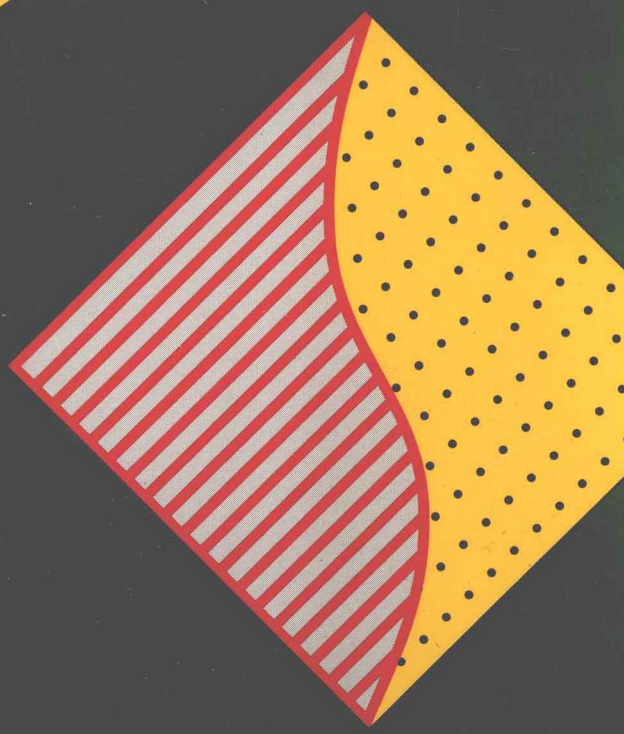
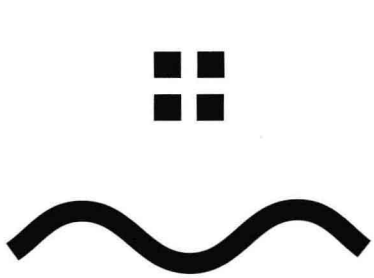
Structured COBOL

THIRD EDITION





Structured COBOL

THIRD EDITION

Gerard A. Paquette

Irwin McGraw-Hill

Boston, Massachusetts Burr Ridge, Illinios Dubuque, Iowa
Madison, Wisconsin New York, New York San Francisco, California St. Louis, Missouri

Irwin/McGraw-Hill
A Division of The McGraw-Hill Companies

Vice President and Publisher *Susan A. Simon*
Acquisitions Editor *Paul Ducham*
Managing Developmental Editor *Linda Meehan Avenarius*
Advertising/Marketing Coordinator *Jennifer Wherry*
Product Development Assistant *Sandy Ludovissy*

Chief Executive Officer *G. Franklin Lewis*
Corporate Senior Vice President and Chief Financial Officer *Robert Chesterman*
Corporate Senior Vice President and President of Manufacturing *Roger Meyer*
Executive Vice President/General Manager, Brown & Benchmark Publishers *Tom Doran*
Executive Vice President/General Manager, Wm. C. Brown Publishers *Beverly Kolz*

The credits section for this book is on page 838, and is considered an extension of the copyright page.

Cover design by Sailer & Cook Creative Services

Library of Congress Catalog Card Number: 93-73870

ISBN 0-697-12394-4

Printed in the United States of America

10 9 8 7 6 5

Preface

This text is written for the student who wants to develop effective problem-solving techniques while learning structured COBOL programming. It contains a number of features that make it stand out from other available texts.

Problem-Solving Procedure

A time-tested problem-solving procedure is presented and explained in the very first chapter, and then the "Turbo Manufacturing Company Employee Listing" program is used immediately to show the procedure in actual use. Every original program presented in the text is also systematically developed following the steps specified in the procedure.

Many textbooks do suggest a systematic approach to programming, but more often than not, the systematic approach is not used later in the text to develop sample programs, no further reference is made to it, and if students use it at all, it is on their own initiative. Since developing an effective approach to problem solving is so crucial to a novice programmer, repeated exposure to a specific method that is used in a consistent manner will go a long way toward achieving that goal. The net result is that the student retains the procedure as a productive "working" tool in later programming situations.

Problem-Oriented Approach

The text follows a problem-oriented approach. This approach allows the development of COBOL statements and features to be presented on a "need to know" basis, where they are applied immediately in the situation being discussed. The sample problems that have been selected as vehicles for the introduction of COBOL language features are presented at the very beginning of the chapter, thus setting the stage for what is to come in the rest of the chapter.

In Chapter 2, the program "Allied Stock Company Phone Sales Report" is presented and developed primarily to expose the students to the major functions of the four divisions of a COBOL program.

In Chapter 3, the same program is used as a vehicle to expand upon the features of the IDENTIFICATION DIVISION, the ENVIRONMENT DIVISION, and the DATA DIVISION. In Chapter 4, it is used to elaborate upon the features of the PROCEDURE DIVISION.

In Chapter 5, a new program, "Reliable Auto Parts Company Payroll Report," is presented and developed as a tool to introduce arithmetic statements, since the preparation of a payroll requires a number of arithmetic operations.

In Chapter 6, the program of Chapter 5 is expanded to prepare a summary payroll report that contains totals and averages for the company. The same program is then revised to include refinements in both the program and the printed report.

In Chapter 7, the program "Jocelyn Originals Company Annual District Sales Report" prepares a summary report by district. Because the records of the input file are in alphabetical order, the district code must be tested so that the sales figures can be added to the appropriate district accumulators. This leads into a discussion of conditional statements—simple, compound, and nested. Once the report has been produced, an alternative approach is developed to illustrate the use of CORRESPONDING statements and qualification of data names. Since the setting of the problem remains the same, students can more easily concentrate their attention on the new features.

In Chapter 8, two programs are presented to illustrate important applications of conditional statements. The "Sportsman Company Sales Analysis Report" program serves as an introduction to control

breaks. One-level control breaks are examined first, and the program is then expanded to illustrate two-level control breaks. The "Sportsman Company Data Validation Report" program is then developed to examine various data validation procedures.

In Chapter 9, the "Monster Burger Stores Weekly Payroll Report" program has been designed to require the use of input-loaded and hard-coded tables. Its development leads into discussions of table organizations, levels in a table, table-loading procedures, and table-data retrieval procedures.

In Chapter 10, variations of the "Sportsman Company Sales Report" program are used to illustrate all the possible combinations of options of the SORT statement. PROCEDURE DIVISION sections are introduced in relation to the SORT statement. A final program is used to illustrate the MERGE statement.

In Chapter 11, the Jocelyn Originals Company provides a setting for the development of programs that create master files on tape or on disks. These files have sequential organization. Indexed files and relative files are also introduced, but complete programs are not included since the sequel to this book examines these two file organizations in detail.

■ Structured Programming Tools

Built into the design phase of the problem-solving procedure is the use of tools that result in structured programs. The system flowchart specifies the overall task of the program, in addition to identifying the required files. The structure chart begins with the overall task as its primary module, and then breaks it down into major subtasks, one per module. These major subtasks are further subdivided as needed, until the question "What has to be done?" is completely answered. The program pseudocode or the program flowchart then details the procedures that answer the question "How is the task to be done?"

In the structure chart, the first-level, or primary, module is numbered 100, the second-level modules are numbered consecutively from left to right in the 200s using increments of 10, the third-level modules are similarly numbered in the 300s, and so on for the remaining levels. If a particular module is needed at different levels in the structure chart, every occurrence of the module is assigned the number of the lowest level where it is needed, and the upper right-hand corner is darkened to clearly indicate its repetitive nature.

The paragraphs of the pseudocode and the modules in the flowchart retain the numeric prefixes and names first assigned in the structure chart. The paragraph names in the PROCEDURE DIVISION are then the same as those of the pseudocode or flowchart. Furthermore, by coding the paragraphs in ascending order of their numeric prefixes, program debugging is greatly facilitated, since the programmer can quickly scan margin A to locate a particular paragraph.

■ COBOL '85 Standards

The text implements the standards of COBOL '85. A student who learns the language as presented in this text will be well-prepared to write programs that include up-to-date syntax and structure. However, since most of COBOL '74 is still supported in the COBOL '85 standards, and since the new features are identified as such within the text, this text remains appropriate for use by students who are currently using a COBOL '74 compiler, and these students will have a handy reference when they first encounter a COBOL '85 compiler. Naturally, the new features cannot be incorporated in programs to be compiled by a '74 compiler, and features that are currently labeled as obsolete may be required entries. (Obsolete entries are not used in the sample programs of this text.)

■ Debugging Activities

Debugging exercises have been included at the end of every chapter. They are listed following the more traditional "review and practice" exercises and before the programming exercises. They include debugging data entry errors, misaligned entries on a printed report, data description errors that do not conform to prepared layouts, compiler-generated syntax errors, coding errors that do not represent the logic specified in either pseudocode or flowchart form, numeric and/or numeric-edited data description coding errors, errors in mathematical statements or expressions, errors related to missing data or incorrect values on a printed report, printed report errors related to incorrect handling of control breaks, coding errors related to table definitions, table loading, and data retrieval, sorting errors due to improperly sequenced keys, and control errors related to the use of sections within the PROCEDURE DIVISION.

These exercises are designed to draw attention to common errors and some less common ones. Students who successfully complete these exercises should be able to avoid making the same types of errors in their own programs.

Programming Assignments

There are four programming assignments at the end of each chapter, beginning with Chapter 3. In the spirit of "leading the student by the hand," the student is presented with the completed design phase of the problem-solving procedure in the first assignment; that is, the input record layout form and the printer spacing chart are prepared, and the system flowchart, the structure chart, and the program pseudocode and flowchart are completely developed. After studying these aids, the student can quickly move to the completion of the task—coding the program, debugging it as needed, and running it with a sample data file.

In the second assignment, the design phase of the problem-solving procedure is completed for the student except for the program pseudocode and the program flowchart. The student's task is to complete the design phase, and then implement the design.

The third and fourth assignments are full-scale assignments. The student must design the program "from scratch," and then carry out the implementation of the design.

Data sets for all programming assignments are provided in Appendix C, and are also available on magnetic disks for the convenience of instructors and students.

Program Documentation

All sample programs contain a number of documentation paragraphs in the IDENTIFICATION DIVISION. The student is aided in verbalizing the nature of the task that the program must accomplish by being expected to properly document the listing of the program. The better the task is defined, the easier it is to map out an appropriate strategy. Few textbooks provide this type of documentation or encourage students to do so.

Listing of Rules

Whenever COBOL language elements and statements are introduced, explained, and illustrated in a section of a chapter, all of the applicable rules are collected and summarized within that same section under a title beginning with "Rules Governing Use of. . . ." Students will find these sections convenient because everything they need to know about the feature or statement is there, assembled in one place. A list of these rules sections, complete with page references, is printed at the front of the book on page ix. The colored tabs that are visible on page edges mark the locations of these rules in the text.

Course Organization

There is ample material in this text for a quarter or a semester's work in a course that might be entitled "Structured COBOL Programming I." It is not intended as a complete treatment of the COBOL language, but rather as an effective teaching/learning tool for the developing programmer. The sequel to this text, *Advanced Structured COBOL,* takes the student into more advanced table handling, file organizations, master file updating procedures, subprograms and nested programs, and also introduces on-line interactive COBOL programs, even though a screen-handling facility is not yet included in ANSI COBOL Standards. A Report Writer supplement is also available upon request.

Ancillaries

The following ancillary materials have been prepared for the convenience of instructors using this text:

a. An Instructor's Manual that contains, for each chapter, an overview of the chapter, a discussion of the particularly significant topics of the chapter, some suggested coordinated activities for the student, answers to all the chapter exercises, solutions to all the debugging activities, and a complete set of solutions for all the programming exercises.

b. A complete set of transparency masters.

c. TestPak, a computerized testing program that an instructor can use in constructing class tests quickly and efficiently. TestPak is a computerized system that enables you to make up customized exams quickly and easily. For each exam, you can select up to 250 questions from the file and either print the test yourself or have Business and Educational Technologies print it.

d. A Data Disk that contains the data files used in the chapter sample programs and in the debugging activities programs, and data files that can be used as test data for all the programming exercises.

e. Source code disks that contain:

i. The source code for all the chapter sample programs.

ii. The source code for the programs used in the debugging activities at the end of each chapter. The source code for the completely debugged programs are also included.

iii. The source code for the programs that must be developed in the programming exercises at the end of each chapter.

These source code disks allow instructors to make changes to existing programs, thereby customizing the programs to their own needs without a major effort on their part.

Software

Structured COBOL can be purchased with the following excellent software packages:

a. The RM/COBOL-85 Educational Version Compiler, which offers complete language functionality, screen handling capabilities, and an application development environment. A Convenience Disk is included in the textbook for students who also purchase the RM/COBOL-85 Compiler for use on their personal computer. This disk contains the source code for the programs in the debugging exercises at the end of each chapter. It also contains all the data sets used in the chapter sample programs and the data sets needed for the programming exercises at the end of each chapter.

b. Micro Focus Personal COBOL, a full-featured, PC-based package that allows students to write, compile, debug, and test complete COBOL programs.

A Final Note

I wish to thank the staff of the Computer Center at the University of Massachusetts–Lowell for their cooperation whenever their technical assistance was needed.

I also wish to thank the members of my family for their patient understanding during the long development process, and in particular, my wife, Vivian, for her constant encouragement and support.

Notes on My Computer

1. Computer name: _____ Operating system: _____

2. Nonnumeric literals must be enclosed in _____ (quotation marks or apostrophes).

3. Implementor-names to be used in the ASSIGN clause are:

_____

_____

_____

4. ANSI COBOL compiler: _____ (1985 or 1974)

5. Procedures to log on to my system:

_____

_____

_____

_____

6. System commands to edit a file:

_____

_____

_____

7. Commands to "run" a program:

_____

_____

_____

Contents

Handy Reference to Rules ix
Preface x

Introduction 2

Historical Development of Computers 5
Historical Development of COBOL 5
Computers 5
Major Computer Components 6
- Input Unit 7
- Processor Unit 7
- Output Unit 8

Computer Languages 8
- Low-Level Languages 8
- Intermediate-Level Languages 9
- High-Level Languages 9

Computer Systems 11
Processing a Program 13
Conclusion 14

1 The Problem-Solving Procedure 16

Objectives You Should Achieve 16
Problem-Solving Procedure 17
Program Illustrating the Problem-Solving Procedure 18
- Problem 20
- Solution 21
  - The Program Design Phase 21
  - The Design Implementation Phase 24

Debugging 34
Data Input to a Cobol Program 34
Important Terms in Chapter 1 36
Exercises 36
Debugging Activities 36
Terminal Exercises 37

2 A First Look at a COBOL Program 38

Objectives You Should Achieve 38
The Problem 39
Program Design Phase 40
- Step 1: Developing a Detailed Understanding of the Problem 40
  - Step 1a. Prepare Layouts of Input Records 40
  - Step 1b. Prepare Layouts of Output Records 41
- Step 2: Planning the Solution to the Problem 42
  - Step 2a. Determine the Data Flow (System Flowchart) 42
  - Step 2b. Develop the Structure Chart 43
  - Step 2c. Write the Program Pseudocode 44
  - Alternate Step 2c. Design the Program Flowchart (Alternative to the Pseudocode) 47

Design Implementation Phase 51
- Step 3: Coding the Program 51
  - Program Coding 52
  - Program Walk-through 52
- Step 4: Keying (or Entering) the Program 56
- Step 5: Debugging the Program 57
  - Step 5a. Correct All Syntax Errors 57
  - Step 5b. Compile the Program 58
- Step 6: Performing a Test Run of the Program 58
  - Step 6a. Run the Program with a Sample Input File 59
  - Step 6b. Check the Output 59
- Step 7: Assembling the Complete Package 64

The Divisions of a COBOL Program 64
- IDENTIFICATION DIVISION 64
- ENVIRONMENT DIVISION 64
- DATA DIVISION 65
  - FILE SECTION 65
  - WORKING-STORAGE SECTION 65
- PROCEDURE DIVISION 66

Basic Logic Structures 67
- Simple Sequence 67
- Selection 68
- Iteration 69

Combinations of the Three Simple Structures 71
The Case Structure 74
Important Terms in Chapter 2 76
Exercises 76
Debugging Activities 84
Terminal Exercises 86

3 The Divisions of a COBOL Program: IDENTIFICATION, ENVIRONMENT, DATA 88

Objectives You Should Achieve 88
User-Defined Names 89
IDENTIFICATION DIVISION 96
- PROGRAM-ID 96
- AUTHOR 97
- INSTALLATION 97
- DATE-WRITTEN 97
- DATE-COMPILED 97
- SECURITY 97
- Use of Asterisk (*) in Program Documentation 97
- Coding the IDENTIFICATION DIVISION 98
- Technical Format Notation 98

ENVIRONMENT DIVISION 99
- CONFIGURATION SECTION 99
- INPUT-OUTPUT SECTION 100
  - FILE-CONTROL 100
- Coding the ENVIRONMENT DIVISION 102

DATA DIVISION 103
- FILE SECTION 103
  - The FD Paragraph 104

The 01 Record Description Entry 105
FILE SECTION Examples 105
Record Field Descriptions 106
More on Numeric, Alphabetic, and Alphanumeric Classes 111
PICTURE-Related Issues 112
First Look at Editing Characters in Output Record Fields 114
Coding the FILE SECTION of the DATA DIVISION 114
WORKING-STORAGE SECTION 116
01 Elementary Items 116
The VALUE Clause 117
An Alternative to the VALUE Clause 120
Records in the WORKING-STORAGE SECTION 121
Adding a Title to the Report 122
Coding the WORKING-STORAGE SECTION 125
The General Format of a Data Item Entry 126
Important Terms in Chapter 3 126
Exercises 127
Debugging Activities 130
Programming Exercises 134
Programming Exercise I 134
Programming Exercise II 137
Programming Exercise III 139
Programming Exercise IV 140

4 The Divisions of a COBOL Program: PROCEDURE DIVISION 142

Objectives You Should Achieve 142
Planning the PROCEDURE DIVISION 143
Paragraph Names 146
Statements and Sentences 146
Coding the PROCEDURE DIVISION 148
The Main-Control Paragraph 100-PRODUCE-PHONE-SALES-REPORT 148
The PERFORM Statement 149
The PERFORM-UNTIL Statement 152
The STOP RUN Statement 156
The Second-level Paragraph 200-START-UP 157
The OPEN Statement 158
The MOVE Statement 159
The INITIALIZE Statement 164
The Second-Level Paragraph 210-PROCESS-PHONE-SALE-RECORD 166
The Second-Level Paragraph 220-FINISH-UP 167
The Third-Level Paragraph 300-WRITE-REPORT-HEADERS 168
The Third-Level Paragraph 310-READ-PHONE-SALE-RECORD 172
The Third-Level Paragraph 320-COMPUTE-PHONE-SALE-AMOUNT 175
The Third-Level Paragraph 330-PREPARE-PHONE-SALE-LINE 178
Conclusion 180
Important Terms in Chapter 4 180
Exercises 180
Debugging Activities 185
Programming Exercises 188
Programming Exercise I 188
Programming Exercise II 192
Programming Exercise III 193
Programming Exercise IV 194

5 The Arithmetic Operations: Addition, Subtraction, Multiplication 196

Objectives You Should Achieve 196
The Problem 197
Program Design Phase 197
Step 1 197
Step 2 198
System Flowchart 198
Structure Chart 199
Program Pseudocode 200
Program Flowchart 205
Design Implementation Phase 208
Step 3 208
Issue 1 208
Issue 2 208
Issue 3 208
Comments on Issue 1: Class of Input Data Item 209
Comments on Issue 2: Editing Fields of Output Records 211
Comments on Issue 3: Class of Working-Storage Data Items 214
Coding the Procedure Division 216
The Procedure 210-PROCESS-EMPLOYEE-RECORD 217
Coding the Procedure 320-COMPUTE-PAYROLL-ITEMS 218
Back to the Paragraph 320-COMPUTE-PAYROLL-ITEMS 226
Back to the Paragraph 320-COMPUTE-PAYROLL-ITEMS 231
Back to the Paragraph 320-COMPUTE-PAYROLL-ITEMS 234
Coding the Procedure 330-PREPARE-EMPLOYEE-PAYLINE 235
Correspondence between the Program Flowchart and the PROCEDURE DIVISION 236
Design Implementation—Continued 240
Steps 4 and 5 240
Step 6 243
Important Terms in Chapter 5 245
Exercises 245
Debugging Activities 250
Programming Exercises 255
Programming Exercise I 255
Programming Exercise II 259
Programming Exercise III 261
Programming Exercise IV 261

6 More Arithmetic: Totals and Averages 262

Objectives You Should Achieve 262
The Problem 263
Program Design Phase 263
Step 1 263
Step 2 264
Design Implementation Phase 270
Step 3 270
Coding the DATA DIVISION 270
Coding the PROCEDURE DIVISION 273
Steps 4 and 5 276
Step 6 276
The COMPUTE Statement 283
Program Refinements 286
The USAGE Clause 286
Grouping Report Headers 288
The Trailing Minus Sign 289
Checking for an Empty Data File 290
Inserting Documentation within the PROCEDURE DIVISION 291
Report Refinements 295
Vertical Spacing 295
"End of Report" Message 296
Limiting the Number of Detail Lines per Page 296

Dating the Report and Numbering the Pages 300
The Page Number 300
The Date 300
A New Module to Initialize Program Variables 304
The Revised Program 305
Important Terms in Chapter 6 319
Exercises 319
Debugging Activities 321
Programming Exercises 326
Programming Exercise I 326
Programming Exercise II 332
Programming Exercise III 335
Programming Exercise IV 335

7 Conditional Statements 336

Objectives You Should Achieve 336
The Problem 337
A Brief Analysis of the Problem 337
Program Design Phase 338
Step 1 338
Step 2 339
System Flowchart 339
Structure Chart 339
Program Pseudocode 342
Design Implementation Phase 354
Steps 3 and 4 354
PROCEDURE DIVISION Considerations 356
The Conditional Statement 356
The CONTINUE Statement 360
The Relational Test 360
Nonnumeric Relational Operands 361
The Condition-Name Test 363
More Condition-Name Examples 364
The SET Statement for Condition-Names 369
The Sign Test 370
The Class Test 371
Negated Test 373
Compound Tests 375
Abbreviated Compound Relational Tests 378
Nested Conditional Statements 379
The EVALUATE Statement 383
Back to the Program 388
Coding the PROCEDURE DIVISION 388
Steps 5 and 6 388
An Alternate Program Development 396
Qualification 396
The MOVE CORRESPONDING Option 398
The ADD CORRESPONDING Option 399
The SUBTRACT CORRESPONDING Option 400
The Alternate Program 400
Important Terms in Chapter 7 405
Exercises 405
Debugging Activities 416
Programming Exercises 420
Programming Exercise I 420
Programming Exercise II 427
Programming Exercise III 430
Programming Exercise IV 431

8 Control Breaks and Data Validation 434

Objectives You Should Achieve 434
Control Breaks 435
The Problem 435
Brief Analysis of the Problem 436
The Design Phase 438
Flagging Summary Levels 438
System Flowchart 439
Structure Chart 439
200 START UP 439
210 PROCESS SALES RECORD 440
220 FINISH UP 440
Program Pseudocode 441
Program Flowchart 447
The Design Implementation Phase 447
Subtotaling Versus Rolling Forward 456
Forcing the Last Store Footing 457
Multiple Control Breaks 458
Structure Chart 460
Program Pseudocode and Program Flowchart 460
The Revised Program 469
Data Validation 475
The Problem 475
Program Design Phase 476
Structure Chart 476
Program Pseudocode 478
Program Flowchart 482
Coding the Program 486
The INSPECT Statement 490
More on the INSPECT Statement 499
More INSPECT Statement Examples 501
Reference Modification 502
The REDEFINES Clause 503
Important Terms in Chapter 8 505
Exercises 505
Debugging Activities 509
Programming Exercises 517
Programming Exercise I 517
Programming Exercise II 525
Programming Exercise III 528
Programming Exercise IV 529

9 Table Handling Using Subscripts 532

Objectives You Should Achieve 532
The Problem 533
A Brief Analysis of the Problem 535
The Design Phase 536
System Flowchart 537
Structure Chart 538
Program Pseudocode 538
Processing a Payroll Record 543
The Program Flowchart 552
Tables in COBOL 557
Defining a Table 557
Storing Values in a Table 559
Hard-Coding a Table—Part I 559
Hard-Coding a Table—Part II 561
Loading a Table 561
Table Organizations 566
Accessing Data in a Table 568
Using Direct Referencing 568
Using Table Look-Up 568
Two-level Tables 573
Loading a Two-Level Table—Scheme A 574
The PERFORM-VARYING-AFTER Statement 576
Loading a Two-Level Table—Scheme B 579
Searching a Two-Level Table 582
Accessing Data from a Two-Level Table 583
The Federal Tax Table 584
Hard-Coding a Two-Level Table 586
Some Technical Considerations 586
The Design Implementation Phase 587
Coding the ENVIRONMENT DIVISION 587
Coding the FILE SECTION of the DATA DIVISION 588
Coding the WORKING-STORAGE SECTION 589
Coding the PROCEDURE DIVISION 590
More to Come 599
SEARCH ALL 600

SEARCH Versus SEARCH ALL 602
Important Terms in Chapter 9 603
Exercises 603
Debugging Activities 609
Programming Exercises 611
Programming Exercise I 611
Programming Exercise II 618
Programming Exercise III 621
Programming Exercise IV 622

10 The SORT and MERGE Statements 624

Objectives You Should Achieve 624
Problem 1—Sorting Only 625
Brief Analysis of the Problem 625
The Structure Chart for Problem 1 626
The Program Pseudocode for Problem 1 626
The Program Flowchart for Problem 1 626
The SORT Statement in Problem 1 626
Coding the Program for Problem 1 630
Problem 2—Sorting and Postsort Processing 631
Brief Analysis of the Problem 632
The Structure Chart for Problem 2 633
The Program Pseudocode and Program Flowchart for Problem 2 633
The SORT Statement in Problem 2 636
Coding the Program for Problem 2 636
Running the Program for Problem 2 640
Problem 3—Sorting and Presort Processing 640
Brief Analysis of the Problem 640
The Structure Chart for Problem 3 641
The Program Pseudocode and Program Flowchart for Problem 3 642
The SORT Statement in Problem 3 642
Coding the Program for Problem 3 644
Running the Program for Problem 3 646
Problem 4—Sorting with Presort and Postsort Processing 646
Brief Analysis of the Problem 646
Option 1 646
Option 2 648
The Program Pseudocode and Program Flowchart for Problem 4 648
The SORT Statement in Problem 4 653
Coding the Program for Problem 4 653
Running the Program for Problem 4 657
Problem 5—Sorting with Multiple Input Files and Postsort Processing 657
Brief Analysis of the Problem 657
Coding the Program for Problem 5 659
The SORT Statement Rules 669
The MERGE Statement 672
A Merge Problem 673
A Brief Analysis of the Problem 673
The Merge Program 679
Internal Sort/Merge vs. External Sort/Merge 685
Important Terms in Chapter 10 685
Exercises 685
Debugging Activities 688
Programming Exercises 692
Programming Exercise I 692
Programming Exercise II 696
Programming Exercise III 698
Programming Exercise IV 698

11 File Handling 700

Objectives You Should Achieve 700
Uses of Sequential Files 701
Creating Sequential Files 701
Some Tape File Characteristics 702
More Tape File Considerations 705
Problem 1 706
Brief Analysis of the Problem 706
The Program Pseudocode 707
FILE-CONTROL Entries 711
Back to the Program 713
Updating the Master Personnel Tape File 718
Comments on the Flowchart 726
Comment 1 726
Comment 2 726
Comment 3 727
Comment 4 727
Comment 5 727
Backing Up the Master File 733
Disk Files 734
Some Disk File Characteristics 734
Creating a Sequential File on Disk 737
Coding the Program 742
Updating Records in a Disk Sequential File 751
Coding the Program 753
I-O Open Mode 753
REWRITE Statement 754
Updating a Disk Sequential File 758
Sequential Files in Brief 758
Indexed Files 758
Accessing Records of an Indexed File 760
Accessing Records Sequentially 760
Accessing Records Randomly 760
Maintaining an Indexed File 761
More to Come on Indexed Files 761
Relative Files 761
Creating a Relative File 761
Loading a Relative File 761
Randomization Procedure 765
Updating a Relative File 766
More to Come on Relative Files 766
Important Terms in Chapter 11 766
Exercises 767
Debugging Activities 770
Programming Exercises 778
Programming Exercise I 778
Programming Exercise II 780
Programming Exercise III 782
Programming Exercise IV 783
Appendix A: COBOL Reference Formats 785
Appendix B: Collating Sequences 802
Appendix C: Data Sets 805
Appendix D: ANSI COBOL 1985 Reserved Words 821
Glossary 823
Index 832

■ Handy Reference to Rules for:

ACCEPT Statement 302
ADD CORRESPONDING Statement 399
ADD Statement 229
BLOCK CONTAINS Clause 704
Class Test 371
CLOSE Statement 168
Compound Tests 376
COMPUTE Statement 285
Condition-Names 364
Data Item Description Entry 126
Data Transfers 160
DISPLAY Statement 304
DIVIDE Statement 275
EVALUATE Statement 387
EXTEND Mode 742
IF-ELSE Statement 359
INITIALIZE Statement 164
INSPECT Statement 499
MOVE CORRESPONDING Statement 398
MULTIPLY Statement 225
OCCURS Clause 586
OPEN Statement 159
PERFORM Statement 156
PERFORM-UNTIL Statement 156
PERFORM-VARYING Statement 563
PERFORM-VARYING-AFTER Statement 576
Qualification 397
READ Statement 173
REDEFINES Clause 504
Reference Modification 503
Relational Test 363
RELEASE STATEMENT 646
RETURN Statement 639
REWRITE Statement 755
SEARCH ALL Statement 602
SEARCH Statement 600
SELECT Statement-Sequential Files 711
SIGN Clause 210
SORT Statement 669
Subscripts 587
SUBTRACT Statement 234
USAGE Clause 287
User-Defined Names 94
VALUE Clause 119
WRITE Statement 171

Structured COBOL

■ Introduction

In this introduction, we take a brief look at the historical development of the modern computer and the COBOL programming language. We identify the major computer components and various languages that programmers use in giving instructions to computers.

If you are now undertaking your first course in computer programming, or if you feel a need to review the topics just listed, you are encouraged to spend an hour browsing through this introduction. ■

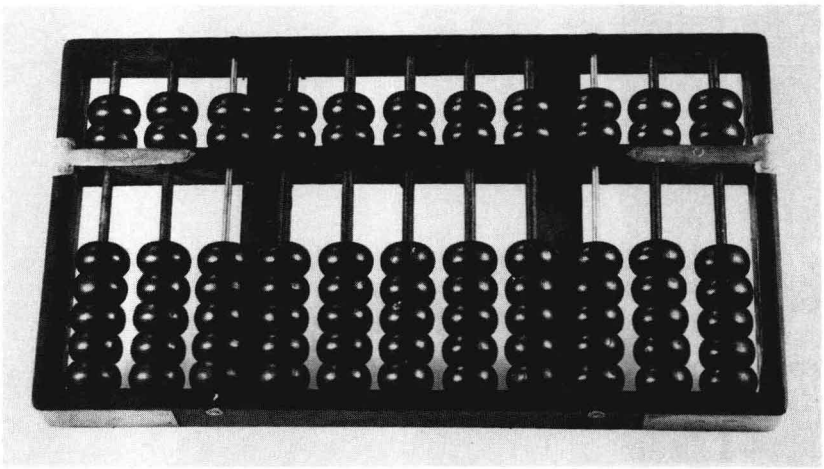
Historical Development of Computers

From time immemorial, people have had a need to gather and disseminate data. The caveman kept a record of his kill by carving notches in his spear; wise men preserved scientific observations for posterity by engraving in clay tablets; and the monks of old transcribed thousands of documents by hand. The development of the printing press greatly increased the speed by which accounts could be disseminated; and now, radio and television with the help of orbiting satellites make it possible to collect data and relay it immediately to any point on the globe.

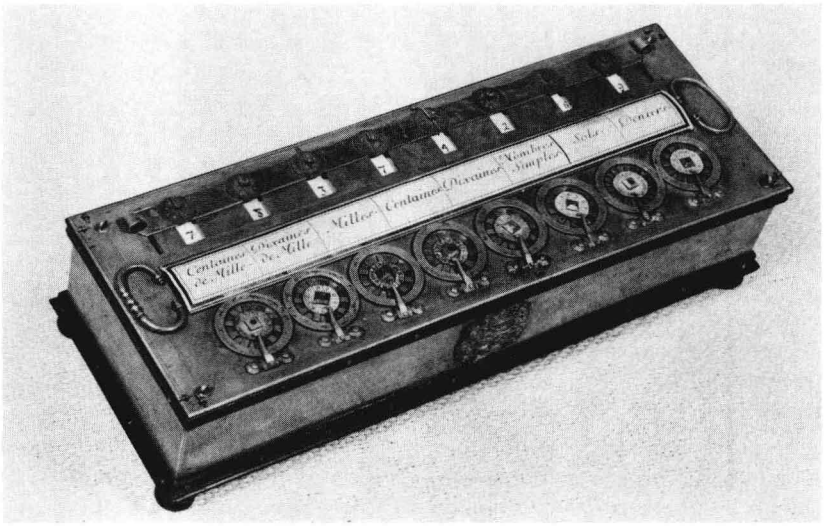
In our need to collect and transmit data, we are constantly searching for better tools that enable us to accomplish our objectives. Historically, the collecting, processing, and dissemination of data have progressed through three stages: manual, mechanical, and now electronic. The earliest simple manual tool used for counting was the abacus, which is still in use today. In an effort to increase the capability of the abacus, the French mathematician Blaise Pascal developed the first mechanical calculator in about 1642. A few years later, the German mathematician Gottfried von Leibnitz also worked on a mechanical calculator that performed all the basic operations of arithmetic. By the mid-1800s, Charles Babbage partially constructed an automatic machine to perform calculations, but it was not until 1885 that William Burroughs developed the first useful and practical automatic calculator. These mechanical calculators had to be operated manually by a crank. They were improved and developed into electrical calculators.

By the mid to late 1930s, the threat of World War II provided additional impetus, and, very importantly, funding for research in the planning and development of electronic machines. Dr. Howard

■ An abacus

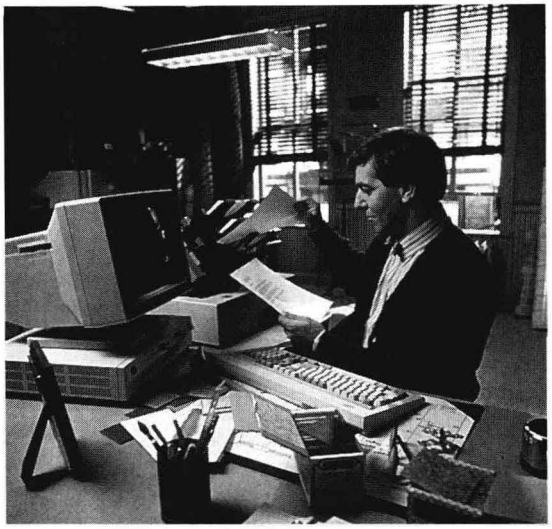


■ Pascal's mechanical calculator

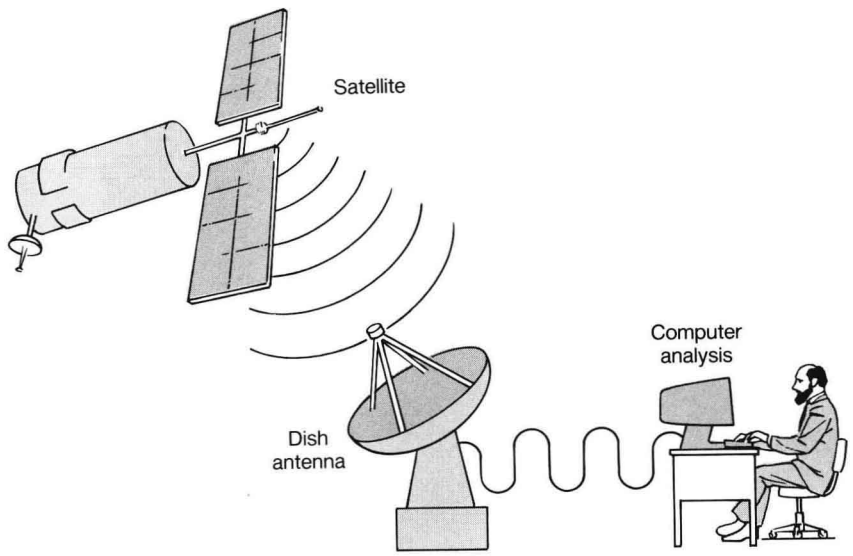


■ Burroughs automatic calculator

■ An IBM personal system/2



■ Transmission of data to and from the computer via satellite



Aiken of Harvard University developed the Mark I in 1944. Dr. John Atanasoff of Iowa State University, with the help of his graduate student Clifford Berry, developed the ABC computer. But it was not until 1946 that the first electronic calculator, known as ENIAC, was produced by J. P. Echert and J. W. Mauchly at the University of Pennsylvania. (It became known much later that the key ideas developed by Mauchly had been obtained from Atanasoff.) From this time on, the development of computers has been so rapid and fantastic that it boggles the mind. John von Neumann developed the concept of *stored programs;* that is, how to keep programs (sets of instructions to the computer) in the computer's memory. Echert and Mauchly produced the first commercial computer, UNIVAC I, in 1950. (The head programmer of the UNIVAC was Dr. Grace Hopper, who later made very important contributions to the development of the COBOL language.)

As the technology improved, hundreds of vacuum tubes were replaced by tiny transistors, miles of wiring were replaced by printed circuits, and room-size computers were replaced by hand-held calculators. Today, the state of the art is still changing and leaping forward.

Historical Development of COBOL

The rapid technological advances of the 1950s made it necessary to increase the efficiency of programming languages. When computers were first brought on the market, all programming was written in *machine language.* This language is heavily dependent on the electronic makeup of a particular computer; hence, programs had to be written for specific computers. Programs prepared for one computer could not be used on another without substantial rewriting at substantial additional expense. Institutions could not share programs unless they had identical computers. If an industry updated its computer hardware, its applications programs had to be rewritten. Such a state of affairs was obviously unacceptable.

In 1959, a number of computer professionals representing the interests of the U.S. government, the world of finance, universities, the insurance industry, and a variety of commercial industries and computer manufacturers agreed to meet for the express purpose of developing a business-oriented programming language that would be machine-independent. This group was called the Conference on Data Systems Languages (CODASYL). Its efforts culminated in the publication in the latter part of 1960 of the first version of COBOL. The word *COBOL* is an acronym for COmmon Business-Oriented Language. Some of the important features CODASYL wanted to incorporate in this language were the following:

1. It was to be machine-independent, thus allowing the same program to be processed on more than one computer. This objective was achieved to a remarkable degree, as you will learn in subsequent chapters.
2. It was to be easily maintainable. It was to allow for expansion, and for the inclusion of additional features to take advantage of technological improvements anticipated in later-generation computers. The subsequent versions COBOL-61, COBOL-61 extended (1963), COBOL Edition 65 (1965), COBOL Edition 1974, and COBOL Edition 1985 attest to the success of this feature.
3. It was to be free of mathematical and scientific symbols. In fact, the syntax, as developed, looks very much like the English language. It has a sentence structure and uses verbs, names, phrases, and clauses. This structure, along with a great versatility in the selection of names, allows programs to be largely self-documenting.

Though CODASYL still assumes the responsibility to update and maintain the COBOL language, the American National Standards Institute (ANSI) entered the picture in 1968, when it established COBOL as a standard language. The versions of COBOL as specified by ANSI are referred to as ANS COBOL. Most computer manufacturers provide COBOL compilers for their computers that comply with the specifications established by ANSI, but they also include extended features that capitalize on the unique features of their hardware configurations.

In programming establishments throughout the country, the 1974 version of COBOL is gradually being replaced by the 1985 version. This text generally adheres to the newest version.

Computers

Today, there are computer systems of all sizes and capabilities. Microprocessors are used in cars to increase efficiency, minimize fuel consumption, display road maps, and control passenger area climate (see Figure i.1). Other microprocessors are used in home heating systems, controlling the amount of heat to be produced in relation to ambient outdoor temperatures according to a programmed time schedule.