

**IBM**<sup>®</sup>

SYSTEM/360 COBOL  
COBOL Program Fundamentals  
Reference Handbook

**Programmed Instruction Course**

**IBM**

**SYSTEM/360 COBOL  
COBOL Program Fundamentals  
Reference Handbook**

**Programmed Instruction Course**

Minor Revision (July 1967)

This publication, R29-0206-1, is a reprint of Form R29-0206-0 incorporating minor editorial changes made on pages dated (7/67). The original publication is not obsoleted.

Copies of this publication can be obtained through IBM Branch Offices.  
Address comments concerning the contents of this publication to:  
IBM DPD Education Development, Education Center, Endicott, New York

## PREFACE

This reference handbook provides useful information for people who want to be able to read COBOL programs with a high degree of comprehension. It is designed to be studied in conjunction with the COBOL Program Fundamentals programmed instruction textbook (Form R29-0205).

The reader of these publications is expected to have prior experience in data processing and computer programming, as well as knowledge of System/360 features, but no prior knowledge of COBOL.

This publication is not intended to provide all of the information a student needs in order to compose original COBOL programs. Additional information for that purpose is given in the next course of this series, Writing Programs in COBOL. The publications that make up that course are a programmed instruction text (Form R29-0210) and a reference handbook (Form R29-0211).

Complete specifications for System/360 COBOL may be found in the reference manual, IBM Operating System/360 COBOL Language (Form C28-6516), or the reference manual IBM System/360 Disk and Tape Operating Systems COBOL Language Specifications (Form C24-3433). The appropriate manual should be selected based on the operative system that you plan to use.

## ACKNOWLEDGEMENT

The following information is reprinted from COBOL Edition 1965, published by the Conference on Data Systems Languages (CODASYL), and printed by the U. S. Government Printing Office.

"Any organization interested in reproducing the COBOL report and specifications in whole or in part, using ideas taken from this report as the basis for an instruction manual or for any other purpose is free to do so. However, all such organizations are requested to reproduce this section as part of the introduction to the document. Those using a short passage, as in a book review, are requested to mention "COBOL" in acknowledgment of the source, but need not quote this entire section.

"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 COBOL 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.

"Procedures have been established for the maintenance of COBOL. Inquiries concerning the procedures for proposing changes should be directed to the Executive Committee of the Conference on Data Systems Languages.

"The authors and copyright holders of the copyrighted material used herein

FLOW-MATIC (Trademark of Sperry Rand Corporation), Programming for the Univac (R) I and II, Data Automation Systems copyrighted 1958, 1959, by Sperry Rand Corporation; IBM Commercial Translator Form No. F28-8013, 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 of similar publications".

## TABLE OF CONTENTS

### Introduction

Introduction to COBOL . . . . .	3
COBOL programming system terms . . . . .	4

### Language Elements

Language elements . . . . .	7	Literals . . . . .	.12
Reserved words . . . . .	8-9	Level numbers . . . . .	.13
Programmer-supplied names . . . . .	.10	Pictures . . . . .	.14
Symbols . . . . .	.11		

### Program Structure and Contents

Program structure . . . . .	16-17
Program contents . . . . .	.18

### Identification Division

Identification division entries . . . . .	.21
--	-----

### Environment Division

Environment division entries . . . . .	25-26
---	-------

### Data Division

Data division entries . . . . .	28-30	File description entry . . . . .	32-34
System/360 COBOL terms for units of data . . . . .	.31	Record descriptions . . . . .	.35
		Item description entries . . . . .	36-42

### Procedure Division

Procedure division entries . . . . .	44-45	Procedural words . . . . .	48-51
Procedures . . . . .	.47	Test conditions . . . . .	52-54
		Flow of control . . . . .	55-57

### Case Study

Case study . . . . .	61-67
----------------------	-------

## Introduction

## INTRODUCTION TO COBOL

Origins. COBOL (COmmon Business Oriented Language) is the result of an effort to establish a standard language for programming computers to do business data processing. The original specifications for COBOL were drawn up in 1959 by representatives of several computer manufacturers and users. The specifications have been revised and improved several times since 1959.

Aims. COBOL is designed for producing source programs that are

- standardized, using standard language elements in standard entry formats within a standard program structure. COBOL endeavors to provide one common language for all computers, regardless of make or model.
- easy to understand, because they are written in English. The bulk of every COBOL program is made up of English words in entries that resemble English sentences. Good COBOL programs are easy to read and comprehend, for non-programmers as well as for programmers.
- oriented to business procedures, not to the technology of computing machinery. This makes it possible for business people who are not computer experts to use COBOL.

Differences. In order to adjust to major differences in computers, certain language differences are allowed in COBOL for individual computer systems, within the framework of one common language. System/360 COBOL is different in some ways from COBOL for other computers. (This handbook is concerned only with System/360 COBOL.)

## COBOL PROGRAMMING SYSTEM TERMS

COBOL program: a source program written in COBOL, from which an object program is compiled.

Object program: the machine language program compiled from a COBOL program.

Compile: to use a computer to produce an object program from a COBOL program. During compilation, listings of the source and object programs are printed, as well as diagnostic messages that pinpoint errors the compiler has discovered in the COBOL program.

COBOL compiler: a program supplied by IBM that directs the computer during compilation.

Source computer: the computer used to compile the object program.

Object computer: the computer used to execute the object program.

## Language Elements

## LANGUAGE ELEMENTS

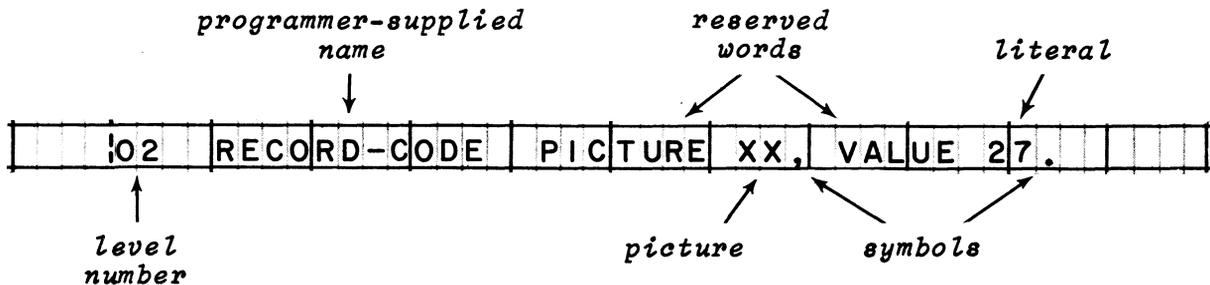
The COBOL language is made up of these elements:

- Reserved words
- Programmer-supplied names
- Symbols
- Literals
- Level numbers
- Pictures

Programmers compose programmer-supplied names, literals, and pictures. Of course, there are rules that govern the choice and arrangement of characters. Within the latitude permitted by the rules, however, programmers are free to compose an almost infinite number of names, literals, and pictures to suit particular needs in programs.

By contrast, the reserved words, symbols, and level numbers are provided in fixed sets, from which programmers select the ones they need. Programmers are not allowed to invent new reserved words, symbols, or level numbers. Even so, there are rules to follow -- for example, in determining which reserved word to use for a particular entry.

Examples of elements. The sample COBOL program entry below contains all six elements.



## RESERVED WORDS

Approximately 250 English words and abbreviations have been set aside to be used only for certain purposes. Special meanings have been preassigned to the reserved words; therefore, the programmer

- does not define reserved words.
- has no way of changing the meanings of reserved words.
- cannot add words of his own to the reserved word list.
- cannot substitute other words for those on the list.
- must not alter or misspell reserved words.
- may use reserved words only for specified purposes.

Types of reserved words. Some of the main types of reserved words are:

- words that identify program units; for instance, SECTION, ENVIRONMENT, and WORKING-STORAGE.
- words that identify or explain parts of entries; for instance, BLOCK, PICTURE, and VALUE.
- words that specify actions to be taken, like READ, MOVE, and ADD.
- words with specific functional meanings, such as NEGATIVE, COMPUTATIONAL, and EQUAL.
- words that represent certain data values; for instance, ZERO, SPACES, and HIGH-VALUE. (See "Figurative constants" below.)

Figurative constants. As a rule, it is up to the programmer to define data items and to supply names for them. However, a few data items with predefined values have been built right into the COBOL language. The names of these built-in data items are reserved words which are called "figurative constants".

The most frequently used figurative constants are ZERO and SPACE. These reserved words (and their plural forms, ZEROS or ZEROES, and SPACES) represent the data characters zero and blank, respectively. The programmer may use these words whenever zero or blank values are required in a program; for instance, he might write MOVE ZEROS TO TOTAL-WAGES in order to put all-zeros into a data item. Similarly, he might write MOVE SPACE TO CONTROL-CODE in order to blank-out a data item.

(Continued on next page)

RESERVED WORDS (continued)

Complete list of reserved words for System/360 COBOL.

ACCEPT	DE	INDEXED	PAGE	SELECT
ACCESS	DECIMAL-POINT	INDICATE	PAGE-COUNTER	SENTENCE
ACTUAL	DECLARATIVES	INITIATE	PERFORM	SEQUENTIAL
ADD	DEPENDING	INPUT	PF	SIZE
ADVANCING	DESCENDING	INPUT-OUTPUT	PH	SORT
AFTER	DETAIL	INSTALLATION	PICTURE	SOURCE
ALL	DIRECT	INTO	PLUS	SOURCE-COMPUTER
ALPHABETIC	DIRECT-ACCESS	INVALID	POSITIVE	SPACE
ALTER	DISPLAY	I-O	PRINT-SWITCH	SPACES
ALTERNATE	DISPLAY-ST	I-O-CONTROL	PROCEDURE	SPECIAL-NAMES
AND	DIVIDE	IS	PROCEED	STANDARD
APPLY	DIVISION		PROCESS	STOP
ARE		JUSTIFIED	PROCESSING	SUBTRACT
AREA	ELSE		PROGRAM-ID	SUM
AREAS	END	KEY	PROTECTION	SYMBOLIC
ASCENDING	ENDING			SYNIN
ASSIGN	ENTER	LABEL	QUOTE	SYSOUT
AT	ENTRY	LABELS	QUOTES	SYSPUNCH
AUTHOR	ENVIRONMENT	LAST		
	EQUAL	LEADING	RANDOM	TALLY
BEFORE	ERROR	LESS	RD	TALLYING
BEGINNING	EVERY	LINE	READ	TERMINATE
BLANK	EXAMINE	LINE-COUNTER	READY	THAN
BLOCK	EXHIBIT	LINES	RECORD	THEN
BY	EXIT	LINKAGE	RECORDING	THRU
		LOCK	RECORDS	TIMES
CALL	FD	LOW-VALUE	REDEFINES	TO
CF	FILE	LOW-VALUES	REEL	TRACE
CH	FILES		RELATIVE	TRACK-AREA
CHANGED	FILE-CONTROL	MODE	RELEASE	TRACKS
CHARACTERS	FILE-LIMIT	MORE-LABELS	REMARKS	TRANSFORM
CHECKING	FILLER	MOVE	REPLACING	TRY
CLOCK-UNITS	FINAL	MULTIPLY	REPORT	TYPE
CLOSE	FIRST		REPORTING	
COBOL	FOOTING	NAMED	REPORTS	UNIT
CODE	FOR	NEGATIVE	RERUN	UNIT-RECORD
COLUMN	FORM-OVERFLOW	NEXT	RESERVE	UNITS
COMMA	FROM	NO	RESET	UNTIL
COMPUTATIONAL		NOT	RESTRICTED	UPON
COMPUTATIONAL-1	GENERATE	NOTE	RETURN	USAGE
COMPUTATIONAL-2	GIVING	NUMERIC	REVERSED	USE
COMPUTATIONAL-3	GO		REWIND	USING
COMPUTE	GREATER	OBJECT-COMPUTER	REWRITE	UTILITY
CONFIGURATION	GROUP	OCCURS	RF	
CONSOLE		OF	RH	VALUE
CONTAINS	HEADING	OH	RIGHT	VARYING
CONTROL	HIGH-VALUE	OMITTED	ROUNDED	
CONTROLS	HIGH-VALUES	ON	RUN	WHEN
COPY	HOLD	OPEN		WITH
CORRESPONDING		OR	SA	WORKING-STORAGE
CREATING	IBM-360	ORGANIZATION	SAME	WRITE
CYCLES	IDENTIFICATION	OTHERWISE	SD	WRITE-ONLY
	IF	OUTPUT	SEARCH	
DATA	IN	OV	SECTION	ZERO
DATE-COMPILED	INCLUDE	OVERFLOW	SECURITY	ZEROES
DATE-WRITTEN				ZEROS

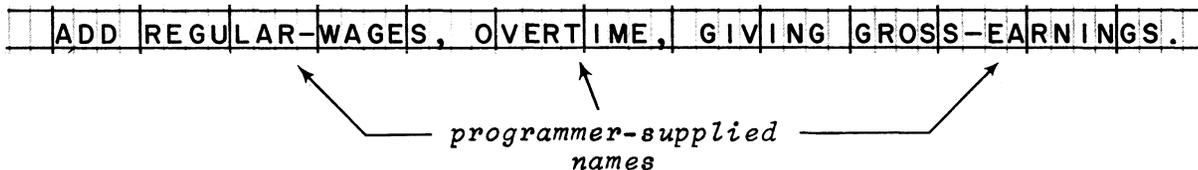
## PROGRAMMER-SUPPLIED NAMES

Names for data items, data conditions, and procedures are supplied by programmers. These names must be defined within the program in which they are used, since, unlike reserved words, they do not have pre-assigned meanings.

### Rules governing programmer-supplied names.

- A name may be as many as 30 characters long.
- It may contain letters, digits, and hyphens.
- Names of procedures may be composed entirely of digits, but names of data items and data conditions must contain at least one letter.
- A name must not begin or end with a hyphen, although there may be hyphens anywhere else in the name.
- Spaces (blanks) must not appear within a name.
- No name may be spelled exactly the same as a reserved word.

### Examples of programmer-supplied names in an entry.



## SYMBOLS

Symbols are special characters which, individually, have particular meanings for the compiler.

Punctuation symbols -- used to punctuate program entries.

.	period	used to terminate entries
,	comma	used to separate operands or clauses in a series
;	semicolon	used to separate clauses in a series
'	quotation mark	used to enclose non-numeric literals
()	parentheses	used to enclose subscripts

Arithmetic symbols -- found in arithmetic formulas.

+	plus	addition; "plus"
-	minus	subtraction; "minus"
*	asterisk	multiplication; "times"
/	slash	division; "divided by"
**	two asterisks	exponentiation; "raised to the power of"
=	equal	"make equal to"
()	parentheses	used to enclose quantities, to control the sequence in which operations are performed

Condition symbols -- found in expressions which involve tests of data conditions.

=	equal	"is equal to"
>	greater than	"is greater than"
<	less than	"is less than"
()	parentheses	used to enclose expressions, to control the sequence in which conditions are evaluated



## LEVEL NUMBERS

Level numbers are found in entries that assign names to data items and data values. The major purpose of level numbers is to designate the levels of data items, in relation to each other. The numbers that may be used are 01 through 49; 77; and 88.

Level numbers 01 through 49 are used to designate the levels of data items that form records. Level number 01 is always assigned to the record as a whole, while 02 through 49 are assigned to items that are parts of records.

Level number 77 is used in entries that describe independent data items, that is, items that are not records or parts of records.

Level number 88 is used in entries that assign names to specific values that data items may assume.

When an entry contains a level number, the level number is always the first element of the entry. The level number is followed either by a programmer-supplied name or the reserved word FILLER.

### Examples of level numbers in entries.

01	TRANSACTION-RECORD.			
02	RECORD-IDENTIFICATION.			
03	RECORD-CODE, PICTURE X.			
88	INCOMING, VALUE 'A'.			
88	OUTGOING, VALUE 'B'.			
03	ACCOUNT-NUMBER, PICTURE 999.			
02	CUSTOMER-IDENTIFICATION.			



## Program Structure and Contents

## PROGRAM STRUCTURE

COBOL programs are composed of entries arranged in divisions, sections, and paragraphs. In general, a division is made up of sections, and a section is made up of paragraphs.

Divisions. All COBOL programs are divided into four separate divisions. The divisions have fixed names -- IDENTIFICATION, ENVIRONMENT, DATA, and PROCEDURE; and always appear in that order in a program.

The beginning of each division is marked by a division header entry, which consists of the name of the division followed by the word DIVISION and a period. A division header always appears on a line by itself.

Sections. Sections are not found in the Identification division. The Environment and Data divisions always contain sections, and the sections in those divisions have fixed names. In the Procedure division, sections are optional; there programmers may, if they wish, create sections and supply names for them.

Each section is identified by a header entry which consists of the section name followed by the word SECTION and a period. A section header usually appears on a line by itself.

Paragraphs. All of the divisions except the Data division contain paragraphs. In the Identification and Environment divisions, the names of all paragraphs are fixed. In the Procedure division, paragraph names are supplied by programmers.

Paragraphs are identified by header entries which consist of a name followed by a period. Paragraph headers do not contain the word PARAGRAPH. Also, a paragraph header does not have to appear on a line by itself; it must be the first entry on a line, but it may be followed on the same line by other entries of that paragraph.

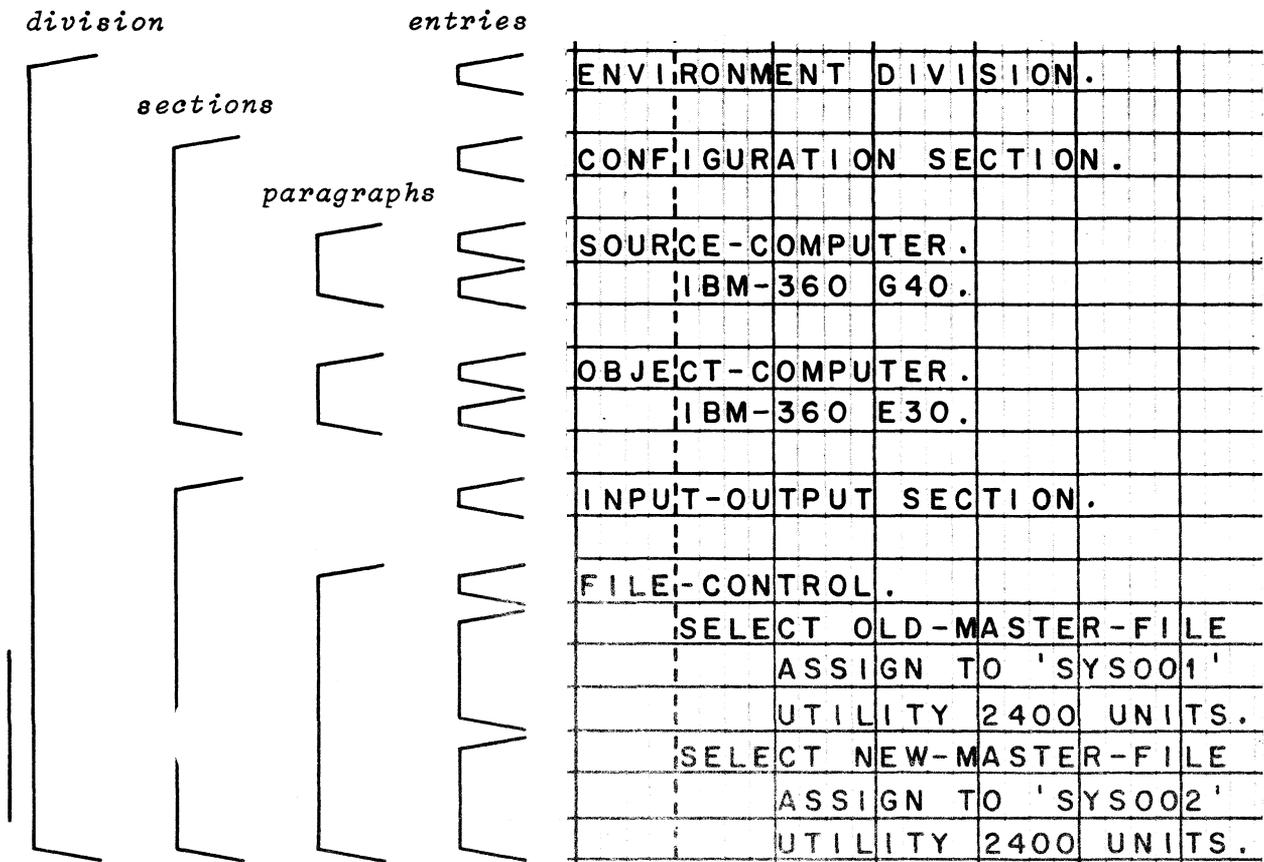
Entries. An entry can be defined as a series of two or more language elements, the last of which is a period. (However, the programmer cannot arbitrarily string together a bunch of elements, and call them an entry. The sequence of elements in each entry is dictated by precise format rules.)

A paragraph header is probably the simplest entry, since it consists of a reserved word or a programmer-supplied name, and a period. Likewise, division and section headers are relatively simple entries. Most entries, though, are longer and more complex.

(Continued on next page)

PROGRAM STRUCTURE (continued)

Sample division with structural units identified.



## PROGRAM CONTENTS

Identification division. The Identification division contains information that identifies the program. It is intended, for the most part, to inform people who read the program.

At the very least, the division states the name of the program. Usually it contains further information about the program, such as when the program was written and who the programmers were. There may also be remarks that explain the data processing job for which the program was written.

Environment division. The Environment division contains information about the equipment that will be used when the object program is compiled and executed. Most importantly, it ties together the devices of the computer system and the data files that will be processed.

The model numbers of the System/360s on which the program will be compiled and run may be given. Each data file, by name, is assigned to an input-output device. Sometimes, special input-output techniques are specified.

Data division. The Data division describes the data to be processed by the object program. It describes the data items that make up each of the files named in the Environment division, and in addition, describes the data items that make up working-storage -- such as constants and work areas.

Entries in this division show how the data items are grouped and organized into records and files. Data names, pictures, and other information about the data items are given.

Procedure division. The Procedure division specifies the actions that are required to process the data. Also, it indicates the order in which the actions are to be carried out, and provides for alternate paths of action under given conditions.

The main types of actions that may be specified are input-output, arithmetic, data movement, and sequence control.

Identification Division



Environment Division



ENVIRONMENT DIVISION ENTRIES (continued)

System/360 model numbers. The Source-Computer and Object-Computer paragraphs may give a model number that consists of a letter followed by a number, for instance, F40. The letter designates the main core storage capacity of the computer, according to the following code:

C = 8,192 bytes	G = 131,072 bytes
D = 16,384	H = 262,144
E = 32,768	I = 524,288
F = 65,536	

The number that follows the letter designates the System/360 model. The number 50, for example, means System/360 Model 50.

Assignment of files to input-output devices. Files are assigned to devices in the File-Control paragraph. For every file, there is an entry that selects the file by name, and assigns it to an "external" name and to a device.

SELECT	PAYROLL,	ASSIGN TO	'SYS001'	UTILITY	2311	UNIT.
--------	----------	-----------	----------	---------	------	-------

*file name*                      *external name*              *device class*      *device number*

- File name is the programmer-supplied name by which the file is referred to in the COBOL program.
- External name (always enclosed by quotation marks) is the name by which the file will be identified on a job control card at the time that the object program is executed.
- Device class can be UTILITY, DIRECT-ACCESS, or UNIT-RECORD. The UTILITY class is composed of machines that can read and write data sequentially -- magnetic tape, disk, drum, and data cell devices. The DIRECT-ACCESS class is composed of machines that can read and write data randomly -- disk, drum, and data cell devices. The UNIT-RECORD class is composed of printers and card read/punches.
- Device number is the IBM number of a specific device. For example, 2311 means the IBM 2311 Disk Storage Drive. An exception to this rule is device number 2400, which stands for any of the magnetic tape units in the IBM 2400 series (2401, 2402, 2403, or 2404). Device number is sometimes omitted.

**Data Division**



DATA DIVISION ENTRIES (continued)

Sample Data division (continued).

FD	PURCHASE-REPORT-FILE								
	RECORDING MODE IS F								
	LABEL RECORDS ARE OMITTED								
	DATA RECORD IS PURCHASE-REPORT-LINE.								
01	PURCHASE-REPORT-LINE.								
02	FILLER				PICTURE	X(10).			
02	COMMODITY-NUMBER				PICTURE	9(5)B99B9(5).			
02	FILLER				PICTURE	X(6).			
02	COMMODITY-NAME				PICTURE	X(30).			
02	FILLER				PICTURE	X(6).			
02	PURCHASE-DATE				PICTURE	99B99B99.			
02	FILLER				PICTURE	X(4).			
02	QUANTITY				PICTURE	Z,ZZZ,ZZ9.			
02	FILLER				PICTURE	X(6).			
02	COST-PER-UNIT				PICTURE	\$\$,\$\$\$\$.99.			
02	FILLER				PICTURE	X(4).			
02	PURCHASE-COST				PICTURE	\$\$,\$\$\$,\$\$\$\$.99.			
02	FILLER				PICTURE	X(14).			
	WORKING-STORAGE SECTION.								
77	OLD-NUMBER				PICTURE	9(12).			
77	QUANTITY-TOTAL				PICTURE	9(7), VALUE ZERO.			
77	PURCHASE-COST-TOTAL				PICTURE	9(7)V99,			
						VALUE ZERO.			

(Continued on next page)



## SYSTEM/360 COBOL TERMS FOR UNITS OF DATA

Item: an area used to contain data of a particular kind. (In general, an "item" is the same as a "field".)

Group item: an item that is composed of smaller items.

Elementary item: an item that is not composed of smaller items.

Independent item: any item that is not a record and not a part of a record. Must be an elementary item. Used as a work area or to contain a constant.

Data record: the most inclusive item, usually (though not always) a group item comprising several related items. Sometimes spoken of as the "logical record". (Whenever the term "record" is used in COBOL, data record is implied -- unless label record is specified.)

Label record: a record that contains information about a file. Label records are normally written in files stored on magnetic tape or direct-access devices. Some files (such as card files) do not have label records.

Block: a unit of data, containing one or more data records, that is transferred to or from main storage at one time by an input-output device. Sometimes spoken of as the "physical record". When data records are stored on magnetic tape or direct-access devices, each block generally contains more than one data record.

File: a collection of related data records. The records in a file may have the same or different lengths and formats.



## FILE DESCRIPTION ENTRY (continued)

RECORDING MODE clause. The RECORDING MODE clause specifies whether the recording mode in this file is V, F, or U. The clause may appear as RECORDING MODE IS U, but it can also be abbreviated to RECORDING MODE U, or simply RECORDING U. The clause may be omitted when the recording mode is V.

- "Recording mode" means the same as "data record format". The three permissible recording modes are V (Variable length), F (Fixed length), and U (Unspecified length).
- Recording mode V is the only mode in which blocks of two or more variable-length records can be handled. However, it is also possible to have just one record per block and to have fixed record lengths in this mode. The distinguishing feature of mode V is that each data record includes a record-length field and each block includes a block-length field. These fields are not described in the Data division, because provision is automatically made for them.
- In recording mode F, all of the records in a file are the same length. Blocks may contain more than one record, and there is generally a fixed number of records per block. In this mode, there are no record-length or block-length fields.
- Mode U records may be either fixed or variable in length; however, there is only one record per block. There are no record-length or block-length fields.

BLOCK CONTAINS clause. The BLOCK CONTAINS clause tells either how many records are in a block, or how many characters are in a block.

When the number of records per block is given, the clause may appear as BLOCK CONTAINS 25 RECORDS, or simply BLOCK 8 RECORDS. If the number of records in a block varies, the clause tells only how many of the longest records would form the longest possible block. The clause may be omitted when there is only one record per block.

When the number of characters per block is given, the clause specifies the number of bytes that the longest block will occupy in storage.

(Continued on next page)

## FILE DESCRIPTION ENTRY (continued)

RECORD CONTAINS clause. The RECORD CONTAINS clause specifies how many characters are in the longest data record in the file. More precisely, it specifies how many bytes the longest record will occupy in storage, for instance, RECORD CONTAINS 140 CHARACTERS. It may also give the range of record sizes, as RECORD CONTAINS 82 TO 540 CHARACTERS, or in abbreviated form RECORD 80 TO 160. This clause may be omitted, since the compiler can determine the size of records from the record descriptions.

LABEL RECORDS clause. A LABEL RECORDS clause is required to appear in every file description entry. The clause may indicate that label records are standard, it may give a name for label records, or it may state that label records are omitted.

- When the clause states that LABEL RECORDS ARE STANDARD, it means that the labels have the standard System/360 label format. In this case, the labels are checked or created automatically, and the label records are not described in the COBOL program.
- When a name is given, for example, LABEL RECORDS ARE BALANCE-TOTALS, it means there are user labels in addition to standard labels. Such additional label records are described in the Linkage section of the Data division (not in the File section), and are processed by "declarative" procedures (separate from the main body of the Procedure division).
- LABEL RECORDS ARE OMITTED means either that the file has no labels at all (as in the case of a card file), or that the file has non-standard labels. (As a rule, non-standard label records are treated as if they were data records. Each label is defined as a separate file, described as a record in the File section, and processed in the main body of the Procedure division.)

DATA RECORD clause. The DATA RECORD clause gives the name of each different kind of record in the file. For example, DATA RECORDS ARE SALES, RETURNS, PAYMENTS, CHARGES. There must be at least one kind of record in the file, and there may be several kinds, so this clause appears in every file description entry. Below the file description entry, there must appear a record description entry for every record that is named in this entry.

## RECORD DESCRIPTIONS

At least one record description will be found below each file description entry. A record description is written for each type of record in the file; for example, if a file contains four types of records, there will be four record descriptions below the file description entry.

A record description shows the structure of the record: the order in which items appear in the record, and how the items are related to each other. For every item, there is an item description entry that begins with the level number of the item.

01	PURCHASING.								
	02	ACCOUNT.							
		03	CATEGORY		PICTURE	XX.			
		03	NUMBER		PICTURE	9(6).			
	02	STATUS-CODE			PICTURE	X.			
	02	TRANSACTION.							
		03	VENDOR.						
			04	NAME	PICTURE	X(25).			
			04	NUMBER	PICTURE	9(6).			
		03	PURCHASE.						
			04	DATE.					
				05	MONTH	PICTURE	99.		
				05	DAY	PICTURE	99.		
				05	YEAR	PICTURE	99.		
			04	AMOUNT	PICTURE	9(4)V99.			

- Record descriptions always begin with an entry for the level 01 item, which is the record as a whole. Usually, the record is subdivided, and entries for items that make up the record appear below the level 01 entry.
- The item description entries are made in the order in which items appear in the record.
- An entry for a group item is followed by entries for the items that make it up. A group item comprises all the items described under it, until a level number equal to or less than the level number of the group item is encountered.
- Item description entries are usually indented to help the reader comprehend the structure of the record. The entries for each level are indented more than the entries for the preceding level. Indenting is not required, however.
- Level 88 (condition name) entries may appear in a record description. These are not item description entries, and can be ignored when the structure of a record is being analyzed.
- Sometimes a record is treated as an elementary item, with no subdivisions. In such a case, the level 01 entry is the only item description entry in the record description.

## ITEM DESCRIPTION ENTRIES

There is a separate item description entry for each item. An item description entry always begins with a level number, followed by either a name or the word FILLER, and usually includes one or more descriptive clauses that begin with words like USAGE, PICTURE, VALUE, OCCURS, or REDEFINES.

02	SALARY,	PICTURE	S9(5)V99,	COMPUTATIONAL-3.
----	---------	---------	-----------	------------------

*item description  
entry*

Level number. A level number is always the first element found in an item description entry.

Level number 01 indicates that the item is a record. A record is generally a group of related items, but it may also be an elementary item.

Level numbers 02 through 49 are used for items that are subdivisions of records.

Level number 77 identifies an independent item, which is an elementary item that is not related to other items. An independent item is not a record, and not part of a record. Level 77 items are found only in the Working-Storage section.

Level number 88 designates a condition-name entry, which strictly speaking, is not an item description at all. Instead, a condition-name entry gives a name to one of the values that the preceding item can assume. Level 88 entries are found after elementary items only; however, sometimes there are two or more consecutive level 88 entries after one elementary item.

03	SHIPMENTS-THIS-MONTH,	PICTURE	9(4).						
----	-----------------------	---------	-------	--	--	--	--	--	--

*level  
number*

(Continued on next page)



ITEM DESCRIPTION ENTRIES (continued)

USAGE clause. The USAGE clause indicates the code that will be used to represent data that is stored in the item. It may be found in descriptions of both elementary and group items. The usage that is specified for a group item applies to all of the items in that group.

The words that specify usage are DISPLAY, COMPUTATIONAL, COMPUTATIONAL-1, COMPUTATIONAL-2, and COMPUTATIONAL-3. The word USAGE, or the words USAGE IS, may also appear in the clause, but these words are optional and generally not written. So if an item's usage is computational, the usage clause will probably consist of the single word, COMPUTATIONAL, but it may also be USAGE COMPUTATIONAL, or USAGE IS COMPUTATIONAL.

The word DISPLAY is also optional, and is often not written. The usage of an elementary item is assumed to be display if some other usage is not specified for it or for any group item that it is part of.

*USAGE clause*

:		03	MINIMUM-BALANCE	COMPUTATIONAL-3
:			PICTURE S9(5).	

What the usage words indicate:

If the usage is	Then the data code is	Which means that
display	external decimal --- also called BCD (binary-coded decimal), or EBCDIC (extended binary coded decimal interchange code)	one character is stored in each byte of the item; if the item is used to store a number, the rightmost byte may contain an operational sign in addition to a decimal digit
computational	binary	one binary digit is stored in each bit of the item, except the leftmost bit, in which the operational sign is stored
computational-1	internal floating-point, short (full word) format	the item has a special format designed for floating-point arithmetic operations; part of the item is stored in binary code, and part in hexadecimal code
computational-2	internal floating-point, long (double word) format	
computational-3	internal decimal --- also called packed decimal	two decimal digits are stored in each byte of the item, except the rightmost byte, in which one digit and the operational sign are stored

(Continued on next page)

ITEM DESCRIPTION ENTRIES (continued)

PICTURE clause. PICTURE clauses are found only in descriptions of elementary items. They are required for all elementary items except those whose usage is computational-1 or computational-2. (These floating-point items are not given pictures because they have a definite storage format.)

A picture always tells how many characters will be stored and what kinds of characters they will be. The characters that an item will contain are represented by picture characters such as X, A, or 9. The picture character 9, for example, represents one decimal digit; so, the picture 999 stands for three decimal digits.

In addition to characters with symbolic meanings, pictures often include numeric literals (unsigned whole numbers only) enclosed in parentheses. Such numbers are a shorthand way of repeating picture characters. The number in parentheses tells how many consecutive times the character in front of the parentheses is repeated; in other words, X(6) is another way of writing XXXXXX, and S9(9) means the same as S999999999.

*PICTURE clause*

	02	Y-T-D-DEMAND, COMPUTATIONAL,	PICTURE	S9(6)V99.
--	----	------------------------------	---------	-----------

How to identify an item from its picture.

If the picture contains	and also (possibly)	For example...	Then the item is called	And will be used to store
one or more Xs		XXX	alphanumeric	characters of any kind; letters, digits, special characters, or spaces
one or more As		A(35)	alphabetic	only letters or spaces
one or more 9s, but no editing symbols	S V P	S9(7)V99	numeric	only digits, and possibly an operational sign
one or more editing symbols; Z * \$ . , DB CR + - 0 B	9 V P	\$\$Z,ZZZ.99	report	numeric data that is edited with spaces or certain special characters when the data is moved into the item
an E, in addition to 9s	+ - .V	+.9(8)E+99	external floating-point	a decimal quantity in an edited floating-point format that includes spaces or certain special characters

Note: Pictures of all kinds of items may contain numbers in parentheses.

(Continued on next page)

ITEM DESCRIPTION ENTRIES (continued)

PICTURE clause (continued)

What some common picture characters mean.

X	Each X stands for one character of any kind -- a letter, digit, special character, or space. The picture X(12) indicates that the item will contain twelve characters, but gives no indication of what characters they will be; all twelve could be spaces, or all could be digits, or there could be a mixture of various kinds of characters.
A	Each A stands for one letter or space.
9	Each 9 stands for one decimal digit. Numbers are always described in terms of the <u>decimal</u> digits they are the equivalent of - even when the data code is <u>binary</u> .
S	S indicates that the number has an operational sign. An "operational" sign tells the computer that the number is negative or positive; it is <u>not</u> a separate character that will print as "+" or "-".
V	V shows the location of an assumed decimal point in the number. An "assumed" decimal point is <u>not</u> a separate character in storage.
P	Each P stands for an assumed zero. Ps are used to position the assumed decimal point away from the actual number. For example, an item whose actual value is 25 will be treated as 25000 if its picture is 99PPPv; or as .00025 if its picture is VPPP99.

How picture and usage are related. Certain combinations of picture and usage are not compatible. For instance, an alphabetic item cannot possibly be computational, because letters cannot be represented in binary; this item -- in fact, any item that will store letters, spaces, or special characters -- must have display usage. Therefore, the following kinds of items can only have display usage: alphanumeric, alphabetic, report, and external floating-point.

Digits, on the other hand, can be stored in any data code; so numeric items can have any usage: display, computational, computational-1, computational-2, or computational-3.

Conversely, these rules mean that a display item might have any kind of picture; but that an item with other than display usage can only have a numeric picture. (Except that computational-1 and computational-2 items have no pictures at all.)

(Continued on next page)

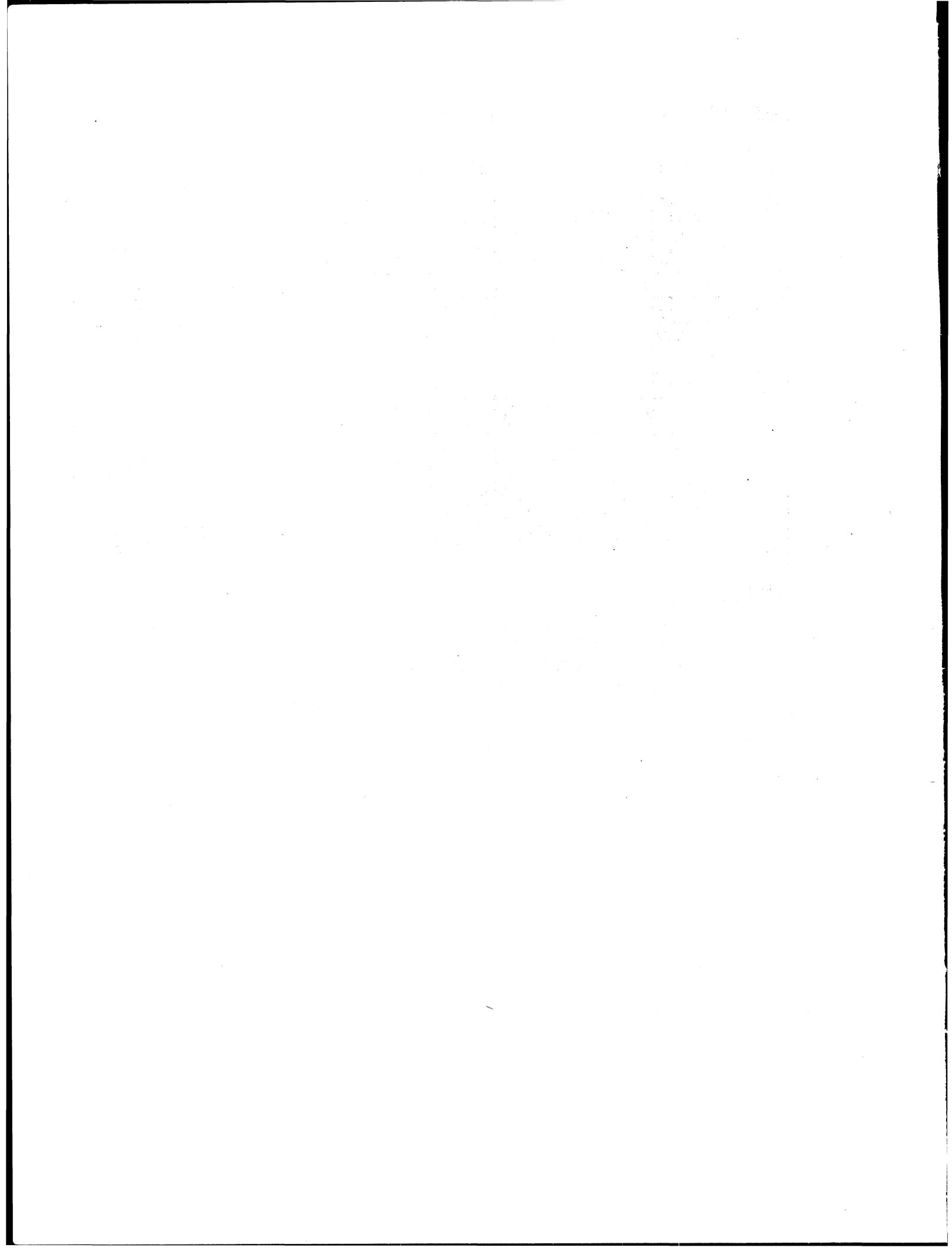




Procedure Division









## PROCEDURAL WORDS

The most commonly used actions are listed below, alphabetically. The function of each word is summarized, and a sample sentence using the word is given.

**ACCEPT.** Input. Obtains up to 80 characters of data from the system input device (which might be a disk file, tape drive, card reader, etc.), or up to 72 characters from the console keyboard. Used to read low-volume data, such as information needed to initialize program switches, balance totals, or serial numbers. Not used to read files of data (see READ).

ACCEPT CONTROL-INFORMATION.
-----------------------------

**ADD.** Arithmetic. Adds two or more numbers. Puts the sum into the data item named after the word TO, unless there is a GIVING clause. If GIVING is specified, the sum is put into the data item named after the word GIVING, and it is edited according to the item's picture; the value of this item is not added. (Numbers can also be added by using the verb COMPUTE.)

ADD REGULAR-PAY, OVERTIME-PAY, BONUS,
GIVING TOTAL-EARNED.

**CLOSE.** Input-output. Terminates the processing of one or more data files. Such actions as checking and creating end-of-file labels are done automatically when a file is closed. Used after processing is finished for any file that was opened (see OPEN).

CLOSE OLD-MASTER-FILE, NEW-MASTER-FILE.
---

**COMPUTE.** Arithmetic. Computes the value of the data item, literal, or formula written to the right of the equal sign; and puts that value into the item named after the verb. Can add, subtract, multiply, divide, or exponentiate numbers; or combine these operations. (May be used in place of the other arithmetic verbs: ADD, SUBTRACT, MULTIPLY, DIVIDE.) Edits the result according to the receiving item's picture.

COMPUTE WITHHOLDING-TAX = .18 * (GROSS -
NUMBER-OF-DEPENDENTS * 13).

(Continued on next page)





PROCEDURAL WORDS (continued)

PERFORM. Sequence control. Causes a branch to a procedure or series of procedures, and following their execution, a return branch to the statement after the PERFORM statement. Sets up the linkage for procedures to serve as closed subroutines. Also used to control the execution of loops. (Contrast with GO TO, which causes a branch but not a return.)

```
PERFORM DISCOUNT-CALCULATION.
```

READ. Input. Makes a data record from an input file available for processing. For sequential files, such as tape and card files, the READ statement contains an AT END clause, which specifies actions that are to be taken after the last record of the file has been processed. A READ statement is valid only if the file is open (see OPEN).

```
READ BEAM-LOADING-FILE;  
AT END, GO TO CLOSE-FILES.
```

STOP. Sequence control. Stops the execution of the object program, either permanently or temporarily. If STOP is followed by the word RUN, execution is stopped permanently; this would be done at the end of the job, or when a serious data error made it impossible to continue the run. If STOP is followed by a literal, such as a message to the operator, the literal is typed out and then execution is delayed until the operator takes required steps; execution is resumed at the statement after the STOP statement.

```
STOP RUN.
```

SUBTRACT. Arithmetic. Subtracts one or more numbers from another number. Puts the difference into the data item named after the word FROM, unless there is a GIVING clause. If GIVING is specified, the difference is put into the data item named after the word GIVING, and it is edited according to the item's picture; the value of this item is not used in the subtraction. (Numbers can also be subtracted by using the verb COMPUTE.)

```
SUBTRACT SAVINGS-BONDS, BLUE-CROSS,  
CHARITY-CONTRIBUTIONS FROM EARNINGS.
```

WRITE. Output. Releases a record for an output file. The actual transfer of this record to an output device may not occur until sometime later; in particular, if there are to be two or more records per block, the record may be held until there are enough records to fill a block. When the record is to be printed or punched, the WRITE statement contains an AFTER or AFTER ADVANCING clause to specify carriage control or stacker selection. A WRITE statement is valid only if the file is open (see OPEN).

```
WRITE INVENTORY-RECORD.
```







## FLOW OF CONTROL

The way in which control flows through procedures in a COBOL program represents the sequence in which instructions in the object program will be executed. How control will flow depends on the kinds of statements in the Procedure division, and their arrangement.

Starting point. Control starts at the first statement of the first procedure in the division, provided that there are no declaratives. If there are any declaratives, control starts at the first procedure after the END DECLARATIVES entry.

- Declaratives are special, optional procedures that are grouped together at the beginning of the Procedure division. Declarative procedures are logically separate from the main body of the Procedure division; that is, they cannot be affected by the flow of control through other procedures.
- If the program contains declaratives, a DECLARATIVES header will appear on the line after the PROCEDURE DIVISION header, and an END DECLARATIVES header will appear on the line after the last declarative procedure.

Sequence. Control automatically flows from one statement to the next statement in sequence, and from one paragraph to the next, except when

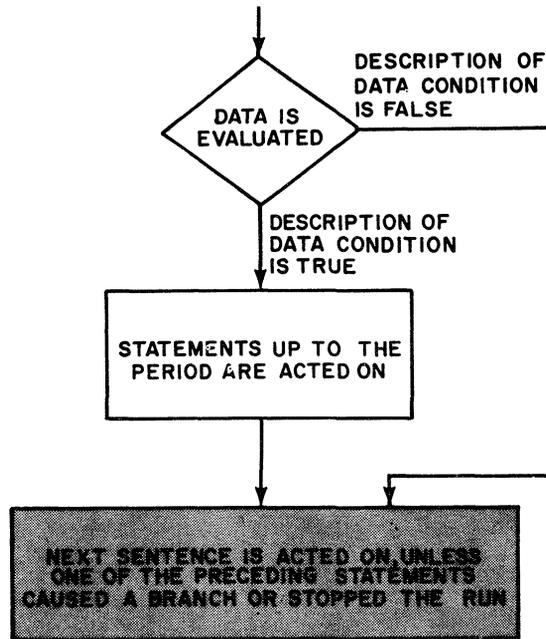
- a GO TO statement causes a branch.
- an IF sentence causes control to jump over certain statements.
- a PERFORM statement gives control temporarily to another procedure.
- a STOP statement causes a delay in execution, or terminates the run.

Branching. When a GO TO or PERFORM statement causes a branch to a procedure, control is transferred to the first statement of that procedure. It is permissible for a GO TO statement to send control back to the beginning of the procedure that the GO TO is part of.

(Continued on next page)

FLOW OF CONTROL (continued)

Flow of control through an IF sentence that does not contain ELSE or OTHERWISE.



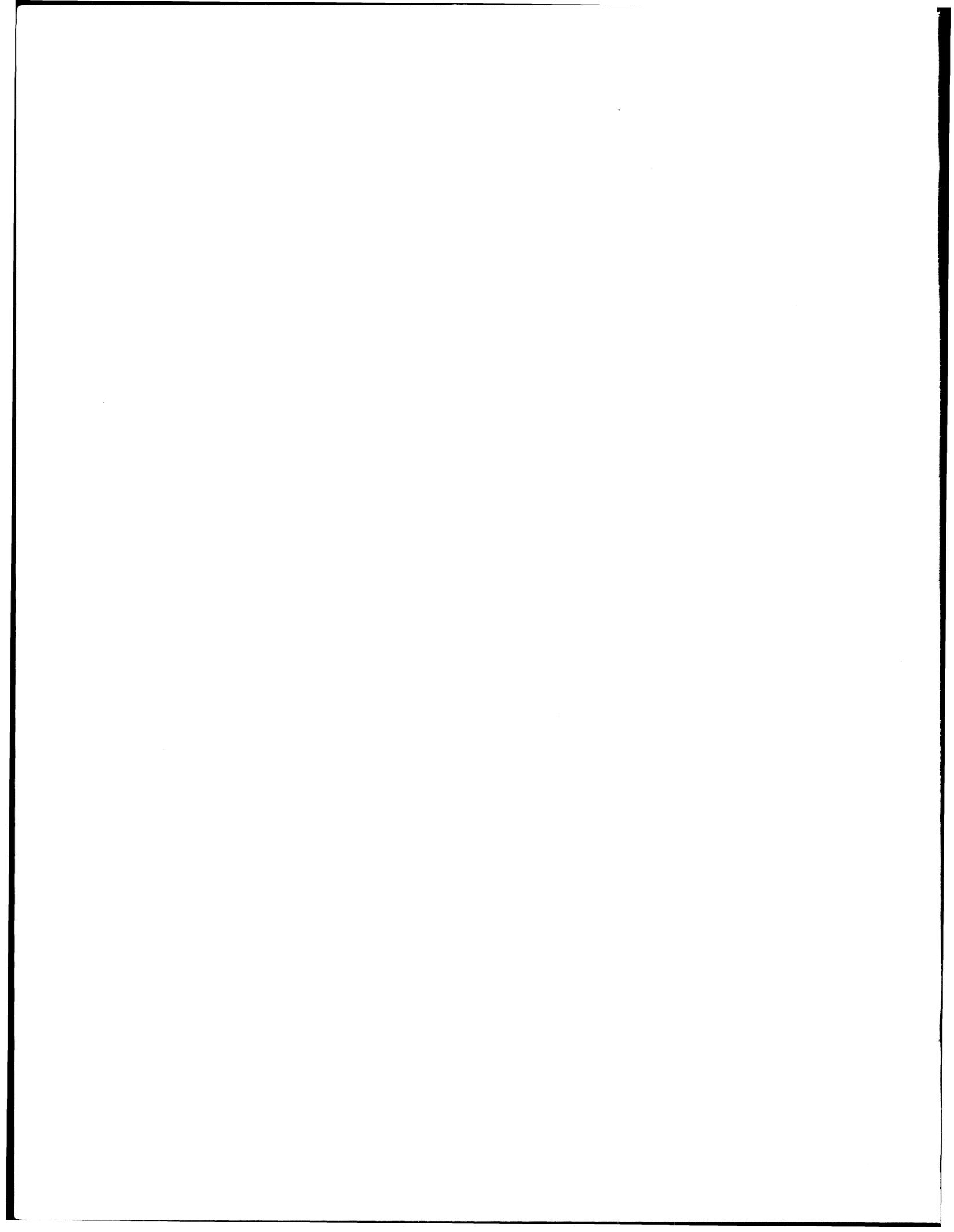
Example.

1	IF BALANCE IS NEGATIVE,
2	MOVE 'BALANCE IS IN YOUR FAVOR'
3	TO MESSAGE-AREA.
4	WRITE CUSTOMER-BILL-RECORD.

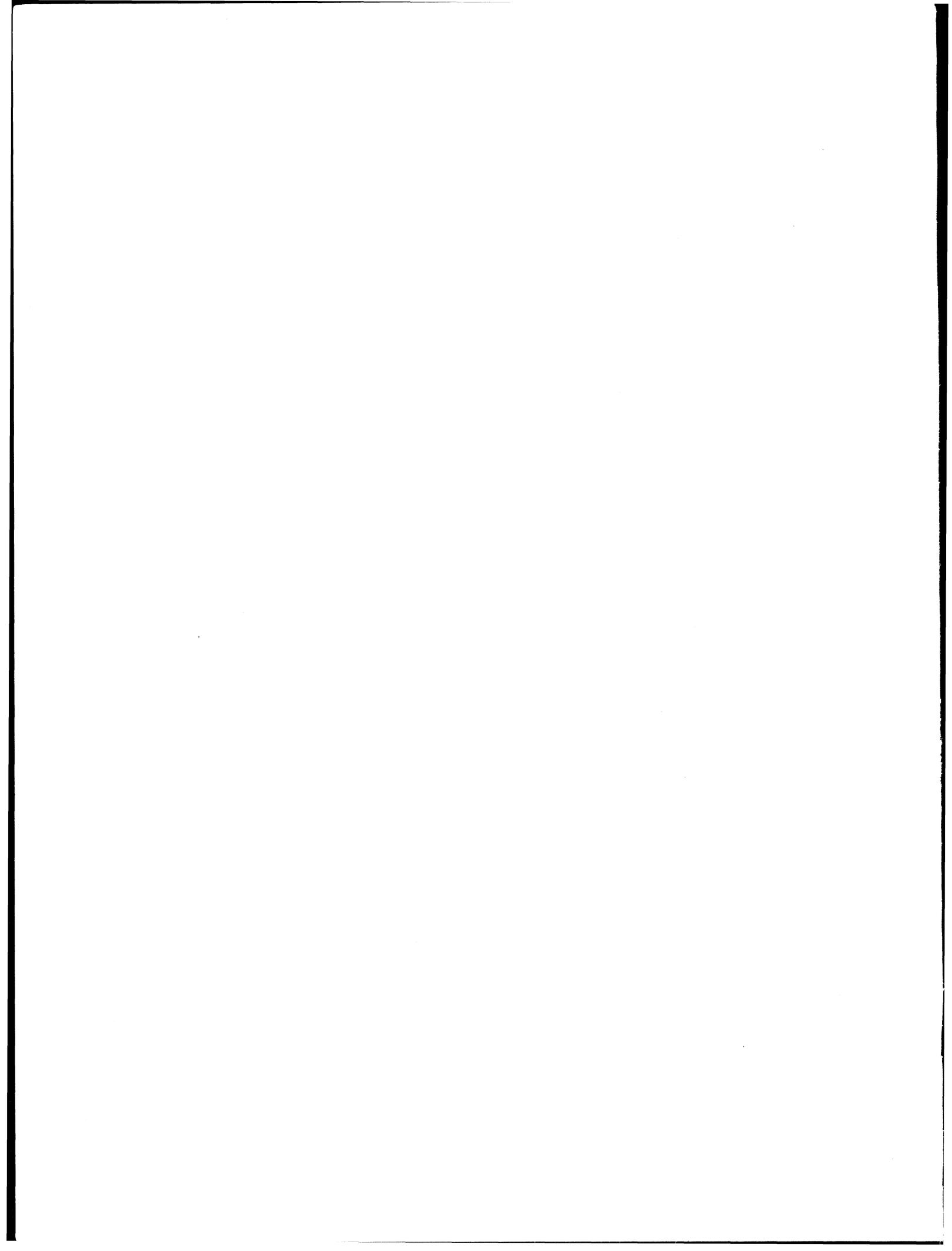
The data item (BALANCE) is evaluated to see if it is negative. If it is negative, the statements on lines 2-3 are acted on, and then control passes to the next sentence (line 4). If BALANCE is either positive or zero, control jumps directly to the next sentence (line 4).

(Continued on next page)





Case Study



CASE STUDY. Customer billing program.

Input. Billing record.

ACCOUNT IDENTIFICATION							CREDIT STATUS	
TYPE OF ACCOUNT	ACCOUNT NUMBER		BILLING CYCLE	CUSTOMER NAME	STREET ADDRESS	CITY-STATE	RATING CODE	PURCHASE LIMIT
	STORE NUMBER	FILE NUMBER						

ACCOUNT HISTORY			LAST YEAR			
YEAR OPENED	YEAR LAST ACTIVE	HIGHEST BALANCE	MONTHS ACTIVE	MONTHS OVER 90	TOTAL PURCHASES	TOTAL RETURNS

THIS YEAR TO DATE				LAST MONTH	
MONTHS ACTIVE	MONTHS OVER 90	TOTAL PURCHASES	TOTAL RETURNS	NUMBER OF TRANSACTIONS	BALANCE FORWARD

THIS MONTH								
BILLING DATE	NUMBER OF TRANSACTIONS	CURRENT BALANCE	PURCHASES		PAYMENTS		CREDITS	
			NUMBER	AMOUNT	NUMBER	AMOUNT	NUMBER	AMOUNT

COLLECTION HISTORY								
RETURNS		OVERDUE BALANCES				LAST PAYMENT		DUNNING CODE
NUMBER	AMOUNT	30 DAY	60 DAY	90 DAY	120 DAY	DATE	AMOUNT	

Output. Monthly statement.

	0				1				2				3				4				5				6				7				8			
	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
1	GENERAL STORES 7200 WEBSTER AVENUE • ENDICOTT • NEW YORK 13760																																			
2																																				
3																																				
4																																				
5																																				
6																																				
7	BALANCE FROM		TRANSACTIONS MADE THIS MONTH												ACCOUNT				BILLING																	
8	LAST BILL		PURCHASES				PAYMENTS				CREDITS				NUMBER				DATE																	
9	\$56.17		\$126.03				\$56.17				\$49.22				365 8004				10 18 65																	
10																																				
11																																				
12																																				
13																																				
14																																				
15	AMOUNT DUE		\$76.81				MRS RALSTON WRAITH JR																													
16																																				
17																																				
18																																				
19																																				
20																																				
21																																				
22																																				

(Continued on next page)



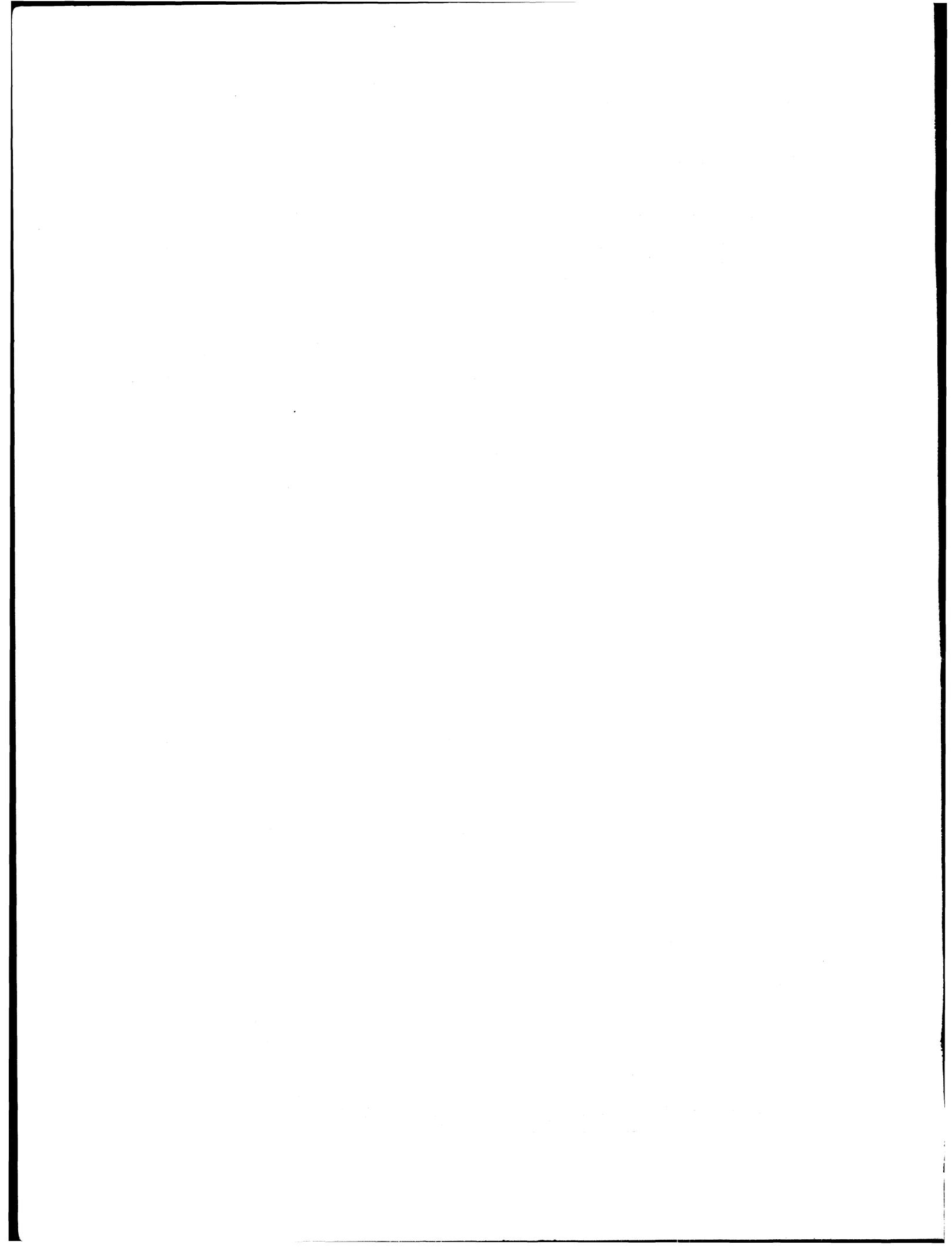












STUDENT'S COMMENT FORM

SYSTEM/360 COBOL - COBOL Program Fundamentals Reference Handbook  
R29-0206-1

Your comments, as well as answers to the following questions, will help us design and administer programmed or self-study courses in a way that better suits your needs. If your answer to a question is "No", or needs further explanation, please use the space provided below.

Comments and suggestions become the property of IBM.

- What is your occupation? \_\_\_\_\_
- Why did you take this course? \_\_\_\_\_

- |  | Yes                      | No                       |
|--|--------------------------|--------------------------|
| ● Did this course, in general, meet your needs?  | <input type="checkbox"/> | <input type="checkbox"/> |
| ● Did an IBM employee serve as your advisor?   | <input type="checkbox"/> | <input type="checkbox"/> |
| ● Did you receive a completion certificate?  | <input type="checkbox"/> | <input type="checkbox"/> |
| ● Did you find the material:<br>Easy to read and understand?   | <input type="checkbox"/> | <input type="checkbox"/> |
| Organized for convenient use?  | <input type="checkbox"/> | <input type="checkbox"/> |
| Well illustrated?  | <input type="checkbox"/> | <input type="checkbox"/> |
| ● Did you feel that any particular topics should be added or emphasized?                                 | <input type="checkbox"/> | <input type="checkbox"/> |
| ● Did you feel that any particular topics should not have been included?                                 | <input type="checkbox"/> | <input type="checkbox"/> |
| ● If you found any technical errors, please list them below, giving form number, page, and frame number. |                          |                          |

Staple

Fold

Fold

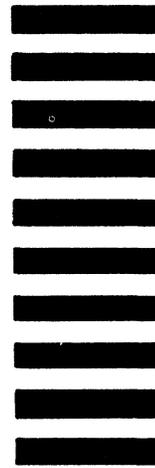
FIRST CLASS  
PERMIT NO. 10  
ENDICOTT, N. Y.

**BUSINESS REPLY MAIL**  
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . . .

IBM Corporation  
1701 North St.  
Endicott, N.Y. 13760

Attention: DP Education Development, Dept. 617



Cut Along Line

Fold

Fold

Additional Comments:

## STUDENT'S COMMENT FORM

SYSTEM/360 COBOL - COBOL Program Fundamentals Reference Handbook  
R29-0206-1

Your comments, as well as answers to the following questions, will help us design and administer programmed or self-study courses in a way that better suits your needs. If your answer to a question is "No", or needs further explanation, please use the space provided below.

Comments and suggestions become the property of IBM.

- What is your occupation? \_\_\_\_\_
- Why did you take this course? \_\_\_\_\_

- |  | Yes                      | No                       |
|--|--------------------------|--------------------------|
| ● Did this course, in general, meet your needs?  | <input type="checkbox"/> | <input type="checkbox"/> |
| ● Did an IBM employee serve as your advisor?   | <input type="checkbox"/> | <input type="checkbox"/> |
| ● Did you receive a completion certificate?  | <input type="checkbox"/> | <input type="checkbox"/> |
| ● Did you find the material:<br>Easy to read and understand?   | <input type="checkbox"/> | <input type="checkbox"/> |
| Organized for convenient use?  | <input type="checkbox"/> | <input type="checkbox"/> |
| Well illustrated?  | <input type="checkbox"/> | <input type="checkbox"/> |
| ● Did you feel that any particular topics should be added or emphasized?                                 | <input type="checkbox"/> | <input type="checkbox"/> |
| ● Did you feel that any particular topics should not have been included?                                 | <input type="checkbox"/> | <input type="checkbox"/> |
| ● If you found any technical errors, please list them below, giving form number, page, and frame number. |                          |                          |

Staple

Fold

Fold

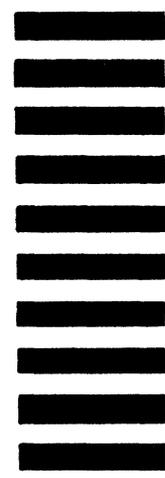
FIRST CLASS  
PERMIT NO. 10  
ENDICOTT, N. Y.

BUSINESS REPLY MAIL  
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation  
1701 North St.  
Endicott, N.Y. 13760

Attention: DP Education Development, Dept. 617

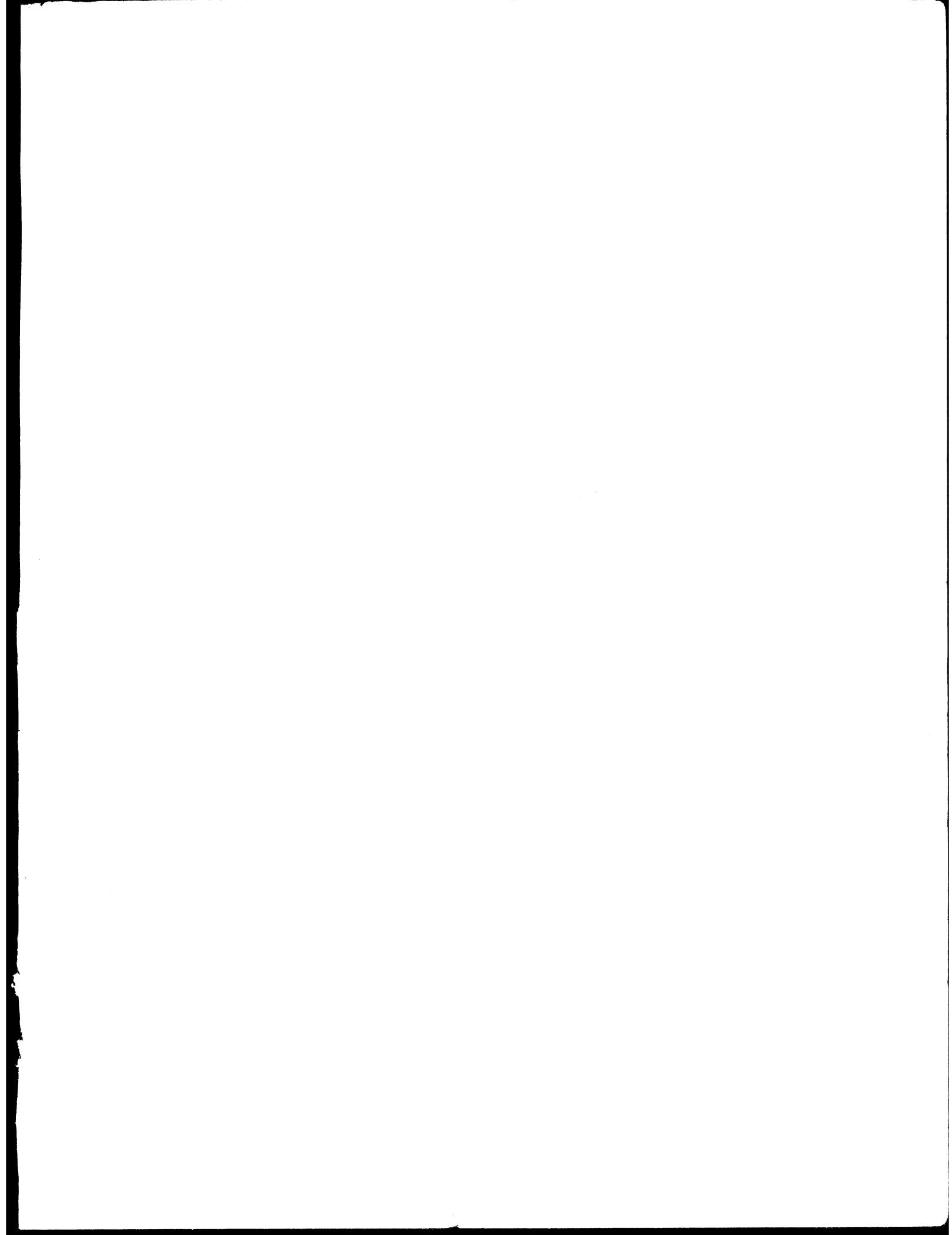


Cut Along Line

Fold

Fold

Additional Comments:



International Business Machines Corporation  
Data Processing Division  
112 East Post Road, White Plains, N.Y. 10601  
[USA Only]

IBM World Trade Corporation  
821 United Nations Plaza, New York, New York 10017  
[International]