Structured ANS COBOL

Part 1: A course for novices using a subset of 1974 and 1985 ANS COBOL

Second edition

Mike Murach
Paul Noll

Structured ANS COBOL

Part 1: A course for novices using a subset of 1974 and 1985 ANS COBOL

Second edition

Mike Murach
Paul Noll

Editorial team

Mike Murach
Paul Noll
Judy Taylor
Pat Bridgemon

Production team

Steve Ehlers
Lori Davis
Carl Kisling

Other products in our COBOL series

Instructor's Guide for *Structured ANS COBOL, Part 1*
Minireel for *Structured ANS COBOL, Part 1*

Structured ANS COBOL, Part 2 by Mike Murach and Paul Noll
Instructor's Guide for *Structured ANS COBOL, Part 2*
Minireel for *Structured ANS COBOL, Part 2*

How to Design and Develop COBOL Programs
by Paul Noll and Mike Murach
The COBOL Programmer's Handbook by Paul Noll and Mike Murach
Instructor's Guide for *How to Design and Develop COBOL Programs*
Minireel for *How to Design and Develop COBOL Programs*

Report Writer by Steve Eckols

CICS for the COBOL Programmer, Part 1 by Doug Lowe
CICS for the COBOL Programmer, Part 2 by Doug Lowe
Instructor's Guide for *CICS for the COBOL Programmer*
Minireel for *CICS for the COBOL Programmer*

IMS for the COBOL Programmer, Part 1 by Steve Eckols

20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Library of Congress Catalog Card Number: 86-61654

ISBN: 0-911625-37-2

Structured ANS COBOL

Mike Murach & Associates, Inc.

4697 West Jacquelyn Avenue
Fresno, California 93722
(209) 275-3335

Contents

Preface			1
Section 1	**Required background**		7
Chapter 1	An introduction to computers, applications, and software		8
	Topic 1	An introduction to computer hardware	9
	Topic 2	An introduction to computer applications	19
	Topic 3	An introduction to computer software	29
Chapter 2	A student's procedure for developing COBOL programs		39
Section 2	**A professional subset of COBOL for report preparation**		53
Chapter 3	An introduction to COBOL: The basic elements		54
	Topic 1	The inventory-listing program	55
	Topic 2	The Identification and Environment Divisions	66
	Topic 3	The Data Division	71
	Topic 4	The Procedure Division	81
	Topic 5	Shop standards for COBOL programs	93
	Topic 6	Compiler dependent code	97
Chapter 4	Building on the COBOL basics		109
	Topic 1	Data Division elements	110
	Topic 2	Procedure Division elements	127
	Topic 3	The investment-listing program	142
	Topic 4	Compiler dependent code	157
Chapter 5	COBOL elements the professionals use		167
	Topic 1	Procedure Division elements	168
	Topic 2	How to use the COPY library	181
	Topic 3	How to use subprograms	185
	Topic 4	The investment-listing program	189
	Topic 5	Compiler dependent code	197

Chapter 6	Completing the professional subset		199
	Topic 1	How to handle one-level tables using indexes	200
	Topic 2	How to read records in indexed files	218
	Topic 3	The investment-listing program	224
	Topic 4	Compiler dependent code	243
Chapter 7	The 1985 COBOL elements for structured programming		245
Section 3	**Program development techniques**		257
Chapter 8	Procedures and JCL for compiling and testing a program		258
	Topic 1	Procedures for compiling and testing a program	259
	Topic 2	JCL for compiling and testing a program on an IBM mainframe under DOS/VSE	267
	Topic 3	JCL for compiling and testing a program on an IBM mainframe under OS/MVS	272
Chapter 9	How to correct compilation diagnostics		277
Chapter 10	How to test and debug a program		288
	Topic 1	How to test a program	289
	Topic 2	How to debug a program	295
Section 4	**Structured programming techniques**		313
Chapter 11	An introduction to structured programming		314
Chapter 12	The structure and logic of report preparation programs		332
Section 5	**Related subjects**		361
Chapter 13	What else an effective COBOL programmer must know		362
Section 6	**Appendixes**		373
Appendix A	A summary of the COBOL elements presented in this book		374
Appendix B	A model report preparation program		387
Appendix C	One chapter-by-chapter case study		405
Appendix D	Four more case studies		415
Index			431

Preface

This book is a long overdue revision of our 1979 book: *Structured ANS COBOL, Part 1*. The first edition was used in dozens of colleges and junior colleges for classroom instruction, and in thousands of businesses for inhouse training. Nevertheless, it needed revision badly, due to changes in COBOL, changes in data processing procedures, and changes in training requirements.

What this book does

The main objective of this book is to teach you how to use a subset of COBOL to develop structured programs that prepare reports. This subset consists of COBOL elements that conform to both the 1974 and the 1985 ANS standards (the standards published by the American National Standards Institute). Although most businesses use 1974 COBOL today, the trend is obviously toward the use of 1985 COBOL. That's why this book teaches you how to write programs for either version of COBOL.

Because we feel it's impossible to teach students how to code structured programs without teaching them how to design programs, this book gives extensive coverage to modern design techniques. Specifically, it shows you how to design a program from the top down using a structure chart, and it shows you how to plan the modules of a program using pseudocode. In addition, this book teaches you how to test a program from the top down. As a result, students who complete this course should not only be able to code COBOL, they should be able to develop programs using effective procedures for design, planning, and testing.

When compared with other introductory COBOL books, this book presents less COBOL than the average book does, but it teaches more about the structure and logic of programs that prepare reports. This makes sense because most students have more trouble with structure and logic than they do with COBOL. As a result, this book is a truer test of a student's programming aptitude. If a student can study this book and do the case studies in appendixes C and D with little outside help, we're confident that he or she has the aptitude required of a professional programmer in industry. On the other hand, if a student has considerable difficulty with the case studies, COBOL programming probably isn't the right vocation for him or her. In general,

we feel that any student who successfully completes this course has the qualifications of an entry-level programmer in industry.

Who this book is for

This book is a first course for anyone who wants to learn how to use COBOL. Since it assumes that you have no data processing experience, the first two chapters present the background you need for COBOL programming. As a result, if you've had experience with computers or programming, you may be able to skip portions of these chapters.

Since this book presents standard COBOL as defined in the 1974 and 1985 standards, it teaches COBOL that can be used on any computer system that supports COBOL. Although standard COBOL has minor variations as you move from one computer system to another, examples are given throughout the book that apply to microcomputers, minicomputers, and IBM mainframes. In general, any program in this text will run on any system that supports COBOL with only a couple of minor changes, and these changes are clearly specified.

Since IBM mainframes are the most widely-used systems today, all of the program examples in this text have been run on an IBM mainframe. As a result, this book is particularly easy to use if you're going to develop your programs on an IBM mainframe. As I said, though, the variations required by other systems are also presented, and they are trivial.

How to use this book

If you're reading this book as part of a course, your instructor should guide you through it. On the other hand, if you're reading this book on your own, you should realize that the chapters don't have to be read in sequence from chapter 1 through chapter 13. Instead, the chapters are grouped into five sections as shown by the table in figure P-1. As you can see, you can read section 3 any time after you complete chapter 3 in section 2, and you can read section 4 any time after you complete chapter 4 in section 2.

This type of organization, which we call *modular organization*, gives you a number of options as you use this book. If, for example, you want to learn all of the COBOL elements in section 2 before you learn program development techniques and structured programming techniques, that's one option. Then, you just read the 13 chapters in sequence. On the other hand, if you want to start the case study in appendix C right after you complete chapter 3, you can study section 3 next to find out how to compile and test your case study program. Similarly, you can study section 4 right after you complete chapter 4 to find out how to design typical report preparation programs.

Section	Chapters	Section title	Prerequisites
1	1-2	Required background	None
2	3-7	A professional subset of COBOL	Section 1
3	8-10	Program development techniques	Chapter 3
4	11-12	Structured programming techniques	Chapter 4
5	13	Related subjects	Sections 1-4

Figure P-1 The basic organization of this book

To help you learn from this book, each topic or chapter is followed by a terminology list and behavioral objectives. If you feel you understand the terms in each terminology list, it's a good indication that you've understood the content of the topic or chapter you've just read. In other words, we don't expect you to be able to define the terms in a list, but you should recognize and understand them. Similarly, if you feel that you can do what each objective requires, it's a good indication that you've learned what we wanted you to learn in each topic or chapter.

To give you a chance to apply your learning, appendix C presents a chapter-by-chapter case study. You can start working on this case study when you complete chapter 3. Then, as you complete each new chapter, the case study asks you to modify or enhance the program that you developed for the last chapter. If you can code and test all phases of this case study so they work correctly, we feel that this book has accomplished its primary objective.

To help you apply your COBOL knowledge to more demanding problems, appendix D presents four more case studies. You can start on these any time after you complete chapter 4. These case studies require you to develop four different types of report preparation programs, programs that require four different types of structure and logic. If you can develop all of these programs, we feel that you have the qualifications of an entry-level programmer in industry.

Related books

This book is only one book in our COBOL training series. *Structured ANS COBOL, Part 2* is an advanced book that starts where this book ends. It teaches an entry-level programmer how to use advanced

COBOL elements to develop batch edit and update programs. Then, *Report Writer* teaches you how to use the Report Writer module of COBOL.

Perhaps the most important book we've ever done for COBOL programmers is called *How to Design and Develop COBOL Programs*. It shows experienced COBOL programmers how to design, code, and test programs that are easy to debug and maintain. And it shows them how to increase their productivity, often by 200 percent or more. As an accompanying reference, we offer the *The COBOL Programmer's Handbook*, which summarizes the procedures and techniques presented in the text. It also presents seven model programs that you can use as guides for developing your own programs. Because this text and handbook present techniques and examples that will help you at any stage of your COBOL training, we recommend that you get them and use them throughout your training and career.

Beyond this, we offer books that teach the COBOL programmer how to use CICS and IMS or DL/I on IBM mainframes. We have books on other subjects that the IBM COBOL programmer must know, like VSAM, TSO, ICCF, and JCL. And we are publishing new books each year. So please check our current catalog for titles that may be of interest to you.

Instructor's materials

If you're an instructor in a school or business, you will probably be interested in the *Instructor's Guide* that is available with this book. It presents complete solutions for the case studies in appendixes C and D. It gives you ideas and summary information for administering a first course in COBOL. And it gives you masters for most of the figures in the text so you can make overhead transparencies from them.

A *minireel* is also available with this course. It is a 1600-bpi tape that contains files of test data, COPY members, and source programs. In short, it provides all of the complete program examples used in this book, as well as all of the files you'll need for running the case study solutions on your system.

Incidentally, we also offer instructor's guides and minireels for other courses in our COBOL series. These courses include *Structured ANS COBOL, Part 2* and *How to Design and Develop COBOL Programs*.

Reference manuals

Although this book represents a complete first course in COBOL, we recommend that you have access to the basic COBOL reference manuals for your system. On an IBM mainframe, two manuals are usually enough for the version of COBOL that you're using: the COBOL reference manual and the programmer's guide for using COBOL. On other systems, one manual is usually enough.

In general, you shouldn't have to refer to these manuals as you do the case studies. Occasionally, though, a problem may come up that is specific to your system, not to standard COBOL. Then, you can research the problem yourself in your system's manuals. Also, as we point out in chapter 13, it's good to page through your system's COBOL manuals at some time during your training to find out what features your version of COBOL provides.

About Paul Noll

Paul Noll originated the program development techniques we recommend in this book when he was working as a training manager for Pacific Telephone back in the mid-1970's. In 1978, he became an independent COBOL consultant. Since then, he has conducted seminars in hundreds of companies throughout the United States and Canada. His books have been used by thousands of programmers around the world, and, based on a COBOL survey we did in 1984, we believe that more than 3000 COBOL shops now use Paul's methods for program development. As far as we can tell, that means that Paul's methods are the most widely-used methods for structured program development. We've used Paul's methods in our own company since 1979, and we're convinced that they're the most effective methods currently available.

Paul is listed as a co-author of this book in the sense that he developed the basic methods that are taught in this book. He also reviewed the manuscript for this book as a double check on its technical accuracy. As a result, we feel that he has made an important contribution to the educational and technical quality of this book.

Conclusion

Paul and I believe that this book will help you learn COBOL better than any competing book or course will. We're confident that you'll learn a usable subset of COBOL from this book and that you'll learn it with maximum efficiency. We're also confident that the case studies will let you discover on your own whether a career in COBOL is right for you.

If you have comments about this book, we welcome them. If you check the last few pages of this book, you'll find a postage-page comment form. You'll also find a postage-paid order form in case you want to order any of our products. We hope you find this book useful, and thanks for being our customer.

Mike Murach
Fresno, California
May 20, 1986

Section 1

Required background

Before you can learn to develop programs in COBOL, you need some data processing background. The two chapters in this section present the minimum background that you need for this programming course. Chapter 1 introduces you to computers, computer applications, and computer programs. Chapter 2 presents a procedure you can use when you develop the COBOL programs required by this course.

Of course, if you already have computing experience or programming experience in another language, you may already know much of the material in this section. If so, you can review the objectives and terminology lists at the end of each chapter or topic to see whether you need to study it.

Chapter 1

An introduction to computers, applications, and software

This chapter consists of three topics that introduce you to computers (*hardware*), computer applications, and computer programs (*software*). If you've had no experience at all with computers or programming, these topics provide the minimum background you need for COBOL programming. On the other hand, if you're already familiar with computers and programming, much of this chapter will be review for you. In that case, you can review the terminology list and objectives at the end of each topic to determine whether or not you need to read the topic.

Topic 1 An introduction to computer hardware

Today, computer systems vary tremendously in size and price. On the low end, you can buy a home computer for less than $2,000. On the high end, a large IBM system may *rent* for over $1,000,000 per month. Nevertheless, the same basic concepts apply to both types of systems. In fact, you can run COBOL programs on both types of systems. This topic introduces you to the *hardware* concepts that apply to all business computers.

All business computer systems today consist of the four components shown in figure 1-1: processor, visual display terminal, disk device, and printer. A large system will have many more than four components, as I'll explain in a moment, but it will have at least one of each of the four components shown. In addition, a computer system may have one or more tape drives and one or more card readers, and it may have any number of special-purpose devices. In simplest terms, you can classify the components of a computer system into two groups: processors and input/output devices.

Processors

The center of a computer system is the *processor*. All the other devices that make up the system are attached to it. Conversationally, it is the "brain" of the system.

In simple terms, a processor consists of two main parts: the central processing unit and main storage. The *central processing unit* (or *CPU*) is a collection of circuits that execute program instructions for calculation and data manipulation. *Main storage* (or *main memory*) is the high-speed, general-purpose electronic storage that contains both the data the CPU operates upon and the program instructions it executes.

The smallest unit of main storage is called a *byte*. In general, a byte of memory can store one character of data, such as the letter K, the digit 3, or the symbol for dollars ($). Later on, you'll learn that numeric data can be stored in more than one form within a byte, so two or more digits can be stored in a single byte. For now, though, just assume that one byte of memory holds one letter, digit, or special character.

To refer to the amount of main storage a system provides, the symbol *K* has traditionally been used. Because the word *kilo* refers to 1,000, one K (or *KB* for *kilobyte*) refers to approximately 1,000 bytes of storage. Thus, "a 128KB system" means a computer system with approximately 128,000 storage positions in its main storage. I say "approximately" because one K is actually 1,024 storage positions.

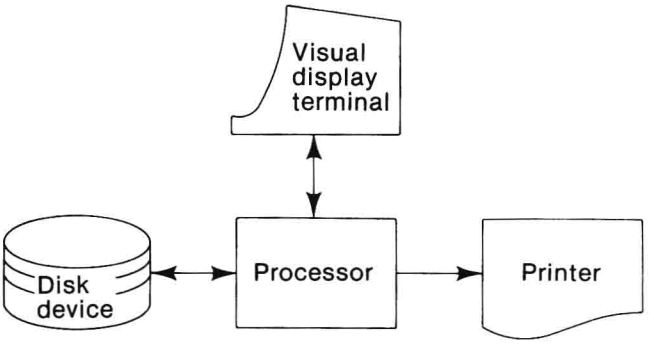

Figure 1-1 The four components of any modern computer system

Today, however, large computers are sold with much larger memories than can be expressed conveniently in K's. For instance, a small computer may have one megabyte of storage (expressed as 1*MB*). The term *megabyte* refers to approximately 1,000,000 bytes of storage. More precisely, a megabyte is 1,024KB, or 1,024 times 1,024 bytes of storage.

Although a processor consists of millions of electronic components, you really don't need to know much more about one than what I've just told you. As far as you're concerned, a processor is a black box that stores programs and data and does what your programs tell it to do.

Input/output devices

The second group of computer components is made up of *input/output devices*, or just *I/O devices*. Input devices send data to the processor, and output devices receive data from it. Some devices can perform both functions.

Because dozens of different devices are available for use within a computer system, many with a variety of special features, I'm not going to try to describe all of the possible I/O devices of a computer system. Instead, I'm going to concentrate on the three I/O devices shown in figure 1-1, because these are the ones you'll be using most frequently. In addition, I'll briefly mention tape drives and card readers: tape drives because you're likely to use one someday, and card readers because of their historical significance.

Disk devices *Disk devices* provide permanent storage for the programs and data of a system. They are both input and output devices because data and programs can be written on them or read from them. Because disk devices allow direct and rapid access to large quantities of data, they are a key component of all modern computer

systems. In contrast to the permanent storage of a disk device, processor memory is only used to store programs *while they are being executed* or to store data *while it is being processed.*

The most common type of disk device is the *disk drive*, a unit that reads and writes data on a *disk pack*. A disk pack, illustrated conceptually in figure 1-2, is a stack of metal platters that are coated with a metal oxide. Data is recorded on one or both sides of each of the platters. A disk pack can be removable or it can be fixed in a permanent, sealed assembly inside the drive.

On each recording surface of a disk pack, data is stored in concentric circles called *tracks*. This is also illustrated conceptually in figure 1-2. Although the number of tracks per recording surface varies by device type, the surface illustrated in figure 1-2 has 200 tracks, numbered from 000 to 199.

The data stored on a track is read by the disk drive's *access mechanism*, which is an assembly that has one *read/write head* for each recording surface as shown in figure 1-2. As you can see, the access mechanism is positioned over the same track on all recording surfaces at the same time. As a result, all of these tracks can be operated upon, one after another, without the access mechanism having to move. Because the access mechanism can be positioned on any track of a device in a fraction of a second, any record on a disk drive can be accessed in an instant.

To specify the capacity of a disk drive, megabytes are normally used. For instance, a disk drive on a personal computer may have a capacity of 10MB, while a disk drive on a minicomputer may have a capacity of 75MB. On the largest computer systems, though, *gigabytes*, or *GB's*, are used to record disk capacities. One GB is equal to 1,000MB, so a GB is approximately one billion bytes of storage. On some systems, a single disk drive can store more than one GB of data.

For the most part, you can think of a disk device as a black box, just as you can a processor. In other words, you don't need to know how many disks it consists of, how many tracks are on each disk, and so on. All you need to know is that your program can write data on the disk device, and it can access and read any of the data on the disk device at a high rate of speed.

Terminals On a modern computer system, *visual display terminals*, or just *terminals*, are used to enter data into the system and to display data stored in the system. As a result, terminals are both input and output devices. Each terminal consists of two parts: a keyboard, which is similar to that of a typewriter, and a display screen, which is somewhat similar to the screen of a TV. Since you've probably used a terminal at one time or another, I won't dwell on one's operational characteristics. If you haven't used one before, you probably will as part of this course.

You should know, however, that terminals are also known by many other names. They can be called *VDT's*, using the acronymn for