

CO1010 – Introduction to COBOL Programming

Course Synopsis	Duration:	Five (5) days.
	Audience:	Application programmers who want to learn COBOL.
	Prerequisites:	Familiarity with z/OS JCL, TSO/ISPF and z/OS Utilities and a programming background is preferred.
	Delivery Method:	Instructor led, Hands-on workshops

Brief Description	This course provides application developers with basic knowledge in developing COBOL programs in a mainframe environment.
--------------------------	---

Course Objectives What You'll Learn	<p>Upon successful completion of this course, the student will be able to:</p> <ul style="list-style-type: none"> • Code and test COBOL batch programs • Understand the principles and practices of "Structured Coding"
--	---

Topics Covered	<p>I. Introduction to IBM Mainframes</p> <ul style="list-style-type: none"> • IBM mainframe processors <ul style="list-style-type: none"> – The basic architecture of a processor – History of System 360/370 family of processors • Input/output devices for IBM mainframes <ul style="list-style-type: none"> – Terminal display devices – Direct access storage devices – Printer devices – Magnetic tape and optical disk devices • Basic features of a mainframe operating system <ul style="list-style-type: none"> – Virtual storage – Multiprogramming – Spooling – Batch processing – Time sharing • Operating systems for IBM mainframes <ul style="list-style-type: none"> – The OS family of operating systems – Other IBM mainframe operating systems – Features of z/OS operating systems – z/OS subsystems and facilities <p>II. How to Compile and Test Programs on an IBM Mainframe</p> <ul style="list-style-type: none"> • Introduction to mainframe program development <ul style="list-style-type: none"> – How a COBOL program is compiled, link-edited, and executed on a mainframe – A general procedure for developing a COBOL program on a mainframe • How to create a source program using ISPF <ul style="list-style-type: none"> – Basic skills for working with ISPF – Creating a partitioned data set – Starting an edit session – Working in the edit data display
-----------------------	---

CO1010 – Introduction to COBOL Programming

II. How to Compile and Test Programs on an IBM Mainframe (Continued)

- Using line commands and primary commands
- Terminating an edit session
- Compile, link-edit, and execute a program
 - Introduction to jobs and Job Control Language
 - Cataloged procedures for COBOL development
 - Executing an existing program
 - Creating a sequential data set
- Using SDSF to work with jobs and job output
 - Basic skills for working with SDSF
 - How to work with jobs
 - How to work with job output
 - How to display job output
- Handling error messages
 - Correcting compile-time error messages
 - Correcting run-time error messages

III. Introduction to COBOL Programming

- COBOL platforms, standards, and compilers
 - COBOL platforms
 - COBOL standards and compilers
- An interactive COBOL program
 - An interactive session
 - The COBOL code
 - Basic coding rules
 - Identification Division
 - Environment Division
- Coding the Working-Storage Section
 - Creating data names
 - Picture clauses
 - Value clauses
 - Coding group items
- Coding the Procedure Division
 - Creating procedure names
 - Accept statements
 - Display statements
 - Move statements
 - Compute statements
 - Arithmetic expressions
 - Add statements
 - If statements
 - Perform statements
 - Perform Until statements
 - Stop Run statement
- Another interactive COBOL program
 - An interactive session
 - The COBOL code

IV. Compile, Test, and Debug a COBOL Program

- Introduction to compiling and testing
 - How a COBOL program is compiled, link-edited, and executed
 - Procedure for developing COBOL programs
- Basic skills for compiling and testing a program
 - Enter and edit a source program
 - Compile and run a program
 - Reviewing messages for compile-time errors
 - Correcting compile-time errors
 - Reviewing messages for a run-time error
 - Correcting a run-time error
 - Debugging tools can make debugging easier
 - Using COBOL statements to get debugging info

V. Writing a Program that Prepares a Report

- A simple report-preparation program
 - The file specifications
 - The report specifications
 - The COBOL code
- Coding Select statements and FD statements
 - Select statements on a mainframe
 - FD statements and record descriptions
 - For disk files
 - For print files
- Coding the Procedure Division
 - Open and Close statements
 - Read statements for sequential disk files
 - Write statements for print files
 - The Current-Date function
 - Accept to get the current date and time
- Enhanced version of the reporting-program
 - The enhanced report specifications
 - Changes to the Data Division code
 - Changes to the Procedure Division code
- Testing a report-preparation program
 - Preparing a test plan and test data Common run-timer

CO1010 – Introduction to COBOL Programming

VI. How to Design, Code, and Test a Structured Program

- Developing a structured program
 - Procedure for developing a structured program
 - The specifications for a summary report-preparation program
 - Designing a structured program
- How to design a structured program
 - Working with a structure chart
 - Naming the modules in a chart
 - Designing the first two levels of a chart
 - Designing the legs of a chart
 - Adding the Read and Write modules to a chart
 - Numbering the modules in a chart
 - Drawing charts that won't fit on one page
 - Alternatives to the structure chart
- When and how to use pseudo-code
 - The basics of using pseudo-code
 - Using pseudo-code for the critical modules of a program
 - Coding each module so it is independent
- Coding and testing a program from the top down
 - Planning the coding and testing sequence
 - Coding the new modules for each phase of testing
 - Coding program stubs
- Code for the summary report-preparation program
 - Data Division code
 - Procedure Division code
 - How COBOL programming compares to object-oriented programming

VII. How to Use the COBOL Features for Structured Coding

- Introduction to structured programming
 - The three valid structures
 - The principles of substitution and combination
 - From theory to practice
 - Shop standards for structured coding
- Coding conditions
 - Relation, sign, and class tests
 - Compound conditions
 - Condition names
- Coding selection and iteration structures
 - If statements

- Nested If statements
- Perform Until statements with tests before and after Inline Perform statements
- Perform statements with Varying and Times clauses
- Using Evaluate statements
 - Using an Evaluate statement instead of nested If statements
 - Using Evaluate statements to implement case structures
 - Complete syntax of Evaluate statement
- Other features for improved coding
 - Set to True statements
 - Structured delimiters
 - Not clauses
- A program that prepares a two-level summary report
 - The program specifications
 - The structure chart
 - COBOL code for primary control module
 - Three paragraphs written with a modern coding style

VIII. Other Ways to Define, Move, and Initialize Fields

- Defining fields in the Data Division
 - When to omit the word Filler
 - Literals and figurative constants
 - Picture, Value, and Blank When Zero clauses
 - Usage clauses
 - Redefines clauses
- How to use Move and Initialize statements
 - Move statements
 - Pictures for alphanumeric and numeric editing
 - Initialize statements
- Usages and data formats on an IBM mainframe
 - IBM mainframe extensions to the standard usages
 - Binary, hex, and EBCDIC notation
 - Primary formats for numeric data
- An enhanced report-preparation program
 - Changes to the one-level summary report
 - Changes to the Working-Storage code
 - Changes to the Procedure Division code

CO1010 – Introduction to COBOL Programming

IX. Arithmetic Statements and Intrinsic Functions

- Coding arithmetic statements
 - Compute statements
 - Arithmetic expressions
 - Add and Subtract statements
 - Multiply and Divide statements
- Using intrinsic functions
 - A summary of the six types of functions
 - How to code any function
 - The mathematical functions
 - The statistical functions
 - The financial functions

X. Working with Dates

- Accept statements and the date functions
 - Accept Date and Accept Time statements
 - Accept Day and Accept Day-Of-Week statements
 - Functions for working with dates
- The Y2K problem and its solutions
 - The Y2K problem
 - The Y2K solutions: date expansion and century windowing
- IBM's Millennium Language Extensions
 - Defining date fields with the Date Format clause
 - Working with date fields in the Procedure Division

XI. Working with Characters

- Reference modification
- Functions for working with characters
- String statements
- Unstring statement
- Inspect statement
 - With the Tallying clause
 - With the Replacing clause
 - With the Tallying and Replacing clauses
 - With the Converting clause
- Two illustrative routines
 - An editing routine
 - A conversion routine

XII. Working with Tables

- Using subscripts to work with one-level tables
 - Defining a one-level table that uses subscripts
 - Initializing a table
 - Defining a table with constant values
 - Referring to entries in a one-level table using subscripts

- Loading a one-level table using subscripts
- Searching a one-level table using subscripts
- Using subscripts to work with multi-level tables
 - Defining a multi-level table
 - Referring to entries in a multi-level table using subscripts
 - Loading a two-level table using subscripts
 - Searching a two-level table using subscripts
- Using indexes to work with tables
 - Defining a table with an index
 - Referring to table entries using an index
 - Loading a table using an index
 - Coding the Set statement
 - Performing a sequential search
 - Performing a binary search
 - Searching a two-level table using indexes
- Working with variable-length tables
 - Defining a variable-length table
 - Loading a variable-length table
 - Searching a variable-length table
- Other features for working with tables
 - Perform Varying statement to vary more than one index or subscript
 - Intrinsic functions with tables

XIII. Copy Members and Subprograms

- Copy members
- The Copy statement
- Guidelines for using copy members
- Subprograms
- Calling a subprogram
- Writing a subprogram
- Example of a calling program and subprogram
- Guidelines for using subprograms

XIV. How to Become an Effective Maintenance Programmer

- Statements and features you should be aware of
 - Perform Thru and Go To statements
 - Go To Depending statements
 - Section names in the Procedure Division
 - Qualification and the Move, Add, and Subtract Corresponding statements
 - Declarative section
- Basic skills for maintaining programs
 - A structured procedure for maintaining a program
 - What to change and where to change it
 - How to document the changes in the source code