COBOL

LEARN COBOL

common business oriented language

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

COBOL stands for Common Business-Oriented Language. The US Department of Defense, in a conference, formed CODASYL (Conference on Data Systems Language) to develop a language for business data processing needs which is now known as COBOL.

COBOL is used for writing application programs and we cannot use it to write system software. The applications like those in defense domain, insurance domain, etc. which require huge data processing make extensive use of COBOL.

Audience

This tutorial is designed for software programmers who would like to learn the basics of COBOL. It provides enough understanding on COBOL programming language from where you can take yourself to a higher level of expertise.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of computer programming terminologies and JCL. A basic understanding of any of the programming languages will help you understand the concepts of COBOL programming and move fast on the learning track.

Copyright & Disclaimer

© Copyright 2014 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

tutorialspoint
SIMPLYEASYLEARNING

tutorialspoint
SIMPLYEASYLEARNING

tutorialspoint
SIMPLYEASYLEARNING

tutorialspoint
SIMPLYEASYLEARNING

tutorialspoint
SIMPLYEASYLEARNING

1. OVERVIEW

Introduction to COBOL

COBOL is a high-level language. One must understand the way COBOL works. Computers only understand machine code, a binary stream of 0s and 1s. COBOL code must be converted into machine code using a **compiler**. Run the program source through a compiler. The compiler first checks for any syntax errors and then converts it into machine language. The compiler creates an output file which is known as **load module**. This output file contains executable code in the form of 0s and 1s.

Evolution of COBOL

During 1950s, when the businesses were growing in the western part of the world, there was a need to automate various processes for ease of operation and this gave birth to a high-level programming language meant for business data processing.

- In 1959, COBOL was developed by CODASYL (Conference on Data Systems Language).
- The next version, COBOL-61, was released in 1961 with some revisions.
- In 1968, COBOL was approved by ANSI as a standard language for commercial use (COBOL-68).
- It was again revised in 1974 and 1985 to develop subsequent versions named COBOL-74 and COBOL-85 respectively.
- In 2002, Object-Oriented COBOL was released, which could use encapsulated objects as a normal part of COBOL programming.

Importance of COBOL

- COBOL was the first widely used high-level programming language. It is an English-like language which is user friendly. All the instructions can be coded in simple English words.
- COBOL is also used as a self-documenting language.
- COBOL can handle huge data processing.

- COBOL is compatible with its previous versions.
- COBOL has effective error messages and so, resolution of bugs is easier.

Features of COBOL

Standard Language

COBOL is a standard language that can be compiled and executed on machines such as IBM AS/400, personal computers, etc.

Business Oriented

COBOL was designed for business-oriented applications related to financial domain, defense domain, etc. It can handle huge volumes of data because of its advanced file handling capabilities.

Robust Language

COBOL is a robust language as its numerous debugging and testing tools are available for almost all computer platforms.

Structured Language

Logical control structures are available in COBOL which makes it easier to read and modify. COBOL has different divisions, so it is easy to debug.

2. ENVIORNMENT SETUP

Installing COBOL on Windows/Linux

There are many Free Mainframe Emulators available for Windows which can be used to write and learn simple COBOL programs.

One such emulator is Hercules, which can be easily installed on Windows by following a few simple steps as given below:

- Download and install the Hercules emulator, which is available from the Hercules' home site : **www.hercules-390.eu**
- Once you have installed the package on Windows machine, it will create a folder like **C:/hercules/mvs/cobol**.
- Run the Command Prompt (CMD) and reach the directory C:/hercules/mvs/cobol on CMD.
- The complete guide on various commands to write and execute a JCL and COBOL programs can be found at:

  www.jaymoseley.com/hercules/installmvs/instmvs2.htm

Hercules is an open-source software implementation of the mainframe System/370 and ESA/390 architectures, in addition to the latest 64-bit z/Architecture. Hercules runs under Linux, Windows, Solaris, FreeBSD, and Mac OS X.

A user can connect to a mainframe server in a number of ways such as thin client, dummy terminal, Virtual Client System (VCS), or Virtual Desktop System (VDS). Every valid user is given a login id to enter into the Z/OS interface (TSO/E or ISPF).

Compiling COBOL Programs

In order to execute a COBOL program in batch mode using JCL, the program needs to be compiled, and a load module is created with all the sub-programs. The JCL uses the load module and not the actual program at the time of execution. The load libraries are concatenated and given to the JCL at the time of execution using **JCLLIB** or **STEPLIB**.

There are many mainframe compiler utilities available to compile a COBOL program. Some corporate companies use Change Management tools like **Endevor**, which compiles and stores every version of the program. This is useful in tracking the changes made to the program.

```
//COMPILE   JOB,CLASS=6,MSGCLASS=X,NOTIFY=&SYSUID
//*
//STEP1     EXEC IGYCRCTL,PARM=RMODE,DYNAM,SSRANGE
//SYSIN     DD DSN=MYDATA.URMI.SOURCES(MYCOBB),DISP=SHR
//SYSLIB    DD DSN=MYDATA.URMI.COPYBOOK(MYCOPY),DISP=SHR
//SYSLMOD   DD DSN=MYDATA.URMI.LOAD(MYCOBB),DISP=SHR
//SYSPRINT  DD SYSOUT=*
//*
```

IGYCRCTL is an IBM COBOL compiler utility. The compiler options are passed using the PARM parameter. In the above example, RMODE instructs the compiler to use relative addressing mode in the program. The COBOL program is passed using the SYSIN parameter. Copybook is the library used by the program in SYSLIB.

Executing COBOL Programs

Given below is a JCL example where the program MYPROG is executed using the input file MYDATA.URMI.INPUT and produces two output files written to the spool.

```
//COBBSTEP  JOB CLASS=6,NOTIFY=&SYSUID
//
//STEP10    EXEC PGM=MYPROG,PARM=ACCT5000
//STEPLIB   DD DSN=MYDATA.URMI.LOADLIB,DISP=SHR
//INPUT1    DD DSN=MYDATA.URMI.INPUT,DISP=SHR
//OUT1      DD SYSOUT=*
//OUT2      DD SYSOUT=*
//SYSIN     DD *
//CUST1     1000
//CUST2     1001
/*
```


tutorialspoint
SIMPLYEASYLEARNING

The load module of MYPROG is located in MYDATA.URMI.LOADLIB. This is important to note that the above JCL can be used for a non-DB2 COBOL module only.

Executing COBOL-DB2 programs

For running a COBOL-DB2 program, a specialized IBM utility is used in the JCL and the program; DB2 region and required parameters are passed as input to the utility.

The steps followed in running a COBOL-DB2 program are as follows:

- When a COBOL-DB2 program is compiled, a DBRM (Database Request Module) is created along with the load module. The DBRM contains the SQL statements of the COBOL programs with its syntax checked to be correct.
- The DBRM is bound to the DB2 region (environment) in which the COBOL will run. This can be done using the IKJEFT01 utility in a JCL.
- After the bind step, the COBOL-DB2 program is run using IKJEFT01 (again) with the load library and the DBRM library as the input to the JCL.

```
//STEP001  EXEC PGM=IKJEFT01
//*
//STEPLIB  DD DSN=MYDATA.URMI.DBRMLIB,DISP=SHR
//*
//input files
//output files
//SYSPRINT DD SYSOUT=*
//SYSABOUT DD SYSOUT=*
//SYSDBOUT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DISPLAY  DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
    DSN SYSTEM(SSID)
    RUN PROGRAM(MYCOBB) PLAN(PLANNAME) PARM(parameters to cobol program) -
    LIB('MYDATA.URMI.LOADLIB')
```


tutorialspoint
SIMPLYEASYLEARNING

```
    END
/*
```

In the above example, MYCOBB is the COBOL-DB2 program run using IKJEFT01. Please note that the program name, DB2 Sub-System Id (SSID), and DB2 Plan name are passed within the SYSTSIN DD statement. The DBRM library is specified in the STEPLIB.

Try it Option Online

You really do not need to set up your own environment to start learning COBOL programming language. Reason is very simple, we have already set up COBOL Programming environment online, so that you can compile and execute all the available examples online at the same time, when you are doing your theory work. This gives you confidence in what you are reading and to check the result with different options. Feel free to modify any example and execute it online.

Try the following example using our **Try it** option available alongside the code in our website.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
PROCEDURE DIVISION.
DISPLAY 'Hello World'.
STOP RUN.
```

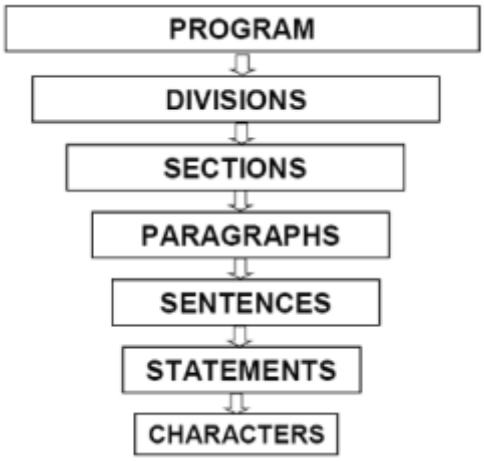
When you compile and execute the above program, it produces the following result:

```
Hello World
```

For some of the examples given in this tutorial, you will find a **Try it** option in our website code sections at the top right corner that will take you to the online compiler. So just make use of it and enjoy your learning. **Try it** option would work only with the code compatible with OpenCOBOL. The programs that require JCL (Input file, Output file or Parameters) for execution would not run on **Try it** option.

3. PROGRAM STRUCTURE

A COBOL program structure consists of divisions as shown in the following image:



A brief introduction of these divisions is given below:

- **Sections** are the logical subdivision of program logic. A section is a collection of paragraphs.
- **Paragraphs** are the subdivision of a section or division. It is either a user-defined or a predefined name followed by a period, and consists of zero or more sentences/entries.
- **Sentences** are the combination of one or more statements. Sentences appear only in the Procedure division. A sentence must end with a period.
- **Statements** are meaningful COBOL statements that perform some processing.
- **Characters** are the lowest in the hierarchy and cannot be divisible.

You can co-relate the above-mentioned terms with the COBOL program in the following example:

```
PROCEDURE DIVISION.
A0000-FIRST-PARA SECTION.
FIRST-PARAGRAPH.
```


tutorialspoint
SIMPLYEASYLEARNING

```
ACCEPT WS-ID            - Statement-1  -----|
MOVE '10' TO WS-ID      - Statement-2       |-- Sentence - 1
DISPLAY WS-ID           - Statement-3  -----|
.
```

Divisions

A COBOL program consists of four divisions.

Identification Division

It is the first and only mandatory division of every COBOL program. The programmer and the compiler use this division to identify the program. In this division, PROGRAM-ID is the only mandatory paragraph. PROGRAM-ID specifies the program name that can consist 1 to 30 characters.

Try the following example using the **Try it** option online.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
PROCEDURE DIVISION.
DISPLAY 'Welcome to Tutorialspoint'.
STOP RUN.
```

Given below is the **JCL** to execute the above COBOL program.

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
Welcome to Tutorialspoint
```

Environment Division

Environment division is used to specify input and output files to the program. It consists of two sections:

- **Configuration section** provides information about the system on which the program is written and executed. It consists of two paragraphs:

tutorialspoint
SIMPLYEASYLEARNING

- Source computer : System used to compile the program.
- Object computer : System used to execute the program.

- **Input-Output section** provides information about the files to be used in the program. It consists of two paragraphs:
  - File control : Provides information of external data sets used in the program.
  - I-O control : Provides information of files used in the program.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. XXX-ZOS.
OBJECT-COMPUTER. XXX-ZOS.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT FILEN ASSIGN TO DDNAME
ORGANIZATION IS SEQUENTIAL.
```

Data Division

Data division is used to define the variables used in the program. It consists of four sections:

- **File section** is used to define the record structure of the file.
- **Working-Storage section** is used to declare temporary variables and file structures which are used in the program.
- **Local-Storage section** is similar to Working-Storage section. The only difference is that the variables will be allocated and initialized every time a program starts execution.
- **Linkage section** is used to describe the data names that are received from an external program.

COBOL Program

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
```


tutorialspoint
SIMPLYEASYLEARNING

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT FILEN ASSIGN TO INPUT.
       ORGANIZATION IS SEQUENTIAL.
       ACCESS IS SEQUENTIAL.
DATA DIVISION.
FILE SECTION.
FD FILEN
01 NAME PIC A(25).
WORKING-STORAGE SECTION.
01 WS-STUDENT PIC A(30).
01 WS-ID PIC 9(5).
LOCAL-STORAGE SECTION.
01 LS-CLASS PIC 9(3).
LINKAGE SECTION.
01 LS-ID PIC 9(5).
PROCEDURE DIVISION.
DISPLAY 'Executing COBOL program using JCL'.
STOP RUN.
```

The **JCL** to execute the above COBOL program is as follows:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
//INPUT DD DSN=ABC.EFG.XYZ,DISP=SHR
```

When you compile and execute the above program, it produces the following result:

```
Executing COBOL program using JCL
```

Procedure Division

Procedure division is used to include the logic of the program. It consists of executable statements using variables defined in the data division. In this division, paragraph and section names are user-defined.

There must be at least one statement in the procedure division. The last statement to end the execution in this division is either **STOP RUN** which is used in the calling programs or **EXIT PROGRAM** which is used in the called programs.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WS-NAME PIC A(30).
01 WS-ID PIC 9(5) VALUE '12345'.
PROCEDURE DIVISION.
A000-FIRST-PARA.
DISPLAY 'Hello World'.
MOVE 'TutorialsPoint' TO WS-NAME.
DISPLAY "My name is : "WS-NAME.
DISPLAY "My ID is : "WS-ID.
STOP RUN.
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
Hello World
My name is : TutorialsPoint
My ID is : 12345
```


tutorialspoint
SIMPLYEASYLEARNING

4. BASIC SYNTAX

Character Set

'Characters' are lowest in the hierarchy and they cannot be divided further. The COBOL Character Set includes 78 characters which are shown below:

Character	Description
A-Z	Alphabets(Upper Case)
a-z	Alphabets (Lower Case)
0-9	Numeric
	Space
+	Plus Sign
-	Minus Sign or Hyphen
*	Asterisk
/	Forward Slash
$	Currency Sign
,	Comma
;	Semicolon

tutorialspoint
SIMPLYEASYLEARNING

.	Decimal Point or Period
"	Quotation Marks
(	Left Parenthesis
)	Right Parenthesis
>	Greater than
<	Less than
:	Colon
'	Apostrophe
=	Equal Sign

Coding Sheet

The source program of COBOL must be written in a format acceptable to the compilers. COBOL programs are written on COBOL coding sheets. There are 80 character positions on each line of a coding sheet.

Character positions are grouped into the following five fields:

Positions	Field	Description
1-6	Column Numbers	Reserved for line numbers.
7	Indicator	It can have Asterisk (*) indicating comments, Hyphen (-) indicating continuation and Slash (/) indicating form feed.

tutorialspoint
SIMPLYEASYLEARNING

8-11	Area A	All COBOL divisions, sections, paragraphs and some special entries must begin in Area A.
12-72	Area B	All COBOL statements must begin in area B.
73-80	Identification Area	It can be used as needed by the programmer.

Example

The following example shows a COBOL coding sheet:

```
000100 IDENTIFICATION DIVISION.
000100
000200 PROGRAM-ID. HELLO.
000101
000250* THIS IS A COMMENT LINE
000102
000300 PROCEDURE DIVISION.
000103
000350 A000-FIRST-PARA.
000104
000400 DISPLAY “Coding Sheet”.                                    000105
000500 STOP RUN.
000106
```

JCL to execute the above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C
//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
Coding Sheet
```


tutorialspoint
SIMPLYEASYLEARNING

Character Strings

Character strings are formed by combining individual characters. A character string can be a

- Comment,
- Literal, or
- COBOL word.

All character strings must be ended with **separators**. A separator is used to separate character strings.

Frequently used separators : Space, Comma, Period, Apostrophe, Left/Right Parenthesis, and Quotation mark.

Comment

A comment is a character string that does not affect the execution of a program. It can be any combination of characters.

There are two types of comments:

Comment Line

A comment line can be written in any column. The compiler does not check a comment line for syntax and treats it for documentation.

Comment Entry

Comment entries are those that are included in the optional paragraphs of an Identification Division. They are written in Area B and programmers use it for reference.

The text highlighted in **Bold** are the commented entries in the following example:

```
000100 IDENTIFICATION DIVISION.
000100
000150 PROGRAM-ID. HELLO.
000101
000200 AUTHOR. TUTORIALSPOINT.
000102
000250* THIS IS A COMMENT LINE
000103
```

```
000300 PROCEDURE DIVISION.
000104

000350 A000-FIRST-PARA.
000105

000360/ First Para Begins - Documentation Purpose
000106

000400  DISPLAY “Comment line”.                                    000107

000500 STOP RUN.
000108
```

JCL to execute above COBOL program:

```
//SAMPLE JOB(TESTJCL,XXXXXX),CLASS=A,MSGCLASS=C

//STEP1 EXEC PGM=HELLO
```

When you compile and execute the above program, it produces the following result:

```
Comment Line
```

End of ebook preview
If you liked what you saw…
Buy it from our store @ **https://store.tutorialspoint.com**

tutorialspoint
SIMPLYEASYLEARNING