Review of COBOL Coding Rules:

Columns	Use	Explanation
1-6	sequence numbers or page and line numbers (optional)	Previously used for sequence-checking when programs were punched into cards. Omit.
7	Continuation, Comment, or starting a new page	(*) for comment (/) printer skip to a new page (-) to continue nonnumeric literals
8-11	Area A	Some entries such as DIVISION, SECTION, and paragraph-names must begin in Area A.
12-72	Area B	Most COBOL entries, including PROCEDURE DIVISION sentences, are coded in Area B.
73-80	Program identification (optional)	Used to identify the program. We will omit this entry.

Type of Cobol Entries

Divisions Sections And Paragraphs begin in Area A	**Divisions** Examples: IDENTIFICATION DIVISION. ENVIRONMENT DIVISION. DATA DIVISION. PROCEEDURE DIVISION. **Sections** Examples: FILE SECTION. WORKING-STORAGE SECTION. **Paragraphs** Examples: PROGRAM-ID. 200-WAGES-ROUTINE.
Statements and Sentences	**Statements and Sentences** Examples: MOVE NAME-IN TO NAME-OUT. ADD AMT-IN TO TOTAL.

Review of Margin rules:

1. DIVISION and SECTION names
    a. Begin in area A.
    b. End with period.
    c. Must appear on a line with no other entries.
2. Paragraph-names (rules : the same rules of forming **data-names** except that a Paragraph-name can be all digits. Paragraph-name must be unique in a COBOL program)
    a. Begin in area A.
    b. End with period, which must always be followed by at least one space.
    c. Must appear on a line by themselves or with other entries.
3. Sentences
    a. Begin in area A.
    b. End with period, which must always be followed by at least one space.
    c. Must appear on a line by themselves or with other entries.
    d. A sentence consists of a statement or series of statements.

RULES FOR INTERPRETING INSTRUCTION FORMATS

1. Uppercase words are COBOL reserved words that have special meaning to the compiler.
2. Underlined words are required in the paragraph.
3. Lowercase words represented user-defined entries :
    1. 1 to 30 character.
    2. letters, digits, and hyphens (-) only.
    3. NO embedded blanks.
    4. At least one alphabetic character.
    5. May not begin or end with a hyphen.
    6. NO COBOL reserved words such as DATA, DIVISION, etc.
4. Braces {} denoted that one of the enclosed items is required.
5. Brackets [] means that clause or paragraph is optional.
6. If punctuation is specified in the format, it is required
7. the use of dots or ellipses (…) means that additional entries of the same type may be included if desired.

1. IDENTIFICATION DIVISION.

This division supplies identifying information abut the program to the computer. The IDENTIFICATION DIVISION is divided into *paragraphs* not sections. It is coded in Area A and each must be followed by a period.

Format

```
IDENTIFICTION DIVISION.
PROGRAM-ID. Program-name.
[AUTHOR. [comment-entry]…]
[INSTALLATION. [comment-entry]…]
[DATE-WRITTEN. [comment-entry]…]
[DATE-COMPILED. [comment-entry]…]
[SECURITY. [comment-entry]…]
```

Example

```
IDENTIFICATION DIVISION.
PROGRAM-ID. PROGRAM1.
AUTHOR. Husain Gholoom.
```

2. ENVIRONMENT DIVISION.

This division is the only division that is machine-dependant division of COBOL program. It supplies the information about the *computer equipment* to be used in the computer.

The Environment Division is the only division that will change significantly if the program is to be run on a different computer.

The ENVIRONMENT DIVISION compose of 2 sections:

a) Configuration Section.

This sections indicates:

1) The Source-Computer that will be used for compiling the program and
2) The Object-Computer that will be used for executing or running the program.

b) Input-Output Section.

This section supplies the information concerning the input and the output devices used in the program. This section consist of FILE-CONTROL paragraph. In this paragraph, a file name is selected for each file to be used in the program.

The FILE-CONTROL consist of SELECT statements, each of which is coded in area B followed by a period.

Format : ENVIRONMENT DIVISION.
[CONFIGURATION SECTION.]
[SOURCE-COMPUTER. [source-computer]]
[OBJECT-COMPUTER. [source-computer]]]
[INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT file-name-1
ASSIGN TO implementers-name-1
.
.
]

Example.

```
ENVIRONMENT  DIVISION.
INPUT-OUTPUT  SECTION.
FILE-CONROL.
      SELECT    Employee-File ASSIGN TO  'c:\Program1.dat'
            ORGANIZATION  IS  LINE  SEQUENTIAL
            ACCESS  MODE  IS   SEQUENTIAL.
      SELECT    Payroll-Listing  ASSIGN TO  'c:\Program1.out'
            ORGANIZATION  IS  LINE  SEQUENTIAL
            ACCESS  MODE  IS   SEQUENTIAL.
```

3. DATA DIVISION.

Part of the COBOL program that defines and describes fields, records and files in storage.
Usually, large applications such as payroll system process large volume of data. Rather than entering data from the keyboard, This data is stored in files. **File** is a collection of records. **Record** is a collection of fields contains a unit of information . **Field** is a group of consecutive positions reserved for an item of data.

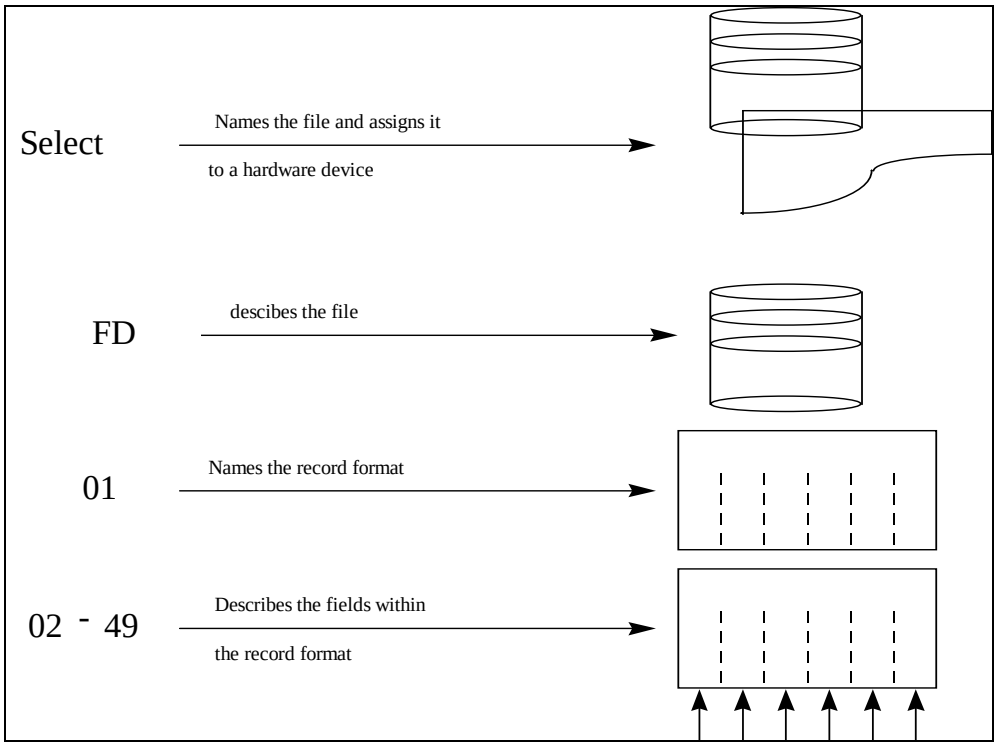
The two main sections of the data division

1. FILE SECTION : defines all input and output files.
2. WORKING-STORAGE SECTION: reserves storage for fields not part of input or output but nonetheless required for processing. These include constants, end-of-file indictors, and work areas.

Format :
```
DATA DIVISION.
[FILE SECTION.
FD file-name.
  LABEL  { RECORD IS      }    { OMITTED   }
         { RECORDS ARE }    { STANDARD }
          [RECORD CONTAINS integer-1 CHARACTERS]
         [BLOCK CONTAINS integer-2   RECORDS]  ]

  01   RECORD-NAME .
       02    field-name  PICTURE  type & size of the field.

 [WORKING-STORAGE SECTION.
     .
     .
     ]
```

Example

```
Data  Division.
     File  Section.
     FD   Student-file        Label  Records  are   standard.
     01   Student-record.
          05   Std-No              PIC   9(5).
          05   Sts-Name            PIC   X(15).
          05   Std-exam            PIC    999v99

WORKING-STORAGE  SECTION.
01    EOF   PIC    XXX  VALUE  'YES'.
```



1. FD must be followed with the file name
2. A Record-Name is coded on the 01 level
3. Other fields are coded in the levels number between 02 and 49.

ELEMENTARY AND GROUP ITEMS:

GROUP ITEM : a fields that is subdivided.(it must not have a picture).
ELEMENTARY ITEM : a fields that is NOT further subdivided (it must has a picture)

PICTURE (PIC) Clauses

1. To specify the type of data contained within an elementary item.
2. To indicate the size of the field.

CHARACTERS USED IN PICTURE CLAUSES:

A For alphabetic
X For alphanumeric
9 For numeric

Data Types : Variables and Constant Data.

Variable Data : the area described by the file description and working-storage section.

Type of constants:

Numeric literal

A numeric literal is a constants used primarily for arithmetic operations.

Rules for forming numeric literal :

1. 1 to 18 digits.
2. A + or - sign may be used, but it must appear to the **left** if the number.
3. A decimal point is permitted within the literal. The decimal point, however, may not be the rightmost character of the literal.

Nonnumeric literal

A nonnumeric or alphanumeric literal is a constant that is used in PROCEDURE DIVISION for all operations except arithmetic.

Rules for forming Nonnumeric literal :

1. The literal must be enclosed in quotation marks. (‘ ‘)
2. From 1 to 160 characters, including spaces, may be used. (only 120 character are permitted for COBOL 74 .)
3. Any character permitted in the COBOL character set may be used except the quotation mark.

Figurative Constant

A figurative constant is a COBOL reserved words that has special significance to the compiler. Such as **ZEROS** and **SPACES**.

Rules for using the WORKING-STORAGE SECTION:

1. The WORKING-STORAGE SECTION follows the FILE SECTION.
2. WORKING-STORAGE SECTION is coded on a line by itself beginning in Area A and ending with a period.
3. A group item that will be subdivided into individual storage areas as needed may then be defined. All necessary fields can be described within this 01-level entry:

```
WORKING-STORAGE SECTION.
01 WS-STORED-AREAS.
    05 ARE-THERE-MORE-RECORDS        PIC X(3).
    05 WS-GROSS-AMT                  PIC 999V99.
```

.
.
.

4. Names associated with group and elementary items must conform to the rules for forming data-names. WS- is frequently used as a prefix to denote fields as WORKING-STORAGE entries.
5. Each elementary item must contains a PIC Clause.
6. Each elementary item may contain an **initial value**, if desired:

```
WORKING-STORAGE SECTION.
01 WS-STORED-AREAS.
    05 ARE-THERE-MORE-RECORDS      PIC X(3)      VALUE
'YES'.
    05 WS-GROSS-AMT                PIC 999V99    VALUE 0.
```

VALUE clause for initializing fields may only be used in the WORKING-STORAGE SECTION, not in the FILE SECTION. Either figurative constants or literals may be used in value clauses.

3. PROCEDURE DIVISION.

The Procedure Division contains the set of instructions to be executed by the computer. Each instruction is executed in the order in which it appears in the coding sheet. First, a file must be opened, Then records are read, and operations are performed on each record. When all records are completed, the input and the output files must be closed and the program is terminated by a stop run command.

1. OPEN statement:

The open statement accesses the input and output files in a program.

Format :

```
OPEN { INPUT file-name-1…    }
     { OUTPUT file-name-2 …}
```

2.READ statement:

transmit data from the input device.

Format :

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

OR

```
READ file-name-1
     AT END
statement-1…

  [END-READ].
```

3. PERFORM……UNTIL

The PERFORM…UNTIL statements is critical for implementing the structure programming technique. First, it transfers control to the procedure or paragraph named. This named paragraph is executed repeatedly until the condition specified is met. When the condition is met, control returns to the statement directly following the PERFORM.

Format:

```
PERFORM  procedure-name-1
   UNTIL  condition-1
[END-PERFORM].
```

4. CLOSE Statement:

A CLOSE statement must be coded at the end of the job after all records have been processed to release these files and deactivate the devices.

Format : **CLOSE file-name-1 … .**

5. STOP RUN Statement.
The STOP RUN instruction tells the computer to terminate the program.

With COBOL 85, when a STOP RUN statement is executed, it will close any files that are still opened. Thus, with COBOL 85 a CLOSE statement is unnecessary.

6. WRITE Statement:
Takes data in the output area defined in the DATA DIVISION and transmit it to the device specified in the ENVIROMENT DIVISION.

Format: **WRITE record-name-1 … .**

7. ARITHMETIC AND CONDITIONAL Verbs:

7.1 ARITHMETIC Verbs:

Format :

ADD {identifier-1 } … TO identifier-2
{literal-1 }

SUBTRACT ý{identifier-1 } FROM identifier-2
{literal-1 }

MULTIPLY { identifier-1 } BY identifier-2
{ literal-1 }

DIVIDE {identifier-1 } INTO identifier-2
{literal-1 }

{ arithmetic expression-1 }
COMPUTE identifier-1 [ROUNDED]… = { literal-1 }
{ identifier-2 }
[ON SIZE ERROR imperative statement]
[END-COMPUTE]

7.2 CONDITIONAL Verbs:

IF (condition)
(statement-1)…
[ELSE
(statement-2)…]
[END_IF]

Example

```
Procedure  Division.
    Open   Input      Student-file
           Output     Student-Transcript
     Perform     Until   ARE-THERE-MORE-RECORDS = 'NO'
        Read     Student-file
                 At End    Move 'NO' to ARE-THERE-MORE-RECORDS
                 Not At  End     Perform    Paragraph-A
      End-Read
  End-Perform

     Close   Student-file
             Student-Transcript
     Stop Run .

Paragraph-A.
     move std-name to std-name-out
     move std-gpa to std-gpa-out
     write std-record  after advancing 2 lines .
```

Create your first program that reads input from data file
And display the out put on the screen

A sample Program Handout [SAMPLE1.CBL] **(page 23)**

A computer center of a large company is assigned the task of calculating weekly wages. The employee name, hourly rate, and number of hours worked are supplied as an input for each employee, and the weekly wages figure is to be computed as follows:

WEEKLY-WAGES = HOURS-WORKED X HOULY-RATE.

Input Record Layout [employee-data] from 'employee.fil' :

Employee-Name (Alphanumeric)	Hours-Worked (Numeric)	Hourly-Rate Dollars | Cents
1 20	21 22	23 24 25

Output Record Layout [payroll-listing] to 'employee.out' :

XXXXXXXXXXXX	99	9.99	999.99
(Name-Out)	Hours-Out	Rate-Out	Weekly-Wage