

# COBOL

---



Jean Longhurst / Audrey Longhurst

---

# ***COBOL***

---

*Jean Longhurst*

Harper College, Palatine, IL

*Audrey Longhurst*

Motorola, Inc.



PRENTICE HALL Englewood Cliffs, New Jersey 07632

*Library of Congress Cataloging-in-Publication Data*

LONGHURST, JEAN.  
COBOL.

Includes index.

1. COBOL (Computer program language) I. Longhurst,  
Audrey. II. Title.  
QA76.73.C25L66 1988 005.13'3 87-35683  
ISBN 0-13-139387-1

Editorial/production supervision: Allison DeFren and Arthur Maisel  
Interior design: Jayne Conte and Maureen Eide  
Cover design: Maureen Eide  
Manufacturing buyers: Barbara Kittle and Lorraine Fumoso  
Page layout: Karen Noferi



© 1989 by Prentice-Hall, Inc.  
A Division of Simon & Schuster  
Englewood Cliffs, New Jersey 07632

*All rights reserved. No part of this book may be  
reproduced, in any form or by any means,  
without permission in writing from the publisher.*

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

**ISBN 0-13-139387-1 01**

Prentice-Hall International (UK) Limited, *London*  
Prentice-Hall of Australia Pty. Limited, *Sydney*  
Prentice-Hall Canada Inc., *Toronto*  
Prentice-Hall Hispanoamericano, S.A., *Mexico*  
Prentice-Hall of India Private Limited, *New Delhi*  
Prentice-Hall of Japan, Inc., *Tokyo*  
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*  
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

# *Preface*

---

Three of the most troublesome topics for beginning COBOL students are programming logic, debugging, and tables. This text attempts to deal more fully with these topics.

The text is built around one business system, an order system. Our objective is to allow the student to become so familiar with the files used they can concentrate on the COBOL instead of a variety of record descriptions. Using a variety of program design tools, the system builds from simple listings through extracts, control breaks, table handling, edits, updates, and sorts.

Debugging is emphasized from diagnostics to data exceptions. For the more common diagnostics a list of items the student should check is provided. For data exceptions and usage emphasis on IBM storage types is used. The use of trace and display are encouraged for solving logic problems.

There never seems to be enough material in a text about tables to satisfy a beginning student. We have tried to include more information about structure, loading, accessing, and printing tables with the hope the student will be able to find an example that fits the current problem they are trying to program.

Material on the 1985 standard is included as the associated topic is covered. It is, however, differentiated so it can be used or not based on its availability.

We wish to thank the following, who reviewed the manuscript of this book: Gail L. Kroepel, DeVry Institute of Technology (Chicago); J. Patrick Fenton, West Valley College (Saratoga, California); Henry J. Walker, State University of New York at Farmingdale; Rose M. Laird, Northern Virginia Community College (Annandale); Barbara Harris, DeVry Institute of Technology (Chicago); Ron Teemley, DeVry Institute of Technology (Irving, Texas); Beverly Bilshausen, College of DuPage (Glen Ellyn, Illinois); Lou R. Goodman, University of Wisconsin-Madison; and E. Gladys Norman, Linn-Benton Community College (Albany, Oregon). We also want to thank members of the staff at Prentice Hall: Dennis Hogan, editor; Jayne Conte and Maureen Eide, designers; and Allison DeFren and Arthur Maisel, production editors.

Jean Longhurst  
Audrey Longhurst

*Note:* COBOL is an industry language and is not the property of any company or group of companies, or of any organization or group of organizations.

No warranty, expressed or implied, is made by any contributor or by the CODASYL Programming Language Committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.

The authors and copyright holders of the copyrighted material used herein

FLOW-MATIC (trademark of Sperry Rand Corporation), Programming for the UNIVAC® I and II, Data Automation Systems copyrighted 1958, 1959, by Sperry Rand Corporation; IBM Commercial Translator Form No. F 28-8013, copyrighted 1959 by IBM; FACT, DSI 27A5260-2760, copyrighted 1960 by Minneapolis-Honeywell

have specifically authorized the use of this material in whole or in part, in the COBOL specifications. Such authorization extends to the reproduction and use of COBOL specifications in programming manuals or similar publications.

IBM is a registered trademark of International Business Machines, Inc.

---

## **BRIEF CONTENTS**

- 1** Introduction to Programming in COBOL
- 2** The Identification, Environment, and Data Divisions
- 3** The Procedure Division
- 4** Extraction
- 5** Control Breaks, I
- 6** Control Breaks, II
- 7** Table Handling, I
- 8** Table Handling, II
- 9** Maintenance of Sequential Files
- 10** Sorting

Appendix A: Reserved Words

Appendix B: COBOL Formats

# Contents

## Preface xv

<b>CHAPTER 1</b>	<b>Introduction to Programming in COBOL 1</b>
	INTRODUCTION TO COBOL 1
	The History of COBOL
	Characteristics of COBOL
	PROGRAM DEVELOPMENT 2
	The Development Process
	<i>Analysis Design Coding Testing</i>
	Analysis Tools and Techniques
	<i>Program Specifications Input/Output Layouts System Flowcharts</i>
	Structured Programming
	Design Tools
	<i>Flowcharts Pseudocode Hierarchy Chart</i>
	DATA ORGANIZATION 14
	File Structure
	<i>Characters Field Record File</i>
	Access Methods
	<i>Sequential Access Nonsequential Access Database</i>
	Storage Media
	<i>Disk Storage Tape Storage Paper Cards</i>
	Physical Organization
	<i>Label Records Blocking</i>
	File Description Techniques
	<i>Input/Output Record Layouts Printer Layout</i>
	COBOL TOPICS 19
	The Compilation Process
	Overview of COBOL
	<i>The Four Divisions COBOL Elements COBOL Coding Forms</i>
	SUMMARY 21
	EXERCISES 22

<b>CHAPTER 2</b>	<b>The Identification, Environment, and Data Divisions 23</b>
	INTRODUCTION 23
	PROGRAM SPECIFICATION: LISTING 23
	THE COBOL PROGRAM (FIRST THREE DIVISIONS) 24
	IDENTIFICATION DIVISION 25
	PROGRAM-ID
	AUTHOR

INSTALLATION	
DATE-WRITTEN	
DATE-COMPILED	
SECURITY	
Comment Lines	
ENVIRONMENT DIVISION	32
CONFIGURATION SECTION	
INPUT-OUTPUT SECTION	
DATA DIVISION	33
FILE SECTION	
File-Description Entry	
LABEL RECORDS Clause	
RECORD CONTAINS Clause	
DATA RECORDS Clause	
Record-Description Entry	
Level Numbers	
Data-Names	
PICTURE Clause	
Data Formats	
WORKING-STORAGE SECTION	
VALUE Clause	
DEBUGGING	47
Spaces and Hyphens	
SUMMARY	49
EXERCISES	51
PROJECTS	53

## CHAPTER 3 The Procedure Division 59

PROGRAM SPECIFICATION: LISTING	59
LOGIC FOR SIMPLE LISTING	59
THE COBOL PROGRAM	69
PROCEDURE DIVISION	72
Organization	
Notation	
COBOL Verbs	
The PERFORM Statement	
The OPEN Statement	
The READ Statement	
The MOVE Statement	
The WRITE Statement	
The CLOSE Statement	
The STOP Statement	
The GO TO Statement	
DEBUGGING	91
Diagnostics	
SUMMARY	93
EXERCISES	95
PROJECTS	98

## CHAPTER 4 Extraction 104

PROGRAM SPECIFICATION: EXTRACT	104
LOGIC FOR EXTRACT PROGRAM	104
THE COBOL PROGRAM	117
NEW COBOL ELEMENTS	121
New WORKING-STORAGE Fields	
Output Editing: Supression of Zeros	
IF/ELSE	
Decision Structures	
The COMPUTE Statement	
Arithmetic Expressions	
Arithmetic Operators	
Priority of Operations	
The ADD Statement	
The SUBTRACT Statement	
The MULTIPLY Statement	
The DIVIDE Statement	
CURRENT-DATE	



PROGRAM SPECIFICATION; LEAD-TIME	143
LOGIC FOR EXTRACT WITH CALCULATIONS	143
THE COBOL PROGRAM	150
NEW COBOL ELEMENTS	154
AND/OR Logic	
DEBUGGING	154
Diagnostics	
EBCDIC	
USAGE IS DISPLAY	
SUMMARY	158
EXERCISES	160
PROJECTS	163

## CHAPTER 5 Control Breaks, I 172

PROGRAM SPECIFICATION: BREAK-1	172
LOGIC FOR SINGLE-LEVEL BREAK AND FINAL TOTALS	172
THE COBOL PROGRAM	181
NEW COBOL ELEMENTS	186
Signed data	
SIGN condition	
Output Editing	
<i>Simple Insertion</i>	
<i>Special Insertion</i>	
<i>Fixed Insertion</i>	
<i>Floating Insertion</i>	
<i>Suppression of Zeros and Replacement with Asterisks</i>	
BLANK WHEN ZERO	
JUSTIFIED RIGHT	
PROGRAM SPECIFICATION: BREAK-2	191
LOGIC FOR SINGLE-LEVEL BREAK WITH TOTALS	192
THE COBOL PROGRAM	202
NEW COBOL ELEMENTS	202
Condition-Names	
DEBUGGING	209
Packed Data	
TRACE	
DISPLAY	
SUMMARY	214
EXERCISES	216
PROJECTS	217

## CHAPTER 6 Control Breaks, II 224

PROGRAM SPECIFICATION: BREAK-3	224
LOGIC FOR MULTIPLE-LEVEL BREAKS WITH TOTALS,	224
THE COBOL PROGRAM	235
PROGRAM SPECIFICATION: BREAK-4	235
LOGIC FOR EXTRACT WITH MULTIPLE-LEVEL BREAKS	
AND TOTALS	242
THE COBOL PROGRAM	25
NEW COBOL ELEMENTS	256
REDEFINES	
DATA RECORDS ARE	
DISPLAY	
ACCEPT	
Qualification	
CORRESPONDING	
SIZE ERROR	
GO TO/DEPENDING ON	

DEBUGGING	262
Data Exceptions	
SUMMARY	264
EXERCISES	265
PROJECTS	268

## CHAPTER 7 **Table Handling, I 275**

PROGRAM SPECIFICATION: TABLE-1	275
LOGIC FOR TABLE STORAGE OF TOTALS	279
THE COBOL PROGRAM	279
NEW COBOL ELEMENTS	295
Table Structure: OCCURS	
<i>One-Dimensional Tables Two-Dimensional Tables Three-Dimensional Tables Accessing Table Data Subscript Range</i>	
Overview of Table Handling	
<i>Loading Tables Accessing Tables Loading an Execution-Time Table</i>	
Writing Out a Table	
<i>PERFORM . . . VARYING Writing a One-Dimensional Table</i>	
<i>Writing a Two-Dimensional Table Writing a Three-Dimensional Table</i>	
DEBUGGING	321
Diagnostics	
USAGE IS COMPUTATIONAL	
<i>Positive Numbers Negative Numbers</i>	
SUMMARY	323
EXERCISES	324
PROJECTS	327

## CHAPTER 8 **Table Handling, II 334**

PROGRAM SPECIFICATION: EDIT	334
LOGIC FOR INPUT-EDIT USING TABLES	336
THE COBOL PROGRAM	351
NEW COBOL ELEMENTS	358
Input Editing	
<i>Class Range Reasonableness Existence Consistency</i>	
Indexing	
<i>The SET Statement</i>	
Loading Tables	
<i>Compile-Time Tables Preprocess Tables</i>	
Searching Tables	
<i>Serial Search Binary Search Programmer-Written Searches</i>	
DEBUGGING	373
Diagnostics	
Testing Considerations	
Test Data	
SUMMARY	374
EXERCISES	376
PROJECTS	377

## CHAPTER 9 **Maintenance of Sequential Files 387**

INTRODUCTION	387
PROGRAM SPECIFICATION: SHIPMENT	387
LOGIC FOR SEQUENTIAL-FILE MAINTENANCE WITH CHANGES ONLY	389

THE COBOL PROGRAM	406
PROGRAM SPECIFICATION: ORDER-MAINT	412
LOGIC FOR SEQUENTIAL-FILE MAINTENANCE WITH ADDS, CHANGES, AND DELETES	414
THE COBOL PROGRAM	440
SUMMARY	448
EXERCISES	448
PROJECTS	449

## CHAPTER 10 **Sorting 460**

SORTS	460
COBOL SORT	462
Sort Files	
Sort Keys	
File Handling Procedures	
INPUT PROCEDURE    USING    OUTPUT PROCEDURE    GIVING	
Combinations	
SAMPLE PROGRAM 1: INPUT PROCEDURE/OUTPUT	
PROCEDURE	467
PROGRAM SPECIFICATION: SORT-1	468
LOGIC FOR INPUT PROCEDURE/OUTPUT PROCEDURE SORT	470
THE COBOL PROGRAM	476
SAMPLE PROGRAM 2: USING/GIVING	476
LOGIC FOR USING/GIVING SORT	481
PROGRAM SPECIFICATION: SORT-2	481
THE COBOL PROGRAM	482
SAMPLE PROGRAM 3: INPUT PROCEDURE/GIVING	485
PROGRAM SPECIFICATION: SORT-3	485
LOGIC FOR INPUT PROCEDURE/GIVING SORT	486
THE COBOL PROGRAM	487
SAMPLE PROGRAM 4: USING/OUTPUT PROCEDURE	493
PROGRAM SPECIFICATION: SORT-4	493
LOGIC FOR USING/OUTPUT PROCEDURE SORT	494
THE COBOL PROGRAM	497
DEBUGGING	508
Diagnostics	
EXHIBIT	
SUMMARY	509
EXERCISES	510
PROJECTS	512

## **Appendix A: Reserved Words 519**

## **Appendix B: COBOL Formats 521**

## **Index 531**

# *Introduction to Programming in COBOL*

This chapter presents introductory material which is necessary to begin learning the COBOL programming language. It includes an introduction to COBOL, program development, data organization, and COBOL topics.

The introduction to COBOL covers the history and characteristics of COBOL.

Program development is the process by which programs are created. Coverage of program development includes the development process, analysis tools and techniques, structured programming, and design tools.

Data organization involves the storage of data on files, including a description of the files, access methods, storage media, physical considerations, and file description techniques.

Included among COBOL topics are basic information about COBOL, the compilation process, and an overview of the COBOL language.

## *INTRODUCTION TO COBOL*

### **The History of COBOL**

The development of COBOL began in 1959 with the creation of the Conference on Data System Languages (CODASYL). CODASYL was created because of a need for a common programming language for use in business applications. CODASYL's objective was to create a language which would not be the property of any particular computer manufacturer, yet could be adopted by a wide range of manufacturers. In 1960 CODASYL published a document which described its newly developed language, COBOL. (COBOL is an acronym for Common Business Oriented Language.)

In 1968 the first standard version of COBOL was approved (USA Standard COBOL 1968). The 1968 standard was revised and updated, and in 1974 a new standard was approved (American National Standard COBOL 1974). In 1985 there was another revision, and a new standard was approved (American National Standard COBOL 1985). It is this revision process which keeps the COBOL language current.

### **Characteristics of COBOL**

COBOL is an English-like programming language. It uses English words and relatively few symbols. Because of this, it is a very readable language. To an extent, nonprogrammers can read and understand a COBOL program. This also allows a COBOL program to be self-documenting. It is usually a simple matter to determine the function of a COBOL program from an examination of the program itself. Therefore, a COBOL program requires little additional documentation.

COBOL is a business-oriented language and has extensive data and file-handling capabilities. COBOL's ability to manipulate data makes it a good language for dealing with the vast amounts of data which businesses process. COBOL's ability to handle numbers is not as strong. Where extensive and complex calculations are required, as in scientific applications, there are other languages which are better suited to the task.

## **PROGRAM DEVELOPMENT**

### **The Development Process**

In creating a computer program, the development process is the same regardless of which programming language is used. The development process may be divided into the phases of (1) analysis, (2) design, (3) coding, and (4) testing. Depending on the size and complexity of the programming project, a different person (or group of people) might be responsible for each of the phases, or one person might complete the entire process alone.

#### **Analysis**

The development process begins with a problem to be solved. The analyst meets with the people who have requested a computerized solution to their problem to determine their requirements. He or she then analyzes the current existing and computerized systems that relate to the problem. With this information, the analyst determines the input, output, and processing requirements for the program. In general, the analyst will produce documents which describe these requirements. These documents usually include program specifications, input layouts, output layouts, and system flowcharts. These documents are described later in this chapter.

#### **Design**

The designer begins with the information gathered during the analysis phase and develops the logic which will be used to solve the problem. The processing requirements are divided into smaller and smaller pieces until the logical flow is described for the entire program.

Several design tools are available which are useful for describing program logic. These tools include flowcharts, pseudocode, Warnier-Orr diagrams, Nassi-Schneiderman charts, Chapin charts, hierarchy charts, and IPO diagrams. The tool used for design may be mandated by the institution; if not, it will depend upon the preferences of the designer.

#### **Coding**

Coding is the process of translating the logic design into programming language code. Each step of the logic must be converted into code which conforms to the rules of the language. In many programming languages, including COBOL, each piece of data used by the program must also be described. If the logic design is complete and comprehensive, coding will consist only of translating the design to fit the syntax of the programming language.

#### **Testing**

After a program is coded, it must be tested in order to discover and correct any errors that might have occurred during the analysis, design, and coding phases. Testing involves converting the program into machine language. In COBOL, a commercially available program (a COBOL compiler) is used to convert the program into machine language. A computer cannot run a program unless it is in machine language. Two types of errors may be uncovered during testing: syntax errors and errors causing incorrect output.

##### **Syntax Errors**

Syntax errors occur when the program code violates a rule of the programming language. A syntax error might be caused by a typographical error, a misspelling, or a misunderstanding of the rules of the language. For example, a syntax error would occur if the word MVOE were used instead of the word MOVE. It may not be possible to run a program that has syntax errors.

## Errors Causing Incorrect Output

Once syntax errors have been removed from the program, the program is executed to determine whether it produces correct output. Test data must be carefully prepared to assure thorough testing of the program. All possible combinations of data, correct and incorrect, should be tested, and the output must be carefully examined to locate any errors which might be present. If the output is incorrect, the program is debugged. Debugging is the process of locating and removing program errors. Incorrect output may be caused by coding errors or by errors in the logic of the program.

## Analysis Tools and Techniques

Analysis begins when a user group (such as the accounting or marketing department) approaches the data processing department and asks for assistance in obtaining information necessary to solve a business problem. The analyst then meets with the user group to determine its requirements and analyzes the systems (both manual and computerized) the group is currently working with. Then the analyst explores the options available to solve the problem and determines which option should be used. The analyst will usually put this solution into writing using program specifications, input/output layouts, and system flowcharts.

### *Program Specifications*

A program specification is a document which is produced during the analysis phase. It describes the requirements of the program to be written and is used as the basis for designing, coding, and testing the program.

The contents of a program specification will vary greatly from installation to installation. Usually, a program specification will include the program name, program function, input and output file descriptions, processing requirements, and output requirements. For example, the order processing department might approach the data processing department with a request for a printed report which will list all orders. The analyst then determines the requirements for the report, determines that the needed data is available on an existing file, and produces the program specification shown in Figure 1-1.

The program specification consists of the following elements:

**Program Name** The program is given a name by which it will be recognized by the computer system. The program name is usually created using a set of naming standards which are in place at the installation.

**Program Function** The function and purpose of the program are described, in order to give an overview of the program.

**File Descriptions** Often, a program must use or create data which exists outside of the program. This data is stored in files. (Files are discussed in greater detail later in the chapter.) Each file which will be used in the program is described in the file description section. The description provides basic information about the file, including the name of the file and whether the file is used as input to the program or as output from the program.

**Processing Requirements** There are no processing requirements in the example of Figure 1-1. When included, however, processing requirements provide an overview of the logic needed in the program, including whatever manipulation of the data and mathematical formulas are necessary.

**Output Requirements** The content and format required for the output are described.

PROGRAM SPECIFICATION

Program Name: LISTING  
Program Function:  
The program will produce a printed listing of all orders from the ORDER FILE.

Input Files:

- I. ORDER-FILE  

INPUT DEVICE: DISK  
FILE ORGANIZATION: SEQUENTIAL  
RECORD LENGTH: 70 BYTES  
FILE SEQUENCE: ASCENDING ON BRANCH /  
SALES REP

Output Files:

- I. PRINT-FILE  

OUTPUT DEVICE: PRINTER  
RECORD LENGTH: 133 BYTES

Output Requirements:

Each record read from the order file should be printed on one line of the output report. All fields from the input record are to be included on the output.  
The following are formatting requirements for the report:

- The first page of the report should contain:
  - (a) a main heading which includes the company name and report name.
  - (b) column headings which describe the items which are printed underneath.
- The detail lines should include all fields from the input.
- The detail lines should be double spaced.
- Do not include provisions for page overflow.

FIGURE 1-1

Input/Output Layouts

Input and output layouts are usually included with the program specification. They describe the data items which are or will be stored on the files. Layouts for the ORDER-FILE and PRINT-FILE of Figure 1-1 are shown in Figures 1-2 and 1-3. Input/output layouts are discussed in greater detail later in the chapter.

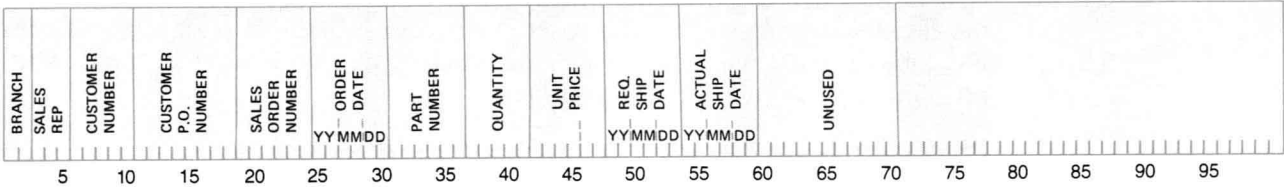


FIGURE 1-2

System Flowcharts

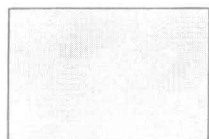
A system flowchart describing the flow of data through the system may be included with the program specification. Each symbol on the flowchart represents a file or







a program. Flowlines and arrows are used to show the relationships between the files and the programs. The following symbols are used in system flowcharting.



Process Symbol

This symbol indicates an entire program. In a system flowchart, the name of the program is written inside the box.



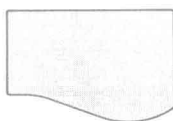
Magnetic Disk Symbol

This symbol is used to indicate a file which is or will be stored on magnetic disk. The name of the file is written inside the symbol.



Magnetic Tape Symbol

This symbol is used to indicate a file which is or will be stored on magnetic tape. The name of the file is written inside the symbol.



Document Symbol

This symbol indicates an output file which is a document (i.e., a printed report) or an input function where the input is obtained from a source document. The name of the document is written inside the symbol.



Flowlines

Flowlines are used to connect the symbols used in a system flowchart. The arrowhead indicates the direction of data flow.

Figure 1-4 shows the system flowchart for the LISTING program of Figure 1-1. A process symbol is used to represent the program. The program name, LISTING, is written inside this symbol. Flowlines are used to connect the input and output files to the program. The input file is stored on magnetic disk and is called the ORDER FILE. The document symbol is used for the output and indicates that the output is in printed form. The output file is called the CURRENT ORDER REPORT.