Welcome to:

Introduction to COBOL Programming

Education

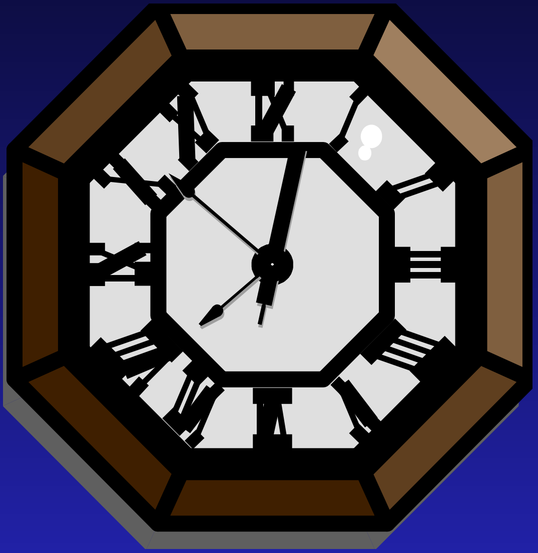
Leveraging Technology with Dedicated Specialists

Introduction to COBOL Programming

- Class Introductions………….
- Your Trainer – Peter Molchan
  - COB100

Introduction to COBOL Programming

- Class Hours
  - Approx 9:00 am. to 4:00 pm.
  - Lunch around 11:30
  - Morning and afternoon break



Introduction to COBOL Programming

- Training Medium
  - Student Workbook
  - Additional Skill-Building Exercises
  - Mainframe Express COBOL Compiler

Introduction to COBOL Programming

- High Level Course Overview
  - COBOL Introduction
  - Structure of a COBOL Program
  - Introduction to Mainframe Express
  - Back to COBOL

Introduction to COBOL Programming

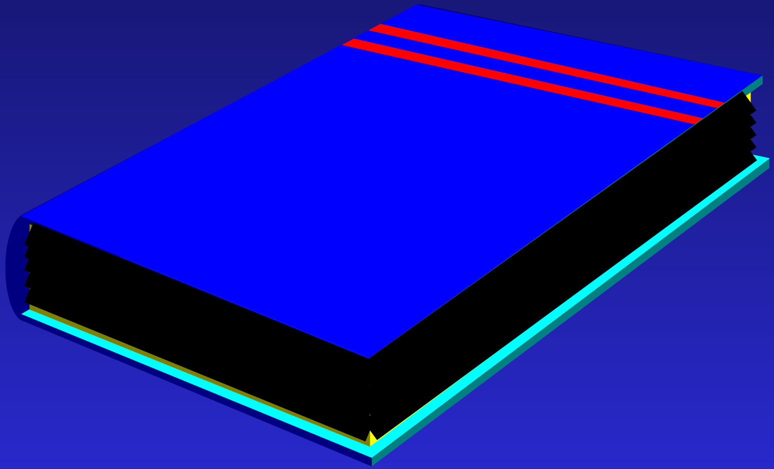
- Course Methodology
  - Lecture
  - Instructor led hands-on instruction
  - Student exercises
  - Case problems
  - Workshop sessions

Introduction to COBOL Programming

- Course Objectives
  - Learn the requirements and syntax of the COBOL language
  - Describe expressions and statements
  - Write File and Data Definition statements
  - Perform Input/Output operations
  - Use arithmetic functions
  - Write basic report programs
  - Use subroutines

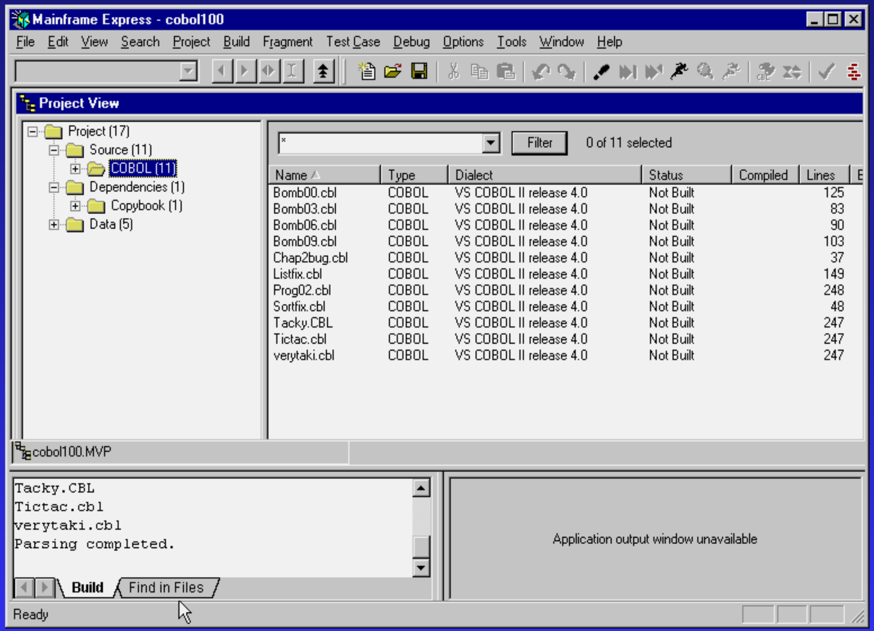
Introduction to COBOL Programming

Course Manual (Student Workbook) TOC



Introduction to COBOL Programming

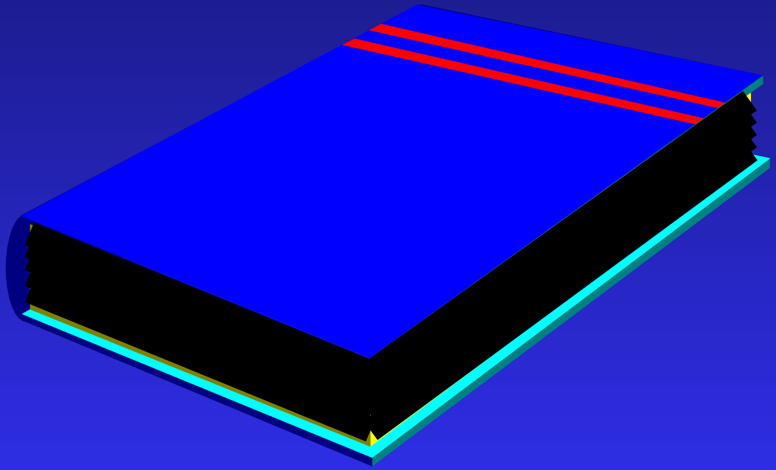
- Mainframe Express
  - Create/Edit Programs
  - Compile Programs
  - Test/Debug Programs
  - Edit Data Files
  - Control Compiler


Mainframe Express - cobol100
File Edit View Search Project Build Fragment Test Case Debug Options Tools Window Help
Project View
Project (17)
Source (11)
COBOL (11)
Dependencies (1)
Copybook (1)
Data (5)
Filter
0 of 11 selected
cobol100.MVP
Build
Find in Files
Application output window unavailable
Ready

Name	Type	Dialect	Status	Compiled	Lines
Bomb00.cbl	COBOL	VS COBOL II release 4.0	Not Built		125
Bomb03.cbl	COBOL	VS COBOL II release 4.0	Not Built		83
Bomb06.cbl	COBOL	VS COBOL II release 4.0	Not Built		90
Bomb09.cbl	COBOL	VS COBOL II release 4.0	Not Built		103
Chap2bug.cbl	COBOL	VS COBOL II release 4.0	Not Built		37
Listfix.cbl	COBOL	VS COBOL II release 4.0	Not Built		149
Prog02.cbl	COBOL	VS COBOL II release 4.0	Not Built		248
Sortfix.cbl	COBOL	VS COBOL II release 4.0	Not Built		48
Tacky.CBL	COBOL	VS COBOL II release 4.0	Not Built		247
Tictac.cbl	COBOL	VS COBOL II release 4.0	Not Built		247
verytaki.cbl	COBOL	VS COBOL II release 4.0	Not Built		247

```
Tacky.CBL
Tictac.cbl
verytaki.cbl
Parsing completed.
```

Chapter 1 COBOL Overview



1.1 Objectives

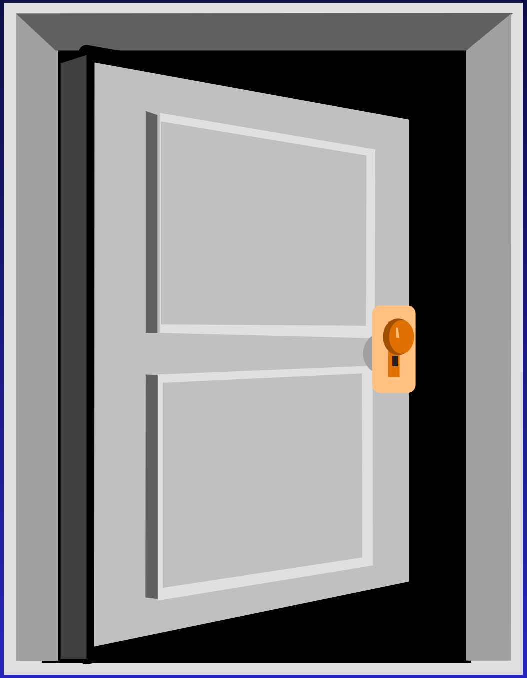
After completing this chapter, you will understand the capabilities and syntax of COBOL programs. Specifically, you will be able to:

- Describe the steps of the Programming Life Cycle
- Describe the function of the four COBOL divisions
- List the advantages and disadvantages of COBOL
- Describe the purpose of the COBOL compiler
- Understand the column structure of COBOL

1.2 Topics to be covered:

Programming life cycle
What is COBOL?
Advantages of COBOL
Limitations of COBOL
COBOL preparation
COBOL structure
COBOL columns
COBOL lines
COBOL syntax

1.2.1 Programming life cycle



Background.........

1.2.1 Programming life cycle

Enterprise Level

Business/data modeling

* Enterprise modeling

Needs analysis

* Feasibility, survey investigation, data gathering & analysis

System design

* Input/output requirements, system controls, databases

Program Development/Maintenance Level

Program development

* Code, compile, link

Testing

* Find the bugs before the bugs find you

Implementation/sign-off

* Conversion, training, auditing, evaluation

Maintenance

* Monitoring, adjustments, upgrades, service requests

1.2.2 What is COBOL?????????

- **<u>Co</u>**mmon **<u>B</u>**usiness **<u>O</u>**riented **<u>L</u>**anguage

- COBOL Roots - Evolution
  - Developed by the Department of Defense in 1959
  - **<u>CO</u>**nference of **<u>DA</u>**ta **<u>SY</u>**stem **<u>L</u>**anguages (CODASYL)
  - Under the guidance of Grace Hopper
  - Conference goals were to develop a language that was:
    - Business Oriented
    - Machine independent
    - English-like
    - Self documenting
  - DOD mandated parameters to software developers
  - Standards were/are maintained/updated by the **<u>A</u>**merican **<u>N</u>**ational **<u>S</u>**tandards **<u>I</u>**nstitute (ANSI)

1.2.3 Advantages of COBOL

- English-like
- Solves Business Problems
- Handles large volumes of data
- Universal and standardized
- Compatible and transportable
- Easy to maintain
- Supports a variety of file organizations

1.2.4 Limitations of COBOL

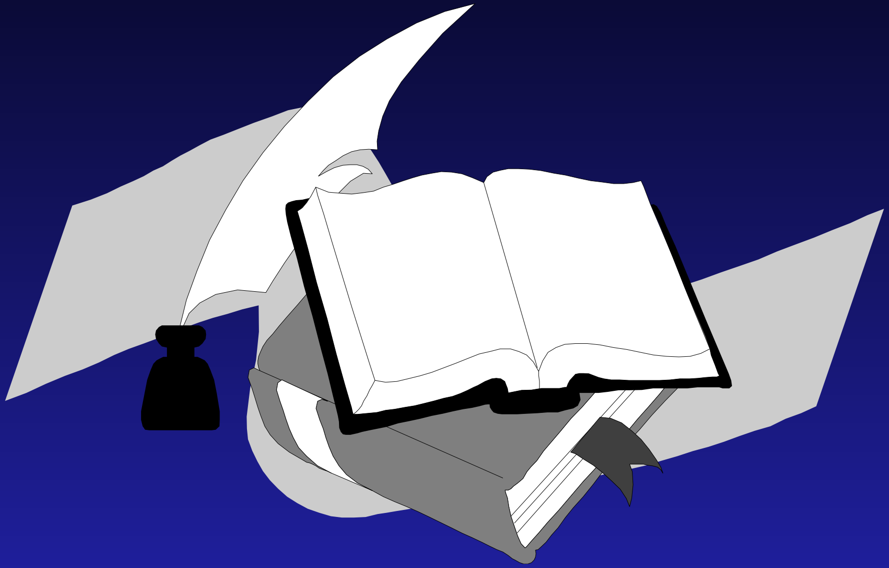
- Requires a compiler
- English like means statements can be very long
- If unstructured, can be very difficult to maintain/debug
- No Relational DBMS verbs - (SQL must be embedded/pre-compiled)

1.2.5 COBOL Preparation

COBOL code must go through a two step process to become executable

- COMPILE
  - Checks for syntax errors
  - Produces source listing of COBOL
  - Produces diagnostic listing
  - Translates COBOL statements to machine language instructions, producing an object program
- LINK
  - Brings COBOL subroutines into program object code
  - Resolves external references of programs external to program object code
  - Produces load module

1.2.6 COBOL Structure



Programming Specifics

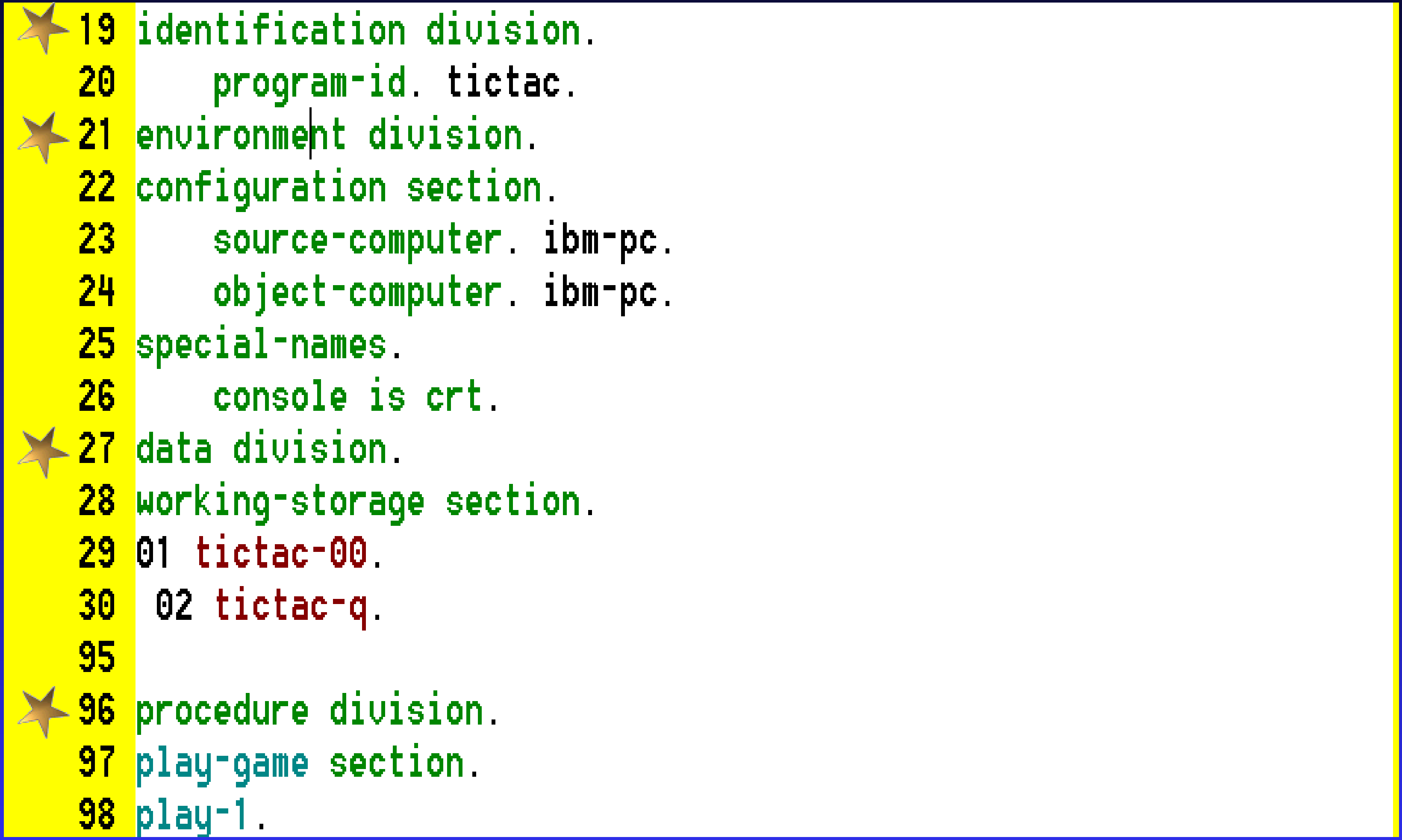
1.2.6 COBOL Structure

COBOL structure - Formal - 4 Divisions

- Identification Division
  - Identifies the program via program name, author, date written, and other pertinent information
- Environment Division
  - Describes computer hardware and external file information
- Data Division
  - Describes input, output, and work files/items
- Procedure Division
  - Contains the logical instructions

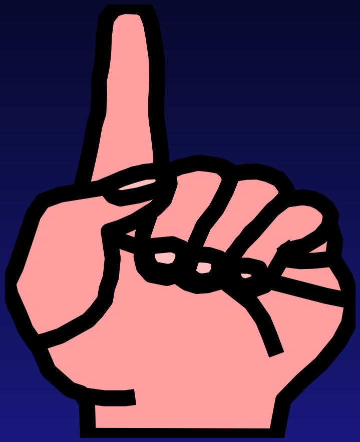
1.2.6 COBOL Structure

COBOL structure - Formal - 4 Divisions



```
identification division.
    program-id. tictac.
environment division.
configuration section.
    source-computer. ibm-pc.
    object-computer. ibm-pc.
special-names.
    console is crt.
data division.
working-storage section.
01 tictac-00.
 02 tictac-q.

procedure division.
play-game section.
play-1.
```

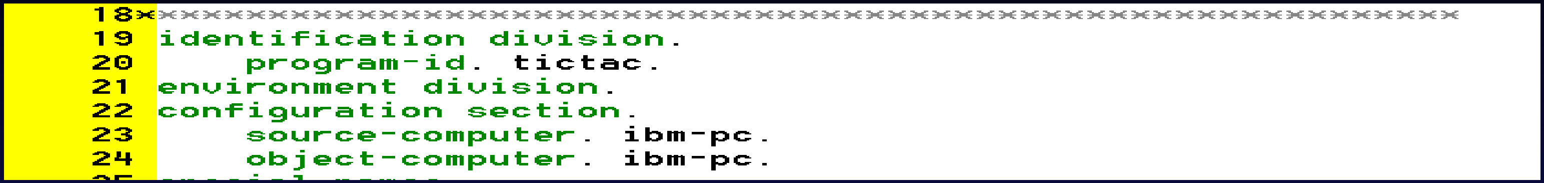
1.2.7 COBOL Columns



Coding Rules………

There are some precise rules governing COBOL coding.

1.2.7 COBOL Columns



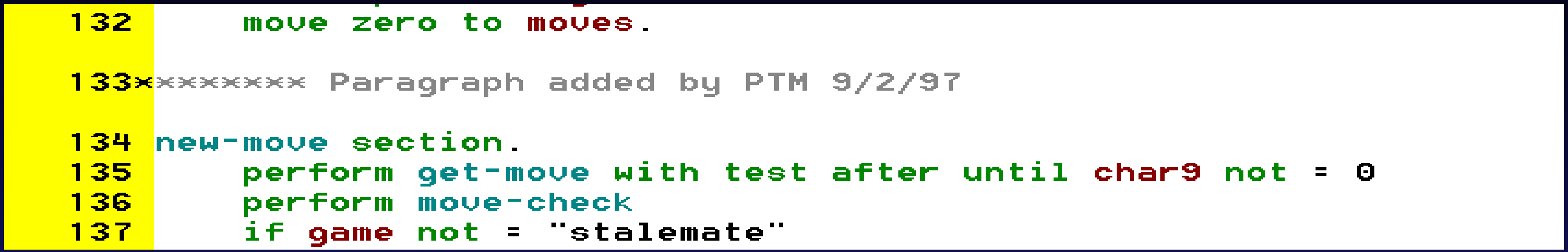
```
*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
identification division.
    program-id. tictac.
environment division.
configuration section.
    source-computer. ibm-pc.
    object-computer. ibm-pc.
```

Sequence Numbers

- Columns 1-6

- A Margin
  - Columns 8-11 - Division names, Section names, Paragraph names, File descriptions, Hi-level data items
- B Margin
  - Columns 12-72
- Identification Code
  - Columns 73-80
- Comment/Continuation
  - Column 7
    - Statements are continued in B margin
    - Splitting of literals requires hyphen in column 7 and a leading quote (’) in the B-margin

1.2.8 COBOL Lines



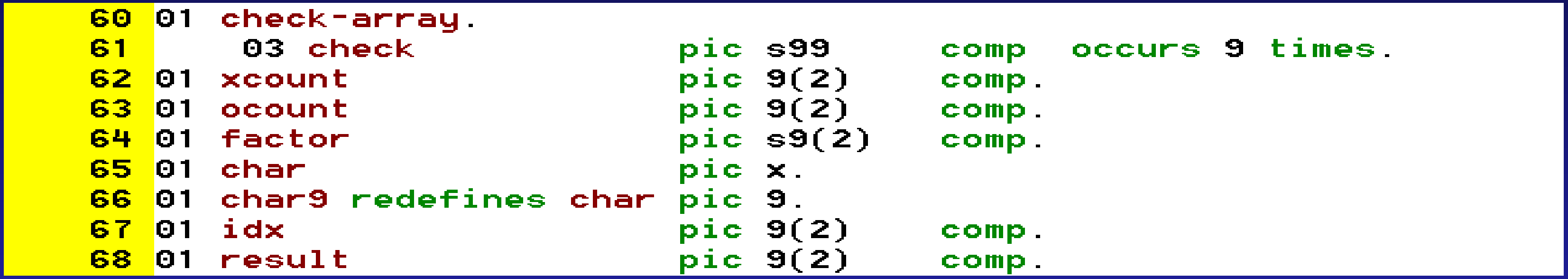
```
          move zero to moves.

      ******** Paragraph added by PTM 9/2/97

       new-move section.
           perform get-move with test after until char9 not = 0
           perform move-check
           if game not = "stalemate"
```

- Blank lines are OK
- * used in Column 7 for comment lines
- Keywords can be used to control the appearance of your ‘post compile’ listing
- Line Skips
  - SKIP1
  - SKIP2
  - SKIP3
- Paper Eject - Start New Page for your listing
  - EJECT

1.2.9 COBOL Syntax

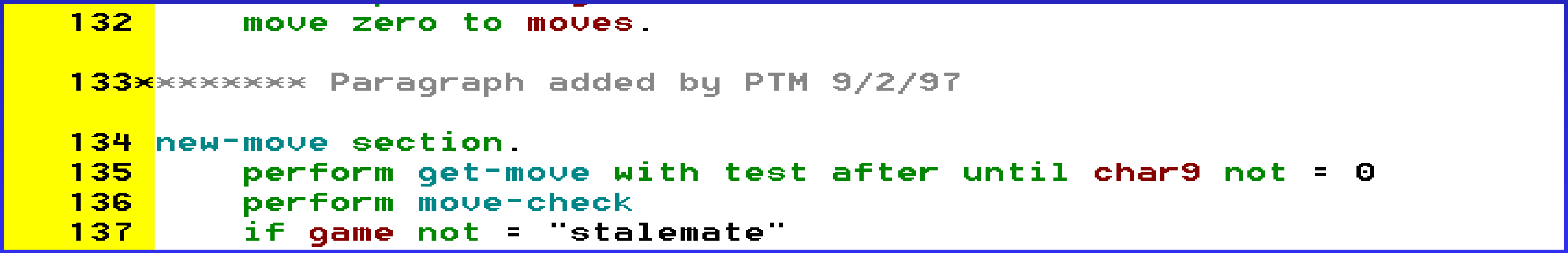
Naming conventions apply to:

Data-names



```
01 check-array.
    03 check                pic s99     comp  occurs 9 times.
01 xcount                   pic 9(2)    comp.
01 ocount                   pic 9(2)    comp.
01 factor                   pic s9(2)   comp.
01 char                     pic x.
01 char9 redefines char pic 9.
01 idx                      pic 9(2)    comp.
01 result                   pic 9(2)    comp.
```

Paragraph-names



```
     move zero to moves.

********* Paragraph added by PTM 9/2/97

 new-move section.
     perform get-move with test after until char9 not = 0
     perform move-check
     if game not = "stalemate"
```

1.2.9 COBOL Syntax

```
01 check-array.
    03 check            pic s99     comp  occurs 9 times.
01 xcount               pic 9(2)    comp.
01 ocount               pic 9(2)    comp.
01 factor               pic s9(2)   comp.
01 char                 pic x.
01 char9 redefines char pic 9.
01 idx                  pic 9(2)    comp.
01 result               pic 9(2)    comp.
```

Rules for forming data-names/paragraph-names

Not permitted

- May NOT be COBOL reserved word (refer to Appendix A in your Manual)
- May NOT contain spaces
- May NOT contain special characters other than hyphen
- May NOT begin or end with hyphen

Permitted

- May contain 1-30 characters
- May consist of alphabet (A-Z), integers (0-9), and hyphens
- Paragraph names may consist entirely of integers, but all other names MUST contain at least one alphabetic character
- SHOULD be different from all other names in THIS program (qualification is possible but not recommended)

1.3 Workshop

DO

- 1.3.1. Review Questions

Skip

- 1.3.2 Exercise

Take a Break

1.3 Workshop

1.

d. DATA DIVISION

c. IDENTIFICATION DIVISION

b. ENVIRONMENT DIVISION

a. PROCEDURE DIVISION

2.

c. A Margin

d. B Margin

e. Identification code

b. Comments/continuation

a. Sequence numbers

3.

X OUTPUT RECORD

X RATE/5

QUANTITY-ON-HAND

X TOTAL#RECORDS

INPUT-REC

X PAY$

52-PICKUP

X SUPER*

GROSS-PROFITS

X PAY_TABLE

1.3 Workshop

4. COBOL Compiler

- Checks for syntax errors
- Produces source listing of COBOL
- Produces diagnostic listing
- Translates COBOL statements to machine language instructions, producing an object program

5. COBOL Advantages

- English-like
- Solves Business Problems
- Handles large volumes of data
- Universal and standardized
- Compatible and transportable
- Easy to maintain
- Supports a variety of file organizations

6. COBOL Disadvantages

- Requires a compiler
- English like means statements can be very long
- If unstructured, can be very difficult to maintain/debug
- No Relational DBMS verbs - (SQL must be embedded/pre-compiled)

Review……

At this point we should be able to:

Describe the steps of the Programming Life Cycle

Describe the function of the four COBOL divisions

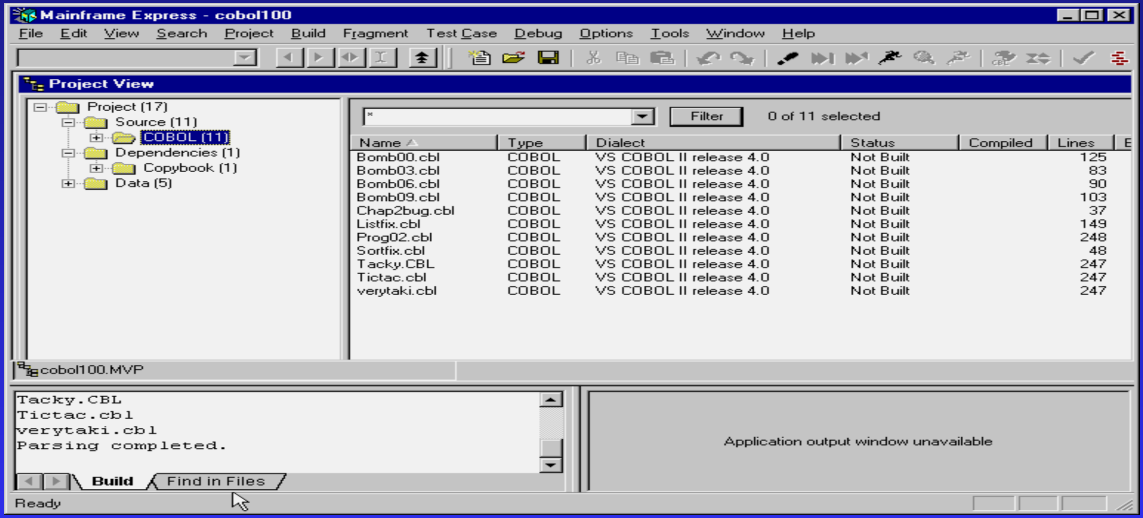
List the advantages and disadvantages of COBOL

Describe the purpose of the COBOL compiler

Understand the column structure of COBOL

Introduction to COBOL Programming

Using the Micro Focus Mainframe Express


Mainframe Express - cobol100
File Edit View Search Project Build Fragment Test Case Debug Options Tools Window Help
Project View
Project (17)
Source (11)
COBOL (11)
Dependencies (1)
Copybook (1)
Data (5)
Filter
0 of 11 selected
cobol100.MVP
Build
Find in Files
Application output window unavailable
Ready

Name	Type	Dialect	Status	Compiled	Lines
Bomb00.cbl	COBOL	VS COBOL II release 4.0	Not Built		125
Bomb03.cbl	COBOL	VS COBOL II release 4.0	Not Built		83
Bomb06.cbl	COBOL	VS COBOL II release 4.0	Not Built		90
Bomb09.cbl	COBOL	VS COBOL II release 4.0	Not Built		103
Chap2bug.cbl	COBOL	VS COBOL II release 4.0	Not Built		37
Listfix.cbl	COBOL	VS COBOL II release 4.0	Not Built		149
Prog02.cbl	COBOL	VS COBOL II release 4.0	Not Built		248
Sortfix.cbl	COBOL	VS COBOL II release 4.0	Not Built		48
Tacky.CBL	COBOL	VS COBOL II release 4.0	Not Built		247
Tictac.cbl	COBOL	VS COBOL II release 4.0	Not Built		247
verytaki.cbl	COBOL	VS COBOL II release 4.0	Not Built		247

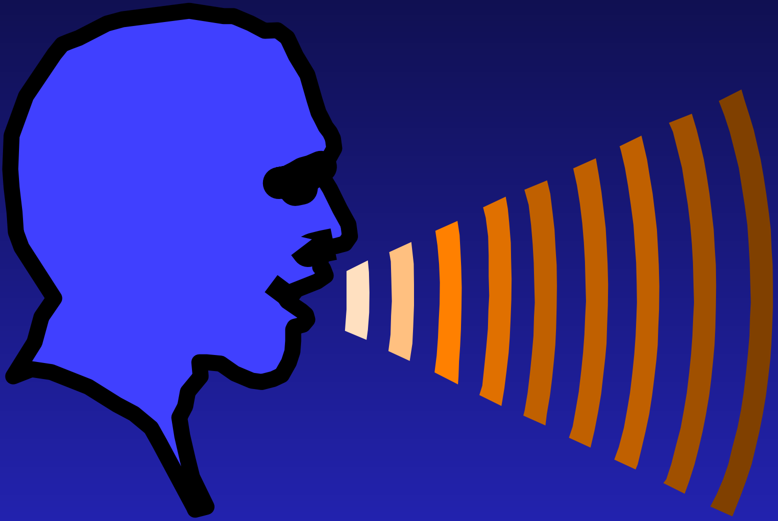
```
Tacky.CBL
Tictac.cbl
verytaki.cbl
Parsing completed.
```

What is the Mainframe Express????

An integrated , graphical COBOL application development toolset which allows you to create, maintain and support:

- Production mainframe applications
- PC-based and GUI-Client/Server applications

COBOL Compiler-Language Dialects



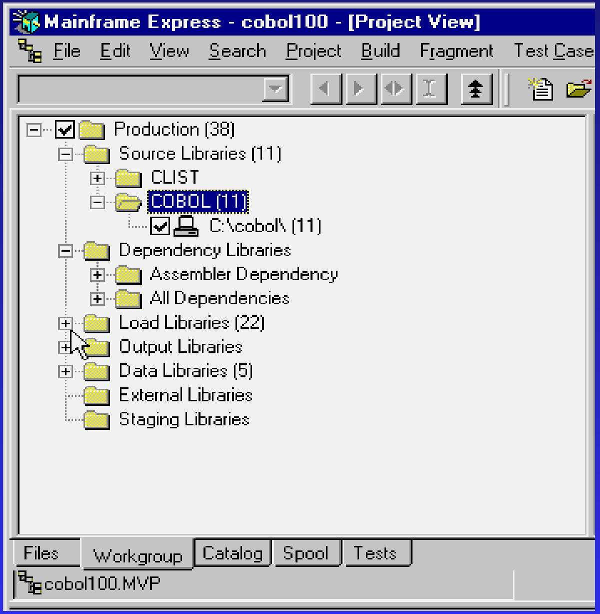
- OSVS
- VSCII
- COBOL370
- SAA-COBOL
- Object COBOL

COBOL Development Tools

↘ ***Edit***	↘ Editor
↘ ***Check***	↘ Compiler
↘ ***Animate***	↘ Testing Environment

Project Organization

- Workgroups are used to group programs, data and related files together for easy access to the testing environment


Mainframe Express - cobol100 - [Project View]
File Edit View Search Project Build Fragment Test Case
Files Workgroup Catalog Spool Tests
cobol100.MVP

```
Production (38)
  Source Libraries (11)
    CLIST
    COBOL (11)
      C:\cobol\ (11)
  Dependency Libraries
    Assembler Dependency
    All Dependencies
  Load Libraries (22)
  Output Libraries
  Data Libraries (5)
  External Libraries
  Staging Libraries
```

Shall we try it out???

Let's Edit, Check and Animate a program……

- Start the MFE
- Open our COB100 Project
  (C:\COBOL\COBOL100.MVP)
- Expand Source Folder and highlight COBOL
- Start our Edit
  - Right Click on TICTAC.CBL
  - Click Edit
  - After the program loads, click Check/Compile
- Shut Down the Workbench

Shall we try it out???

Let’s Edit, Check and Animate a program……

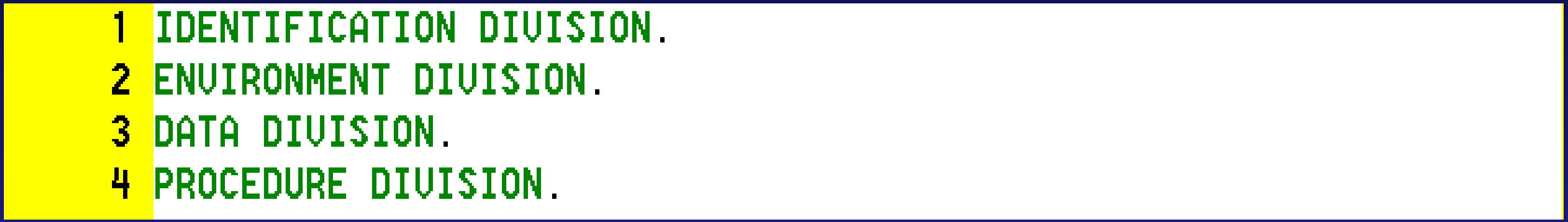
- Select Run from the Debug Menu
- Select the TSO Tab
- Enter CALL TICTAC

Now its your turn…….

Repeat the test of TICTAC.CBL on your own……..

Let’s code something new……..

- In the COBOL100 Project
  - From the File Menu select New
  - Select Source File
- You should be in an Edit Session - Code the following……..



```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
```

- When you finish coding, click
  - Save as ,under the File Menu
  - Right mouse button in the edit area and Select Add to Project
  - Check
- Clean-up any errors……ask for Help if needed…..

Let's debug a program……..

- Start the MFE
- Load COBOL100 Project
- Start our Edit/Compile Session
  - Double Click on TACKY.CBL to edit
  - Click Check/Compile
- When you encounter the first compiler error, click Zoom to finish the Compile
- Fix the Program Bug
- Recheck the Program

Quiz time…….

When using Mainframe Express, the cycle of Edit, Compile Test is referred to as:

- Edit, Compile, Debug

Name a few of the existing COBOL compiler dialects.

- OSVS, VSCII, ANSI85, SAA-COBOL, Object COBOL

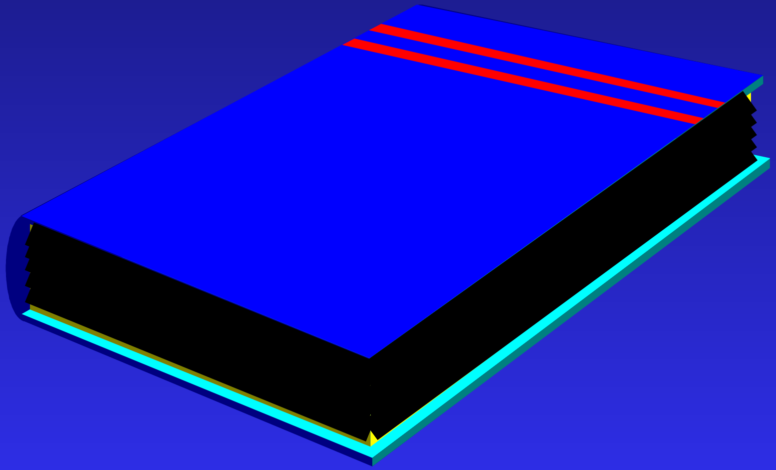
The program VERYTAKI.CBL has several errors in it. See if you can rise to the challenge and get a clean compile…….

Review……

At this point we should be able to:

- Describe the steps of the Programming Life Cycle
- Describe the function of the four COBOL divisions
- List the advantages and disadvantages of COBOL
- Describe the purpose of the COBOL compiler
- Understand the column structure of COBOL
- ***Use the Micro Focus Workbench to Edit, Syntax Check and Animate a program***

Chapter 2 Program and File Definitions



2.1 Objectives

After completing this chapter, you will understand the three COBOL divisions used to identify the program and its files (Identification, Environment, and Data Division). Specifically, you will be able to:

- Code an identification division
- Code an environment division
- Code a data division
- Tell whether statements belong in the A-margin or B-margin
- Write a record description for a file
- Process literals and figurative constants
- Describe the mainframe COBOL compiler

2.2 Topics to be covered:

- Identification division
- Environment division
- Data division
- File description
- PICTURE clause
- USAGE clause
- VALUE clause
- Literals and figurative constants
- Copy statement
- COBOL compiler and options

2.2.1 Identification Division

Documents program name and origin

- PROGRAM-ID
  - Required
  - 1-30 characters
  - Only first 8 used to uniquely identify program
- AUTHOR
- INSTALLATION
- DATE-WRITTEN
- DATE-COMPILED
- SECURITY

2.2.1 Identification Division

Example:

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. HL2COB1.
000300 AUTHOR. ALFRED E NEWMAN SALES X-9876.
000400 INSTALLATION. COMPANY B.
000500 DATE-WRITTEN. JANUARY, 1990.
000600 DATE-COMPILED.
000700 SECURITY. UNCLASSIFIED.
```

Note the use of periods

2.2.1 Work Assignment

- **Use Maineframe Express to create a new file called PROGRAM1.CBL**
- **Code the example (in your book). Use applicable notations for Program-Id, etc.**

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. PROGRAM1.
000300 AUTHOR. ALFRED E NEWMAN SALES X-9876.
000400 INSTALLATION. COMPANY B.
000500 DATE-WRITTEN. JANUARY, 1990.
000600 DATE-COMPILED.
000700 SECURITY. UNCLASSIFIED.
```

- **Check your program for syntax errors**

2.2.2 Environment Division

made up of 2 sections

CONFIGURATION SECTION

Describes computer on which program is compiled and executed

- SOURCE-COMPUTER
- OBJECT-COMPUTER

INPUT-OUTPUT SECTION

Relates each program file with external hardware device via

- FILE-CONTROL statement
  - SELECT program-file ASSIGN TO jcl-external-name
    - jcl-external-name
      - class indicator (2)
      - organization indicator (1)
      - external name (1-8)

2.2.2 Environment Division

Example:

```
000800 ENVIRONMENT DIVISION.
000900 CONFIGURATION SECTION.
001000 SOURCE-COMPUTER.  IBM-370.
001100 OBJECT-COMPUTER.  IBM-370.
001200 INPUT-OUTPUT SECTION.
001300 FILE-CONTROL.
001400 SELECT SALES-FILE-IN ASSIGN TO UT-S-SALESIN.
001500 SELECT REPORT-FILE-OUT ASSIGN TO UT-S-RPTOUT.
```

UT - Unit Tape

S - Sequential

2.2.2 Work Assignment

Add the following code to PROGRAM1.CBL

```
000800 ENVIRONMENT DIVISION.
000900 CONFIGURATION SECTION.
001000 SOURCE-COMPUTER.  IBM-370.
001100 OBJECT-COMPUTER.  IBM-370.
001200 INPUT-OUTPUT SECTION.
001300 FILE-CONTROL.
001400 SELECT SALES-FILE-IN ASSIGN TO UT-S-SYSUT1.
```

Check your program for syntax errors

****** Syntax errors for missing FD's are OK for now**

2.2.3 Data Division

Contains detailed information about all data used by your program

FILE SECTION

- describes external data

WORKING-STORAGE SECTION

- describes internal data

2.2.3 Data Division/File Section

FILE SECTION

FD File Descriptors (Logical File Definitions) - one for each file in the program

FD SALES-FILE-IN	(Describes the Data file named in the SELECT statement)
LABEL RECORDS ARE STANDARD	(Throwback to tape storage - records were either labeled or unlabeled - STANDARD for disk storage)
RECORDING MODE IS F	(fixed/variable record length)
RECORD CONTAINS 80 CHARACTERS	(# of bytes in the record)
BLOCK CONTAINS 0 RECORDS	(# of records in a block of records)
DATA RECORD IS SALES-RECORD.	(data name of the record)
01 SALES-RECORD PICTURE X(80).	(refers back to the DATA-RECORD data-name and defines the record layout)

2.2.3 Data Division

Example:

```
001600 DATA DIVISION.
001700 FILE SECTION.
001800 FD SALES-FILE-IN
001900     LABEL RECORDS ARE STANDARD
002000     RECORDING MODE IS F
002100     RECORD CONTAINS 80 CHARACTERS
002200     BLOCK CONTAINS 0 RECORDS
002300     DATA RECORD IS SALES-RECORD.
002400 01 SALES-RECORD                PICTURE X(80).
002500 WORKING-STORAGE SECTION.
002600 77 END-OF-FILE-SWITCH PICTURE X VALUE 'N'.
002700 01 PRINT-CONTROL.
002800     05 LINE-COUNTR             PICTURE 9(2) VALUE 99.
002900     05 PAGE-COUNTR             PICTURE 9(4) VALUE 0.
003000     05 LINES-PER-PAGE          PICTURE 9(2) VALUE 60.
```

2.2.3 Work Assignment

Add the following code to PROGRAM1.CBL

```
001600 DATA DIVISION.
001700 FILE SECTION.
001800 FD SALES-FILE-IN
001900     LABEL RECORDS ARE STANDARD
002000     RECORDING MODE IS F
002100     RECORD CONTAINS 78 CHARACTERS
002200     BLOCK CONTAINS 0 RECORDS
002300     DATA RECORD IS SALES-RECORD.
002400 01 SALES-RECORD                PICTURE X(78).
002500 WORKING-STORAGE SECTION.
002600 77 END-OF-FILE-SWITCH PICTURE X VALUE 'N’.
002700 01 PRINT-CONTROL.
002800     05 LINE-COUNTR             PICTURE 9(2) VALUE 99.
002900     05 PAGE-COUNTR             PICTURE 9(4) VALUE 0.
003000     05 LINES-PER-PAGE          PICTURE 9(2) VALUE 60.
```

2.2.4 Variable Length Records

RECORDING MODE IS V

RECORD CONTAINS largest #

DATA RECORDS ARE

Example:

```
DATA DIVISION.
FILE SECTION.
FD SALES-FILE-IN
LABEL RECORDS ARE STANDARD
RECORDING MODE IS V
RECORD CONTAINS 90 CHARACTERS
BLOCK CONTAINS 0 RECORDS
DATA RECORDS ARE        REGION-1-RECORD
                        REGION-2-RECORD
                        REGION-3-RECORD.
01  REGION-1-RECORD    PICTURE X(80).
01  REGION-2-RECORD    PICTURE X(40)
01  REGION-3-RECORD    PICTURE X(90).
```

2.2.5 Describing Data

- ***File: group of related records***
- File description area (FD)
  - Code an FD for each file referenced by the program
  - FD coded in the A margin
  - File name coded in the B margin
  - File Parameters coded in the B margin
  - File name must match SELECT statement in Environment Division
- ***Record: group of related fields***
- Record name
  - Follow each FD (external record description)
  - Also appear in Working-Storage (internal to the program)
  - Described as an 01 level
  - Code in A margin
- ***Field: item used for one piece of data***
- Field within record
  - 02-49 level
  - Data Name or FILLER may be used
  - Code in B margin
- Elementary item
  - One field
  - Code in A margin
  - 01 LINE-COUNTR PICTURE 9(2) VALUE 99.
- Group item
  - Higher level item composed of one or more lower level elementary items
  - 01 PRINT-CONTROL.

```
    05 LINE-COUNTR          PICTURE 9(2) VALUE 99.
    05 PAGE-COUNTR          PICTURE 9(4) VALUE 0.
```

2.2.5 Describing Data - Example

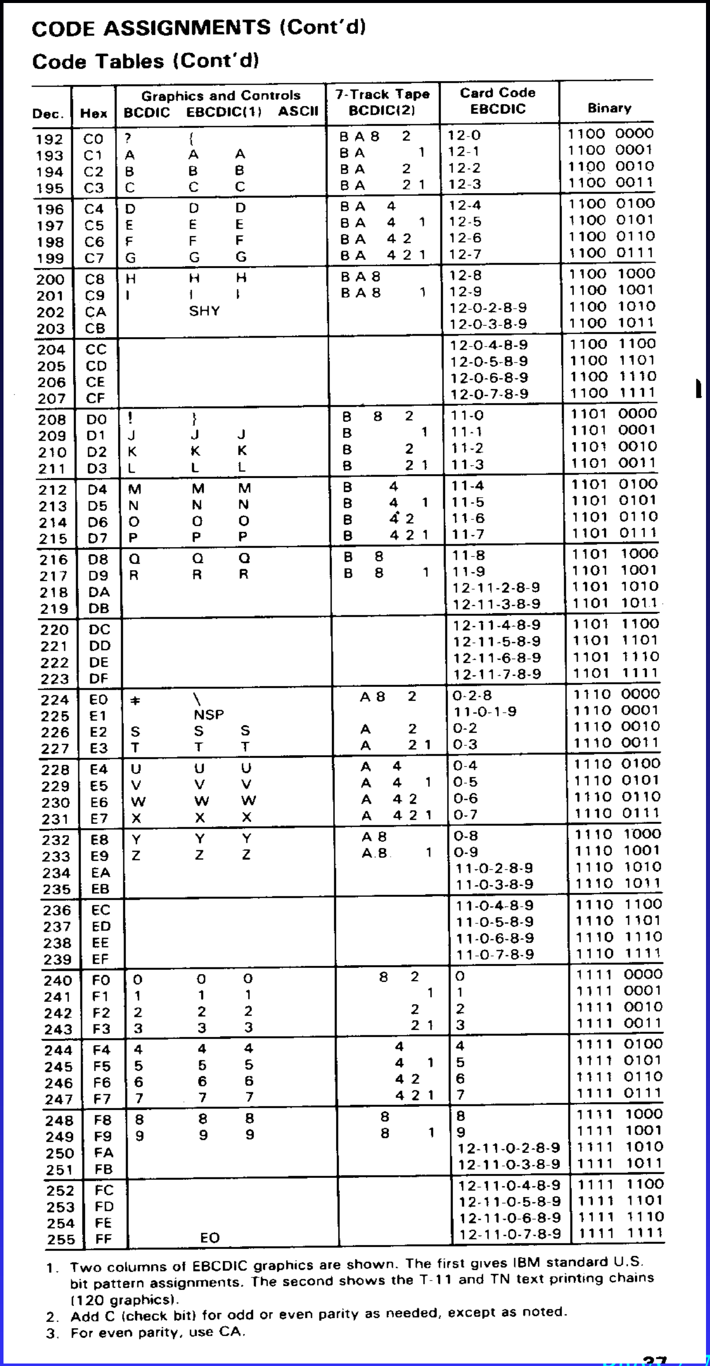
```
1      8   12
       A    B
```

```
001600 DATA DIVISION.
001700 FILE SECTION.
001800 FD  SALES-FILE-IN
001900     LABEL RECORDS ARE STANDARD
002000     RECORDING MODE IS F
002100     RECORD CONTAINS 80 CHARACTERS
002200     BLOCK CONTAINS 0 RECORDS
002300     DATA RECORD IS SALES-RECORD.
002400 01  SALES-RECORD               PICTURE X(80).
002500 WORKING-STORAGE SECTION.
002600 77 END-OF-FILE-SWITCH PICTURE X VALUE 'N'.
002700 01 PRINT-CONTROL.
002800    05 LINE-COUNTR              PICTURE 9(2) VALUE 99.
002900    05 PAGE-COUNTR              PICTURE 9(4) VALUE 0.
003000    05 LINES-PER-PAGE           PICTURE 9(2) VALUE 60.
```

2.2.5 Data Representation

- A discussion about how data is represented
  - binary
  - hex
  - bits and bytes
  - halfwords and words
    - 2bytes, 4 bytes
  - etc…………..

2.2.5 Data Representation



CODE ASSIGNMENTS (Cont'd)

Code Tables (Cont'd)

Dec.	Hex	Graphics and Controls BCDIC	EBCDIC(1)	ASCII	7-Track Tape BCDIC(2)	Card Code EBCDIC	Binary
192	C0	?	{		B A 8 2	12-0	1100 0000
193	C1	A	A	A	B A 1	12-1	1100 0001
194	C2	B	B	B	B A 2	12-2	1100 0010
195	C3	C	C	C	B A 2 1	12-3	1100 0011
196	C4	D	D	D	B A 4	12-4	1100 0100
197	C5	E	E	E	B A 4 1	12-5	1100 0101
198	C6	F	F	F	B A 4 2	12-6	1100 0110
199	C7	G	G	G	B A 4 2 1	12-7	1100 0111
200	C8	H	H	H	B A 8	12-8	1100 1000
201	C9	I	I	I	B A 8 1	12-9	1100 1001
202	CA		SHY			12-0-2-8-9	1100 1010
203	CB					12-0-3-8-9	1100 1011
204	CC					12-0-4-8-9	1100 1100
205	CD					12-0-5-8-9	1100 1101
206	CE					12-0-6-8-9	1100 1110
207	CF					12-0-7-8-9	1100 1111
208	D0	!	}		B 8 2	11-0	1101 0000
209	D1	J	J	J	B 1	11-1	1101 0001
210	D2	K	K	K	B 2	11-2	1101 0010
211	D3	L	L	L	B 2 1	11-3	1101 0011
212	D4	M	M	M	B 4	11-4	1101 0100
213	D5	N	N	N	B 4 1	11-5	1101 0101
214	D6	O	O	O	B 4 2	11-6	1101 0110
215	D7	P	P	P	B 4 2 1	11-7	1101 0111
216	D8	Q	Q	Q	B 8	11-8	1101 1000
217	D9	R	R	R	B 8 1	11-9	1101 1001
218	DA					12-11-2-8-9	1101 1010
219	DB					12-11-3-8-9	1101 1011
220	DC					12-11-4-8-9	1101 1100
221	DD					12-11-5-8-9	1101 1101
222	DE					12-11-6-8-9	1101 1110
223	DF					12-11-7-8-9	1101 1111
224	E0	‡	\		A 8 2	0-2-8	1110 0000
225	E1		NSP			11-0-1-9	1110 0001
226	E2	S	S	S	A 2	0-2	1110 0010
227	E3	T	T	T	A 2 1	0-3	1110 0011
228	E4	U	U	U	A 4	0-4	1110 0100
229	E5	V	V	V	A 4 1	0-5	1110 0101
230	E6	W	W	W	A 4 2	0-6	1110 0110
231	E7	X	X	X	A 4 2 1	0-7	1110 0111
232	E8	Y	Y	Y	A 8	0-8	1110 1000
233	E9	Z	Z	Z	A 8 1	0-9	1110 1001
234	EA					11-0-2-8-9	1110 1010
235	EB					11-0-3-8-9	1110 1011
236	EC					11-0-4-8-9	1110 1100
237	ED					11-0-5-8-9	1110 1101
238	EE					11-0-6-8-9	1110 1110
239	EF					11-0-7-8-9	1110 1111
240	F0	0	0	0	8 2	0	1111 0000
241	F1	1	1	1	1	1	1111 0001
242	F2	2	2	2	2	2	1111 0010
243	F3	3	3	3	2 1	3	1111 0011
244	F4	4	4	4	4	4	1111 0100
245	F5	5	5	5	4 1	5	1111 0101
246	F6	6	6	6	4 2	6	1111 0110
247	F7	7	7	7	4 2 1	7	1111 0111
248	F8	8	8	8	8	8	1111 1000
249	F9	9	9	9	8 1	9	1111 1001
250	FA					12-11-0-2-8-9	1111 1010
251	FB					12-11-0-3-8-9	1111 1011
252	FC					12-11-0-4-8-9	1111 1100
253	FD					12-11-0-5-8-9	1111 1101
254	FE					12-11-0-6-8-9	1111 1110
255	FF		EO			12-11-0-7-8-9	1111 1111

1. Two columns of EBCDIC graphics are shown. The first gives IBM standard U.S. bit pattern assignments. The second shows the T-11 and TN text printing chains (120 graphics).
2. Add C (check bit) for odd or even parity as needed, except as noted.
3. For even parity, use CA.

2.2.6 Picture Clause

- PICTURE (PIC) describes data TYPE and LENGTH
  - A - alphabetic
  - 9 - numeric
  - X- alphanumeric
  - V- implied decimal
  - S - sign (optional) used to capture +,- values

Examples:

```
Ol CUSTOMER-RECORD.
   05 LAST-NAME                PIC X(20).
   05 FIRST-NAME               PIC X(15).
   05 PHONE.
           10 AREA             PIC 9(3).
           10 EXCHANGE         PIC 9(3).
           10 EXTENSN          PIC 9(4).
           05 BILL-AMOUNT      PIC S9(3)V99.
           05 FILLER           PIC X(30).
```

2.2.7 Usage Clause

- USAGE (optional) describes how data is stored
  - INDEX
    - Used in table handling
  - DISPLAY
    - One character per byte ("print format") - default
  - COMPUTATIONAL (COMP) - binary
    - 1st position contains operational sign
    - 1-4 digits = 2 bytes (halfword)
    - 5-9 digits = 4 bytes (fullword)
    - 10-18 digits = 8 bytes (2 fullwords)

2.2.7 Usage Clause

- USAGE (optional) describes how data is stored
  - COMPUTATIONAL-1 (COMP-1) - short precision floating point
    - 4 bytes (fullword)
  - COMPUTATIONAL-2 (COMP-2) - long precision floating point
    - 8 bytes (doubleword)
  - COMPUTATIONAL-3 (COMP-3) - packed decimal format
    - 2 digits per byte
  - NOTE: COMP-1, COMP-2, COMP-3 are IBM extensions of ANS 74

2.2.8 Display Data

PIC X(3) VALUE ‘ABC’ USAGE DISPLAY.
PIC X(3) VALUE ‘ABC’.

| C1 | C2 | C3 |

PIC 9(4) VALUE 1234 USAGE DISPLAY.
PIC 9(4) VALUE 1234.

| F1 | F2 | F3 | F4 |

PIC S9(4) VALUE 1234 USAGE DISPLAY.
PIC S9(4) VALUE 1234.

| F1 | F2 | F3 | C4 |

PIC S9(4) VALUE -1234 USAGE DISPLAY.
PIC S9(4) VALUE -1234.

| F1 | F2 | F3 | D4 |

2.2.9 Packed data

```
PIC 9(3) VALUE 123  USAGE COMP-3.
PIC 9(3) VALUE 123  COMP-3.

| 12 | 3F |

PIC S9(4) VALUE 123 USAGE COMP-3.
PIC S9(4) VALUE 123 COMP-3.

| 00 | 12 | 3C |

PIC S9(4) VALUE -123 USAGE COMP-3.
PIC S9(4) VALUE -123 COMP-3.

| 00 | 12 | 3D |
```

2.2.10 Binary Data

```
PIC 9(4) VALUE 10 USAGE COMP.
PIC 9(4) VALUE 10 COMP.

| 00 | 0A |

PIC 9(4) VALUE 123 USAGE COMP.
PIC 9(4) VALUE 123 COMP.

| 00 | 7B |
```

2.2.11 VALUE Clause

Optional

- initializes memory
- Can't be used in file section
- Only used with elementary items

EXAMPLES

```
01 PRINT-CONTROL.
    05 LINE-COUNTER          PIC 9(2) VALUE 99.
    OS PAGE-COUNTER          PIC 9(4) VALUE ZERO.
    05 LINES-PER-PAGE        PIC 9(2) VALUE 60.
    01 HEADING-l.
    05 FILLER                PIC X(30) VALUE SPACES.
    OS FILLER                PIC X(16) VALUE 'CUSTOMER LISTING'.
    05 FILLER                PIC X(30) VALUE SPACES.
    05 PAGE-NUMBER           PIC 9(4).
```

2.2.12 Numeric Literals

- Syntax
  - 1-8 digits
  - Optional decimal (any position except last)
  - Optional sign (+ or -) - must be first

    * if sign not used, compiler assumes value is positive
  - Cannot enclose in quotes
- Storage

```
05 PAGE-COUNTER PIC 9(4) VALUE 10.
 | F0 | F0 | F1 | F0 |
05 BILL-AMOUNT      PIC 9(3)V99 VALUE 10.
 | F0 | F1 | F0 | F0 | F0 |
05 CASH-VALUE PIC S9(3)V99 VALUE +14.32.
 | F0 | F1 | F4 | F3 | C2 |
05 DEBIT-AMOUNT PIC S9(3)V99 VALUE -394.13
 | F3 | F9 | F4 | F1 | D3 |
```

2.2.13 Alphanumeric Literals

- Syntax
    - 1-120 characters
    - enclosed in quotes
    - may contain any character (except quotes)
- Storage

```
05 HEADING-3 PIC X(8) VALUE 'PAGE'.

   | P  | A  | G  | E  |    |    |    |    |
   | D7 | C1 | C7 | C5 | 40 | 40 | 40 | 40 |
05 HEADING-NUMBER PIC X(7) VALUE '898'.

   | 8  | 9  | 8  |    |    |    |    |
   | F8 | F9 | F8 | 40 | 40 | 40 | 40 |
```

2.2.14 Figurative Constants

- Compiler generated - *refer to course manual*

- do not enclose in quotes

  LOW-VALUE
  LOW-VALUES
  HIGH-VALUE
  HIGH-VALUES
  SPACE
  SPACES
  QUOTE
  QUOTES
  ZERO
  ZEROS
  ZEROES
  A L 'x'

2.2.15 COPY statement

. Brings in externally stored COBOL code at compile time

- Usually Data Division
- Saves time
- Reduces errors
- Pulled in at compile time from COBOL libraries
- Must use ‘LIB' compiler option and //SYSLIB DD statement
- COPY module-name

```
COPY SALEREC.
```

2.2.16 Compiling procs for JCL

- Purpose - check syntax of COBOL statements
  - COBUC
    - Compile using standard COBOL compiler
  - COB2UC
    - Compile using VS COBOL 11 compiler

2.2.17 Compiler files

- STEPLIB
    - Points to location of compiler program
- SYSIN
    - Compiler input - points to data set containing COBOL source
- SYSUT1
    - Compiler workspace needed by compiler
- SYSPRINT
    - Compiler report output - storage map, listings, messages
- SYSLIN
    - Object data set as output from compiler
- SYSPUNCH
    - Object data set as output from compiler
- SYSLIB
    - Optional user COBOL source libraries (for COPY command)

2.2.18 Compiling your program - mainframe

- ISPF Option 5 (BATCH)
  - Choose which COBOL compiler you want
  - Fill in a valid job card
  - Fill in options (remembered from session to session)
  - Press ENTER or PF3, as instructed to submit batch compile
  - Browse the output using Option S (SDSF)

2.2.19 Compiler Options

- Controls the outputs of the compiler
- Specified in the PARM= field of your JCL
- In Micro Focus, right click on the program to access check/compile options
  - Note: for animating (testing) a program in this class you will have to add the following compiler directive (option) to access data files
    - **ASSIGN 'EXTERNAL'**

2.2.20 Compiler JCL

- Refer to text for samples……...

2.3 Workshop

DO ALL

- 2.3 Review Questions

Extra Fun

- Debug Chap2bug.cbl

2.3 Review Questions

1

A DIVISION	A SECTION
A Level 01 entry	B Level 05 entry
B SELECT	A FD
B BLOCK CONTAINS	B PICTURE

2

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

2.3 Review Questions

3

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

4

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

2.3 Review Questions

5

| E7 | E8 | E9 | 40 |

| F3 | F8 | D4 |

| 95 | 30 | 0F |

| 00 | 03 | 7C |

| 00 | 64 |

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

2.3 Review Questions

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

2.3 Debug Chap2bug.cbl

Problems

FILE-SECTION.	Should be FILE SECTION.

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

Review……

At this point we should be able to:

- Describe the steps of the Programming Life Cycle
- Describe the function of the four COBOL divisions
- List the advantages and disadvantages of COBOL
- Describe the purpose of the COBOL compiler
- Understand the column structure of COBOL
- Use the Micro Focus Workbench to Edit, Syntax Check and Animate a program
- ***Code an identification division***
- ***Code an environment division***
- ***Code a data division***
- ***Tell whether statements belong in the A-margin or B-margin***
- ***Write a record description for a file***
- ***Process literals and figurative constants***
- ***Describe the mainframe COBOL compiler***

2.1 Objectives

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

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

3.2 Topics to be covered:

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

3.2.1 Procedure Division

- Statements
  - Combination of Words & Symbols causing Action

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

  - Sentences

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

3.2.1 Paragraphs

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

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

 PROCESS-EACH-RECORD.
```

3.2.2 Input/Output Statements

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

3.2.3 OPEN Statement

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

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

3.2.4 CLOSE Statement

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

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

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

3.2.5 READ Statement

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

3.2.5 READ Statement

- Examples

```
READ IN-EMP-FILE

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

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

3.2.6 WRITE Statement

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

3.2.6 WRITE Statement

- Examples

```
WRITE NEW-MASTER-RECORD.

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

WRITE REPORT-RECORD AFTER ADVANCING 2 LINES
```

3.2.7 ACCEPT Statement

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

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

3.2.8 DISPLAY Statement

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

```
DISPLAY 'TOTAL RECORDS = ' WS-TOTAL-RECORDS
```

3.2.9 MOVE Statement

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

3.2.9 MOVE Statement Examples

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

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

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

3.2.9 MOVE Statement Examples

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

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

 _________________
| A | R | I | S | T |
```

3.2.9 MOVE Statement Examples

```
01 INPUT-FIRST-NAME          PIC X(4).
01 OUTPUT-FIRST-NAME         PIC X(8)
    MOVE INPUT-FIRST-NAME  TO  OUTPUT-FIRST-NAME
                                         JUSTIFIED RIGHT

| A | R | I | S |

|   |   |   |   | J | A | C | K |
```

3.2.10 Numeric MOVE Statement Examples

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

```
| 5 | 6 |7 | 8 | 9 |
```

```
| 7 | 8 |9 |
```

3.2.10 Numeric MOVE Statement Examples

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

| 5 | 6 |7 | 8 | 9 |

| 6 | 7 |8 |

3.2.10 Numeric MOVE Statement Examples

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

```
| 0 | 0 |0 | 0 | 0 |

| 0 | 0 |0 |

| 00 |0 0 |00 |0 C |
```

3.2.12 GOBACK Statement

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

3.2.13 STOP RUN Statement

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

3.3 Workshop

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

```
SELECT SALES-FILE-IN ASSIGN TO UT-S-SYSUT1

          ORGANIZATION IS LINE SEQUENTIAL.
```

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

3.3 Workshop

1

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

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

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

3.3 Workshop

2

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

3

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

4

```
| F0 | F2 | F3 | F8 | F7 |
```

3.3 Workshop

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

    BLOCK CONTAINS 0 RECORDS
    DATA RECORD IS SALES-RECORD.
01  SALES-RECORD    PIC X(78).
```

```
WORKING-STORAGE SECTION.
77 END-OF-FILE-SWITCH          PICTURE X VALUE 'N'.

01  DATA-RECORD.
    05  DR-ITEM        PIC 9.
    05  DR-LASTNAME    PIC X(20).
    05  DR-FIRSTNAME   PIC X(10).
    05  DR-STREET      PIC X(20).
    05  DR-CITY        PIC X(10).
    05  DR-STATE       PIC A(2).
    05  DR-ZIP         PIC 9(5).
    05  DR-AMOUNT      PIC 9(5)V99.
    05  DR-SALESCODE   PIC 9(3).
PROCEDURE DIVISION.
MAIN.
    OPEN INPUT SALES-FILE-IN.
    READ SALES-FILE-IN INTO DATA-RECORD.
    DISPLAY DATA-RECORD.
    CLOSE SALES-FILE-IN.
```

3.3 Workshop

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

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

Review……

At this point we should be able to:

- Describe the steps of the Programming Life Cycle
- Describe the function of the four COBOL divisions
- List the advantages and disadvantages of COBOL
- Describe the purpose of the COBOL compiler
- Understand the column structure of COBOL
- Use the Micro Focus Workbench to Edit, Syntax Check and Animate a program
- Code an identification division
- Code an environment division
- Code a data division
- Tell whether statements belong in the A-margin or B-margin
- Write a record description for a file
- Process literals and figurative constants
- Describe the mainframe COBOL compiler
- ***Code file 1/0 statements (OPEN, CLOSE, READ, WRITE)***
- ***Code special 1/0 statements (ACCEPT, DISPLAY)***
- ***Perform basic data transfer (MOVE)***
- ***Detect when an end-of-file condition is reached***
- ***Create a simple COBOL program using TSO/ISPF, Micro Focus***
- ***End the program as needed (GOBACK, STOP RUN)***
- ***Compile, link, and test a simple COBOL program***
- ***Understand the function of an optimizer***

4.1 Objectives

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

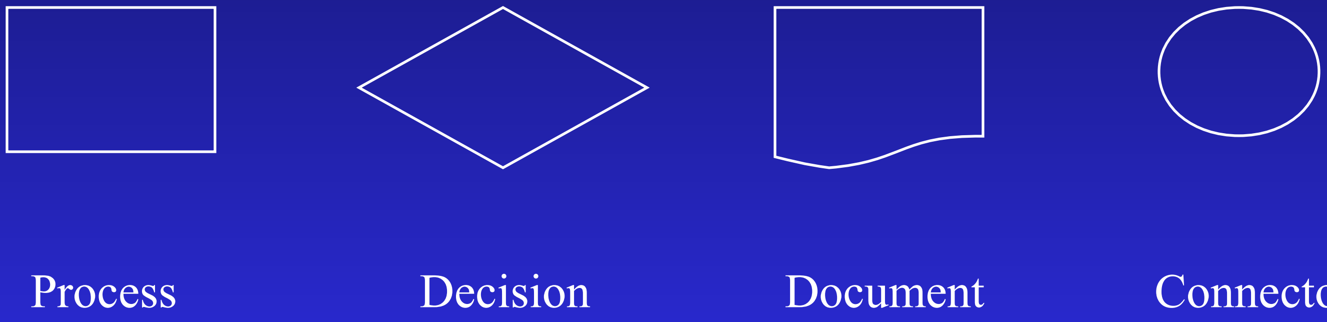
- Flowcharting Overview
- Test to determine proper action
- Unconditionally branch to another part of the Procedure Division
- Execute sequence, selection, and iteration in a COBOL program.
- Validate data for numeric contents
- Test logical conditions using AND, OR, and NOT
- Use condition names to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

4.2 Topics to be Covered

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

4.2.0 Flowcharting

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



4.2.1 GO TO

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

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

4.2.3 PERFORM

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

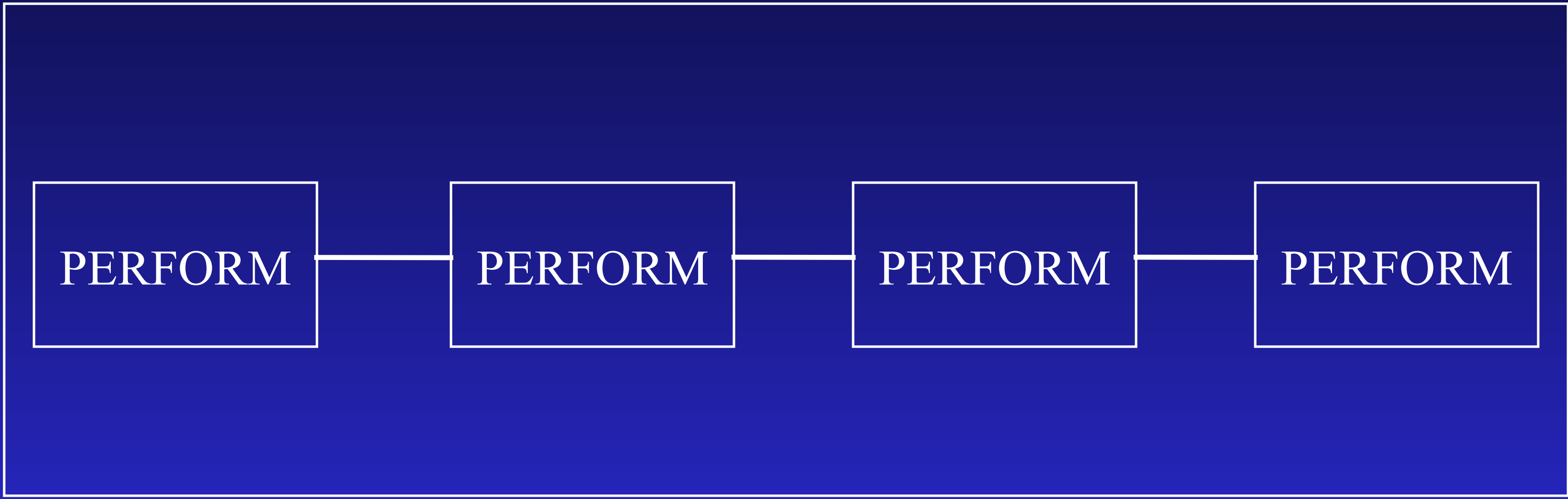
4.2.3 PERFORM

- Example

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

4.2.3 PERFORM

- Sequence Structure
  - TOP-LEVEL paragraph is an example



4.2.4 PERFORM times

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

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


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

4.2.5 PERFORM Thru

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

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

4.2.6 Exit.

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

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

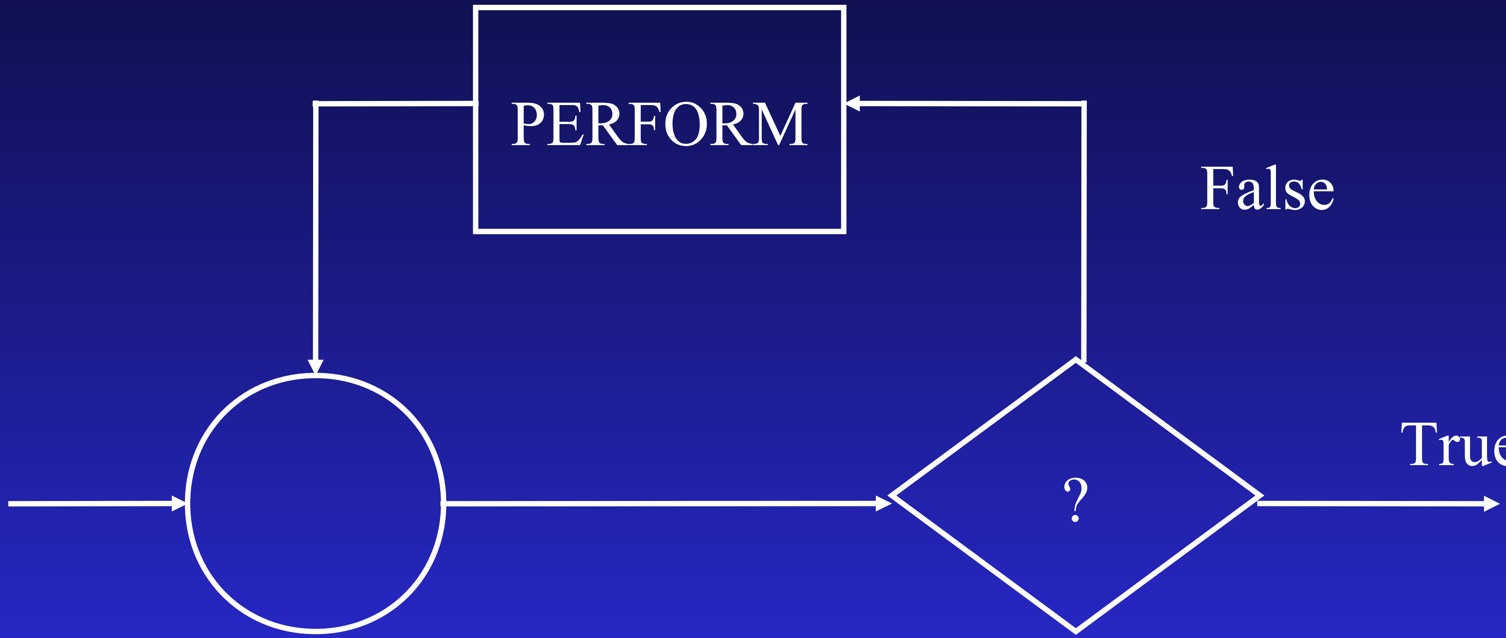
4.2.7 PERFORM Until

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

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

4.2.7 PERFORM Until

- Example of Iteration Structure



4.2.8 Condition Names

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

4.2.8 Condition Names

- Example

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

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

4.2.8 Condition Names

- Example

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

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

4.2.9 IF-THEN-ELSE

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

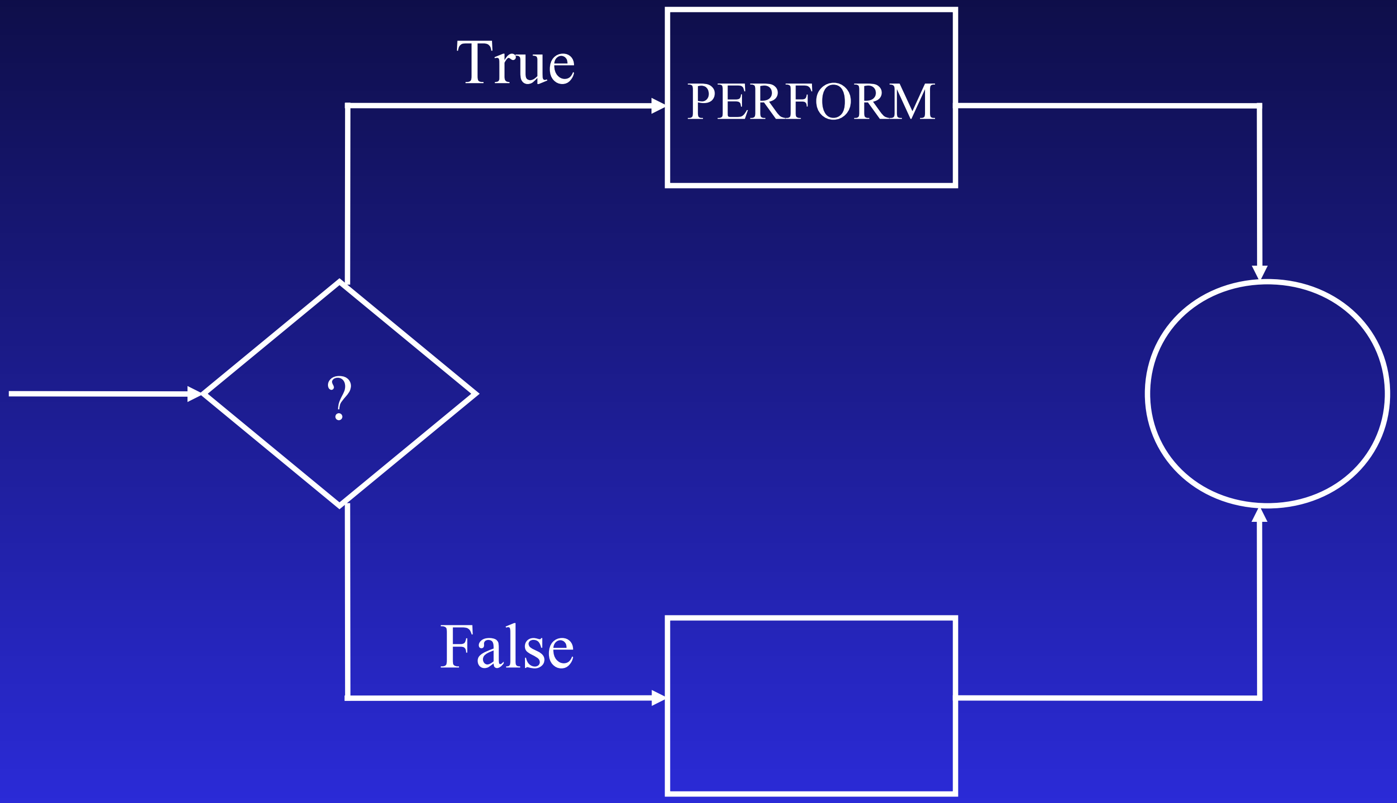
4.2.9 IF-THEN-ELSE

- Syntax

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

4.2.10 IF-THEN-ELSE

- Selection Structure



4.2.10 Class Condition

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

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

4.2.11 Sign Condition

- IF field {IS} {NOT} {POSITIVE}

```
                                  {NEGATIVE}
                                  {ZERO}
```

- Example

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

4.2.12 Relation condition

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

- Example

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

4.2.13 Condition-name condition

- IF {NOT} condition
- Example

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

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

4.2.15 Compound and negated IF-THEN-ELSE

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

4.2.15 Compound and negated IF-THEN-ELSE

- Examples

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

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

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

4.2.15 Compound and negated IF-THEN-ELSE

- Example

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

4.2.16 Truth tables

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

4.3 Workshop

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

Do *not* do page 4-29

The following replaces page 4-29

1. Make a copy of your existing program1.cbl and call it program2.cbl
2. Expand the Procedure Division to test each salesperson code to be sure it is numeric. It should only print (display) if it is numeric. Compile and test the program - one record has a non-numeric salescode.
3. Add an error counter and add 1 to the counter in the Procedure Division for each record with a non-numeric sales code. Display this counter value (should be 1) at the end of processing. Compile and test.
4. Restructure your process record routine to only print records that contain an amount sold greater than 0. Use an 88 level to test this condition. (Note: this is not an error condition). Compile and test. 1 record has a 0 value in its amount.
5. Now, lets expand processing to also write our data to an output file. Here's what you need to do…..
   - Add a select statement for the new file….
     - SELECT SALES-FILE-OUT ASSIGN TO PRNTFILE.
   - Add an FD for the new file…...
     ```
     FD  SALES-FILE-OUT
         LABEL RECORDS ARE STANDARD
         RECORDING MODE IS F
         RECORD CONTAINS 133 CHARACTERS
         BLOCK CONTAINS 0 RECORDS
         DATA RECORD IS REPORT-RECORD.
     01  REPORT-RECORD   PIC X(133).
     ```
   - Add a record description in Working-Storage for your output record
     ```
     01  SALES-REPORT.
         05  SR-LASTNAME    PIC X(20).
         05  SR-FIRSTNAME   PIC X(10).
         05  FILLER      PIC X(2).
         05  SR-SALESCODE   PIC X(3).
         05  FILLER      PIC X(2).
         05  SR-AMOUNT      PIC $$$$9.99.
     ```
   - Add the following code to your program just before or just after you Display the record
     ```
     MOVE DR-LASTNAME TO SR-LASTNAME
     MOVE DR-FIRSTNAME TO SR-FIRSTNAME
     MOVE DR-SALESCODE TO SR-SALESCODE
     MOVE DR-AMOUNT TO SR-AMOUNT
     WRITE REPORT-RECORD FROM SALES-REPORT
     DISPLAY DATA-RECORD
     ```
   - Compile and test. Your Display to the Screen should be the same. Verify that the records have been written to your output file by editing the file REPORT.DAT.

4.3 Workshop

1. c.

2. a. c. d. e. only b. is bad

3.
```
IF COUNTER-3 EQUAL 5
    THEN WRITE OUTPUT-RECORD
        ELSE DISPLAY COUNTER-3.
```

4.
```
IF CURRENT-SALES GREATER THAN 5000.00
    THEN PERFORM DOUBLE-AGENT-COMMISSN.
```

5.
```
IF CUST-AGE GREATER THAN 62
    0R (CITY EQUAL ‘TALLAHASSEE’ AND STATE EQUAL ‘FL’)
     PERFORM CALC-RTN.
```

6.
```
IF NOT MANAGER
    THEN PERFORM BONUS-RTN.
```

7.
```
05 INPUT STATE             PIC X(2).
    88 MASSACHUSETTS          VALUE ‘MA’.
    88 NEWYORK                VALUE ‘NY’.
```

8. c. sequence   b. selection   a. iteration

4.3 Workshop - Program2.cbl

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

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

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

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

Review……

At this point we should be able to:

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