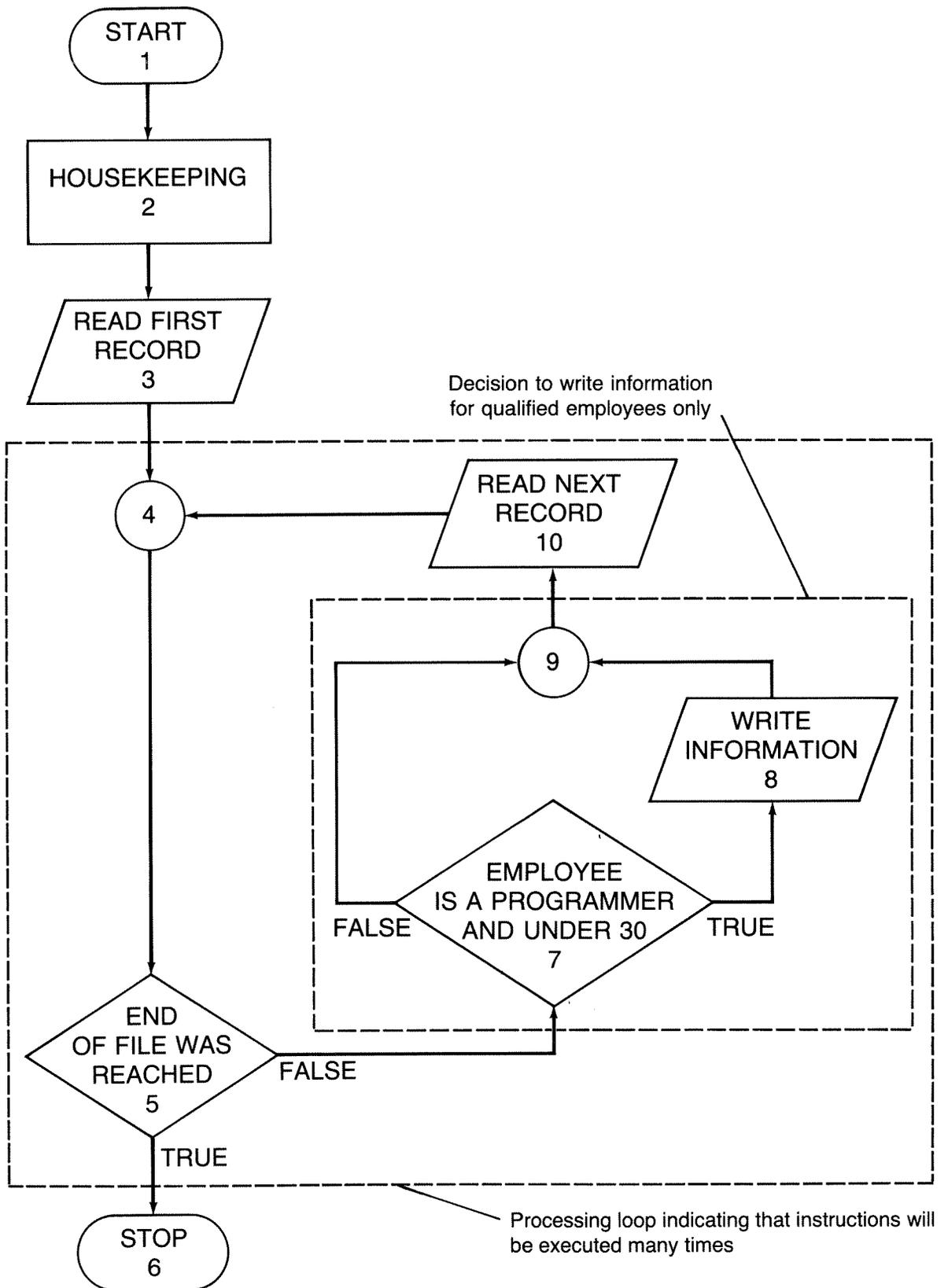
DO NOTHING

READ NEXT RECORD

END DO

STOP

FLOWCHARTS



ELEMENTARY FILE PROCESSING PROGRAM

```

000100 IDENTIFICATION DIVISION.
000110 PROGRAM-ID.  FIRSTTRY.
000120 AUTHOR.      MARION MILGROM.
000130
000140 ENVIRONMENT DIVISION.
000150 CONFIGURATION SECTION.
000160 SOURCE-COMPUTER. TRS-80.
000170 OBJECT-COMPUTER. TRS-80.
000180
000190 INPUT-OUTPUT SECTION.
000200 FILE-CONTROL.
000210     SELECT EMPLOYEE-FILE
000220         ASSIGN TO INPUT "FIRSTTRY/DAT".
000230     SELECT PRINT-FILE
000240         ASSIGN TO PRINT "FIRSTTRY/TXT".
000250
000260 DATA DIVISION.
000270 FILE SECTION.
000280 FD EMPLOYEE-FILE
000290     LABEL RECORDS ARE STANDARD
000300     RECORD CONTAINS 80 CHARACTERS
000310     DATA RECORD IS EMPLOYEE-RECORD.
000320 01 EMPLOYEE-RECORD.
000330     05 EMP-NAME                PIC X(20).
000340     05 EMP-TITLE               PIC X(10).
000350     05 EMP-AGE                 PIC 99.
000360     05 FILLER                  PIC XX.
000370     05 EMP-SALARY              PIC 9(5).
000380     05 FILLER                  PIC X(41).
000390
000400 FD PRINT-FILE
000410     LABEL RECORDS ARE STANDARD
000420     RECORD CONTAINS 132 CHARACTERS
000430     DATA RECORD IS PRINT-LINE.
000440 01 PRINT-LINE.
000450     05 PRINT-NAME              PIC X(25).
000460     05 FILLER                  PIC XX.
000470     05 PRINT-AGE               PIC 99.
000480     05 FILLER                  PIC X(3).
000490     05 PRINT-SALARY           PIC 9(5).
000500     05 FILLER                  PIC X(95).
000510
000520 WORKING-STORAGE SECTION.
000530 77 WS-DATA-REMAINS-SWITCH    PIC X(3)    VALUE SPACES.
000540
000550 PROCEDURE DIVISION.
000560 SELECT-PROGRAMMERS.
000570     OPEN INPUT EMPLOYEE-FILE
000580         OUTPUT PRINT-FILE.
000590     MOVE SPACES TO PRINT-LINE.
000600     MOVE "SALRY REPORT FOR PROGRAMMERS UNDER 30" TO PRINT-LINE.
000610     WRITE PRINT-LINE
000620         AFTER ADVANCING 2 LINES.
000630     READ EMPLOYEE-FILE
000640         AT END MOVE "NO" TO WS-DATA-REMAINS-SWITCH.
000650     PERFORM PROCESS-RECORDS
000660         UNTIL WS-DATA-REMAINS-SWITCH = "NO".
000670     CLOSE EMPLOYEE-FILE
000680         PRINT-FILE.
000690     STOP RUN.
000700
000710 PROCESS-RECORDS.
000720     IF EMP-TITLE = "PROGRAMMER" AND EMP-AGE < 30
000730         MOVE SPACES TO PRINT-LINE
000740         MOVE EMP-NAME TO PRINT-NAME
000750         MOVE EMP-AGE TO PRINT-AGE
000760         MOVE EMP-SALARY TO PRINT-SALARY
000770         WRITE PRINT-LINE
000780             AFTER ADVANCING 2 LINES.
000790
000800     READ EMPLOYEE-FILE
000810         AT END MOVE "NO" TO WS-DATA-REMAINS-SWITCH.

```

Required for file processing

Filename also specified in FD, OPEN, and CLOSE statements

FD is required for every file

EMP-NAME is in positions 1-20

EMP-AGE is in positions 31 and 32

PICTURE clauses sum to 132

Files are opened before processing

Salary is misspelled

Initial read

Files are closed prior to termination

Executed only when IF is satisfied

Single period terminates IF statement

Reads every record but the first

COBOL STATEMENTS REQUIRED
FOR FILE PROCESSING

ENVIRONMENT DIVISION

SELECT

DATA DIVISION

FD

PROCEDURE DIVISION

OPEN

CLOSE

READ

WRITE

THE **SELECT** STATEMENT TIES A PROGRAMMER-
CHOSEN FILE NAME TO A SYSTEM NAME. FOR
EXAMPLE:

```
SELECT EMPLOYEE-FILE  
    ASSIGN TO INPUT "EMPLOYEE/DAT".
```

```
SELECT REPORT-FILE  
    ASSIGN TO PRINT "ERROR/TXT".
```

AN **FD** (FILE DESCRIPTION) IS REQUIRED FOR EVERY FILE IN A PROGRAM. FOR EXAMPLE:

```
FD  EMPLOYEE-FILE
    LABEL RECORDS ARE OMITTED
    RECORD CONTAINS 80 CHARACTERS
    DATA RECORD IS EMPLOYEE-RECORD.
01  EMPLOYEE-RECORD.
    05  EMP-NAME           PIC X(20).
    05  EMP-SALARY        PIC 9(5).
```

NOTE: THE PICTURE CLAUSE DESCRIBES THE TYPE AND LENGTH OF ALL FIELDS WITHIN A RECORD.

ALL FILES MUST BE **OPENED** PRIOR TO PROCESSING
AND **CLOSED** BEFORE PROGRAM TERMINATION.

OPEN INPUT EMPLOYEE-FILE
OUTPUT REPORT-FILE.

CLOSE EMPLOYEE-FILE, REPORT-FILE.

NOTE: THE TYPE OF FILE, *INPUT* OR *OUTPUT*, IS
MENTIONED IN THE OPEN, BUT NOT IN THE
CLOSE, STATEMENT.

THE **READ** AND **WRITE** STATEMENTS ARE USED FOR INPUT AND OUTPUT, RESPECTIVELY.

READ EMPLOYEE-FILE

AT END MOVE "NO" TO DATA-SWITCH.

WRITE PRINT-LINE

AFTER ADVANCING 2 LINES.

NOTES: 1) A FILE OF N RECORDS IS READ $N + 1$ TIMES.

2) SYNTACTICALLY, COBOL *READS A FILE*, BUT *WRITES A RECORD*.

COBOL IMPLEMENTATION OF A LOOP:

INCORRECT

PERFORM PROCESS-RECORDS
UNTIL EOF-SWITCH = "YES".

PROCESS-RECORDS.
READ INPUT-FILE
AT END MOVE "YES" TO EOF-SWITCH.

COBOL IMPLEMENTATION OF A LOOP

CORRECT:

```
READ INPUT-FILE
    AT END MOVE "YES" TO EOF-SWITCH.
PERFORM PROCESS-RECORDS
    UNTIL EOF-SWITCH = "YES".
```

.

.

.

```
PROCESS-RECORDS.
```

.

.

.

```
READ INPUT-FILE
```

```
    AT END MOVE "YES" TO EOF-SWITCH.
```

TRUE/FALSE REVIEW

- 1) A FILE IS A SET OF RECORDS.
- 2) A RECORD IS A SET OF FILES.
- 3) THE COMPUTER IS A PERFECT SPELLER AND AUTOMATICALLY CORRECTS SPELLING ERRORS.
- 4) A FIELD CONTAINS ONE OR MORE RECORDS.
- 5) PSEUDOCODE MUST BE WRITTEN ACCORDING TO PRECISE SYNTACTICAL RULES.
- 6) PSEUDOCODE SERVES THE SAME FUNCTION AS A FLOWCHART.
- 7) A COBOL PROGRAM OFTEN CONTAINS TWO DISTINCT **READ** STATEMENTS.
- 8) THE **READ** STATEMENT TYPICALLY CONTAINS AN **AT END** CLAUSE.
- 9) A FILENAME MAY NOT APPEAR IN MORE THAN TWO STATEMENTS IN THE SAME PROGRAM.
- 10) EVERY PROGRAM MUST HAVE A FILE SECTION.

CHAPTER THREE:
COBOL ON THE TRS-80® MICROCOMPUTER

COMPILATION

COBOL VERSUS MACHINE LANGUAGE

TRS-80 COBOL COMMANDS FOR COMPILATION AND
EXECUTION

FILENAME AND EXTENSIONS

TRSDOS COMMANDS AND UTILITIES

CEDIT

 INSERT MODE

 EDIT MODE

 COMMAND MODE

PRINT

COMPILER — A COMPUTER PROGRAM WHICH
TRANSLATES A HIGHER-LEVEL LANGUAGE, SUCH AS
COBOL, INTO A MACHINE LANGUAGE.

COBOL VERSUS MACHINE LANGUAGE:

A PROGRAM WRITTEN IN A HIGHER-LEVEL LANGUAGE IS SHORTER AND EASIER TO FOLLOW THAN ONE WRITTEN IN A MACHINE LANGUAGE.

COBOL:

COMPUTE X = (A + B) * C

MACHINE LANGUAGE:

LOAD	100
ADD	200
MULTIPLY	300
STORE	400

(THIS EXAMPLE ASSUMES THAT THE LOCATIONS OF A, B, C, AND X ARE 100, 200, 300, AND 400, RESPECTIVELY.)

TRS-80 COBOL COMMANDS FOR COMPILING A PROGRAM:

RSCOBOL *FILENAME OPTIONS*

OPTIONS INCLUDE:

- L** — COMPILER LISTING
- O** — OBJECT MODULE
- P** — SENDS LISTING TO THE PRINTER
- T** — DISPLAYS LISTING ON CRT
- X** — PRODUCES A CROSS REFERENCE

EXAMPLE:

RSCOBOL FIRSTTRY L X T

TRS-80 COBOL COMMANDS FOR EXECUTING A PROGRAM:

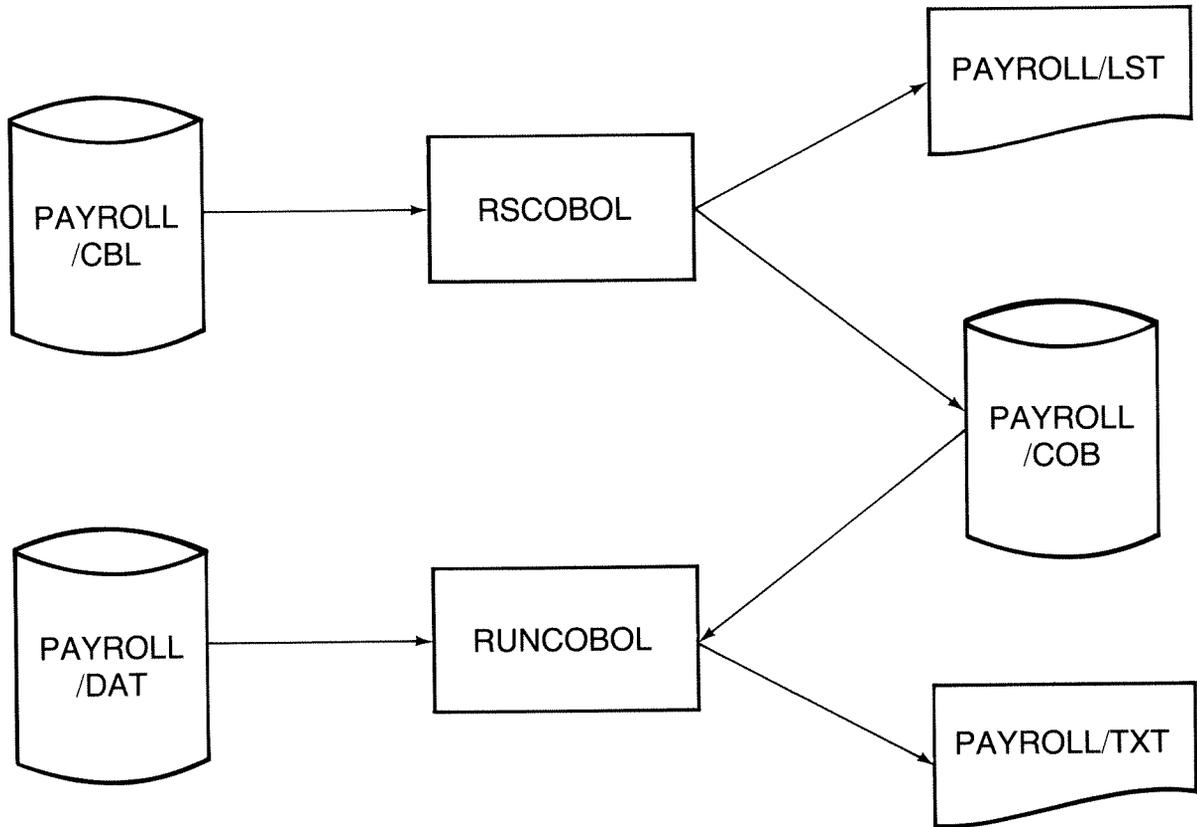
RUNCOBOL *FILENAME*

EXAMPLE:

RUNCOBOL FIRSTTRY

NOTE: EXECUTION SHOULD NOT BE ATTEMPTED UNTIL THE SOURCE PROGRAM FIRSTTRY/CBL HAS COMPILED CLEANLY AND PRODUCED THE OBJECT MODULE FIRSTTRY/COB.

COMPILING AND EXECUTING A PROGRAM



FILENAMES AND EXTENSIONS

ALL FILES ARE SPECIFIED BY A FILENAME AND EXTENSION, WITH THE FOLLOWING CONVENTIONS:

THE FOLLOWING EXTENSIONS ARE TRS-80 CONVENTIONS:

CBL — FOR EXAMPLE, **FIRSTTRY/CBL**, A COBOL SOURCE PROGRAM.

COB — FOR EXAMPLE, **FIRSTTRY/COB**, THE OBJECT (EXECUTABLE) MODULE PRODUCED BY COMPILATION.

LST — FOR EXAMPLE, **FIRSTTRY/LST**, THE OUTPUT (LISTING) OF A COMPILATION, AND USUALLY PRODUCED AS HARD COPY.

THE FOLLOWING EXTENSIONS ARE AUTHOR CONVENTIONS:

DAT — FOR EXAMPLE, **FIRSTTRY/DAT**, A DATA FILE WHICH IS READ DURING EXECUTION.

TXT — FOR EXAMPLE, **FIRSTTRY/TXT**, THE REPORT PRODUCED BY EXECUTING THE OBJECT MODULE.

AS CAN BE SEEN, THE SAME FILENAME CAN EXIST WITH MULTIPLE EXTENSIONS.

TRSDOS COMMANDS AND UTILITIES

AN *OPERATING SYSTEM* IS A SET OF COMPUTER PROGRAMS SUPPLIED BY THE COMPUTER MANUFACTURER, WHICH PROVIDE FOR THE EFFICIENT OPERATION OF THE COMPUTER.

THE TRSDOS OPERATING SYSTEM CONTAINS MANY COMMANDS AND UTILITY PROGRAMS WITH WHICH THE COBOL PROGRAMMER SHOULD BE FAMILIAR. THESE INCLUDE:

- FORMAT** — FOR EXAMPLE, **FORMAT :1 (ID = TRSDOS, PW = PASSWORD, FULL)** MUST BE USED ON ALL BLANK DISKETTES PRIOR TO SAVING ANY INFORMATION.
- BACKUP** — FOR EXAMPLE, **BACKUP** (THE PROGRAM PROMPTS THE USER FOR SUBSEQUENT RESPONSES). BACKUP COPIES ALL FILES FROM ONE DISKETTE TO ANOTHER.
- COPY** — FOR EXAMPLE, **COPY PAYROLL/CBL TO PAYROLL/CBL:1**. COPIES A *SINGLE* PROGRAM FROM ONE DISKETTE TO ANOTHER.

- KILL** — FOR EXAMPLE, **KILL PAYROLL/CBL**. DELETES A *SINGLE* FILE FROM A DISKETTE.
- PURGE** — FOR EXAMPLE, **PURGE** (THE SYSTEM CONTINUALLY PROMPTS THE USER WITH FILENAMES FOR POSSIBLE DELETION). PROVIDES FOR RAPID DELETION OF MANY OR ALL FILES ON A DISKETTE.
- RENAME** — FOR EXAMPLE, **RENAME OLD/CBL TO NEW/CBL**. RENAMES A FILE.
- DIR** — FOR EXAMPLE, **DIR** OR **DIR :1**. LISTS THE DISKETTE DIRECTORY.
- FORMS** — FOR EXAMPLE, **FORMS** (THE SYSTEM PROMPTS FOR ADDITIONAL COMMANDS). SHOULD BE USED PRIOR TO PRINTING.
- I** — FOR EXAMPLE, **I**. *MUST* BE USED PRIOR TO SWAPPING DISKETTES.

CEDIT - COBOL SOURCE PROGRAM EDITOR

- USED TO CREATE AND MODIFY COBOL PROGRAMS
- THREE MODES OF OPERATION

INSERT

EDIT

COMMAND

- INVOKED SIMPLY BY THE COMMAND **CEDIT**, FOLLOWING **TRSDOS READY** MESSAGE

REGARDLESS OF THE PARTICULAR MODE, (COMMAND, INSERT, OR EDIT), THE PROGRAMMER MUST ADHERE TO SPECIFIED COLUMNS IN ACCORDANCE WITH COBOL SYNTAX:

NOTE: THE TAB KEY MAY PROVE QUITE USEFUL

<i>COLUMNS</i>	<i>PURPOSE</i>
1-6	SEQUENCE NUMBERS (GENERATED AUTOMATICALLY BY CEDIT)
7	COMMENTS, CONTINUATION, OR /
8-11	A MARGIN
12-72	B MARGIN
73-80	PROGRAM-ID (OF LITTLE USE)

CERTAIN ENTRIES ARE REQUIRED TO BEGIN IN THE “A MARGIN.” THESE INCLUDE: DIVISION AND SECTION HEADERS, PARAGRAPH NAMES, FD’S, 01’S, AND 77 LEVEL ENTRIES. ANY ENTRY NOT REQUIRED TO BEGIN IN THE “A MARGIN” BEGINS IN OR PAST COLUMN 12.

INSERT MODE

USED TO CREATE PROGRAMS OR TO ADD NEW LINES TO EXISTING PROGRAMS.

ENTERED FROM COMMAND MODE THROUGH **I** AND **R** COMMANDS

SYNTAX: **I** *STARTING-LINE, INCREMENT*

FOR EXAMPLE:

I 100, 10

I 200 (USES CURRENT INCREMENT)

I ,10 (STARTS FROM CURRENT LINE)

SYSTEM WILL NOT PERMIT PROGRAMMER TO INSERT AN EXISTING LINE. IF THIS IS ATTEMPTED, IT PRODUCES THE MESSAGE "NO ROOM BETWEEN LINES".

LINE CONFLICT IS SOLVED BY AUTOMATICALLY RENUMBERING (**A**) OR RENUMBERING (**N**) COMMANDS.

EDIT MODE:

USED TO MAKE CHANGES TO A SPECIFIED LINE.

AVOIDS MULTIPLE **C** AND/OR **X** COMMANDS TO THE SAME LINE

HAS SEVERAL SUBCOMMANDS, INCLUDING:

- D** — DELETES A CHARACTER
- I** — ENTERS INSERTION MODE ON A CHARACTER BASIS
- C** — CHANGES A CHARACTER (PRECEDED BY A NUMBER; FOR EXAMPLE, **3C** WILL CHANGE THE NEXT 3 CHARACTERS)

SPECIAL KEYS

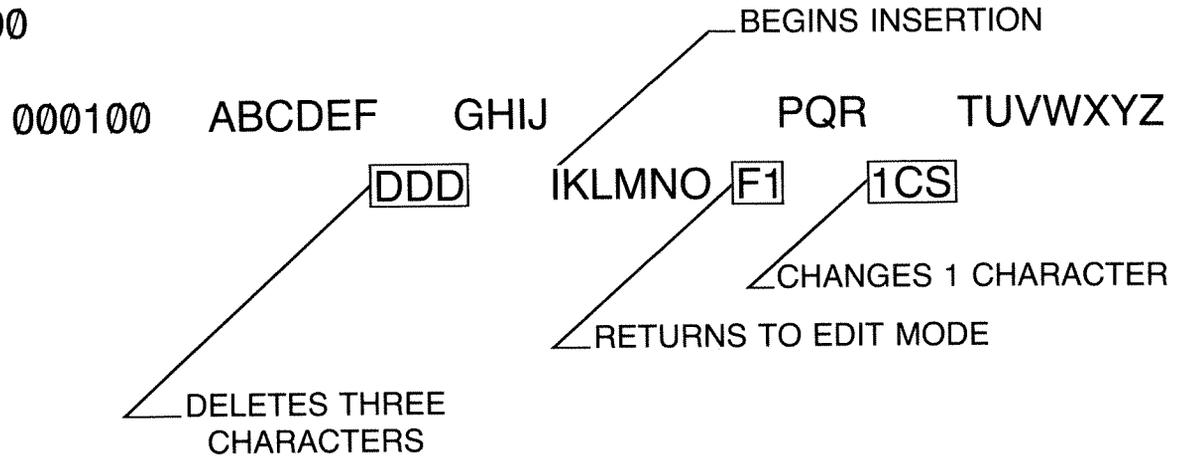
- F1** — ESCAPES FROM INSERTION SUB-MODE AND RETURNS TO EDIT MODE
- ENTER** — ENDS EDITING AND RETURNS TO COMMAND MODE

EXAMPLE OF EDIT MODE:

OLD LINE:

000100 ABCDEFFFGHIIJPQRXTUVWXYZ

E 100



NEW LINE:

000100 ABCDEFGHIJKLMNOPQRSTUVWXYZ

COMMAND MODE:

A — AUTOMATIC RENUMBERING TO AVOID LINE COLLISIONS IN THE **I** OR **R** COMMANDS

B — DISPLAYS THE FIRST (BEGINNING) LINE (ON MODELS II, 12 AND 16 ONLY)

C — UNCONDITIONAL REPLACEMENT (CHANGE) OF ONE CHARACTER STRING BY ANOTHER

C/OLD-STRING/NEW-STRING/

C/OLD/NEW/3

C/OLD//

D — DELETES A LINE OR LINES

D 100

D 100:200

D 5:*

E — ENTERS EDIT MODE, WHICH WAS PREVIOUSLY EXPLAINED

F — FINDS A CHARACTER STRING. UPON COMPLETION, THE CURRENT LINE IS SET TO THE LINE OF THE LAST FIND.

F/STRING/

F/STRING/*

F/X/3

I — ENTERS THE INSERT MODE (TO ESCAPE
PRESS EITHER ESC OR BREAK KEY)

I 100

I 100, 1

I

L — LOADS A CBL SOURCE FILE

L PAYROLL

L PAYROLL :1

N — RENUMBERS ALL LINES IN THE TEXT

N 100

N 100, 20

P — PRINTS THE SPECIFIED LINE OR LINES (IF
LINE-RANGE IS OMITTED, THE NEXT 20 LINES
ARE PRINTED.)

P 500

P 100:200

P 200:*

P

Q — QUILTS CEDIT AND RETURNS TO TRSDOS

R — REPLACES SPECIFIED LINE AND CONTINUES
IN INSERT MODE (EXIT INSERT MODE BY
HITTING EITHER ESC OR BREAK KEYS)

R 100

R 100, 2

S — ACCEPTS A TRSDOS COMMAND AND
RETURNS TO CEDIT

S DIR

T — DISPLAYS TOP LINE (ON
MODEL III AND 4 ONLY)

W — WRITES CURRENT TEXT (WORKFILE) AS A
PERMANENT FILE WITH EXTENSION CBL

W PAYROLL

X — CONDITIONAL REPLACEMENT OF ONE
CHARACTER STRING BY ANOTHER

X/OLD-STRING/NEW-STRING/

X/OLD/NEW/3

X/OLD//

TRSDOS PRINT UTILITY

- USED TO PRINT COMPILER LISTINGS, COBOL SOURCE PROGRAMS, OR OUTPUT FROM EXECUTED PROGRAMS.
- THE **FORMS** COMMAND SHOULD BE EXECUTED PRIOR TO **PRINT**.
- EXAMPLES:

PRINT FIRSTTRY/LST

PRINT FIRSTTRY/CBL

PRINT FIRSTTRY/TXT

TRUE/FALSE REVIEW

- 1) **RSCOBOL** AND **RUNCOBOL** ARE EQUIVALENT COMMANDS.
- 2) **CBL** IS THE EXTENSION FOR A COBOL SOURCE PROGRAM.
- 3) A CROSS-REFERENCE LISTING IS ALWAYS PROVIDED WITH A COBOL COMPILE.
- 4) THE COMMAND **X/PRT/PRINT/** ALWAYS CHANGES **PRT** TO **PRINT**.
- 5) THE COMMAND **C/PRT/PRINT/10** CHANGES **PRT** TO **PRINT** IN LINE 10.
- 6) THE COMMAND **RSCOBOL PAYROLL L** WILL CREATE THE FILES **PAYROLL/COB** AND **PAYROLL/LST**.
- 7) THE COMMAND **P 100:300** PRINTS 20 LINES.
- 8) STATEMENT NUMBERS CAN NEVER BE CHANGED.
- 9) A COBOL PROGRAM IS REQUIRED TO CREATE A DATA FILE.
- 10) THE UTILITY **COBOLPRT** IS USED TO PRINT A COBOL PROGRAM.

CHAPTER FOUR: THE COBOL LANGUAGE

COBOL NOTATION

IDENTIFICATION DIVISION

ENVIRONMENT DIVISION

DATA DIVISION

FILE SECTION, WORKING-STORAGE SECTION,
GROUP VERSUS ELEMENTARY ITEMS (PICTURE
CLAUSE, LEVEL NUMBERS, VALUE CLAUSE)

PROCEDURE DIVISION

ADD, SUBTRACT, MULTIPLY, DIVIDE, COMPUTE,
OPEN, CLOSE, READ, WRITE, IF, PERFORM, MOVE,
STOP RUN

NUMERIC VERSUS EDITED FIELDS

PAYROLL PROGRAM

COBOL NOTATION

UPPER-CASE LETTERS	=	RESERVED WORDS
lower-case letters	=	PROGRAMMER SUPPLIED NAME
<u>UNDERLINE</u>	=	REQUIRED RESERVED WORD
BRACKETS ([])	=	OPTIONAL ENTRY
BRACES ({ })	=	CHOICE
THREE PERIODS (...)	=	REPETITION OF LAST SYNTACTICAL UNIT

COBOL NOTATION (CONTINUED)

WRITE record-name $\left[\begin{array}{l} \text{BEFORE} \\ \text{AFTER} \end{array} \right] \text{ADVANCING} \left\{ \begin{array}{l} \text{integer LINES} \\ \text{PAGE} \end{array} \right\}$

ALL OF THE FOLLOWING ARE ACCEPTABLE:

WRITE PRINT-LINE.

WRITE PRINT-LINE AFTER ADVANCING 2 LINES.

WRITE PRINT-LINE AFTER 2.

WRITE PRINT-LINE BEFORE ADVANCING PAGE.

IDENTIFICATION DIVISION.

PROGRAM-ID.	FIRSTTRY.
AUTHOR.	ROBERT GRAUER.
INSTALLATION.	TANDY CORPORATION.
DATE-WRITTEN.	OCTOBER 31, 1983.
SECURITY.	TOP SECRET.

THE PROGRAM-ID PARAGRAPH IS THE ONLY REQUIRED ENTRY.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. TRS-80.

OBJECT-COMPUTER. TRS-80.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT EMPLOYEE-FILE

ASSIGN TO INPUT "PAYROLL/DAT".

SELECT PRINT-FILE

ASSIGN TO PRINT "PAYROLL/TXT".

DATA DIVISION

FILE SECTION

- CONTAINS AN **FD** (FILE DESCRIPTION) FOR EVERY FILE SPECIFIED IN A SELECT STATEMENT
- DEFINES ALL DATA NAMES WHICH APPEAR IN ANY FILE AS TO TYPE, LENGTH, AND POSITION WITHIN A RECORD.

WORKING-STORAGE SECTION

- DEFINES ALL DATA NAMES WHICH WERE NOT PREVIOUSLY DESCRIBED IN THE FILE SECTION. THESE DATA NAMES TYPICALLY INCLUDE COUNTERS, PROGRAM SWITCHES, AND VARIOUS PRINT LINES.

GROUP ITEMS VERSUS ELEMENTARY ITEMS

A GROUP ITEM IS SUBDIVIDED INTO ONE OR MORE ELEMENTARY ITEMS. IT NEVER HAS A PICTURE CLAUSE

AN ELEMENTARY ITEM IS NOT FURTHER SUBDIVIDED AND ALWAYS HAS A PICTURE CLAUSE

LEVEL NUMBERS ARE USED TO INDICATE THE HIERARCHICAL RELATIONSHIP BETWEEN GROUP AND ELEMENTARY ITEMS

THE PICTURE CLAUSE INDICATES THE TYPE AND SIZE OF A FIELD (FOR EXAMPLE, **PIC X(15)**, **PIC 9(4)**, ETC.)

EXAMPLE 1:

```
01  EMPLOYEE-RECORD.  
    05  EMP-NAME.  
        10  EMP-FIRST-NAME      PIC X(10).  
        10  EMP-LAST-NAME       PIC X(15).  
    05  EMP-SALARY              PIC 9(5).  
    05  EMP-DATE-OF-BIRTH.  
        10  EMP-BIRTH-MONTH     PIC 99.  
        10  EMP-BIRTH-YEAR      PIC 99.  
    05  FILLER                  PIC X(46).
```

EXAMPLE 2:

```
01  EMPLOYEE-RECORD.  
    04  EMP-NAME.  
        08  EMP-FIRST-NAME      PICTURE X(10).  
        08  EMP-LAST-NAME       PICTURE X(15).  
    04  EMP-SALARY              PICTURE 9(5).  
    04  EMP-DATE-OF-BIRTH.  
        08  EMP-BIRTH-MONTH     PICTURE 9(2).  
        08  EMP-BIRTH-YEAR      PICTURE 9(2).  
    04  FILLER                  PICTURE X(46).
```

EXAMPLE 3:

WORKING-STORAGE SECTION.

```
77  END-OF-FILE-SWITCH    PIC X(3)    VALUE SPACES.

01  PROGRAM-COUNTERS.
     05  TOTAL-GROSS-PAY  PIC 9(6)    VALUE ZEROS.
     05  TOTAL-NET-PAY   PIC 9(6)    VALUE ZEROS.

01  HEADING-LINE-ONE.
     05  FILLER           PIC X(5)    VALUE SPACES.
     05  FILLER           PIC X(4)    VALUE "NAME".
     05  FILLER           PIC X(13)   VALUE SPACES.
     05  FILLER           PIC X(6)    VALUE "SALARY".
     05  FILLER           PIC X(104)  VALUE SPACES.

01  DASHED-LINE.
     05  FILLER           PIC X(40)   VALUE ALL "-".
```

NOTE THE USE OF FIGURATIVE CONSTANTS (SPACES AND ZEROS) AND THE WORD "ALL" IN CONJUNCTION WITH THE VALUE CLAUSE.

PROCEDURE DIVISION

ARITHMETIC

ADD

SUBTRACT

MULTIPLY

DIVIDE

COMPUTE

I/O OPERATIONS

OPEN

CLOSE

READ

WRITE

DECISIONS

IF

LOOPING

PERFORM

DATA TRANSFER AND EDITING

MOVE

PROGRAM TERMINATION

STOP RUN

ADD

THE STATEMENT:

ADD A B *GIVING* C.

TAKES THE VALUE OF **A**, ADDS IT TO **B**, AND PUTS THE RESULT IN **C**. FOR EXAMPLE, IF A, B, AND C ARE INITIALLY 20, 25, AND 30, C WILL HAVE A FINAL VALUE OF 45.

THE STATEMENT:

ADD A B *TO* C.

TAKES THE VALUE OF **A**, ADDS IT TO **B**, ADDS THE RESULT TO **C** AND PUTS THE FINAL SUM IN **C**. USING THE VALUES OF 20, 25, AND 30 AS ABOVE, C WILL HAVE A FINAL VALUE OF 75.

SUBTRACT

THE STATEMENT:

SUBTRACT A *FROM* B.

SUBTRACTS THE VALUE OF **A** FROM THE VALUE OF **B**, AND LEAVES THE RESULT IN **B**. FOR EXAMPLE, IF A AND B ARE INITIALLY 4 AND 6, THEY WILL ASSUME FINAL VALUES OF 4 AND 2, RESPECTIVELY.

THE STATEMENT:

SUBTRACT A FROM B *GIVING* C.

SUBTRACTS **A** FROM **B** AND PUTS THE RESULT IN **C**. (THE VALUES OF A AND B ARE UNCHANGED.) USING THE SAME NUMBERS AS ABOVE, THE FINAL VALUES OF A, B, AND C ARE 4, 6, AND 2, RESPECTIVELY.

MULTIPLY

THE STATEMENT

MULTIPLY A *BY* B.

MULTIPLIES THE VALUE OF **A** BY THE INITIAL VALUE OF **B**, AND PUTS THE RESULT IN **B**. GIVEN INITIAL VALUES OF 10 AND 20 FOR A AND B, RESPECTIVELY, THE FINAL VALUES WILL BE 10 AND 200.

THE STATEMENT

MULTIPLY A BY B *GIVING* C.

MULTIPLIES **A** BY **B** AND PUTS THE RESULT IN **C**.
(THE VALUES OF A AND B ARE UNCHANGED.)
USING THE SAME NUMBERS AS ABOVE, THE FINAL VALUES OF A, B, AND C ARE 10, 20, AND 200, RESPECTIVELY.

DIVIDE

THE STATEMENT

DIVIDE A *INTO* B.

TAKES THE VALUE OF **A**, DIVIDES IT INTO **B**, AND LEAVES THE RESULT IN **B**. (IN OTHER WORDS, B IS DIVIDED BY A.) IF A AND B ARE INITIALLY 3 AND 6, THEY WILL ASSUME FINAL VALUES OF 3 AND 2, RESPECTIVELY.

THE STATEMENT

DIVIDE B BY A *GIVING* C.

DIVIDES **B** BY **A** AND PUTS THE RESULT IN **C**. (THE VALUES OF A AND B ARE UNCHANGED.) USING THE SAME NUMBERS AS ABOVE, A, B, AND C WILL HAVE FINAL VALUES OF 3, 6, AND 2, RESPECTIVELY.

COMPUTE

THE **COMPUTE** STATEMENT CAN COMBINE SEVERAL ARITHMETIC OPERATIONS INTO A SINGLE STATEMENT. FOR EXAMPLE:

COMPUTE X = (A - B) / (C + D)

VERSUS:

SUBTRACT B FROM A GIVING NUMERATOR.

ADD C D GIVING DENOMINATOR.

DIVIDE NUMERATOR BY DENOMINATOR GIVING X.

COMPUTE (CONTINUED)

RULES OF PRECEDENCE ARE REQUIRED TO ELIMINATE AMBIGUITY IN THE COMPUTE STATEMENT. GIVEN VALUES OF 1, 2, AND 3 FOR A, B, AND C, RESPECTIVELY, IS D EQUAL TO 9 OR 7?

$$\text{COMPUTE } D = A + B * C$$

THE ANSWER IS 7, BECAUSE OF THE HIERARCHY OF OPERATIONS:

- 1) MULTIPLY OR DIVIDE
- 2) ADD OR SUBTRACT
- 3) LEFT TO RIGHT, IF A TIE.

PARENTHESES CAN *CLARIFY* AND IN SOME CASES *ALTER* THE NORMAL SEQUENCE OF OPERATIONS. WHICH STATEMENT IS EQUIVALENT TO THE ABOVE?

$$\text{COMPUTE } D = (A + B) * C$$

OR

$$\text{COMPUTE } D = A + (B * C)$$

INPUT/OUTPUT STATEMENTS

OPEN INPUT EMPLOYEE-FILE
OUTPUT PRINT-FILE.

READ EMPLOYEE-FILE
AT END MOVE "NO" TO DATA-REMAINS-SWITCH.

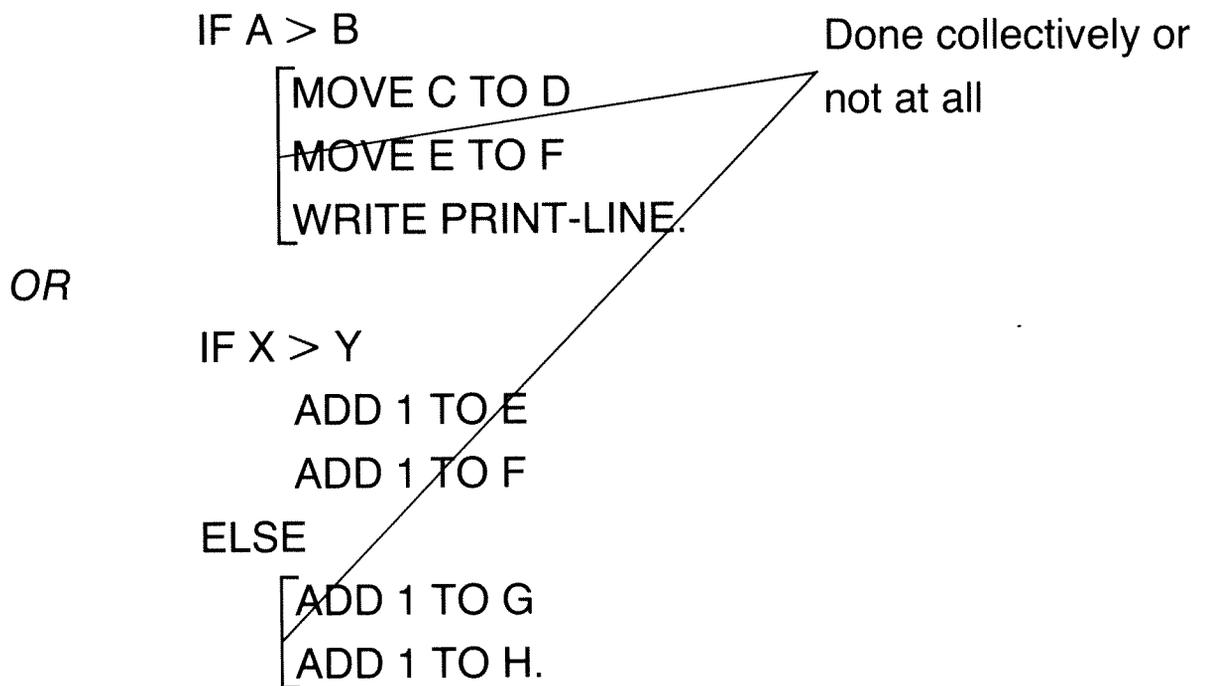
WRITE PRINT-RECORD
AFTER ADVANCING 2 LINES.

CLOSE EMPLOYEE-FILE
PRINT-FILE.

IF STATEMENT (FOR DECISIONS)

IF condition statement-1[ELSE statement-2]

STATEMENT-1 IS TERMINATED BY A PERIOD OR AN ELSE. IN OTHER WORDS, SEVERAL ACTIONS MAY BE TAKEN WHEN A CONDITION IS TRUE. FOR EXAMPLE:

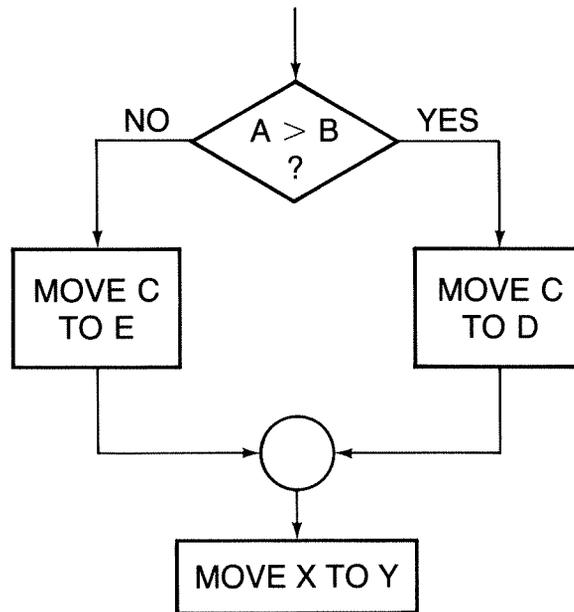


IF STATEMENT (CONTINUED)

THE **IF** MAY BE CODED WITH OR WITHOUT AN **ELSE**

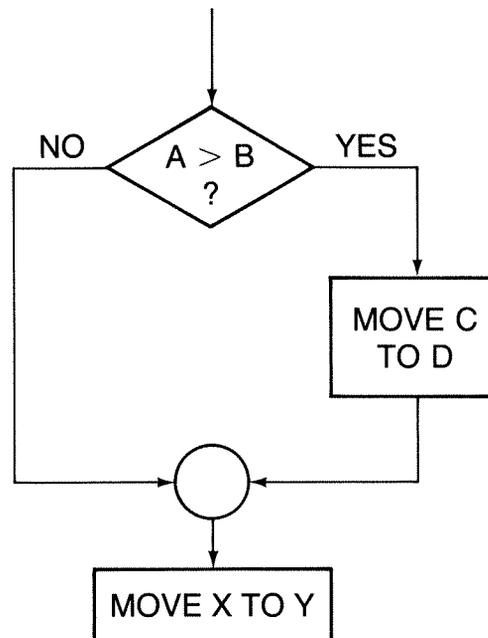
EXAMPLE 1:

```
IF A > B
    MOVE C TO D
ELSE
    MOVE C TO E.
MOVE X TO Y.
```



EXAMPLE 2:

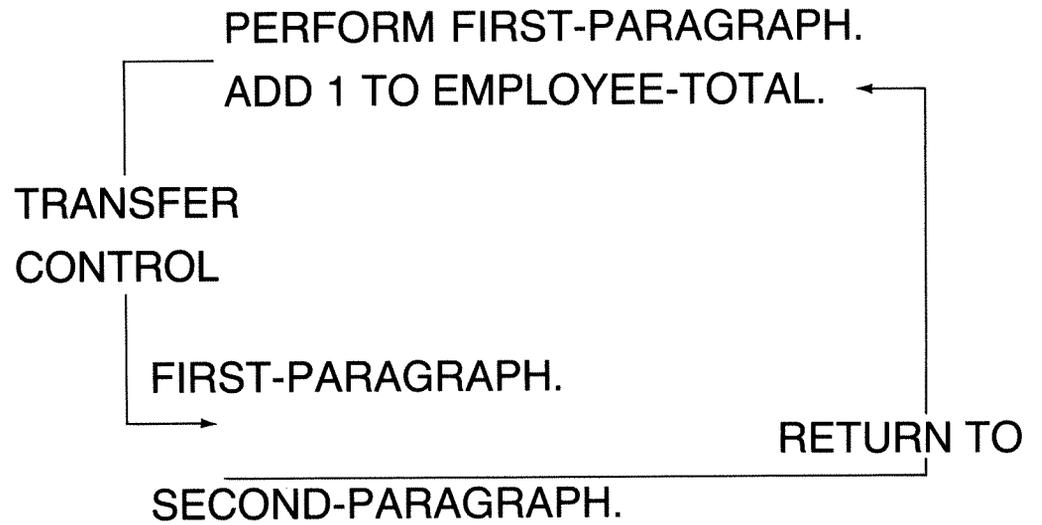
```
IF A > B
    MOVE C TO D.
MOVE X TO Y.
```



PERFORM STATEMENT

PERFORM paragraph-name [UNTIL condition]

FOR EXAMPLE:



PERFORM STATEMENT (CONTINUED)

THE PERFORM STATEMENT IS USED TO PROCESS A FILE UNTIL NO MORE DATA REMAINS:

 READ INPUT-FILE

 AT END MOVE "NO" TO DATA-SWITCH.

 PERFORM PROCESS-RECORDS

 UNTIL DATA-SWITCH = "NO".

 .

 .

 .

PROCESS-RECORDS.

 .

 .

 .

 READ INPUT-FILE

 AT END MOVE "NO" TO DATA-SWITCH.

MOVE STATEMENT

MOVE { identifier-1
literal } TO identifier-2

THE MOVE STATEMENT TRANSFERS DATA BETWEEN ONE STORAGE LOCATION AND ANOTHER.

FOR EXAMPLE:

MOVE 15000 TO BEGINNING-SALARY.

MOVE "PAYROLL REPORT" TO PRINT-LINE.

MOVE BEGINNING-SALARY TO PRINT-SALARY.

NUMERIC VERSUS EDITED FIELDS

- ALL COMPUTATIONS ARE DONE ON NUMERIC FIELDS (THOSE WHOSE PICTURE CLAUSES CONTAIN ONLY 9'S AND AN OPTIONAL IMPLIED DECIMAL POINT).
- THE NUMERIC FIELD IS MOVED TO AN EDITED FIELD PRIOR TO PRINTING.

CONSIDER:

```
01  NUMERIC-FIELDS.  
    05  HOURS          PIC 99.  
    05  RATE           PIC 99V99.  
    05  PAY            PIC 999V99.  
01  PRINT-LINE.  
    05  PRT-HOURS     PIC Z9.  
    05  PRT-RATE      PIC $$9.99.  
    05  PRT-PAY       PIC $999.99.  
  
    COMPUTE PAY = HOURS * RATE.  
    MOVE PAY TO PRT-PAY.
```

USE OF EDITING SYMBOLS

	<i>SOURCE FIELD</i>		<i>RECEIVING FIELD</i>	
	<u>PICTURE</u>	<u>CONTENTS</u>	<u>PICTURE</u>	<u>CONTENTS</u>
A.	9(4)	0678	9(4)	0678
B.	9(4)	0678	\$9(4)	\$0678
C.	9(4)	0678	\$\$\$\$\$	\$678
D.	9(4)V99	123456	9(4).99	1234.56
E.	9(4)V99	123456	\$9(4).99	\$1234.56
F.	9(4)V99	123456	\$9,999.99	\$1,234.56
G.	9(4)	0008	\$\$,\$\$\$	\$8
H.	9(4)V9	12345	9(4)	1234
I.	9(4)V9	12345	9(4).99	1234.50
J.	9(4)	0012	\$ZZZ9	\$ 12

- NOTES: 1) DECIMAL ALIGNMENT IS ALWAYS MAINTAINED IN ANY NUMERIC MOVE.
- 2) MULTIPLE DOLLAR SIGNS PROVIDE A FLOATING DOLLAR SIGN.
- 3) A "Z" SUPPRESSES ZERO AND CAN PROVIDE A FIXED DOLLAR SIGN.

STOP RUN STATEMENT

STOP RUN TERMINATES PROGRAM EXECUTION.

CONTROL IS RETURNED TO TRSDOS.

AT LEAST ONE STOP RUN MUST APPEAR IN EVERY
COBOL PROGRAM.

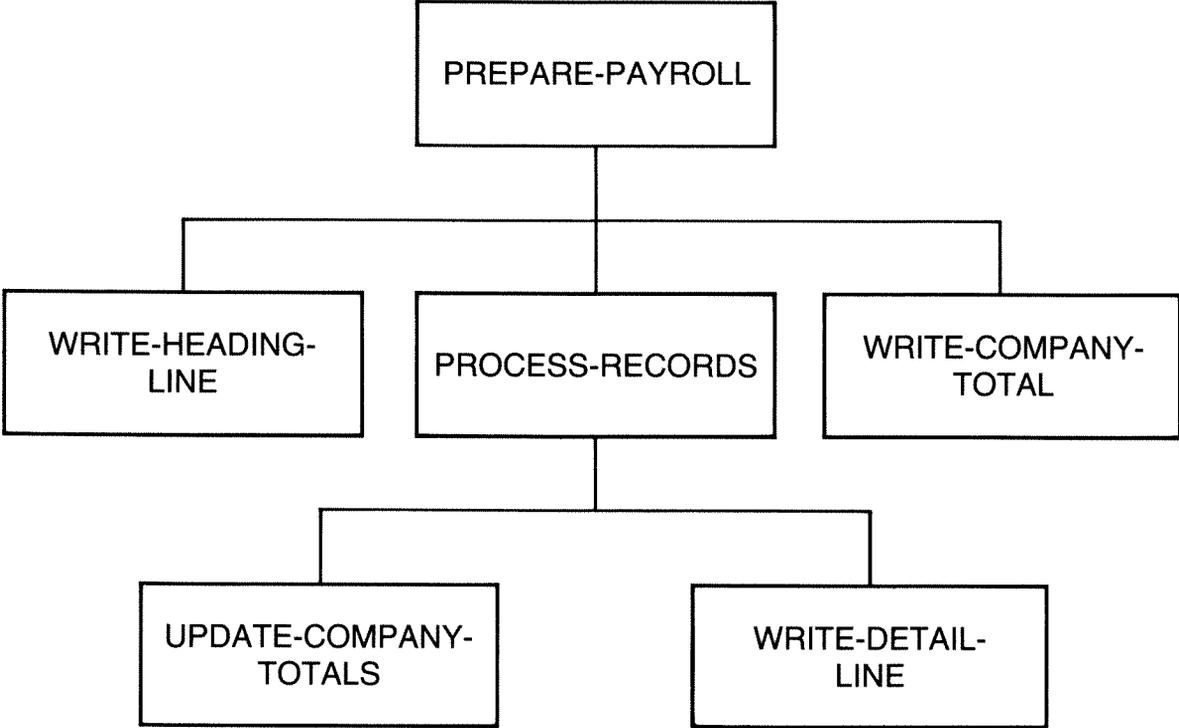
STOP RUN DOES NOT HAVE TO BE THE LAST LINE IN
THE PROGRAM.

PAYROLL PROBLEM

DEVELOP A COBOL PROGRAM TO PROCESS A FILE OF EMPLOYEE RECORDS AND PRODUCE A PAYROLL REPORT. INCOMING RECORDS CONTAIN THE FOLLOWING FIELDS: NAME, REGULAR HOURS WORKED, OVERTIME HOURS WORKED, AND HOURLY RATE. YOUR PROGRAM IS TO ACCOMPLISH ALL OF THE FOLLOWING:

- 1) PRINT AN APPROPRIATE HEADING LINE.
- 2) CALCULATE REGULAR PAY, OVERTIME PAY, AND TOTAL PAY FOR EACH EMPLOYEE.
- 3) MAINTAIN COMPANY TOTALS FOR THESE ITEMS AND PRINT AN APPROPRIATE TOTAL LINE AT THE END OF PROCESSING.

FUNCTIONAL ORGANIZATION



NOTE:
THE BOXES IN A HIERARCHY CHART CORRESPOND ONE-TO-ONE WITH THE PARAGRAPHS IN A PROGRAM.

PAYROLL PROGRAM

```

000100 IDENTIFICATION DIVISION.
000110 PROGRAM-ID.    PAYROLL.
000120 AUTHOR.       R GRAUER.
000130
000140 ENVIRONMENT DIVISION.
000150 CONFIGURATION SECTION.
000160 SOURCE-COMPUTER. TRS-80.
000170 OBJECT-COMPUTER. TRS-80.
000180
000190 INPUT-OUTPUT SECTION.
000200 FILE-CONTROL.
000210     SELECT EMPLOYEE-FILE
000220         ASSIGN TO INPUT "PAYROLL/DAT".
000230     SELECT PRINT-FILE
000240         ASSIGN TO PRINT "PAYROLL/TXT".
000250
000260 DATA DIVISION.
000270 FILE SECTION.
000280 FD EMPLOYEE-FILE
000290     LABEL RECORDS ARE OMITTED
000300     RECORD CONTAINS 80 CHARACTERS
000310     DATA RECORD IS EMPLOYEE-RECORD.
000320 01 EMPLOYEE-RECORD.
000330     05 EMP-NAME.
000340         10 EMP-LAST-NAME      PIC X(15).
000350         10 EMP-FIRST-NAME     PIC X(10).
000360     05 EMP-HOURS-WORKED.
000370         10 EMP-REG-HOURS      PIC 99.
000380         10 EMP-OVERTIME-HOURS PIC 99.
000390     05 EMP-RATE               PIC 99V99.
000400     05 FILLER                 PIC X(47).
000410
000420 FD PRINT-FILE
000430     LABEL RECORDS ARE STANDARD
000440     RECORD CONTAINS 132 CHARACTERS
000450     DATA RECORD IS PRINT-LINE.
000460 01 PRINT-LINE              PIC X(132).
000470
000480 WORKING-STORAGE SECTION.
000490 77 WS-DATA-REMAINS-SWITCH  PIC X(3)    VALUE SPACES.
000500
000510 01 IND-COMPUTATIONS.
000520     05 IND-REGULAR-PAY       PIC 9(4)V99.
000530     05 IND-OVERTIME-PAY     PIC 9(4)V99.
000540     05 IND-GROSS-PAY        PIC 9(4)V99.
000550
000560 01 COMPANY-TOTALS.
000570     05 CO-REGULAR-PAY       PIC 9(6)V99  VALUE ZEROS.
000580     05 CO-OVERTIME-PAY     PIC 9(6)V99  VALUE ZEROS.
000590     05 CO-GROSS-PAY        PIC 9(6)V99  VALUE ZEROS.
000600
000610 01 HEADING-LINE.
000620     05 FILLER                PIC X(8)    VALUE SPACES.
000630     05 FILLER                PIC X(4)    VALUE "NAME".
000640     05 FILLER                PIC X(9)    VALUE SPACES.
000650     05 FILLER                PIC X(4)    VALUE "RATE".
000660     05 FILLER                PIC X(4)    VALUE SPACES.
000670     05 FILLER                PIC X(9)    VALUE "REG HOURS".
000680     05 FILLER                PIC X(4)    VALUE SPACES.
000690     05 FILLER                PIC X(9)    VALUE "O/T HOURS".
000700     05 FILLER                PIC X(4)    VALUE SPACES.
000710     05 FILLER                PIC X(7)    VALUE "REG PAY".
000720     05 FILLER                PIC X(4)    VALUE SPACES.
000730     05 FILLER                PIC X(7)    VALUE "O/T PAY".
000740     05 FILLER                PIC X(4)    VALUE SPACES.
000750     05 FILLER                PIC X(9)    VALUE "GROSS PAY".
000760     05 FILLER                PIC X(46)   VALUE SPACES.
000770
000780 01 DASHED-LINE.
000790     05 ROW-OF-DASHES        PIC X(90)   VALUE ALL "-".
000800     05 FILLER                PIC X(42)   VALUE SPACES.
000810
000820 01 DETAIL-LINE.
000830     05 FILLER                PIC X(2).
000840     05 DET-LAST-NAME        PIC X(15).
000850     05 FILLER                PIC X(2).
000860     05 DET-RATE              PIC $$$99.
000870     05 FILLER                PIC X(8).
000880     05 DET-REG-HOURS        PIC Z9.
000890     05 FILLER                PIC X(10).

```

Input File as it exists on disk

Output File as it exists on disk

Only elementary items have picture clause

Implied decimal point

A printer typically contains 132 print positions

Company totals are initialized to zero

Heading line established through VALUE clauses

Edit picture

PAYROLL PROGRAM (CONTINUED)

```

000900 05 DET-OVERTIME-HOURS PIC Z9.
000910 05 FILLER PIC X(6).
000920 05 DET-REGULAR-PAY PIC $Z,ZZ9.99.
000930 05 FILLER PIC X(3).
000940 05 DET-OVERTIME-PAY PIC $Z,ZZ9.99.
000950 05 FILLER PIC X(2).
000960 05 DET-GROSS-PAY PIC $Z,ZZ9.99.
000970 05 FILLER PIC X(47).
000980
000990 01 TOTAL-LINE.
001000 05 FILLER PIC X(6) VALUE SPACES.
001010 05 FILLER PIC X(6) VALUE "TOTALS".
001020 05 FILLER PIC X(41) VALUE SPACES.
001030 05 TOTAL-REGULAR-PAY PIC $Z,ZZ9.99.
001040 05 FILLER PIC X(3) VALUE SPACES.
001050 05 TOTAL-OVERTIME-PAY PIC $Z,ZZ9.99.
001060 05 FILLER PIC X(2) VALUE SPACES.
001070 05 TOTAL-GROSS-PAY PIC $Z,ZZ9.99.
001080 05 FILLER PIC X(47) VALUE SPACES.
001090
001100 PROCEDURE DIVISION.
001110 PREPARE-PAYROLL.
001120 OPEN INPUT EMPLOYEE-FILE
001130 OUTPUT PRINT-FILE.
001140 PERFORM WRITE-HEADING-LINE.
001150 READ EMPLOYEE-FILE
001160 AT END MOVE "NO" TO WS-DATA-REMAINS-SWITCH.
001170 PERFORM PROCESS-RECORDS
001180 UNTIL WS-DATA-REMAINS-SWITCH = "NO".
001190 PERFORM WRITE-COMPANY-TOTALS.
001200 CLOSE EMPLOYEE-FILE
001210 PRINT-FILE.
001220 STOP RUN.
001230
001240 WRITE-HEADING-LINE.
001250 WRITE PRINT-LINE FROM HEADING-LINE
001260 AFTER ADVANCING PAGE
001270 WRITE PRINT-LINE FROM DASHED-LINE
001280 AFTER ADVANCING 1 LINE.
001290
001300 PROCESS-RECORDS.
001310 MULTIPLY EMP-REG-HOURS BY EMP-RATE GIVING IND-REGULAR-PAY.
001320 COMPUTE IND-OVERTIME-PAY
001330 = EMP-OVERTIME-HOURS * EMP-RATE * 1.5.
001340 ADD IND-REGULAR-PAY IND-OVERTIME-PAY GIVING IND-GROSS-PAY.
001350
001360 PERFORM UPDATE-COMPANY-TOTALS.
001370
001380 PERFORM WRITE-DETAIL-LINE.
001390
001400 READ EMPLOYEE-FILE
001410 AT END MOVE "NO" TO WS-DATA-REMAINS-SWITCH.
001420
001430 UPDATE-COMPANY-TOTALS.
001440 ADD IND-REGULAR-PAY TO CO-REGULAR-PAY.
001450 ADD IND-OVERTIME-PAY TO CO-OVERTIME-PAY.
001460 ADD IND-GROSS-PAY TO CO-GROSS-PAY.
001470
001480 WRITE-DETAIL-LINE.
001490 MOVE SPACES TO DETAIL-LINE.
001500 MOVE EMP-LAST-NAME TO DET-LAST-NAME.
001510 MOVE EMP-RATE TO DET-RATE.
001520 MOVE EMP-REG-HOURS TO DET-REG-HOURS.
001530 MOVE EMP-OVERTIME-HOURS TO DET-OVERTIME-HOURS.
001540 MOVE IND-REGULAR-PAY TO DET-REGULAR-PAY.
001550 MOVE IND-OVERTIME-PAY TO DET-OVERTIME-PAY.
001560 MOVE IND-GROSS-PAY TO DET-GROSS-PAY.
001570 WRITE PRINT-LINE FROM DETAIL-LINE
001580 AFTER ADVANCING 2 LINES.
001590
001600 WRITE-COMPANY-TOTALS.
001610 WRITE PRINT-LINE FROM DASHED-LINE
001620 AFTER ADVANCING 1 LINE.
001630 MOVE CO-REGULAR-PAY TO TOTAL-REGULAR-PAY.
001640 MOVE CO-OVERTIME-PAY TO TOTAL-OVERTIME-PAY.
001650 MOVE CO-GROSS-PAY TO TOTAL-GROSS-PAY.
001660 MOVE TOTAL-LINE TO PRINT-LINE.
001670 WRITE PRINT-LINE
001680 AFTER ADVANCING 2 LINES.

```

Use of editing

Initial read

Causes output to begin on a new page

Arithmetic Statements

Last statement of performed routine is another read

Increments company totals

Builds detail line

TRUE/FALSE REVIEW

- 1) **PROGRAM-ID** IS THE ONLY REQUIRED PARAGRAPH IN THE **IDENTIFICATION DIVISION**.
- 2) BOTH **GIVING** AND **TO** MAY APPEAR IN THE SAME **ADD** STATEMENT.
- 3) EVERY ELEMENTARY ITEM HAS A **PICTURE** CLAUSE.
- 4) A DATA NAME AT THE 10 LEVEL MAY OR MAY NOT HAVE A **PICTURE** CLAUSE.
- 5) **PIC 9(3)** AND **PICTURE IS 999** ARE EQUIVALENT ENTRIES.
- 6) THE **IF** STATEMENT MUST CONTAIN AN **ELSE** CLAUSE.
- 7) ONLY ONE STATEMENT IS EXECUTED WHEN AN **IF** STATEMENT IS TRUE.
- 8) **STOP RUN** IS THE LAST LINE OF EVERY COBOL PROGRAM.
- 9) A FILENAME MAY NOT APPEAR IN MORE THAN TWO STATEMENTS IN THE SAME PROGRAM.
- 10) IN A **COMPUTE** STATEMENT WITH NO PARENTHESES, MULTIPLICATION IS ALWAYS DONE BEFORE ADDITION.

CHAPTER FIVE: DEBUGGING

COMPILATION ERRORS

COMMON COMPILATION ERRORS

OMITTED PERIOD OR HYPHEN

INADVERTENT USE OF RESERVED WORD

MISSPELLED RESERVED WORD

CONFLICTING **RECORD CONTAINS** CLAUSE

CONFLICTING **PICTURE** AND **VALUE** CLAUSE

NON-UNIQUE DATA NAMES

READING A RECORD OR WRITING A FILE

INVALID **PICTURE** FOR NUMERIC ENTRY

EXCEEDING COLUMN 72

OMITTED SPACE BEFORE OR AFTER ARITHMETIC
OPERATOR

EXECUTION ERRORS

COMMON EXECUTION ERRORS

MISSING OR EXTRA PERIOD

MISSING OR EXTRA PARENTHESES

IMPROPER USE OF PRIMING **READ**

FAILURE TO INITIALIZE A COUNTER

RECEIVING FIELD TOO SMALL TO ACCOMMODATE
SENDING FIELD

INCONSISTENT DECIMAL IN CALCULATED AND
PRINTED FIELDS

COMPILATION ERRORS:

- OCCUR DURING THE TRANSLATION OF COBOL TO MACHINE LANGUAGE
- ARE THE RESULT OF AN ERROR IN SYNTAX: FOR EXAMPLE, A MISSPELLED WORD, A MISSING PERIOD, AND SO ON

AN ERROR IN ONE STATEMENT CAN CAUSE ERRORS IN OTHER SEEMINGLY CORRECT STATEMENTS. FOR EXAMPLE, MISSPELLING “**ENVIRONMENT**” CAUSES **SELECT STATEMENTS** TO BE IN ERROR, WHICH IN TURN INVALIDATES **FD’S**, RESULTING IN INVALID PROCEDURE REFERENCES TO ALL **FILENAMES** AND ASSOCIATED **DATA NAMES**.

ALL COMPILATION ERRORS SHOULD BE REMOVED BEFORE EXECUTION IS ATTEMPTED.

MOST COMPILATION ERRORS ARE NOTED DIRECTLY UNDER THE STATEMENT IN QUESTION.

THESE DIAGNOSTIC MESSAGES USUALLY OCCUR IN PAIRS — THE FIRST INDICATES THE NATURE OF THE PROBLEM; THE SECOND SHOWS THE POINT WHERE COMPILATION CONTINUED. FOR EXAMPLE:

05	CO REG-PAY	PIC 9(6)	VALUE ZERO.
	\$	\$	

****1) SYNTAX

****2) SCAN RESUME

COMMON COMPILATION ERRORS:

OMITTED PERIOD OR HYPHEN

INADVERTENT USE OF RESERVED WORD

MISSPELLED RESERVED WORD

CONFLICTING **RECORD CONTAINS** CLAUSE

CONFLICTING **PICTURE** AND **VALUE** CLAUSE

NON-UNIQUE DATA NAMES

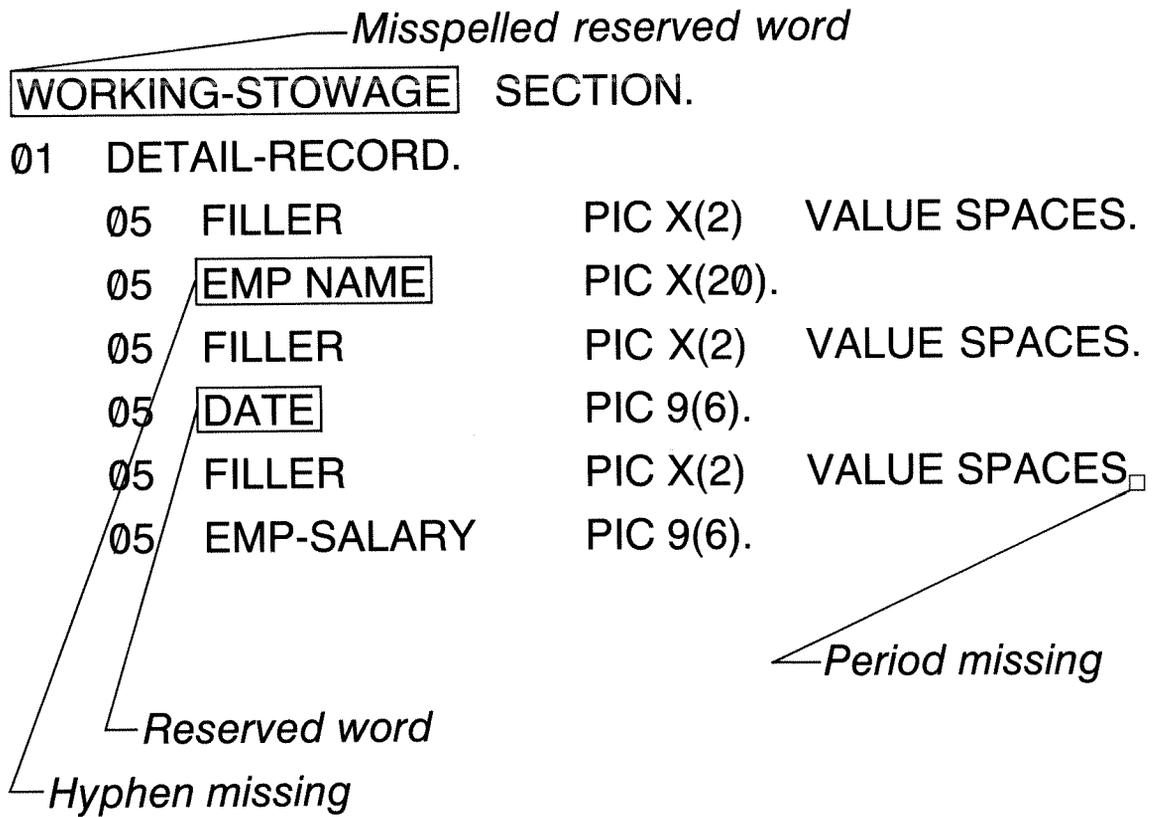
READING A RECORD OR WRITING A FILE

INVALID **PICTURE** FOR NUMERIC ENTRY

EXCEEDING COLUMN 72

OMITTED SPACE BEFORE OR AFTER ARITHMETIC
OPERATOR

OMITTED PERIOD OR HYPHEN, INADVERTENT USE OF
RESERVED WORD, MISSPELLED RESERVED WORD



CONFLICTING **RECORD CONTAINS** CLAUSE

FD PRINT-FILE

LABEL RECORDS ARE OMITTED

RECORD CONTAINS 132 CHARACTERS

DATA RECORD IS PRINT-LINE.

01 PRINT-LINE.

05 FILLER PIC X(10).

05 PR-NAME PIC X(20).

05 FILLER PIC X(2).

05 PR-SALARY PIC \$\$,\$\$9.99.

05 FILLER PIC X(90).

9 Positions

ALL **PICTURE** CLAUSES MUST SUM TO 132. THIS CAN BE TRICKY BECAUSE OF THE PRESENCE OF EDITING CHARACTERS (SAMPLE ENTRIES TOTAL 131).

THIS ERROR APPEARS AS "FILE RECORD SIZE ERROR" AT END OF LISTING.

CONFLICTING **PICTURE** AND **VALUE** CLAUSE

NUMERIC PICTURES REQUIRE NUMERIC VALUES;
NON-NUMERIC PICTURES NEED NON-NUMERIC VALUES.
MOREOVER, THE NUMBER OF CHARACTERS IN A **VALUE**
CLAUSE MUST BE CONSISTENT WITH THE
CORRESPONDING **PICTURE**.

CORRECT ENTRIES

05	NUMERIC-ENTRY	PIC 9(3)	VALUE 123.
05	NON-NUMERIC-ENTRY	PIC X(3)	VALUE "ABC".
05	GOOD-LENGTH	PIC X(4)	VALUE "WXYZ".

INVALID ENTRIES

05	NUMERIC-ENTRY	PIC 9(3)	VALUE "123".
05	NON-NUMERIC-ENTRY	PIC X(3)	VALUE 123.
05	BAD-LENGTH	PIC X(3)	VALUE "ABCD".

NON-UNIQUE DATA NAMES

01 EMPLOYEE-RECORD.

05 NAME

05 ADDRESS

.

.

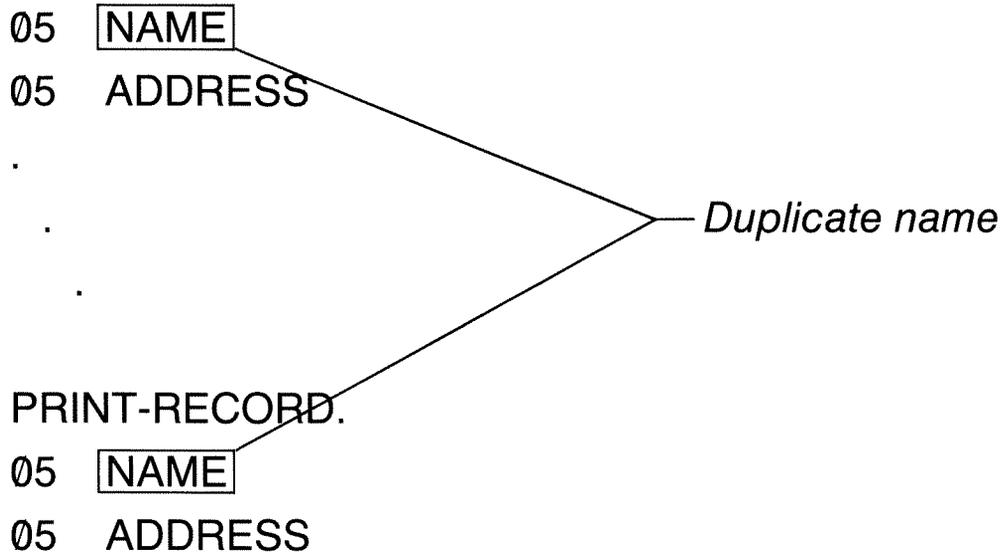
.

01 PRINT-RECORD.

05 NAME

05 ADDRESS

Duplicate name

A diagram illustrating a naming conflict. Two records are shown: 'EMPLOYEE-RECORD.' and 'PRINT-RECORD.'. Each record has a 'NAME' field (indicated by a box) and an 'ADDRESS' field. Lines from the 'NAME' field of the 'EMPLOYEE-RECORD.' and the 'NAME' field of the 'PRINT-RECORD.' converge towards a label on the right that says 'Duplicate name'.

THIS PROBLEM IS NEATLY AVOIDED BY USING A TWO- OR THREE-CHARACTER PREFIX FOR EACH RECORD. (FOR EXAMPLE: EMP-NAME, EMP-ADDRESS, PR-NAME, PR-ADDRESS, ETC.)

READING A RECORD OR WRITING A FILE

FD EMPLOYEE-FILE

.

.

.

DATA RECORD IS EMPLOYEE-RECORD.

FD PRINT-FILE

.

.

.

DATA RECORD IS PRINT-RECORD.

CORRECT ENTRIES:

READ EMPLOYEE-FILE . . .

WRITE EMPLOYEE-RECORD . . .

INVALID ENTRIES:

READ EMPLOYEE-RECORD . . .

WRITE EMPLOYEE-FILE . . .

INVALID PICTURE FOR NUMERIC ENTRY

05	GROSS-PAY	PIC 9(4)V99.
05	PRINT-GROSS-PAY	PIC \$9,999.99.

CORRECT ENTRIES:

COMPUTE GROSS-PAY = HOURS * RATE.

MOVE GROSS-PAY TO PRINT-GROSS-PAY.

INVALID ENTRY:

COMPUTE PRINT-GROSS-PAY = HOURS * RATE.

DATA NAMES APPEARING IN A COMPUTE STATEMENT MUST BE DEFINED WITH NUMERIC PICTURES. THEY MAY SUBSEQUENTLY BE MOVED TO EDITED FIELDS FOR PRINTING.

EXCEEDING COLUMN 72

WORKING-STORAGE SECTION.

05 TOTAL-SALARIES PIC 9(6).

PROCEDURE DIVISION.

ADD THIS-SALARY...TO TOTAL-SALARIES.
\$ UNDEFINED

Column 72

TOTAL-SALARI IS FLAGGED AS AN UNDEFINED DATA NAME BECAUSE THE COMPILER INTERPRETS ONLY COLUMNS 1-72.

OMITTED SPACE BEFORE OR AFTER
ARITHMETIC OPERATOR

THE ARITHMETIC OPERATORS *, /, +, AND -
ALL REQUIRE A SPACE BEFORE AND AFTER THEM.

CORRECT ENTRY:

COMPUTE OVERTIME-PAY = HOURS-OVERTIME *
HOURLY-RATE * 1.5.

INCORRECT ENTRY

COMPUTE OVERTIME-PAY = HOURS-OVERTIME*
HOURLY-RATE*1.5.

LISTING (CONTINUED)

TRS-80 Model II COBOL (RM/COBOL 1.3B)
SOURCE FILE: PAYCOMP

4/18/81 12.05.04 PAGE 6
OPTION LIST: T L=1

ADDRESS SIZE DEBUG ORDER TYPE NAM

READ ONLY BYTE SIZE = >0200

READ/WRITE BYTE SIZE = >0398

OVERLAY SEGMENT BYTE SIZE = >0000

TOTAL BYTE SIZE = >0598

14 ERRORS
11 WARNINGS

————— Compilation summary

EXECUTION ERRORS:

THE COMPUTER WAS ABLE TO EXECUTE THE ENTIRE PROGRAM, BUT THE CALCULATED RESULTS ARE DIFFERENT FROM WHAT THE PROGRAMMER INTENDED,

OR

THE COMPUTER IS UNABLE TO EXECUTE A PARTICULAR INSTRUCTION AND COMES TO A PREMATURE END OF JOB: FOR EXAMPLE, TRYING TO READ A FILE THAT ISN'T THERE.

COMMON EXECUTION ERRORS:

MISSING OR EXTRA PERIOD

MISSING OR EXTRA PARENTHESES

IMPROPER USE OF PRIMING **READ**

FAILURE TO INITIALIZE A COUNTER

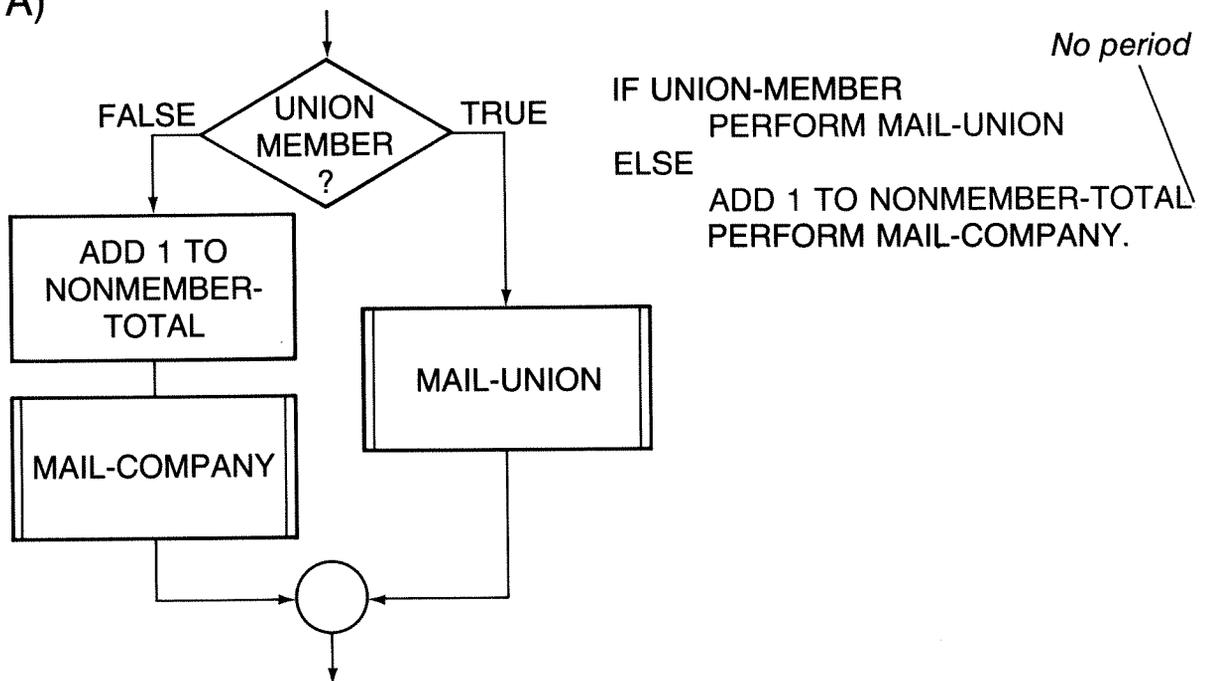
RECEIVING FIELD TOO SMALL TO ACCOMMODATE
SENDING FIELD

INCONSISTENT DECIMAL IN CALCULATED AND
PRINTED FIELDS

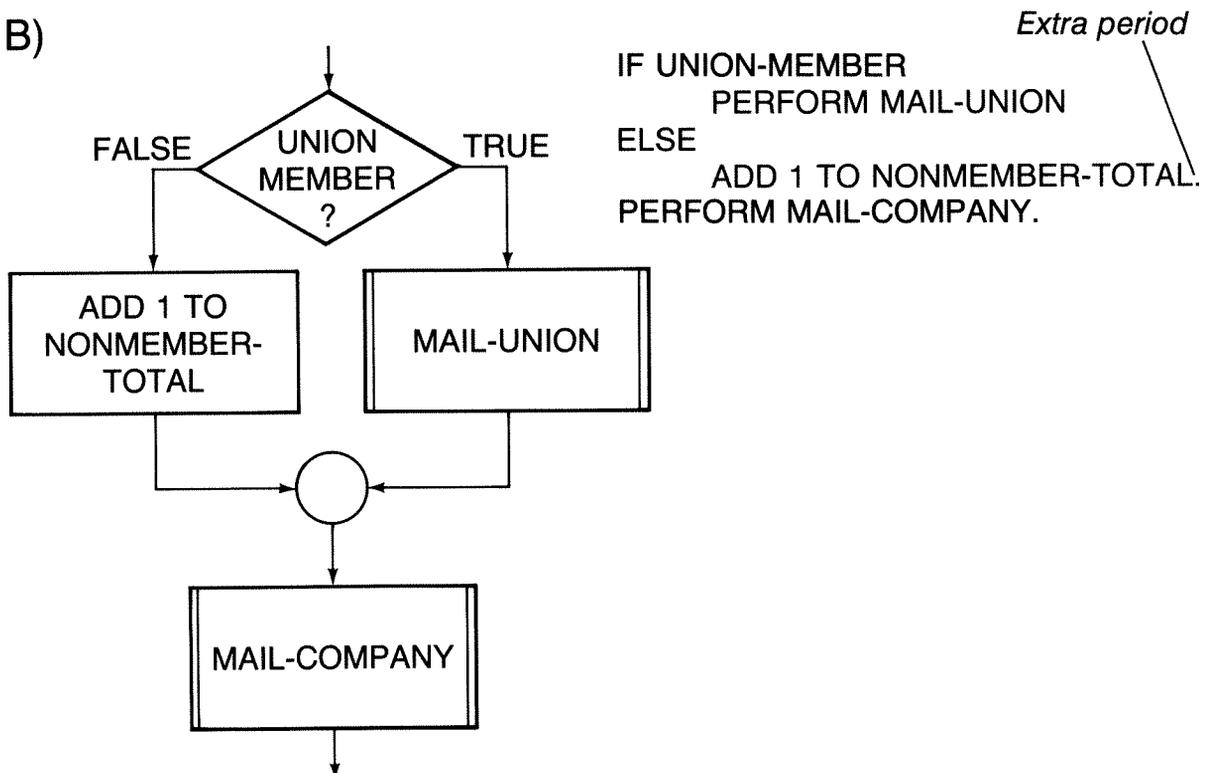
MISSING OR EXTRA PERIOD

THE PERIOD HAS A SIGNIFICANT EFFECT. CONSIDER THE DIFFERENCE BETWEEN

A)



B)



MISSING OR EXTRA PARENTHESES:

PARENTHESES CLARIFY, AND SOMETIMES ALTER, THE
NORMAL HIERARCHY OF OPERATIONS:

MULTIPLICATION OR DIVISION, ADDITION OR
SUBTRACTION, AND LEFT TO RIGHT IF A TIE.

THUS:

$$\text{COMPUTE } X = A + B * C$$

IS NOT THE SAME AS:

$$\text{COMPUTE } X = (A + B) * C$$

IMPROPER USE OF PRIMING **READ**:

INCORRECT:

PERFORM PROCESS-RECORDS
UNTIL EOF-SWITCH = "YES".

.
.
.

PROCESS-RECORDS.

READ INPUT-FILE
AT END MOVE "YES" TO EOF-SWITCH.

.
.
.

CORRECT:

READ INPUT-FILE
AT END MOVE "YES" TO EOF-SWITCH.
PERFORM PROCESS-RECORDS
UNTIL EOF-SWITCH = "YES".

.
.
.

PROCESS-RECORDS.

.
.
.

READ INPUT-FILE
AT END MOVE "YES" TO EOF-SWITCH.

FAILURE TO INITIALIZE A COUNTER

COUNTERS CAN BE INITIALIZED IN A **VALUE** CLAUSE OR WITH A **MOVE** STATEMENT.

WHEN A COUNTER IS NOT PROPERLY INITIALIZED, THE PROGRAMMER CANNOT BE SURE OF ITS INITIAL VALUE.

FAILURE TO INITIALIZE A COUNTER CAN CAUSE THE COUNTER TO CONTAIN AN UNEXPECTED VALUE.

RECEIVING FIELD TOO SMALL TO ACCOMMODATE
SENDING FIELD:

05	TOTAL-GROSS-PAY	PIC 9(5).
05	PRINT-TOTAL-PAY	PIC \$,\$,\$9.

MOVE TOTAL-GROSS-PAY TO PRINT-TOTAL-PAY

TOTAL-GROSS-PAY CAN CONTAIN NUMBERS AS LARGE AS 99,999. PRINT-TOTAL-PAY CANNOT PRINT ANYTHING LARGER THAN \$9,999.

IN AN EXTREME CASE, IF TOTAL-GROSS-PAY HAD A VALUE OF 10,000, PRINT-TOTAL-PAY WOULD APPEAR AS \$0.

INCONSISTENT DECIMAL IN CALCULATED AND PRINTED
FIELDS:

05	GROSS-PAY	PIC 9(4).
05	PRINT-GROSS-PAY	PIC \$9,999.99.

.

.

.

COMPUTE GROSS-PAY = HOURS * RATE.
MOVE GROSS-PAY TO PRINT-GROSS-PAY.

PRINT-GROSS-PAY WILL NEVER HAVE ANY CENTS
BECAUSE GROSS-PAY WAS DEFINED WITHOUT A
DECIMAL POINT.

TRUE/FALSE REVIEW

- 1) DATA NAMES MAY CONTAIN BLANKS.
- 2) IF A PROGRAM COMPILES CORRECTLY, IT MUST EXECUTE CORRECTLY.
- 3) IN COBOL, THE **READ** STATEMENT REQUIRES A FILE NAME.
- 4) COMPILATION STOPS AS SOON AS ONE ERROR IS FOUND.
- 5) ALL COMPILATION ERROR MESSAGES APPEAR DIRECTLY UNDER THE STATEMENT IN ERROR.
- 6) AN ERROR IN ONE STATEMENT MAY CAUSE ERRORS IN OTHER APPARENTLY CORRECT STATEMENTS.
- 7) IT IS IMPOSSIBLE TO EXECUTE A PROGRAM WITH COMPILATION ERRORS.
- 8) THE COMPILER WILL FLAG ALL LOGIC ERRORS.
- 9) DATA NAMES IN A COMPUTE STATEMENT MUST BE DEFINED WITH NUMERIC **PICTURES**.
- 10) ONE COMPILE MAY PRODUCE FILES WITH EXTENSIONS OF **LST** AND **COB**.

CHAPTER SIX: ADVANCED FEATURES

IF STATEMENT

SIGNIFICANCE OF A PERIOD

COMPOUND **IF** STATEMENTS

CONDITION NAMES (88-LEVEL ENTRIES)

NESTED **IFS**

PERFORM

PARAGRAPHS VS SECTIONS

THRU

VARYING/UNTIL

DISPLAY

ACCEPT

READ INTO

WRITE FROM

ROUNDED AND SIZE ERROR OPTIONS

DUPLICATE DATA NAMES

QUALIFICATION

MOVE CORRESPONDING

INSPECT

TABLES

INITIALIZATION

TABLE LOOKUPS

CAR BILLING PROBLEM

IF STATEMENT

IF condition statement-1 [ELSE] statement-2

NOTES:

- 1) THE CONDITION MAY BE RELATIONAL, (=, >, ETC.), COMPOUND (AND, OR), OR AN 88-LEVEL ENTRY.
- 2) EITHER STATEMENT-1 OR STATEMENT-2 MAY BE ANOTHER **IF**, GIVING RISE TO A NESTED **IF**.
- 3) STATEMENT-1 IS TERMINATED BY A PERIOD OR AN **ELSE**.

SIGNIFICANCE OF A PERIOD

(executed collectively or not at all)

IF A = B

MOVE C TO D

ADD 1 TO X

COMPUTE Z = (2 * Y) + 3.

IF LOCATION-CODE = 100

ADD 1 TO LOCATION-COUNTER

PERFORM PROCESS-LOCATION-TOTALS

ELSE

ADD 1 TO OUT-OF-STATE-TOTALS

PERFORM INVALID-LOCATION-ROUTINE

PERFORM WRITE-ERROR-MESSAGE.

PERFORM READ-NEXT-RECORD.

(executed regardless of location-code)

COMPOUND IF STATEMENTS

IF A > B OR C = D AND E = F . . .

THE STATEMENT IS TRUE IF:

1) A > B

OR 2) C = D AND E = F

THIS IS BECAUSE A HIERARCHY OF OPERATIONS EVALUATES **AND** BEFORE **OR**. HOWEVER, PARENTHESES CAN:

1) CLARIFY: **IF A > B OR (C = D AND E = F) . . .**

OR 2) ALTER: **IF (A > B OR C = D) AND E = F . . .**

THE NORMAL SEQUENCE OF LOGICAL OPERATIONS:

1. ARITHMETIC EXPRESSIONS
2. RELATIONAL OPERATORS
3. NOT CONDITION
4. AND
5. OR

CONDITION NAMES
(88 LEVEL ENTRIES)

05	YEAR-CODE	PIC 9.
88	FRESHMAN	VALUE 1.
88	SOPHOMORE	VALUE 2.
88	JUNIOR	VALUE 3.
88	SENIOR	VALUE 4.
88	UPPERCLASSMAN	VALUES ARE 3, 4.
88	VALID-CODES	VALUES ARE 1 THROUGH 4.

EQUIVALENT ENTRIES:

IF FRESHMAN . . .

IF YEAR-CODE = 1 . . .

IF UPPERCLASSMAN . . .

IF YEAR-CODE = 3 OR YEAR-CODE = 4 . . .

IF VALID-CODES . . .

IF YEAR-CODE > 0 AND YEAR-CODE < 5 . . .

NESTED IFS

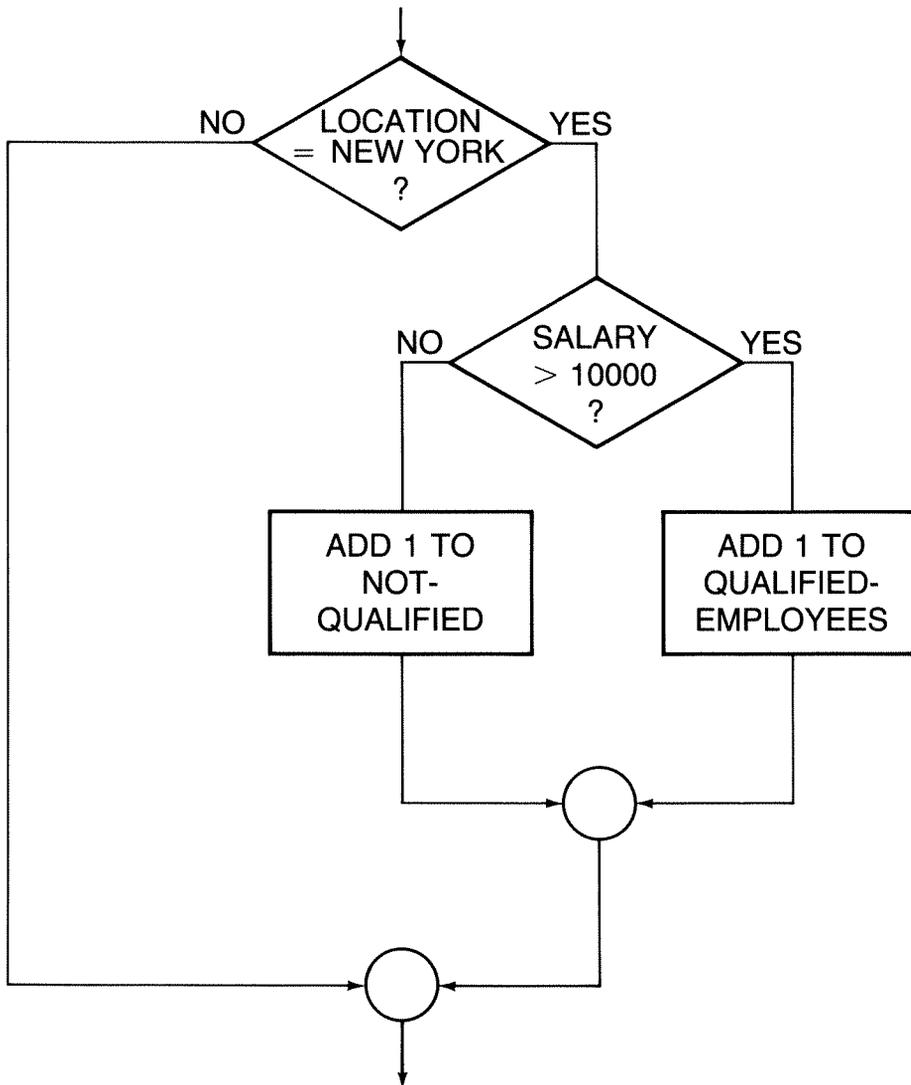
IF condition { statement-1
NEXT SENTENCE } [ELSE { statement-2
NEXT SENTENCE }]

A NESTED **IF** IS DEFINED AS TWO OR MORE **IFS** WITHIN A SENTENCE. IT RESULTS WHEN EITHER STATEMENT-1 OR STATEMENT-2 IS ITSELF ANOTHER **IF** STATEMENT.

THE **ELSE** CLAUSE IS ASSOCIATED WITH THE CLOSEST PREVIOUS UNPAIRED **IF**. NESTED **IFS** SHOULD ALWAYS BE INDENTED IN A MANNER CONSISTENT WITH COMPILER INTERPRETATION. THIS IS NOT A COBOL REQUIREMENT PER SE, BUT RATHER A CHARACTERISTIC OF GOOD PROGRAMS.

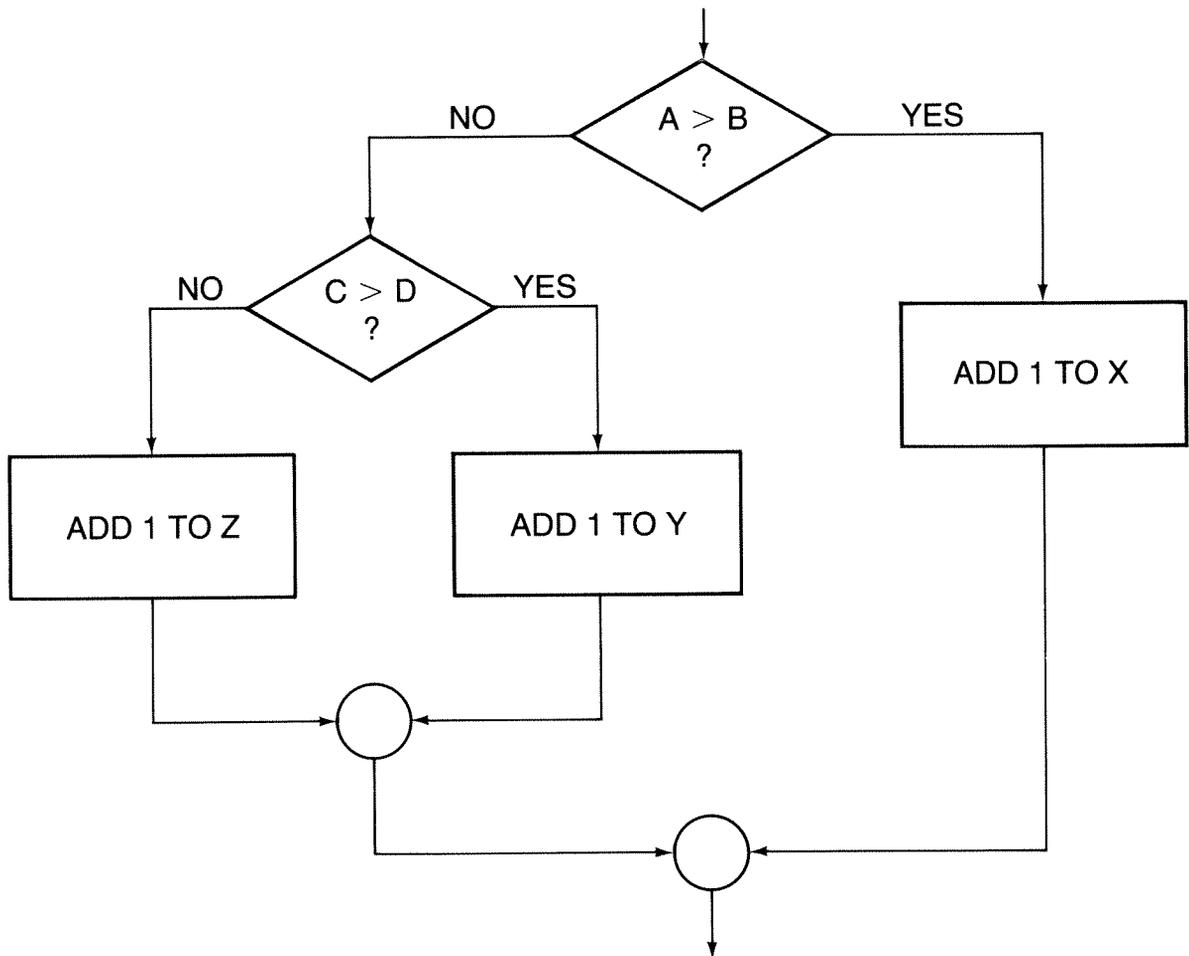
CONSIDER . . .

NESTED IF



```
IF LOCATION = "NEW YORK"  
  IF SALARY > 10000  
    ADD 1 TO QUALIFIED-EMPLOYEES  
  ELSE  
    ADD 1 TO NOT-QUALIFIED.
```

NESTED IF



```
IF A > B
    ADD 1 TO X
ELSE
    IF C > D
        ADD 1 TO Y
    ELSE
        ADD 1 TO Z.
```

PERFORM

1) PERFORM procedure-name-1 [THRU procedure-name-2]

2) PERFORM procedure-name-1 [THRU procedure-name-2]

$\left\{ \begin{array}{l} \text{identifier} \\ \text{integer} \end{array} \right\} \underline{\text{TIMES}}$

3) PERFORM procedure-name-1 [THRU procedure-name-2]

UNTIL condition

4) PERFORM procedure-name-1 [THRU procedure-name-2]

VARYING identifier-1 FROM $\left\{ \begin{array}{l} \text{literal-2} \\ \text{identifier-2} \end{array} \right\}$

BY $\left\{ \begin{array}{l} \text{literal-3} \\ \text{identifier-3} \end{array} \right\} \underline{\text{UNTIL}} condition$

THE PROCEDURE MAY BE EITHER A SECTION OR A PARAGRAPH.

A *PARAGRAPH* CONTAINS ONE OR MORE SENTENCES AND ENDS WHEN THE NEXT PARAGRAPH BEGINS.

A *SECTION* CONTAINS ONE OR MORE PARAGRAPHS AND ENDS WHEN THE NEXT SECTION BEGINS.

THE **THRU** CLAUSE IS OPTIONAL, AND EXPLICITLY DELINEATES THE RANGE OF THE **PERFORM**. THUS, ALL FORMS OF **PERFORM** CAN INVOKE MORE THAN ONE PARAGRAPH.

ANY OF THREE OPTIONS, **TIMES**, **UNTIL**, OR **VARYING/FROM/BY/UNTIL**, CAN CAUSE A PROCEDURE TO BE EXECUTED MANY TIMES.

PARAGRAPHS VERSUS SECTIONS

MOVE ZERO TO X.

PERFORM A.

STOP RUN.

A SECTION.

ADD 1 TO X.

FIRST-PAR.

ADD 1 TO X.

SECOND-PAR.

ADD 2 TO X.

*executed by **PERFORM A** above.*

B SECTION.

.

.

.

VERSUS

MOVE ZERO TO Y.

PERFORM A.

STOP RUN.

A.

ADD 1 TO Y.

*executed by **PERFORM A** above.*

FIRST-PAR.

ADD 1 TO Y.

SECOND-PAR.

ADD 1 TO Y.

B.

.

.

.

PERFORM/VARYING/UNTIL

THE **UNTIL** CONDITION IS TESTED *BEFORE* A PROCEDURE IS PERFORMED. CONSEQUENTLY, IF THE CONDITION IS SATISFIED INITIALLY, THE **PERFORM** IS NEVER INVOKED. FOR EXAMPLE:

**MOVE "YES" TO SWITCH.
PERFORM PROCESS-RECORDS
UNTIL SWITCH = "YES".**

THE **VARYING/UNTIL** OPTION INCREMENTS, TESTS, THEN BRANCHES. CONSEQUENTLY, PARAGRAPH-A IS EXECUTED TWICE RATHER THAN THREE TIMES:

**PERFORM PARAGRAPH-A
VARYING SUBSCRIPT FROM 1 BY 1
UNTIL SUBSCRIPT = 3.**

DISPLAY

THE **DISPLAY** STATEMENT SENDS OUTPUT DIRECTLY TO THE CRT. IT AVOIDS THE NECESSITY OF AN **FD** AND ASSOCIATED RECORD FORMATS

AN ABBREVIATED FORMAT IS:

$$\begin{array}{l} \underline{\text{DISPLAY}} \left\{ \begin{array}{l} \text{identifier-1} \\ \text{literal-1} \end{array} \right\} \left[\underline{\text{LINE}} \left\{ \begin{array}{l} \text{identifier-2} \\ \text{literal-2} \end{array} \right\} \right] \\ \left[\underline{\text{POSITION}} \left\{ \begin{array}{l} \text{identifier-3} \\ \text{literal-3} \end{array} \right\} \right] \\ \left[\underline{\text{REVERSE}} \right] \dots \end{array}$$

A GIVEN DISPLAY STATEMENT MAY CONTAIN MULTIPLE IDENTIFIERS, LITERALS, OR ANY COMBINATION.

EXAMPLES:

DISPLAY "SOCIAL SECURITY NUMBER = "
SOC-SEC-NUM.

DISPLAY "GOOD MORNING" LINE 12 POSITION 20
REVERSE.

ACCEPT

THE **ACCEPT** STATEMENT TAKES INPUT FROM THE KEYBOARD AND TRANSFERS IT TO THE DESIGNATED DATA NAME, WITHOUT THE NECESSITY OF AN **FD** AND ASSOCIATED RECORD FORMATS.

IT PROVIDES LINE- AND POSITION-CAPABILITY SIMILAR TO THE **DISPLAY** STATEMENT. CONSIDER:

$$\begin{array}{l} \underline{\text{ACCEPT}} \text{ identifier-1 } \left[\underline{\text{LINE}} \left\{ \begin{array}{l} \text{identifier-2} \\ \text{literal-1} \end{array} \right\} \right] \\ \underline{\text{POSITION}} \left[\left\{ \begin{array}{l} \text{identifier-3} \\ \text{literal-2} \end{array} \right\} \right] \left[\underline{\text{REVERSE}} \right] \dots \end{array}$$

WHEN AN **ACCEPT** STATEMENT IS EXECUTED, PROCESSING IS SUSPENDED UNTIL THE USER SUPPLIES A RESPONSE AND PRESSES THE **ENTER** KEY.

THE **ACCEPT** STATEMENT IS OFTEN PRECEDED BY A **DISPLAY** STATEMENT TO PROMPT THE USER

EXAMPLE:

DISPLAY "WHAT IS YOUR NAME?"

LINE 3 POSITION 8 REVERSE.

ACCEPT USER-NAME.

DATE OF EXECUTION

THE STATEMENT

ACCEPT DATE-WORK-AREA FROM DATE

OBTAINS THE DATE OF EXECUTION FROM THE RESERVED WORD **DATE**, AND STORES IT IN A PROGRAMMER-DEFINED DATA NAME (IN THIS CASE, **DATE-WORK-AREA**).

THE LATTER MUST BE IN THE FORM: **YYMMDD**. FOR EXAMPLE:

```
01  DATE-WORK-AREA.  
  
    05  TODAYS-YEAR          PIC 99.  
    05  TODAYS-MONTH        PIC 99.  
    05  TODAYS-DAY          PIC 99.
```

THE IMPORTANCE OF A CORRECT RESPONSE TO THE DATE QUESTION WHEN POWERING UP THE MACHINE IS NOW APPARENT.

CALCULATION OF AGE

$$\begin{aligned}\text{COMPUTE AGE} &= (\text{TODAYS-YEAR} - \text{BIRTH-YEAR}) \\ &+ (\text{TODAYS-MONTH} - \text{BIRTH-MONTH}) / 12\end{aligned}$$

EXAMPLE 1:

$$\text{TODAYS DATE} = 9/83$$

$$\text{BIRTH DATE} = 3/77$$

$$\begin{aligned}\text{AGE} &= (83 - 77) + (9 - 3)/12 \\ &= 6 + 6/12 \\ &= 6 \frac{1}{2}\end{aligned}$$

EXAMPLE 2:

$$\text{TODAYS DATE} = 4/83$$

$$\text{BIRTH DATE} = 10/73$$

$$\begin{aligned}\text{AGE} &= (83 - 73) + (4 - 10)/12 \\ &= 10 + (-6/12) \\ &= 9 \frac{1}{2}\end{aligned}$$

READ INTO

READ file-name [INTO identifier]

[AT END imperative-statement]

THUS:

READ EMPLOYEE-FILE INTO WS-AREA
AT END MOVE "YES" TO EOF-SWITCH.

IS EQUIVALENT TO:

READ EMPLOYEE-FILE
AT END MOVE "YES" TO EOF-SWITCH.
MOVE EMPLOYEE-RECORD TO WS-AREA.

WRITE FROM

WRITE record-name FROM identifier

$\left[\begin{array}{l} \underline{\text{BEFORE}} \\ \underline{\text{AFTER}} \end{array} \right] \text{ ADVANCING } \left\{ \begin{array}{l} \text{integer LINES} \\ \underline{\text{PAGE}} \end{array} \right\}$

THUS:

WRITE PRINT-LINE FROM HEADING-LINE
AFTER ADVANCING PAGE.

IS EQUIVALENT TO:

MOVE HEADING LINE TO PRINT-LINE.
WRITE PRINT-LINE
AFTER ADVANCING PAGE.

ROUNDED AND SIZE ERROR OPTIONS

THE COMPLETE SYNTAX OF THE **COMPUTE** VERB IS:

COMPUTE identifier-1 [ROUNDED] = arithmetic-expression

[ON SIZE ERROR imperative statement]

THE **ROUNDED** CLAUSE WILL ROUND RATHER THAN TRUNCATE THE LAST SIGNIFICANT DIGIT AFTER COMPUTATION.

THE **SIZE ERROR** CLAUSE EXECUTES THE SPECIFIED STATEMENT IF THE RESULT OF THE COMPUTATION EXCEEDS THE LARGEST VALUE FOR IDENTIFIER-1.

ROUNDED AND **SIZE ERROR** ARE ALSO PERMITTED WITH **ADD**, **SUBTRACT**, **MULTIPLY**, AND **DIVIDE**.

EXAMPLE:

COMPUTE GROSS-PAY ROUNDED = HOURS * RATE
ON SIZE ERROR PERFORM ERROR-ROUTINE.

DUPLICATE DATA NAMES

01 INPUT-RECORD.

05 NAME PIC X(25).

05 SOC-SEC-NUM PIC 9(9).

05 ADDRESS.

10 STREET PIC X(40).

10 CITY-STATE PIC X(20).

10 ZIP-CODE PIC 9(5).

duplicate name

01 OUTPUT-RECORD.

04 SOC-SEC-NUM PIC 9(9).

04 FILLER PIC X(3).

04 NAME PIC X(25).

04 FILLER PIC X(3).

04 ADDRESS

08 STREET PIC X(38).

08 CITY-STATE PIC X(18).

08 ZIP-CODE PIC 9(5).

QUALIFICATION

- THE STATEMENT BELOW IS AMBIGUOUS:

MOVE NAME TO PRT-FIELD.

- QUALIFICATION SOLVES THE PROBLEM:

MOVE NAME **OF** INPUT-RECORD TO PRT-FIELD.

OR

MOVE NAME **IN** INPUT-RECORD TO PRT-FIELD.

- QUALIFICATION MAY BE REQUIRED OVER TWO LEVELS:

MOVE STREET OF ADDRESS OF INPUT-RECORD TO . . .

NOT

MOVE STREET OF ADDRESS TO . . .

- QUALIFICATION MAY SKIP THE INTERMEDIATE LEVEL:

MOVE STREET OF INPUT-RECORD TO . . .

MOVE CORRESPONDING

- EQUIVALENT TO SEVERAL INDIVIDUAL MOVES.

MOVE CORRESPONDING INPUT-RECORD TO
PRINT-RECORD.

IS EQUIVALENT TO:

MOVE NAME OF INPUT-RECORD TO NAME OF
OUTPUT-RECORD.

MOVE SOC-SEC-NUM OF INPUT-RECORD TO
SOC-SEC-NUM OF OUTPUT-RECORD.

MOVE STREET OF INPUT-RECORD TO STREET OF
OUTPUT-RECORD.

MOVE CITY-STATE OF INPUT-RECORD TO
CITY-STATE OF OUTPUT-RECORD.

MOVE ZIP-CODE OF INPUT-RECORD TO ZIP-CODE
OF OUTPUT-RECORD.

- IN EVALUATING THE **MOVE CORRESPONDING**, THE DATA NAME ITSELF IS CRUCIAL. LEVEL NUMBER, PICTURE CLAUSE, OR POSITION WITHIN THE RECORD DO NOT MATTER.
- **CORR** IS EQUIVALENT TO **CORRESPONDING**.

INSPECT

CONVENIENT WAY OF INSERTING HYPHENS IN A
SOCIAL SECURITY NUMBER, OR SLASHES IN A DATE.

01 INPUT-RECORD.

05 IN-SOC-SEC-NUM PIC 9(9).

05 IN-DATE PIC 9(6).

01 PRINT-RECORD.

05 PR-SOC-SEC-NUM PIC 999B99B9999.

05 PR-DATE PIC 99B99B99.

MOVE IN-SOC-SEC-NUM TO PR-SOC-SEC-NUM.

INSPECT PR-SOC-SEC-NUM

REPLACING ALL " " BY "-".

MOVE IN-DATE TO PR-DATE.

INSPECT PR-DATE

REPLACING ALL " " BY "/".

TABLES

TABLES ARE DEFINED WITH AN **OCCURS** CLAUSE IN THE DATA DIVISION. THE **OCCURS** CLAUSE MAY OCCUR AT EITHER THE GROUP OR ELEMENTARY LEVEL AS WILL BE SHOWN.

WE WILL ALSO CONSIDER THE TABLE INITIALIZATION VIA **REDEFINES** AND **VALUE** CLAUSES, AND A TABLE LOOKUP PROCEDURE.

TABLE INITIALIZATION

01 MAJOR-VALUE.

05	FILLER	PIC X(14)	VALUE "1234ACCOUNTING "
05	FILLER	PIC X(14)	VALUE "1400BIOLOGY "
05	FILLER	PIC X(14)	VALUE "1976CHEMISTRY "
05	FILLER	PIC X(14)	VALUE "2100CIVIL ENG "
05	FILLER	PIC X(14)	VALUE "2458E. D. P. "
05	FILLER	PIC X(14)	VALUE "3245ECONOMICS "
05	FILLER	PIC X(14)	VALUE "3960FINANCE "
05	FILLER	PIC X(14)	VALUE "4321MANAGEMENT"
05	FILLER	PIC X(14)	VALUE "4999MARKETING "
05	FILLER	PIC X(14)	VALUE "5400STATISTICS "

01 MAJOR-TABLE REDEFINES MAJOR-VALUE.

05	MAJORS	OCCURS 10 TIMES.
10	MAJOR-CODE	PIC X(4).
10	MAJOR-NAME	PIC X(10).

TABLE INITIALIZATION

THE **OCCURS** CLAUSE ALLOCATES SPACE FOR A TABLE.

THE **VALUE** CLAUSE INITIALIZES SPECIFIC STORAGE LOCATIONS.

THE **REDEFINES** CLAUSE ASSIGNS ANOTHER NAME TO A PREVIOUSLY ALLOCATED MEMORY LOCATION. IT IS NECESSARY BECAUSE THE SAME ENTRY CANNOT CONTAIN BOTH A **VALUE** AND AN **OCCURS** CLAUSE.

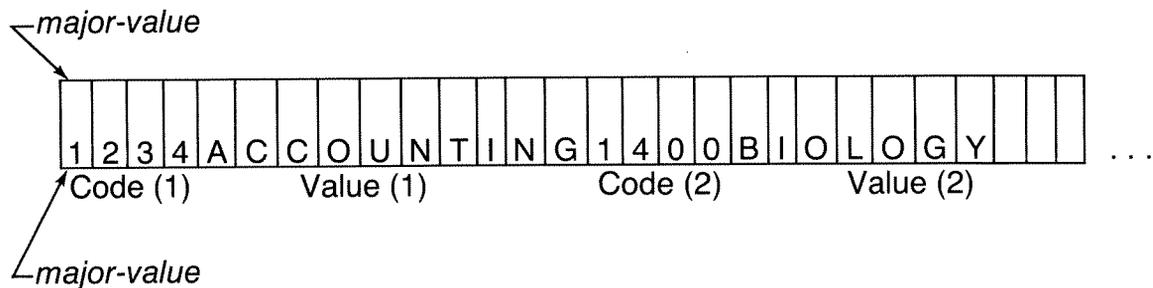


TABLE LOOKUPS

WORKING-STORAGE SECTION.

01 TABLE-PROCESSING-ELEMENTS.

05 WS-MAJOR-SUB PIC 9(4).

05 WS-FOUND-MAJOR-SWITCH PIC X(3) VALUE "NO".

PROCEDURE DIVISION.

MOVE 1 TO WS-MAJOR-SUB.

MOVE "NO" TO WS-FOUND-MAJOR-SWITCH.

PERFORM 030-FIND-MAJOR

UNTIL WS-FOUND-MAJOR-SWITCH = "YES".

030-FIND-MAJOR.

ensures that table size is not
exceeded

IF WS-MAJOR-SUB > 10

MOVE "YES" TO WS-FOUND-MAJOR-SWITCH

MOVE "UNKNOWN" TO HDG-MAJOR

ELSE

IF ST-MAJOR-CODE = MAJOR-CODE (WS-MAJOR-SUB)

MOVE "YES" TO WS-FOUND-MAJOR-SWITCH

MOVE MAJOR-NAME (WS-MAJOR-SUB) TO

HDG-MAJOR

ELSE

ADD 1 TO WS-MAJOR-SUB.

CAR BILLING PROBLEM

- INPUT** - INCOMING RECORDS CONTAIN THE FOLLOWING FIELDS: CUSTOMER NAME, CAR TYPE (C, I, OR F), CAR MAKE (BUICK, TOYOTA, ETC.), DAYS RENTED, AND MILES DRIVEN
- PROCESSING** - 1) VALIDATE CAR TYPE AND MAKE, REJECTING RECORDS WITH INVALID FIELDS
- 2) CALCULATE THE CUSTOMER BILL AS A FUNCTION OF CAR TYPE, DAYS RENTED, AND MILES DRIVEN:
- | | |
|--------------|----------------------|
| COMPACT | - \$11/DAY, 12¢/MILE |
| INTERMEDIATE | - \$12/DAY, 14¢/MILE |
| FULL SIZE | - \$13/DAY, 16¢/MILE |
- OUTPUT** - LIMIT 5 CUSTOMERS PER PAGE WITH SUITABLE HEADING LINES

CAR BILLING PROBLEM

```

000100 IDENTIFICATION DIVISION.
000110 PROGRAM-ID. CARS.
000120 AUTHOR. R GRAUER.
000130
000140 ENVIRONMENT DIVISION.
000150 CONFIGURATION SECTION.
000160 SOURCE-COMPUTER. TRS-80.
000170 OBJECT-COMPUTER. TRS-80.
000180
000190 INPUT-OUTPUT SECTION.
000200 FILE-CONTROL.
000210 SELECT CAR-RENTAL-FILE
000220 ASSIGN TO INPUT "CARS/DAT".
000230 SELECT PRINT-FILE
000240 ASSIGN TO PRINT "CARS/TXT".
000250
000260 DATA DIVISION.
000270 FILE SECTION.
000280 FD CAR-RENTAL-FILE
000290 LABEL RECORDS ARE OMITTED
000300 RECORD CONTAINS 50 CHARACTERS
000310 DATA RECORD IS CAR-RENTAL-RECORD.
000320 01 CAR-RENTAL-RECORD PIC X(50).
000330
000340 FD PRINT-FILE
000350 LABEL RECORDS ARE STANDARD
000360 RECORD CONTAINS 132 CHARACTERS
000370 DATA RECORD IS PRINT-LINE.
000380 01 PRINT-LINE PIC X(132).
000390
000400 WORKING-STORAGE SECTION.
000410 01 PROGRAM-SWITCHES.
000420 05 WS-DATA-REMAINS-SWITCH PIC X(2) VALUE SPACES.
000430 05 NO-MORE-RECORDS VALUE "NO".
000440 05 WS-CAR-MAKE-SWITCH PIC X(3) VALUE SPACES.
000450 05 88 CAR-CODE-EXPANDED VALUE "YES".
000460
000470 01 PROGRAM-SUBSCRIPTS.
000480 05 CAR-MAKE-SUBSCRIPT PIC 99.
000490
000500 01 DATE-WORK-AREA.
000510 05 TODAYS-YEAR PIC 99.
000520 05 TODAYS-MONTH PIC 99.
000530 05 TODAYS-DAY PIC 99.
000540
000550 01 WS-RECORD-IN.
000560 05 SOC-SEC-NUM PIC 9(9).
000570 05 CUST-NAME PIC X(25).
000580 05 DATE-RETURNED PIC 9(6).
000590 05 CAR-TYPE PIC X.
000600 05 88 COMPACT VALUE "C".
000610 05 88 INTERMEDIATE VALUE "I".
000620 05 88 FULL-SIZE VALUE "F".
000630 05 88 VALID-TYPE-CODE VALUES "C" "I" "F".
000640 05 CAR-MAKE PIC X(3).
000650 05 88 VALID-MAKE-CODE
000660 VALUES ARE "BUI" "CAD" "CHE" "CHR" "DAT"
000670 "FOR" "OLD" "PON" "TOY" "VOL".
000680 05 DAYS-RENTED PIC 99.
000690 05 MILES-DRIVEN PIC 9(4).
000700
000710 01 WS-PRINT-LINE.
000720 05 FILLER PIC X(4).
000730 05 SOC-SEC-NUM PIC 99B99B9999.
000740 05 FILLER PIC X(4).
000750 05 CUST-NAME PIC X(25).
000760 05 FILLER PIC X(2).
000770 05 CAR-TYPE PIC X.
000780 05 FILLER PIC X(4).
000790 05 PRINT-CAR-MAKE PIC X(10).
000800 05 FILLER PIC XX.
000810 05 DAYS-RENTED PIC Z9.
000820 05 FILLER PIC X(4).
000830 05 MILES-DRIVEN PIC ZZZ9.
000840 05 FILLER PIC X(4).
000850 05 CUSTOMER-BILL PIC $$,$$9.99.
000860 05 FILLER PIC X(46).
000861
000870 01 CAR-MAKE-VALUES.

```

Definition of 88 level entries

Use of blank as editing character

CAR BILLING (CONTINUED)

```

000880 05 FILLER PIC X(13) VALUE "BUICK"
000890 05 FILLER PIC X(13) VALUE "CADILLAC"
000900 05 FILLER PIC X(13) VALUE "CHEVROLET"
000910 05 FILLER PIC X(13) VALUE "CHRYSLER"
000920 05 FILLER PIC X(13) VALUE "DATSUN"
000930 05 FILLER PIC X(13) VALUE "FORD"
000940 05 FILLER PIC X(13) VALUE "OLDSMOBILE"
000950 05 FILLER PIC X(13) VALUE "PONTIAC"
000960 05 FILLER PIC X(13) VALUE "TOYOTA"
000970 05 FILLER PIC X(13) VALUE "VOLKSWAGON"
000980
000990 01 CAR-MAKE-TABLE REDEFINES CAR-MAKE-VALUES.
001000 05 CAR-CODE-AND-VALUE OCCURS 10 TIMES.
001010 10 CAR-CODE PIC X(3).
001020 10 EXPANDED-CAR-MAKE PIC X(10).
001030
001040 01 PAGE-AND-LINE-COUNTERS.
001050 05 WS-PAGE-COUNT PIC 9(4) VALUE ZEROS.
001060 05 WS-LINE-COUNT PIC 9(4) VALUE 6.
001070
001080 01 BILLING-CONSTANTS.
001090 05 WS-MILEAGE-RATE PIC 9V99.
001100 05 WS-DAILY-RATE PIC 99V99.
001110 05 WS-CUSTOMER-BILL PIC 9(4)V99.
001120
001130 01 HEADING-LINE-ONE.
001140 05 FILLER PIC X(75) VALUE SPACES.
001150 05 FILLER PIC X(4) VALUE "PAGE".
001160 05 WS-PAGE-PRINT PIC Z(4).
001170 05 FILLER PIC X(49) VALUE SPACES.
001180
001190 01 HEADING-LINE-TWO.
001200 05 FILLER PIC X(25) VALUE SPACES.
001210 05 TITLE-INFORMATION PIC X(27)
001220 VALUE "JESSICA'S CAR RENTAL AGENCY".
001230 05 FILLER PIC X(2) VALUE SPACES.
001240 05 TITLE-DATE.
001250 10 TITLE-MONTH PIC 99.
001260 10 FILLER PIC X VALUE "/".
001270 10 TITLE-DAY PIC 99.
001280 10 FILLER PIC X VALUE "/".
001290 10 TITLE-YEAR PIC 99.
001300 05 FILLER PIC X(70) VALUE SPACES.
001310
001320 01 HEADING-LINE-THREE.
001330 05 FILLER PIC X(5) VALUE SPACES.
001340 05 FILLER PIC X(11) VALUE " ACCT #".
001350 05 FILLER PIC X(5) VALUE SPACES.
001360 05 FILLER PIC X(4) VALUE "NAME".
001370 05 FILLER PIC X(19) VALUE SPACES.
001380 05 FILLER PIC X(4) VALUE "TYPE".
001390 05 FILLER PIC X(4) VALUE SPACES.
001400 05 FILLER PIC X(4) VALUE "MAKE".
001410 05 FILLER PIC X(6) VALUE SPACES.
001420 05 FILLER PIC X(4) VALUE "DAYS".
001430 05 FILLER PIC X(2) VALUE SPACES.
001440 05 FILLER PIC X(5) VALUE "MILES".
001445 05 FILLER PIC X(8) VALUE SPACES.
001450 05 FILLER PIC X(6) VALUE "AMOUNT".
001460 05 FILLER PIC X(45) VALUE SPACES.
001470
001480 PROCEDURE DIVISION.
001490 100-PREPARE-CAR-BILLING-REPORT.
001500 ACCEPT DATE-WORK-AREA FROM DATE.
001510 OPEN INPUT CAR-RENTAL-FILE
001520 OUTPUT PRINT-FILE.
001530 READ CAR-RENTAL-FILE INTO WS-RECORD-IN
001540 AT END MOVE "NO" TO WS-DATA-REMAINS-SWITCH.
001550 PERFORM 200-PROCESS-RECORDS
001560 UNTIL NO-MORE-RECORDS.
001570 CLOSE CAR-RENTAL-FILE
001580 PRINT-FILE.
001590 STOP RUN.
001600
001610 200-PROCESS-RECORDS.
001620 IF VALID-TYPE-CODE AND VALID-MAKE-CODE
001630 PERFORM 300-COMPUTE-CAR-BILL
001640 PERFORM 400-WRITE-DETAIL-LINE
001650 ELSE
001660 DISPLAY "ERROR IN INCOMING CAR TYPE OR CAR MAKE "

```

Table initialization

Definition of a table

Definition of multiple heading lines

Obtains date of execution

Compound test with 88 level entries

CAR BILLING (CONTINUED)

```

001670      CUST-NAME OF WS-RECORD-IN.
001680
001690      READ CAR-RENTAL-FILE INTO WS-RECORD-IN
001700      AT END MOVE "NO" TO WS-DATA-REMAINS-SWITCH.
001710
001720 300-COMPUTE-CAR-BILL.
001730      IF COMPACT
001740          MOVE .12 TO WS-MILEAGE-RATE
001750          MOVE 11.00 TO WS-DAILY-RATE
001760      ELSE
001770          IF INTERMEDIATE
001780              MOVE .14 TO WS-MILEAGE-RATE
001790              MOVE 12.00 TO WS-DAILY-RATE
001800          ELSE
001810              MOVE .16 TO WS-MILEAGE-RATE
001820              MOVE 13.00 TO WS-DAILY-RATE.
001830
001840      COMPUTE WS-CUSTOMER-BILL ROUNDED =
001850          MILES-DRIVEN OF WS-RECORD-IN * WS-MILEAGE-RATE
001860          + DAYS-RENTED OF WS-RECORD-IN * WS-DAILY-RATE
001870      ON SIZE ERROR
001880          DISPLAY "RECEIVING FIELD TOO SMALL FOR AMOUNT DUE"
001890          CUST-NAME OF WS-RECORD-IN.
001900
001910 400-WRITE-DETAIL-LINE.
001920      IF WS-LINE-COUNT > 5
001930          PERFORM 600-WRITE-HEADING.
001940          MOVE SPACES TO WS-PRINT-LINE.
001950          MOVE CORRESPONDING WS-RECORD-IN TO WS-PRINT-LINE.
001960          INSPECT SOC-SEC-NUM OF WS-PRINT-LINE
001970              REPLACING ALL " " BY "-".
001980          MOVE WS-CUSTOMER-BILL TO CUSTOMER-BILL.
001990
002000          MOVE "NO" TO WS-CAR-MAKE-SWITCH.
002010          PERFORM 500-EXPAND-CAR-MAKE-CODE
002020              VARYING CAR-MAKE-SUBSCRIPT FROM 1 BY 1
002030              UNTIL CAR-MAKE-SUBSCRIPT > 10
002040              OR CAR-CODE-EXPANDED.
002050
002060          WRITE PRINT-LINE FROM WS-PRINT-LINE
002070          AFTER ADVANCING 2 LINES.
002080          ADD 1 TO WS-LINE-COUNT.
002090
002100 500-EXPAND-CAR-MAKE-CODE.
002110      IF CAR-MAKE = CAR-CODE (CAR-MAKE-SUBSCRIPT)
002120          MOVE EXPANDED-CAR-MAKE (CAR-MAKE-SUBSCRIPT)
002130          TO PRINT-CAR-MAKE
002140          MOVE "YES" TO WS-CAR-MAKE-SWITCH.
002150
002160 600-WRITE-HEADING.
002170      ADD 1 TO WS-PAGE-COUNT.
002180      MOVE 1 TO WS-LINE-COUNT.
002190      MOVE WS-PAGE-COUNT TO WS-PAGE-PRINT.
002200      WRITE PRINT-LINE FROM HEADING-LINE-ONE
002210      AFTER ADVANCING PAGE.
002220      MOVE TODAYS-DAY TO TITLE-DAY.
002230      MOVE TODAYS-MONTH TO TITLE-MONTH.
002240      MOVE TODAYS-YEAR TO TITLE-YEAR.
002250      WRITE PRINT-LINE FROM HEADING-LINE-TWO
002260      AFTER ADVANCING 2 LINES.
002270      WRITE PRINT-LINE FROM HEADING-LINE-THREE
002280      AFTER ADVANCING 2 LINES.
002290

```

Use of qualification

Nested IF determines appropriate rate

ROUNDED and SIZE ERROR options

INSPECT statement inserts hyphens in social security number

Use of PERFORM VARYING

Table lookup is controlled by PERFORM VARYING

Resets line count

TRUE/FALSE REVIEW

- 1) EITHER **OF** OR **IN** CAN QUALIFY DATA NAMES.
- 2) **ROUNDED** AND **SIZE ERROR** ARE MANDATORY IN A **COMPUTE** STATEMENT.
- 3) FOR THE **CORRESPONDING** OPTION TO WORK, BOTH DATA NAMES MUST BE AT THE SAME LEVEL.
- 4) QUALIFICATION OVER A SINGLE LEVEL WILL ALWAYS REMOVE AMBIGUITY OF DATA NAMES.
- 5) THE SAME ENTRY MAY CONTAIN BOTH AN **OCCURS** CLAUSE AND A **PICTURE** CLAUSE.
- 6) THE SAME ENTRY MAY NOT HAVE BOTH AN **OCCURS** CLAUSE AND A **VALUE** CLAUSE.
- 7) ONE STATEMENT CANNOT HAVE TWO **IFS** AND ONE **ELSE**.
- 8) CONDITION NAMES ARE ALSO KNOWN AS 77-LEVEL ENTRIES.
- 9) THE SAME **PERFORM** CAN BE USED TO INITIALIZE AND INCREMENT A SUBSCRIPT.
- 10) THE CONDITION PORTION OF AN **IF** MAY BE A COMPOUND CONDITION.

CHAPTER SEVEN: PROGRAMMING STYLE

CODING GUIDELINES

CHOOSE MEANINGFUL NAMES

PREFIX ENTRIES WITHIN THE SAME 01

USE FUNCTIONAL PARAGRAPHS

ELIMINATE 77-LEVEL ENTRIES

SPACE ATTRACTIVELY

INDENT

RESTRICT SUBSCRIPTS TO A SINGLE USE

USE APPROPRIATE COMMENTS

PERFORM PARAGRAPHS, NOT SECTIONS

AVOID CONSTANTS

KEEP IT SIMPLE

USE 88-LEVEL ENTRIES

STRUCTURED PROGRAMMING

A GOOD PROGRAM IS ONE THAT CAN BE EASILY READ
AND MAINTAINED BY SOMEONE OTHER THAN THE
ORIGINAL AUTHOR.

- SPACE ATTRACTIVELY

VERTICAL ALIGNMENT OF PICTURE CLAUSES

USE OF BLANK LINES

BEFORE PARAGRAPH HEADERS

BEFORE **01** ENTRIES

BEFORE SPECIFIC VERBS (FOR EXAMPLE, **IF**)

BEFORE **FD'S**

BEFORE SECTION HEADERS

USE OF EJECT (/ IN COLUMN 7)

- CHOOSE MEANINGFUL DATA NAMES

POOR CHOICE

SWITCH-ONE

TOTAL-1

IMPROVED CHOICE

END-OF-TRANSACTION-FILE-SWITCH

TOTAL-EMPLOYEE-GROSS-PAY

- PREFIX ENTRIES WITHIN THE SAME 01

POOR CHOICE

01 EMPLOYEE-RECORD

05 SOC-SEC-NUMBER

05 NAME

05 ADDRESS

IMPROVED CHOICE

01 EMPLOYEE-RECORD

05 EMP-SOC-SEC-NUMBER

05 EMP-NAME

05 EMP-ADDRESS

- USE FUNCTIONAL PARAGRAPHS

PARAGRAPH NAMES SHOULD CLEARLY INDICATE
THE FUNCTION OF A PARAGRAPH

USE A VERB, ADJECTIVE OR TWO, AND AN OBJECT
SEQUENCE PARAGRAPHS

POOR CHOICE

0005-MAINLINE
A010-READ-AND-WRITE
READ-TRANSACTION-FILE

IMPROVED CHOICE

A010-WRITE-NEW-MASTER-RECORD
1000-PRODUCE-ERROR-REPORT
2000-READ-TRANSACTION-FILE

► ELIMINATE 77-LEVEL ENTRIES

POOR CODE

```
77  COUNTER-ONE      PIC 9(3)  VALUE ZEROS.  
77  COUNTER-TWO     PIC 9(3)  VALUE ZEROS.  
77  COUNTER-THREE  PIC 9(3)  VALUE ZEROS.
```

IMPROVED CODE

```
01  RECORD-COUNTERS.  
    05  NUMBER-OF-RECORDS-READ  PIC 9(3)  VALUE ZEROS.  
    05  NUMBER-OF-GOOD-RECORDS  PIC 9(3)  VALUE ZEROS.  
    05  NUMBER-OF-BAD-RECORDS   PIC 9(3)  VALUE ZEROS.
```

- INDENT

POOR CODE

PERFORM INITIALIZE-TABLE VARYING
LOCATION-SUB FROM 1 BY 1 UNTIL
LOCATION-SUB > 3.

READ EMPLOYEE-FILE AT END MOVE "YES" TO
END-EMPLOYEE-SWITCH.
WRITE PRINT-LINE AFTER ADVANCING PAGE.

IMPROVED CODE

PERFORM INITIALIZE-TABLE
VARYING LOCATION-SUB FROM 1 BY 1
UNTIL LOCATION-SUB > 3.

READ EMPLOYEE-FILE
AT END MOVE "YES" TO END-EMPLOYEE-SWITCH.

WRITE PRINT-LINE
AFTER ADVANCING PAGE.

NESTED **IF**S HAVE SPECIAL REQUIREMENTS

- 1) INDENT SUCCESSIVE **IF**S 4 COLUMNS.
- 2) PUT THE WORD **ELSE** ON A LINE BY ITSELF, AND DIRECTLY UNDER ITS ASSOCIATED **IF**.
- 3) INDENT DETAIL LINES FOR BOTH **IF** AND **ELSE** FOUR COLUMNS.

FOR EXAMPLE:

```
IF CD-SEX IS EQUAL TO "M"  
    IF CD-AGE IS GREATER THAN 30  
        MOVE CD-NAME TO MALE-OVER-30  
        ADD 1 TO NUMBER-QUALIFIED-MALES  
  
    ELSE  
        MOVE CD-NAME TO PRT-NAME  
        ADD 1 TO MALE-UNDER-30.
```

- RESTRICT SUBSCRIPTS TO A SINGLE USE

POOR CODE

77 SUBSCRIPT PIC 9(4).

.
.
.

PERFORM COMPUTE-SALARY-HISTORY
VARYING SUBSCRIPT FROM 1 BY 1
UNTIL SUBSCRIPT > 3.

PERFORM FIND-MATCHING-TITLE
VARYING SUBSCRIPT FROM 1 BY 1
UNTIL SUBSCRIPT > 100.

IMPROVED CODE

01 PROGRAM-SUBSCRIPTS.

05 TITLE-SUBSCRIPT PIC 9(4).

05 SALARY-SUBSCRIPT PIC 9(4).

PERFORM COMPUTE-SALARY-HISTORY
VARYING SALARY-SUBSCRIPT FROM 1 BY 1
UNTIL SALARY-SUBSCRIPT > 3.

PERFORM FIND-MATCHING-TITLE
VARYING TITLE-SUBSCRIPT FROM 1 BY 1
UNTIL TITLE-SUBSCRIPT > 100.

- USE APPROPRIATE COMMENTS

AVOID REDUNDANCY

- * CALCULATE NET PAY

COMPUTE NET-PAY = GROSS PAY – FEDERAL-TAX.

AVOID INACCURACY.

ASSUME OTHER PROGRAMMERS KNOW COBOL AS WELL AS YOU.

EXPLAIN WHY YOU ARE DOING SOMETHING, RATHER THAN WHAT YOU ARE DOING.

- PERFORM *PARAGRAPHS*, NOT *SECTIONS*

PROCEDURE DIVISION.

MAINLINE SECTION.

MOVE ZEROS TO X.

PERFORM A.

PERFORM B.

PERFORM C.

PERFORM D.

STOP RUN.

A SECTION.

ADD 1 TO X.

B.

ADD 1 TO X.

C.

ADD 1 TO X.

D.

ADD 1 TO X.

WHAT IS THE FINAL VALUE OF X?

- AVOID CONSTANTS

POOR CODE

PERFORM COMPUTE-STATE-TOTALS
VARYING STATE-SUBSCRIPT FROM 1 BY 1
UNTIL STATE-SUBSCRIPT > 50.

COMPUTE AVERAGE-STATE-POPULATION =
TOTAL-POPULATION / 50.

IMPROVED CODE

PERFORM COMPUTE-STATE-TOTALS
VARYING STATE-SUBSCRIPT FROM 1 BY 1
UNTIL STATE-SUBSCRIPT >
NUMBER-OF-STATES.

COMPUTE AVERAGE-STATE-POPULATION =
TOTAL-POPULATION /
NUMBER-OF-STATES.

- KEEP IT SIMPLE

POOR CODE

IF HOURS-WORKED > 48

 COMPUTE GROSS-PAY

 = 52 * HOURLY-RATE

 + (HOURS-WORKED - 48) * HOURLY-RATE * 2.

IMPROVED CODE

IF HOURS-WORKED > 48

 COMPUTE GROSS-PAY

 = 40 * HOURLY-RATE

 + 8 * HOURLY-RATE * 1.5

 + (HOURS-WORKED - 48) * HOURLY-RATE * 2.

- USE 88-LEVEL ENTRIES

POOR CODE

IF (LOCATION-CODE = 48 OR
LOCATION-CODE = 65)
AND POLITICAL-PARTY = "D" . . .

IMPROVED CODE

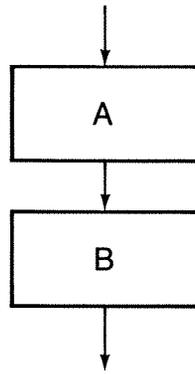
05	LOCATION-CODE	PIC 99.
88	MIAMI	VALUE 48.
88	TAMPA	VALUE 65.
88	FLORIDA	VALUES ARE 48, 65.
05	POLITICAL-PARTY	PIC X.
88	DEMOCRAT	VALUE "D".
88	REPUBLICAN	VALUE "R".

IF FLORIDA AND DEMOCRAT . . .

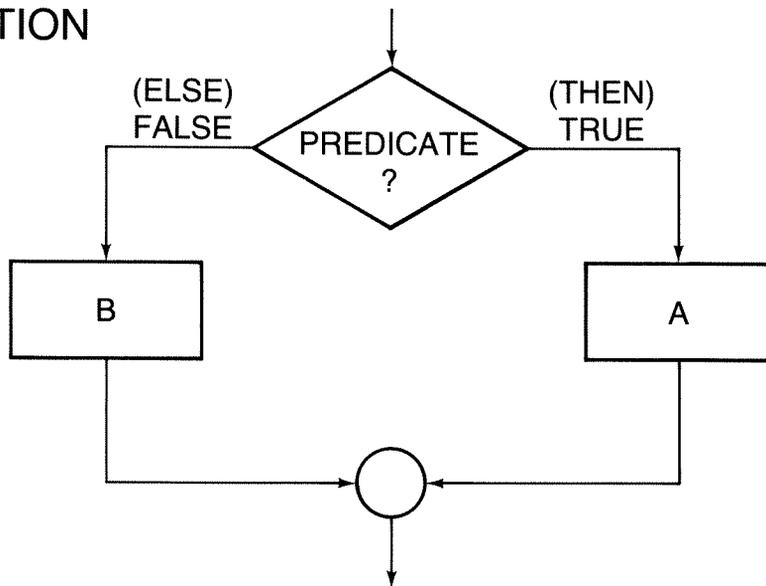
STRUCTURED PROGRAMMING:

- HAS AS ITS OBJECTIVE, PROGRAMS WHICH ARE EASY TO READ AND MAINTAIN.
- DEVELOPS PROGRAMS WHICH CONSIST SOLELY OF THREE KINDS OF BUILDING BLOCKS - SEQUENCE, SELECTION, AND ITERATION.
- REQUIRES THAT EACH MODULE (OR BUILDING BLOCK) HAVE ONLY ONE ENTRY AND ONE EXIT POINT.
- ALLOWS THE BASIC MODULES TO BE COMBINED IN ENDLESS VARIETY TO PRODUCE ANY REQUIRED LOGIC.

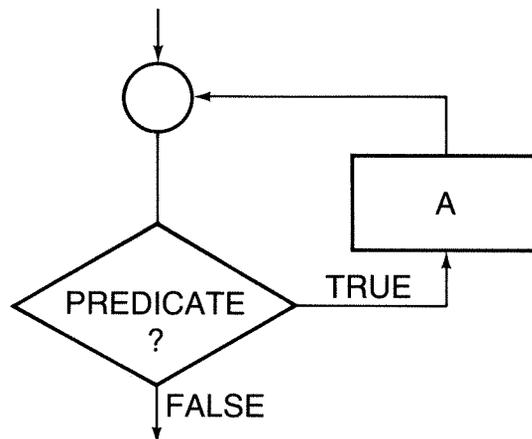
SEQUENCE



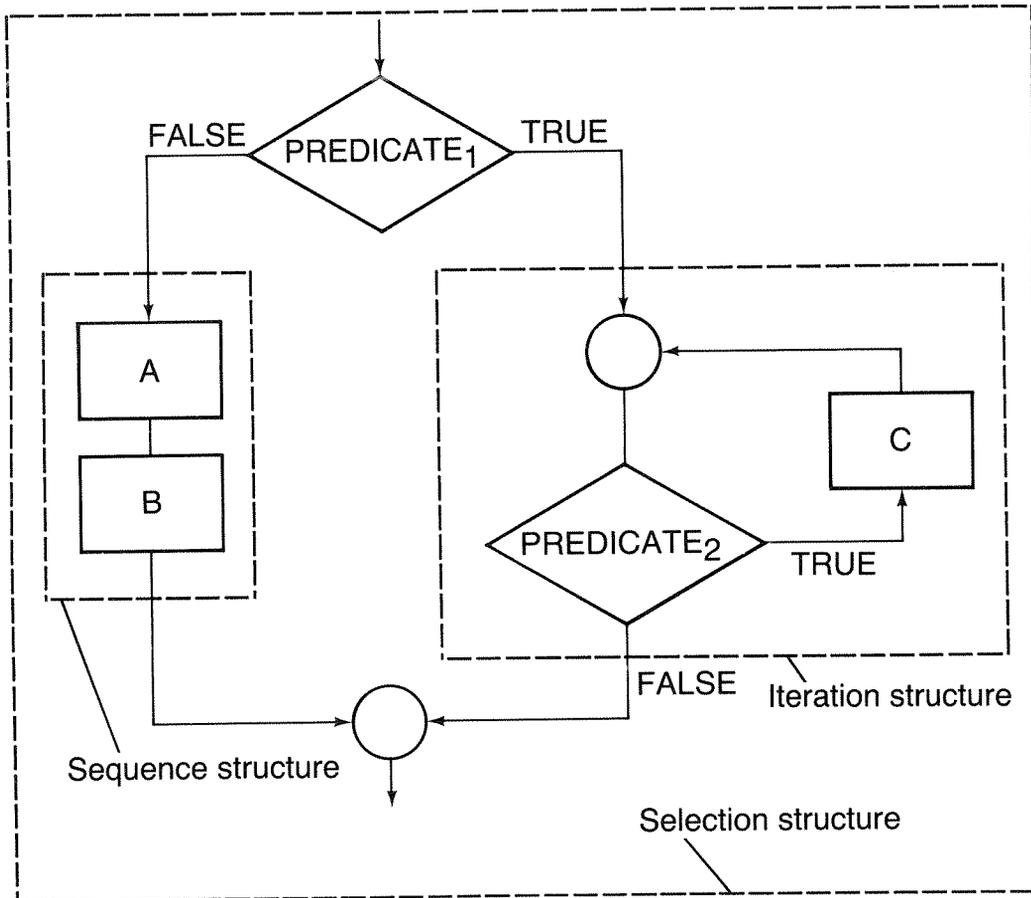
SELECTION



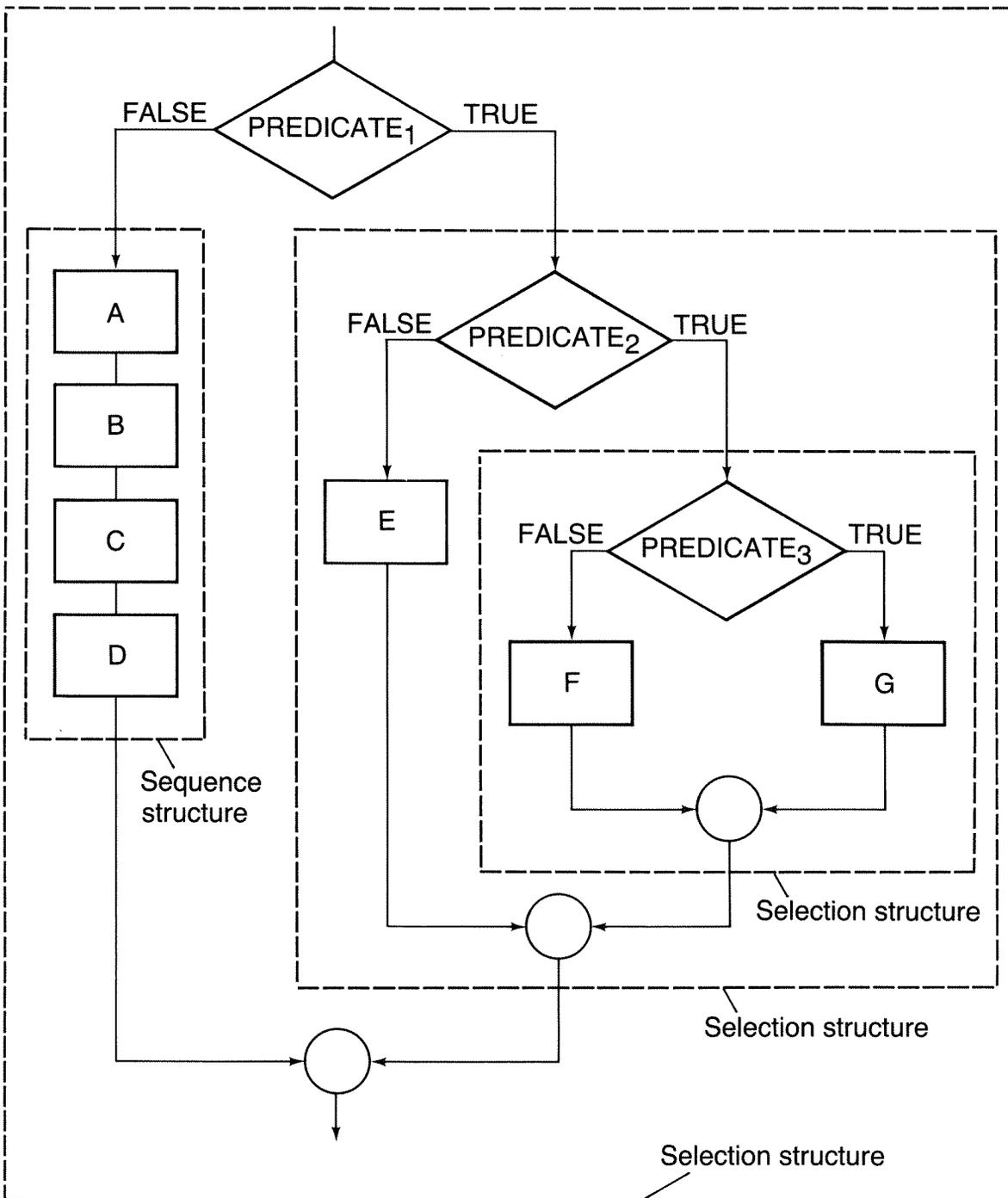
ITERATION



ONE SELECTION, ONE ITERATION, AND ONE SEQUENCE



THREE SELECTION AND ONE SEQUENCE



PAYROLL PROGRAM REWRITTEN

```

000100 IDENTIFICATION DIVISION.
000110 PROGRAM-ID. PAYROLL2.
000120 AUTHOR. R GRAUER.
000130
000140 ENVIRONMENT DIVISION.
000150 CONFIGURATION SECTION.
000160 SOURCE-COMPUTER. TRS-80.
000170 OBJECT-COMPUTER. TRS-80.
000180
000190 INPUT-OUTPUT SECTION.
000200 FILE-CONTROL.
000210 SELECT EMPLOYEE-FILE
000220 ASSIGN TO INPUT "PAYROLL/DAT".
000230 SELECT PRINT-FILE
000240 ASSIGN TO PRINT "PAYROLL2/TXT".
000250
000260 DATA DIVISION.
000270 FILE SECTION.
000280 FD EMPLOYEE-FILE
000290 LABEL RECORDS ARE OMITTED
000300 RECORD CONTAINS 80 CHARACTERS
000310 DATA RECORD IS EMPLOYEE-RECORD.
000320 01 EMPLOYEE-RECORD.
000330 05 EMP-NAME.
000340 10 EMP-LAST-NAME PIC X(15).
000350 10 EMP-FIRST-NAME PIC X(10).
000360 05 EMP-HOURS-WORKED.
000370 10 EMP-REG-HOURS PIC 99.
000380 10 EMP-OVERTIME-HOURS PIC 99.
000390 05 EMP-RATE PIC 99V99.
000400 05 FILLER PIC X(47).
000410
000420 FD PRINT-FILE
000430 LABEL RECORDS ARE STANDARD
000440 RECORD CONTAINS 132 CHARACTERS
000450 DATA RECORD IS PRINT-LINE.
000460 01 PRINT-LINE PIC X(132).
000470
000480 WORKING-STORAGE SECTION.
000490 77 WS-DATA-REMAINS-SWITCH PIC X(3) VALUE SPACES.
000500
000510 01 DATE-WORK-AREA.
000520 05 TODAYS-YEAR PIC 99.
000530 05 TODAYS-MONTH PIC 99.
000540 05 TODAYS-DAY PIC 99.
000550
000560 01 IND-COMPUTATIONS.
000570 05 IND-REGULAR-PAY PIC 9(4)V99.
000580 05 IND-OVERTIME-PAY PIC 9(4)V99.
000590 05 IND-GROSS-PAY PIC 9(4)V99.
000600 05 IND-FEDERAL-TAX PIC 9(4)V99.
000610 05 IND-NET-PAY PIC 9(4)V99.
000620
000630 01 COMPANY-TOTALS.
000640 05 CO-REGULAR-PAY PIC 9(6)V99 VALUE ZEROS.
000650 05 CO-OVERTIME-PAY PIC 9(6)V99 VALUE ZEROS.
000660 05 CO-GROSS-PAY PIC 9(6)V99 VALUE ZEROS.
000670 05 CO-FEDERAL-TAX PIC 9(6)V99 VALUE ZEROS.
000680 05 CO-NET-PAY PIC 9(6)V99 VALUE ZEROS.
000690
000700 01 PAGE-AND-LINE-COUNTERS.
000710 05 WS-PAGE-COUNT PIC 9(4) VALUE ZEROS.
000720 05 WS-LINE-COUNT PIC 9(4) VALUE 4.
000730
000740 01 HEADING-LINE-ONE.
000750 05 FILLER PIC X(4).
000760 05 HDG-MONTH PIC Z9.
000770 05 FILLER PIC X VALUE "/".
000780 05 HDG-DAY PIC Z9.
000790 05 FILLER PIC X VALUE "/".
000800 05 HDG-YEAR PIC 99.
000810 05 FILLER PIC X(40) VALUE SPACES.
000820 05 FILLER PIC X(8) VALUE "PAYROLL ".
000830 05 FILLER PIC X(6) VALUE "REPORT".
000840 05 FILLER PIC X(40) VALUE SPACES.
000850 05 FILLER PIC X(4) VALUE "PAGE".
000860 05 HDG-PAGE-NUMBER PIC Z(4).
000870 05 FILLER PIC X(18) VALUE SPACES.
000880
000890 01 HEADING-LINE-TWO.

```

Data names within an FD have common prefix

Holds date of execution

Common prefix

PICTURE and VALUE clauses are vertically aligned

Page and line counters are required for page heading routine

REWRITTEN PROGRAM (CONTINUED)

```

000900 05 FILLER PIC X(8) VALUE SPACES.
000910 05 FILLER PIC X(4) VALUE "NAME".
000920 05 FILLER PIC X(9) VALUE SPACES.
000930 05 FILLER PIC X(4) VALUE "RATE".
000940 05 FILLER PIC X(4) VALUE SPACES.
000950 05 FILLER PIC X(9) VALUE "REG HOURS".
000960 05 FILLER PIC X(4) VALUE SPACES.
000970 05 FILLER PIC X(9) VALUE "O/T HOURS".
000980 05 FILLER PIC X(4) VALUE SPACES.
000990 05 FILLER PIC X(7) VALUE "REG PAY".
001000 05 FILLER PIC X(4) VALUE SPACES.
001010 05 FILLER PIC X(7) VALUE "O/T PAY".
001020 05 FILLER PIC X(4) VALUE SPACES.
001030 05 FILLER PIC X(11) VALUE "GROSS PAY".
001040 05 FILLER PIC X(2) VALUE SPACES.
001050 05 FILLER PIC X(7) VALUE "FED TAX".
001060 05 FILLER PIC X(5) VALUE SPACES.
001070 05 FILLER PIC X(7) VALUE "NET PAY".
001080 05 FILLER PIC X(23) VALUE SPACES.
001090
001100 01 DASHED-LINE.
001110 05 ROW-OF-DASHES PIC X(111) VALUE ALL "-".
001120 05 FILLER PIC X(21) VALUE SPACES.
001130
001140 01 DETAIL-LINE.
001150 05 FILLER PIC X(2).
001160 05 DET-LAST-NAME PIC X(15).
001170 05 FILLER PIC X(2).
001180 05 DET-RATE PIC $$$9.99.
001190 05 FILLER PIC X(8).
001200 05 DET-REG-HOURS PIC Z9.
001210 05 FILLER PIC X(10).
001220 05 DET-OVERTIME-HOURS PIC Z9.
001230 05 FILLER PIC X(6).
001240 05 DET-REGULAR-PAY PIC $,$$9.99.
001250 05 FILLER PIC X(3).
001260 05 DET-OVERTIME-PAY PIC $,$$9.99.
001270 05 FILLER PIC X(2).
001280 05 DET-GROSS-PAY PIC $,$$9.99.
001290 05 FILLER PIC X(3).
001300 05 DET-FEDERAL-TAX PIC $,$$9.99.
001310 05 FILLER PIC X(3).
001320 05 DET-NET-PAY PIC $,$$9.99.
001330 05 FILLER PIC X(23).
001340
001350 01 TOTAL-LINE.
001360 05 FILLER PIC X(6) VALUE SPACES.
001370 05 FILLER PIC X(6) VALUE "TOTALS".
001380 05 FILLER PIC X(41) VALUE SPACES.
001390 05 TOTAL-REGULAR-PAY PIC $,$$9.99.
001400 05 FILLER PIC X(3) VALUE SPACES.
001410 05 TOTAL-OVERTIME-PAY PIC $,$$9.99.
001420 05 FILLER PIC X(2) VALUE SPACES.
001430 05 TOTAL-GROSS-PAY PIC $,$$9.99.
001440 05 FILLER PIC X(3) VALUE SPACES.
001450 05 TOTAL-FEDERAL-TAX PIC $,$$9.99.
001460 05 FILLER PIC X(3) VALUE SPACES.
001470 05 TOTAL-NET-PAY PIC $,$$9.99.
001480 05 FILLER PIC X(47) VALUE SPACES.
001490
001500 PROCEDURE DIVISION.
001510 0100-PREPARE-PAYROLL.
001520 PERFORM 0200-GET-DATE.
001530 OPEN INPUT EMPLOYEE-FILE
001540 OUTPUT PRINT-FILE.
001550 READ EMPLOYEE-FILE
001560 AT END MOVE "NO" TO WS-DATA-REMAINS-SWITCH.
001570 PERFORM 0300-PROCESS-RECORDS
001580 UNTIL WS-DATA-REMAINS-SWITCH = "NO".
001590 PERFORM 1000-WRITE-COMPANY-TOTALS.
001600 CLOSE EMPLOYEE-FILE
001610 PRINT-FILE.
001620 STOP RUN.
001630
001640 0200-GET-DATE.
001650 ACCEPT DATE-WORK-AREA FROM DATE.
001660 MOVE TODAYS-YEAR TO HDG-YEAR.
001670 MOVE TODAYS-MONTH TO HDG-MONTH.
001680 MOVE TODAYS-DAY TO HDG-DAY.
001690

```

Blank lines are inserted before 01 entries

Datanames within an 01 entry have a common prefix

Subservient clauses are indented

ACCEPT statement to obtain date of execution

REWRITTEN PROGRAM (CONTINUED)

```
001700 0300-PROCESS-RECORDS.
001710 PERFORM 0400-COMPUTE-GROSS-PAY.
001720 PERFORM 0500-COMPUTE-FEDERAL-TAX.
001730 PERFORM 0600-COMPUTE-NET-PAY.
001740 PERFORM 0700-UPDATE-COMPANY-TOTALS.
001750
001760 IF WS-LINE-COUNT > 3
001770     PERFORM 0800-WRITE-HEADING-LINE.
001780 PERFORM 0900-WRITE-DETAIL-LINE.
001790 ADD 1 TO WS-LINE-COUNT.
001800 READ EMPLOYEE-FILE
001810     AT END MOVE "NO" TO WS-DATA-REMAINS-SWITCH.
001820
001830 0800-WRITE-HEADING-LINE.
001840     ADD 1 TO WS-PAGE-COUNT.
001850     MOVE 1 TO WS-LINE-COUNT.
001860     MOVE WS-PAGE-COUNT TO HDG-PAGE-NUMBER.
001870     WRITE PRINT-LINE FROM HEADING-LINE-ONE
001880         AFTER ADVANCING PAGE.
001890     WRITE PRINT-LINE FROM HEADING-LINE-TWO
001900         AFTER ADVANCING 4 LINES.
001910     WRITE PRINT-LINE FROM DASHED-LINE
001920         AFTER ADVANCING 1 LINE.
001930
001940 0400-COMPUTE-GROSS-PAY.
001950     MULTIPLY EMP-REG-HOURS BY EMP-RATE GIVING IND-REGULAR-PAY.
001960     COMPUTE IND-OVERTIME-PAY
001970         = EMP-OVERTIME-HOURS * EMP-RATE * 1.5.
001980     ADD IND-REGULAR-PAY IND-OVERTIME-PAY GIVING IND-GROSS-PAY.
001990
002000 0500-COMPUTE-FEDERAL-TAX.
002010     COMPUTE IND-FEDERAL-TAX = .16 * IND-GROSS-PAY.
002020     IF IND-GROSS-PAY > 160
002030         COMPUTE IND-FEDERAL-TAX
002040             = IND-FEDERAL-TAX + .02 * (IND-GROSS-PAY - 160).
002050
002060     IF IND-GROSS-PAY > 200
002070         COMPUTE IND-FEDERAL-TAX
002080             = IND-FEDERAL-TAX + .02 * (IND-GROSS-PAY - 200).
002090
002100 0600-COMPUTE-NET-PAY.
002110     COMPUTE IND-NET-PAY = IND-GROSS-PAY - IND-FEDERAL-TAX.
002120
002130 0700-UPDATE-COMPANY-TOTALS.
002140     ADD IND-REGULAR-PAY TO CO-REGULAR-PAY.
002150     ADD IND-OVERTIME-PAY TO CO-OVERTIME-PAY.
002160     ADD IND-GROSS-PAY TO CO-GROSS-PAY.
002170     ADD IND-FEDERAL-TAX TO CO-FEDERAL-TAX.
002180     ADD IND-NET-PAY TO CO-NET-PAY.
002190
002200 0900-WRITE-DETAIL-LINE.
002210     MOVE SPACES TO DETAIL-LINE.
002220     MOVE EMP-LAST-NAME TO DET-LAST-NAME.
002230     MOVE EMP-RATE TO DET-RATE.
002240     MOVE EMP-REG-HOURS TO DET-REG-HOURS.
002250     MOVE EMP-OVERTIME-HOURS TO DET-OVERTIME-HOURS.
002260     MOVE IND-REGULAR-PAY TO DET-REGULAR-PAY.
002270     MOVE IND-OVERTIME-PAY TO DET-OVERTIME-PAY.
002280     MOVE IND-GROSS-PAY TO DET-GROSS-PAY.
002290     MOVE IND-FEDERAL-TAX TO DET-FEDERAL-TAX.
002300     MOVE IND-NET-PAY TO DET-NET-PAY.
002310
002320     WRITE PRINT-LINE FROM DETAIL-LINE
002330         AFTER ADVANCING 2 LINES.
002340
002350 1000-WRITE-COMPANY-TOTALS.
002360     WRITE PRINT-LINE FROM DASHED-LINE
002370         AFTER ADVANCING 1 LINE.
002380     MOVE CO-REGULAR-PAY TO TOTAL-REGULAR-PAY.
002390     MOVE CO-OVERTIME-PAY TO TOTAL-OVERTIME-PAY.
002400     MOVE CO-GROSS-PAY TO TOTAL-GROSS-PAY.
002410     MOVE CO-FEDERAL-TAX TO TOTAL-FEDERAL-TAX.
002420     MOVE CO-NET-PAY TO TOTAL-NET-PAY.
002430
002440     WRITE PRINT-LINE FROM TOTAL-LINE
002450         AFTER ADVANCING 2 LINES.
```

Test to invoke heading routine

WS-LINE-COUNT is reset to 1 in heading routine

WS-LINE-COUNT is incremented for each detail line

Paragraph names are sequenced and functional

New functions in expanded payroll

Existing modules have been expanded

Blank lines are inserted before paragraph names

TRUE/FALSE REVIEW

- 1) COBOL REQUIRES THAT PARAGRAPH NAMES BE SEQUENCED.
- 2) BLANK LINES ARE NOT PERMITTED IN A COBOL PROGRAM.
- 3) A SLASH IN COLUMN 7 CAUSES THE NEXT LINE IN A LISTING TO BEGIN ON A NEW PAGE.
- 4) **0100-READ-AND-COMPUTE** IS A GOOD NAME FOR A COBOL PARAGRAPH.
- 5) COMMENTS ARE INDICATED BY AN * IN COLUMN 7.
- 6) A PROGRAM CANNOT HAVE TOO MANY COMMENTS.
- 7) IT IS IMPOSSIBLE TO WRITE A PROGRAM WITHOUT 77-LEVEL ENTRIES.
- 8) TWO- AND THREE-LETTER DATA NAMES ARE DESIRABLE TO REDUCE PROGRAMMER CODING TIME.
- 9) INDENTATION IS A WASTE OF TIME SINCE IT HAS NO EFFECT ON THE COMPILER.
- 10) COBOL REQUIRES ALL **PICTURE** CLAUSES TO BEGIN IN THE SAME COLUMN.

**APPENDIX:
MULTIPLE CHOICE REVIEW QUESTIONS**

1. Which command is used to renumber a program?
 - a) B
 - b) N
 - c) R
 - d) S

2. The command P 100
 - a) Prints 100 lines.
 - b) Prints line 100.
 - c) Prints line 100 and the next 19 lines.
 - d) None of the above.

3. The COBOL compiler is invoked by the command
 - a) CEDIT
 - b) RUNCOBOL
 - c) RSCOBOL
 - d) COBPRT

4. COB is the extension associated with:
 - a) a COBOL source program.
 - b) a COBOL data file.
 - c) an object program (i.e. the output of a compile).
 - d) a report printed by a COBOL program.

5. An ANS COBOL program, written for a TRS-80, should also run on:
 - a) A Univac, Honeywell, or IBM machine provided each has a COBOL compiler.
 - b) Any machine which has a COBOL compiler.
 - c) Both a and b above.
 - d) None of the above.

6. The four divisions in a COBOL program may:
 - a) Appear in any order.
 - b) Appear in any order so long as the Identification Division is first.
 - c) Appear in any order so long as the Procedure Division is first.
 - d) Must appear in the order Identification, Environment, Data and Procedure.
 - e) Must appear in the order Identification, Data, Environment, and Procedure.

7. All of the following are advantages of a computer except:
 - a) Its ability to 'think' for itself.
 - b) Its speed.
 - c) Its accuracy.

8. The 'logic' of a COBOL program is:
 - a) Contained in the Procedure Division only.
 - b) Contained in the Data Division only.
 - c) Contained in any or all of the divisions.
 - d) Contained in both the Procedure and Data Division.

9. If a COBOL program contains exactly 50 statements, the generated machine language program will most likely contain:
 - a) Exactly 50 statements.
 - b) Fewer than 50 statements.
 - c) An even multiple of 50 statements.
 - d) Many more than 50 statements.

10. The COBOL compiler:
 - a) Translates COBOL into machine language.
 - b) Is a computer program.
 - c) Is a piece of software.
 - d) All of the above.

11. A decision is indicated in a flowchart by:
 - a) A rectangle.
 - b) A triangle.
 - c) A 'diamond' shaped block.
 - d) A circle.

12. Numeric literals may contain:
 - a) Numbers.
 - b) Letters.
 - c) Numbers, letters, and special characters.

13. Non-numeric literals may contain:
 - a) Numbers.
 - b) Letters.
 - c) Special characters.
 - d) Reserved words.
 - e) All of the above.

14. Filenames are likely to be contained in:
 - a) All divisions.
 - b) All divisions except the Identification Division.
 - c) Only the Data Division.
 - d) Only the Data and Procedure Divisions.

15. Which of the following are valid as data names:
 - a) SUM-OF-X
 - b) IDENTIFICATION-DIVISION
 - c) ABCDEFGHIJKLMNOPQRSTUVWXYZ
 - d) All of the above.
 - e) Only a and c above.

16. A MOVE statement:
 - a) May begin in or past column 12.
 - b) May begin in or past the A margin.
 - c) Must start in column 12.
 - d) May start anywhere except in column 1.
 - e) Must end before column 62.

17. Which of the following must begin in the A margin:
- Division headers.
 - Section headers.
 - Paragraph names.
 - 01 entries.
 - All of the above.
18. A Programmer defined data name cannot:
- Contain more than 30 characters.
 - Be a reserved word.
 - End on a hyphen.
 - Contain all numbers.
 - All of the above.
19. A clean compile; i.e., a compilation with no errors, means that:
- Subsequent execution results will be correct.
 - Only that the COBOL program has been successfully translated into machine language.
 - The corresponding flowchart must be correct.
 - All of the above.
20. All of the following are valid numeric literals except:
- 123
 - 123.45
 - "\$123.45"
 - 0123.45
21. Braces: i.e., { }, indicate:
- Optional reserved words.
 - One of the enclosed items must be selected.
 - Required reserved words.
 - Programmer supplied information.
 - None of the above.
22. The use of parentheses in a COMPUTE statement:
- Is prohibited.
 - Clarifies, but never alters, the order of operation.
 - Clarifies, and always alters, the order of operation.
 - Clarifies, and sometimes alters, the order of operation.
23. The type of file; e.g., INPUT or OUTPUT is:
- Specified in an OPEN statement.
 - Specified in a CLOSE statement.
 - Specified in both an OPEN and CLOSE statement.
 - Specified in neither an OPEN nor CLOSE statement.

24. Every COBOL program must have:
- a) Exactly one STOP statement.
 - b) At least one STOP statement.
 - c) A STOP statement as the last line in the Procedure Division.
 - d) A maximum of 5 STOP statements.
25. The COBOL SELECT statement:
- a) Is for documentation only and serves no useful purpose.
 - b) Appears in the Data Division.
 - c) Ties a programmer chosen file-name to a system name.
 - d) Appears in the File-Control Paragraph.
 - e) Both c and d above.
26. All of the following paragraphs belong in the Identification Division except:
- a) PROGRAM-ID.
 - b) AUTHOR.
 - c) SOURCE-COMPUTER.
 - d) INSTALLATION.
27. In describing a record layout (i.e., entries under an 01 record), the programmer may choose any level numbers from:
- a) 02 to 49 inclusive.
 - b) 02 to 77 inclusive.
 - c) 02 to 49 inclusive and 77.
 - d) The set 05, 10, 15, 20, 25, 30, 35, 40, 45, 50.
28. A 'filename' appears in all of the following statements except:
- a) OPEN.
 - b) CLOSE.
 - c) SELECT.
 - d) READ.
 - e) WRITE.
29. The Working-Storage Section:
- a) Appears after the File Section.
 - b) May contain 01 entries.
 - c) May contain VALUE clauses.
 - d) May contain PICTURE clauses with an assumed decimal point.
 - e) All of the above.
30. An entry prefixed by the level number 05 will:
- a) Always have a PICTURE clause.
 - b) Always be an elementary item.
 - c) Always appear in the File Section.
 - d) All of the above.
 - e) None of the above.

31. A group item:
 - a) Never has a PICTURE clause.
 - b) Must be an 01 entry.
 - c) May sometimes have both a PICTURE and a VALUE clause.
 - d) Will always have at least four data names defined under it.

32. An elementary item:
 - a) Never has a PICTURE clause.
 - b) Can never be an 01 entry.
 - c) May sometimes have both a PICTURE and a VALUE clause.
 - d) May sometimes have additional data names defined under it.

33. The following are all valid PICTURE clauses except:
 - a) PIC 999
 - b) PIC \$\$\$
 - c) PIC \$99
 - d) PIC 99¢

34. The Procedure Division:
 - a) May contain several paragraphs.
 - b) Must contain at least 3 paragraphs.
 - c) Cannot have a single COBOL statement continue from line to line.
 - d) Must contain at least one occurrence of each of the following verbs: MOVE, ADD, OPEN, CLOSE, and COMPUTE.

35. All of the following will produce compilation errors except:
 - a) Reading a record.
 - b) Writing a file.
 - c) Attempting to do arithmetic on a field defined as alphanumeric.
 - d) Omitting a MOVE statement prior to a WRITE.

36. Which of the following will cause a compilation error:

a) 05 TITLE-FIELD	PIC X(5) VALUE "COBOL".
b) 05 IDENTIFICATION	PIC X(5) VALUE "COBOL".
c) 05 NUMERIC-FIELD	PIC X(5).
d) 05 ALPHA-FIELD	PIC 9(5).

37. If one misspelled 'ENVIRONMENT' as the only error in a COBOL program:
 - a) There would be no problem as the compiler would guess what the programmer intended.
 - b) A single diagnostic would result.
 - c) The program would compile almost perfectly, and probably would go to the execution phase.
 - d) Many compilation errors would result.

38. Which of the following would cause compilation error(s)?
 - a) WORKING-STORAGE-SECTION.
 - b) ENVIROMENT DIVISION
 - c) START (as a paragraph name)
 - d) All of the above.

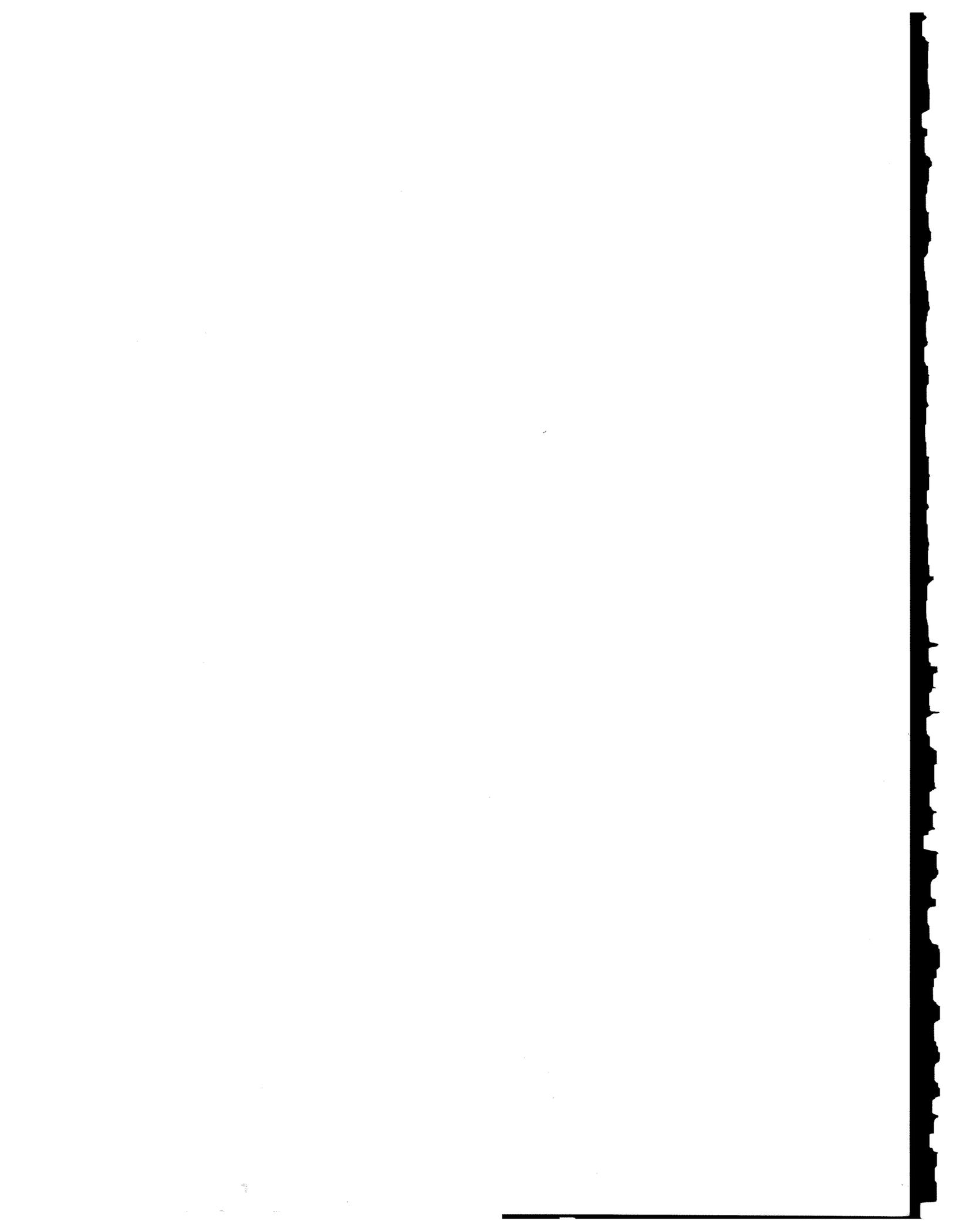
39. Data names within the same COBOL program:
- Must be unique.
 - May be qualified by OF only.
 - May be qualified by IN only.
 - May be qualified by either OF or IN.
 - None of the above.
40. The statement 'IF EXEMPT, PERFORM EXEMPT-ROUTINE' is:
- Syntactically invalid.
 - Valid only if EXEMPT is defined as an 88 level entry.
 - Valid regardless of how EXEMPT is defined.
 - None of the above.
41. The statement: IF X > Y OR X = Z AND X < W is considered true if:
- X > Y.
 - X = Z AND X < W.
 - X < W.
 - All of the above.
 - Either a or b above.
42. The statement:
- ```

IF A > B
 IF C > D
 MOVE X TO Y
 MOVE S TO T
ELSE
 MOVE X TO Z.

```
- Is syntactically invalid.
  - Is syntactically valid, but misleading because the ELSE is not shown under the appropriate IF.
  - Will move X to Z if A not greater than B.
  - Will always move X to Y if A is greater than B.
43. The statement: 'READ CARD-FILE INTO WS-CARD-AREA . . .':
- Will cause the input record to appear in two places.
  - Will cause the input record to appear in WS-CARD-AREA only.
  - Combines the effects of both a READ and a MOVE statement.
  - Both a and c above.
  - Both b and c above.
44. The PERFORM verb:
- Can be used for sections only.
  - Can be used for paragraphs only.
  - Must have an associated THRU clause.
  - Cannot cause the execution of more than a single paragraph.
  - None of the above.

45. DATE is:
- A COBOL reserved word.
  - A programmer-defined data name.
  - The current Julian date.
  - A five character area in the form mm/yy.
46. The DISPLAY verb:
- Permits both a literal and a programmer-defined data name to appear in the same sentence.
  - Permits several data names to appear in the same sentence.
  - Does not require a corresponding record description in the Data Division.
  - Can direct output directly to the console.
  - All of the above.
47. Which of the following is a valid example of a subscripted entry:
- HOURLY-RATE (3).
  - HOURLY-RATE ( 3).
  - HOURLY-RATE(3).
  - HOURLY-RATE (3 ).
48. The statement "PERFORM PAR-A VARYING WS-SUB FROM 1 BY 1 UNTIL WS-SUB = 10" will:
- Execute PAR-A 10 times.
  - Execute PAR-A 9 times.
  - Execute PAR-A once.
  - Is an invalid form of the PERFORM verb.
49. Given the entries:
- |                                 |            |
|---------------------------------|------------|
| 05 SALES-TABLE OCCURS 12 TIMES. |            |
| 10 SALES-VOLUME                 | PIC 9(6).  |
| 10 SALES-MONTH                  | PIC X(10). |
- A total of 16 bytes is defined in storage for SALES-TABLE.
  - A total of 192 bytes is defined in storage for SALES-TABLE.
  - The first 72 bytes of SALES-TABLE relate to SALES-VOLUME, the last 120 bytes relate to SALES-MONTH.
  - Both b and c above.
50. The following are all valid forms of the PERFORM verb except:
- PERFORM PAR-A NUMBER-TRANSACTION TIMES.
  - PERFORM PAR-A.
  - PERFORM PAR-A TWICE.
  - PERFORM PAR-A VARYING WS-SUB FROM 1 BY 1 UNTIL WS-SUB = 10.





**RADIO SHACK**  **A DIVISION OF TANDY CORPORATION**

**U.S.A.: FORT WORTH, TEXAS 76102**  
**CANADA: BARRIE, ONTARIO L4M 4W5**

---

**TANDY CORPORATION**

| AUSTRALIA                                      | BELGIUM                                    | U. K.                                             |
|------------------------------------------------|--------------------------------------------|---------------------------------------------------|
| 91 KURRAJONG ROAD<br>MOUNT DRUITT, N.S.W. 2770 | PARC INDUSTRIEL DE NANINNE<br>5140 NANINNE | BILSTON ROAD WEDNESBURY<br>WEST MIDLANDS WS10 7JN |