## SYSTEM/360 COBOL

## COBOL Program Fundamentals

Reference Handbook

Programmed Instruction Course

# SYSTEM/360 COBOL COBOL Program Fundamentals Reference Handbook 

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.

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 acknowledgement 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, l959, 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



Reserved words. . . . . . . 8-9
Programmer-supplied names . . 10
Symbols

## 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
System/360 COBOL terms for units of data . . . . . 31

Literals. . . . . . . . . . 12
Level numbers . . . . . . . . 1313

Pictures.
Pictures. ..... 14

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 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

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.


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.

RESERVED WORDS (continued)

Complete list of reserved words for System/360 COBOL.

| ACCEPT | DE |  | PAGE | SELECT |
| :--- | :--- | :--- | :--- | :--- |
| ACCESS | DECIMAL-POINT | INDEXED | 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 |  |  | PROCESSING | SUBTRACT |
| AREA | ELSE |  | PROGRAM-ID | SUM |
| AREAS | END |  |  | SROTECTION |

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 preassigned 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 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 <br> clauses in a series |
| ; semicolon | used to separate clauses in a <br> series |
| () parentheses mark | used to enclose non-numeric literals |

Arithmetic symbols -- found in arithmetic formulas.

| + | plus |
| :--- | :--- |
| - | minus |
| $*$ | asterisk |
| $/$ | slash |
| ** | two asterisks |
| = |  |
| () parentheses |  |

addition; "plus"
subtraction; "minus"
multiplication; "times"
division; "divided by"
exponentiation; "raised to the power of"
"make equal to"
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
> greater than
< less than
() parentheses
```

"is equal to"
"is greater than"
"is less than"
used to enclose expressions,
to control the sequence in which
conditions are evaluated

## LITERALS

A literal is an actual value used in a program. Literals are supplied by programmers; but unlike a programmer-supplied name, a literal describes itself and needs no separate definition in the program. The two main types of literals are "numeric" and "non-numeric" literals.

Rules governing numeric literals.

- A numeric literal may be made up of digits, a plus sign or a minus sign, and a decimal point.
- It may contain as many as 18 digits.
- If there is a sign it must be the leftmost character of the literal.
- If there is no sign, the number is assumed to be positive.
- A decimal point may appear anywhere in the number except as the rightmost character. Whole numbers are written without decimal points.

An example of a numeric literal in an entry.
$\square$
numeric
literal

Rules governing non-numexic literals.

- A non-numeric literal is always enclosed by quotation marks. The quotation marks are not part of the literal.
- It may be as many as 120 characters long.
- It may contain any character except a quotation mark. So, digits, letters, spaces, and all special characters except one, may be found in non-numeric literals.

An example of a non-numeric literal in an entry.

non-numeric
Ziteral

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.


Pictures describe certain characteristics of data items, such as

- how many characters an item contains.
- whether the characters are numeric, alphabetic, or alphanumeric.
whether the item has a sign.
where an assumed decimal point is located.
- what editing (deletion, replacement, or insertion of characters) is to be done to form the item.

Each picture is a string of characters, and may be from 1 to 30 characters long. Pictures are composed of one or more of the characters listed below. (Each character or pair of characters has a symbolic meaning; the meanings of some of these characters are explained later in this book.)

Pictures may also contain numeric literals (unsigned whole numbers only) enclosed in parentheses. The literals provide an abbreviated way of repeating a picture character; for instance, $X(20)$ means the same as 20 Xs in a row.

In an entry, a picture is always preceded by the word PICTURE or by the words PICTURE IS.

Examples of pictures in entries.


Program Structure and Contents

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 divisjon, 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 haader 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.

Sample division with structural units identified.


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

Unquestionably the simplest division, the Identification division is only required to have the following three entries:

- Division header
- PROGRAM-ID paragraph header
- Program name, enclosed in quotation marks

The division may also contain up to six additional paragraphs. Fixed names are provided for these paragraphs (AUTHOR, INSTALLATION, DATEWRITTEN, DATE-COMPILED, SECURITY, and REMARKS), but the programmer is free to write any number of entries in each paragraph, and to give any desired information in the entries. The entries may contain any characters and words, including reserved words.

## Sample Identification division.

| IDENITIFIC | CATI | ON D 1 | IVIS | 10 N. |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | , |  |  |  |  |  |  |  |  |  |
| PROGIRAM-I | 1 D. |  |  |  |  |  |  |  |  |  |
| I'PURC | CHAS | $E^{\prime}$ |  |  |  |  |  |  |  |  |
| 1! |  |  |  |  |  |  |  |  |  |  |
| AUTH'OR. |  |  |  |  |  |  |  |  |  |  |
| 'PAUL | MEL | I CHAR | R. |  |  |  |  |  |  |  |
| -1] |  |  |  |  |  |  |  |  |  |  |
| InStiAllat | TION |  |  |  |  |  |  |  |  |  |
| IPURCH | HASI | NG R | ECORD | DS D | EPAR | TMEN | T. |  |  |  |
| $\cdots$ |  |  |  |  |  |  |  |  |  |  |
| DATEEWRIT | TTEN |  |  |  |  |  |  |  |  |  |
| IORIG | INAL | PROG | GRAM | WR I | TTEN | JUL | Y, 196 | 62. |  |  |
| THIS | IS | REVIS | SION | 4, | WRIT | TEN | SEPTE | EMBE | R, 19 | 65 |
| 1 |  |  |  |  |  |  |  |  |  |  |
| REMAIRKS |  |  |  |  |  |  |  |  |  |  |
| IOUTPU | UT O | F TH | 15 P | ROGR | AM 1 | $S$ A | REPOR | RT 0 | F |  |
| IALL P | PUIRC | HASES | S FOR | R TH | E PR | EVIOU | US MO | ONTH |  |  |
| THE R | REPO | RT I S | $S$ RE | QUIR | ED B | Y TH | E 5TH | H DA | $Y$ |  |
| IOF EA | ACH | MONT | H, AN | ND 1 | S DI | STRI | BUTED | D TO |  |  |
| 'ALL P | PURC | HASIN | NG R | EPRE | SENT | ATIV | ES. | 1 NPU | T |  |
| IIS TH | HE P | URCH | ASING | G RE | CORD | S CA | RD F | +LE, |  |  |
| ISORTE | ED B | Y PU\|R | RCHA\|S | SE D | ATE | W/TH\| | IN CIO | OMMO | D I TY |  |

## Environment Division

The Environment division may have two sections, the Configuration section and the Input-Output section. The Configuration section identifies the source and object computers. The Input-Output section (required in all programs that process input or output files) assigns files to input-output devices and may specify special input-output techniques.

## Sample Environment division.

| ENVIIRONM | ENT | DIVI | SION |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| -1 |  |  |  |  |  |  |  |  |  |  |  |
| CONFII GURA | ATION | N SE | CTIO | N. |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |
| SOURICE-CO | OMP U | TER. |  |  |  |  |  |  |  |  |  |
| 1 BM - 3 | 360 | F30. |  |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |
| OB JEICT - CO | OMP U | TER. |  |  |  |  |  |  |  |  |  |
| '1BM- | 36 | F 30. |  |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |
| INPUIT-OUT | TPUT | SEC | TION | , |  |  |  |  |  |  |  |
| I |  |  |  |  |  |  |  |  |  |  |  |
| FILEE-CON | TROL | . |  |  |  |  |  |  |  |  |  |
| I'SELEC | CT PU | URCH | ASIN | G-FIL | LE |  |  |  |  |  |  |
|  | ASSI | GN T | 0 'S | YSO15 | $5^{\prime}$ U | NIT- | RECOR | RD 2 | 540 R | UNI | T. |
| ISELE | CT PU | URCH | ASE- | REPOR | RT-F | 1 LE |  |  |  |  |  |
| 1 | ASSI | GN T | 0 's | YSO27 | $7{ }^{\prime}$ | NIT- | RECO | RD 1 | 403 | UNIT | - |
| - |  |  |  |  |  |  |  |  |  |  |  |
| 1-0-CONT | ROL. |  |  |  |  |  |  |  |  |  |  |
| 'APPLY | $Y$ LA | ST-L | I NE | TO FO | ORM- | O VER | F LOW |  |  |  |  |
| 1 | ON P | URCH | ASE- | REPOR | RT-F | 1 LE. |  |  |  |  |  |

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.


- 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.

The most commonly used sections of the Data division are the File section and the Working-Storage section. The File section contains descriptions of the data files to be processed by the program, and descriptions of the data records in those files. The Working-Storage section contains descriptions of work areas and constants.

Sample Data division.


Sample Data division (continued).

| FD IPURC | HASE | -REP | ORT- | F I LE |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| IRECOR | RD I N | G MO | DE 1 | S F |  |  |  |  |  |  |  |  |
| LABEL | L RE | CORD | $S$ ARE | E OM | ITTE | D |  |  |  |  |  |  |
| DATA | REC | ORD | $1 S$ P | URCH | ASE- | REPOR | R T-L | INE. |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| 01 'PURC | HASE | -REP | ORT-L | L INE |  |  |  |  |  |  |  |  |
| 102 | F I L | ER |  |  |  | PICTU | URE | X (10 | $)$. |  |  |  |
| 102 | C OMM | OD IT | Y - NUM | MBER |  | PICTU | URE | 9 (5) | B99B | 9(5) |  |  |
| 102 | F I LL | ER |  |  |  | PICTU | URE | $\mathrm{X}(6)$ |  |  |  |  |
| 102 | COMM | ODIT | Y - NA | ME |  | PICTU | URE | X (30 | $)$. |  |  |  |
| 102 | FILL | ER |  |  |  | PICTU | URE | $x(6)$ | - |  |  |  |
| 102 | PURC | HASE | -DAT | E |  | PICTU | URE | 9989 | 9 B99 | . |  |  |
| 102 | FILL | ER |  |  |  | PICTU | URE | $\mathrm{X}(4)$ | . |  |  |  |
| 102 | QUAN | T I TY |  |  |  | PICTU | URE | Z, Z Z | Z, Z Z | 9. |  |  |
| 102 | FILL | ER |  |  |  | PICTU | URE | X ( 6 ) | . |  |  |  |
| 102 | Cost | -PER | -UN I | T |  | PICTU | URE | \$ \$ , \$ | \$\$.9 | 9. |  |  |
| 102 | FILL | ER |  |  |  | PICTU | URE | X(4) | - |  |  |  |
| 102 | PURC | HASE | -cos | T |  | P ICTU | URE | \$ \$ , \$ | \$ \$ , \$ | \$ \$. 9 | 9. |  |
| !02 | FILL | ER |  |  |  | PICT | URE | X ( 14 | 1. |  |  |  |
| -1 |  |  |  |  |  |  |  |  |  |  |  |  |
| WORKIING- | STOR | A G E | SECT | 10 N. |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| $77 \quad$ OLD- | NUMB | ER |  |  |  | PICTU | URE | 9112 | $)$ |  |  |  |
| 77 QQUAN | TITY | -TOT | AL |  |  | PICTUR | URE | $9(7)$ | , VA | L UE | ZERO |  |
| $77 \quad$ PURC | HASE | -cos | T-TO | TAL |  | PICTU | URE | $9(7)$ | V99, |  |  |  |
| 1 |  |  |  |  |  |  | VALU | E ZE\| | RO. |  |  |  |

Another sample Data division, with entry types identified.


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.

For each file, there is a file description entry in the File section of the Data division. The entry consists of a level indicator and the file name, plus clauses that describe how many records or characters a block contains, how many characters are in a record, what the recording mode is, whether there are label records, and what data records are in the file. The file description entry will be followed by a record description for each type of record in that file.


Level indicator. File description entries always begin with a level indicator. A level indicator is a special, two-letter reserved word. The level indicator that is used most often is $F D$, which stands for File Description.

File name. The programmer-supplied name of the file always follows the level indicator.

## 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.

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 tgpes 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 PURC | HAS I | NG. |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 102 | ACCO | UNT. |  |  |  |  |  |  |  |  |
| 1 | 03 | CATE | G ORY |  |  | PICTI | URE | $\mathrm{x} \times$. | 1 |  |
| 1 | 03 | NUMB | ER |  |  | PICT | URE | $9(6)$ |  |  |
| 102 | STAT | US-C | ODE |  |  | PICT | URE | X . |  |  |
| 102 | TRAN | SACT | ION. |  |  |  |  |  |  |  |
| 1 | 03 | VEND | OR. |  |  |  |  |  |  |  |
| 1 |  | 04 | NAME |  |  | PICTU | URE | $\times 125$ | ) |  |
| 1 |  | 04 | NUMB | ER |  | PICT | URE | 9 (6) |  |  |
| 1 | 03 | PURC | HASE |  |  |  |  |  |  |  |
| 1 |  | 04 | DATE |  |  |  |  |  |  |  |
| 1 |  |  | 05 | MONT | H | PICT | URE | 99. |  |  |
| I |  |  | 05 | DAY |  | PICT | URE | 99. |  |  |
| 1 |  |  | 05 | YEAR |  | PICT | URE | 99. |  |  |
| 1 |  | 04 | AMOU | NT |  | PICTI | URE | 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.

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.


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 conditionname 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.

(Continued on next page)

## ITEM DESCRIPTION ENTRIES (continued)

Name or FILLER. Every item description entry contains either a programmer-supplied name or the reserved word FILLER, following the level number.

A name is a data name if it follows level numbers 01-49 or 77; or it is a condition name if it follows level number 88.

FILLER may be found in place of a name, following level numbers 01-49. FILLER is not a name, and therefore, cannot be used to refer to an item in a procedure. It is used in descriptions of items that will not be referred to, usually because the items will not contain any information, or because the information will not be processed.


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.


## What the usage words indicate:

| If the usage is | Then the data code is | Which means that |
| :---: | :---: | :---: |
| display | external decimal --also called BCD (binarycoded 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 |
| computational-2 | internal floating-point, long (double word) format | and part in hexadecimal code |
| 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)

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 $\mathrm{X}, \mathrm{A}$, 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 consecuti.ve times the character in front of the parentheses is repeated; in other words, $\mathrm{X}(6)$ is another way of writing XXXXXX, and S9(9) means the same as S999999999.


How to identify an item from its picture.

| If the picture contains | $\begin{aligned} & \text { and also } \\ & \text { (possibly) } \end{aligned}$ | For example... | Then the item is called | And will be used to store |
| :---: | :---: | :---: | :---: | :---: |
| one or more xs |  | \|xx| | | 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 | $\begin{gathered} \mathrm{s} \\ \mathrm{v} \end{gathered}$ | S9(7) 17 | numeric | only digits, and possibly an operational sign |
| one or more editing symbols $+{ }^{-} \mathrm{DB}_{\mathrm{O}}^{\mathrm{CR}}{ }_{\mathrm{B}}$ | $\begin{aligned} & \mathrm{g} \\ & \stackrel{\mathrm{v}}{\mathrm{p}} \end{aligned}$ | \$zz, \|zzz.199 | report | numeric data that is edited with spaces or certain special characters when the data is moved into the item |
| $\begin{aligned} & \text { an } E, \\ & \text { in addition } \\ & \text { to } 9 \mathbf{s} \end{aligned}$ | $\pm$ $\pm$ v | +.918)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 $\mathrm{X}(12)$ 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 decimal digits they are the equivalent of - even when the data code is binary. |
| 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 not 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 not 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.)

VALUE clause. The VALUE clause consists of the word VALUE, or the words VALUE IS, followed by a literal.

This clause is mainly used to assign initial values to certain elementary data items, in particular, to constants. The value assigned to an item remains constant during the execution of the object program, unless it is changed by a procedure in the program. (This use of the VALUE clause is permitted in the Working-Storage section, but not in the File section.)


A different use of the VALUE clause is found in level 88 (conditionname) entries. There the clause specifies the data value which the condition name will represent. (This use of the VALUE clause is permitted in both the File section and the Working-Storage section.)


OCCURS clause. An OCCURS clause indicates that an item is repeated with no change in its usage or picture. It specifies the number of times the item is repeated, for instance, OCCURS 100 TIMES, or just, OCCURS 100. OCCURS clauses are used to define groups of identical items that will store related information, such as data tables.

(Continued on next page)

REDEFINES clause. A REDEFINES clause signifies that this entry gives another name and description to an item that was just previously described. The word REDEFINES is written right after the data name, and is followed by the name of the item that is being redefined.

When an item is redefined, a new area of storage is not set aside for it; instead, the same area of storage may be called by this name as well as by the previous name. Redefining is done when different names or descriptions are desired for the same item, for instance, when the item may hold different kinds of information, or may be processed in different ways, under different conditions.

"LEDGER" will occupy the same storage area as "ACCOUNT"

Procedure Division

The Procedure division generally consists of a series of paragraphs, which may optionally be grouped into sections. Each paragraph has a programmer-supplied name, and may contain a varying number of entries.

Sample Procedure division.


Sample Procedure division (continued).


A procedure is a paragraph or section in the Procedure division. Procedures contain two kinds of entries: header entries and sentences. Hence, any entry (in a procedure) that is not a header is a sentence. A sentence, in turn, is composed of one or more statements that specify actions. Most statements begin with a verb that indicates the action to be taken (for instance, MOVE, ADD, READ, GO, or STOP); some statements begin with the word IF, which calls for the testing of a condition.

- The shortest procedure is a paragraph consisting of a header entry and one sentence.

IINITIIALIZE. IMOVE ZZRIOS TO CONTROL-TOTALI.III

A longer procedure may contain several sentences. The paragraph header may optionally appear on a line by itself.


- A sentence may contain only one statement (as in the above examples), or more than one statement (as shown below).

- Every procedure has a programmer-supplied name, which is given in the header entry. This name is used to refer to the procedure, for instance, when specifying a branch to the procedure.

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 lowvolume data, such as information needed to initialize program switches, balance totals, or serial numbers. Not used to read files of data (see READ).


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 | REGUL | LAR- | PAY |  | ETIM |  | E-PA |  | Y, | BONUS |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 1 | GIVIN | NG T | OTAL | -EA | NED. |  |  |  |  |  |  |  |  |  |

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).

| ICLOSE OLID-MA\|STER|-FILIE. N|EW-M|ASTE|R-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.


DISPLAY. Output. Puts data out on the system output device (which might be a disk file, tape drive, etc.), for printing later. Can also punch out data, or type it on the console typewriter. Data displayed cannot exceed 120 characters when put on the system output device, but cannot exceed 72 characters if punched or typed. Used for low-volume output, such as exception records or messages to the operator. Not generally used to write files of data (see WRITE).


DIVIDE. Arithmetic. Divides one number into another. Puts the quotient into the data item named after the word INTO, unless there is a GIVING clause. If GIVING is specified, the quotient 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 division. (Numbers can also be divided by using the verb COMPUTE.)


GO TO. Sequence control. Causes a branch to a procedure in the program. The normal flow of control is resumed at the beginning of that procedure. (Contrast with PERFORM, which causes a return branch as well.)

| GO TO GET-RECORD. | II |  |  |  |  |  |  |  | $\mid$ |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- | :--- |

IF. Sequence control. Causes alternate paths of action to be taken, depending on whether it finds the description of a data condition to be true or false when it evaluates the data. The word IF is followed by the description of the condition, followed by the actions to be taken if the description of the condition is true. Then, optionally, may come to the word ELSE or OTHERWISE, followed by the actions to be taken if the description of the condition is false.


MOVE. Data movement. Moves data from one area of main storage to another. Converts the data (for instance, from decimal to binary) if required to fit the description of the receiving item. Edits the data (inserts, deletes, or replaces characters) if the picture of the receiving item calls for it.


MULTIPLY. Arithmetic. Multiplies one number by another. Puts the product into the data item named after the word BY, unless there is a GIVING clause. If GIVING is specified, the product 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 multiplied. (Numbers can also be multiplied by using the verb COMPUTE.)


NOTE. Program comments. Allows programmers to write explanatory sentences which will be printed in the program listing, but which serve no other purpose during compilation. A note can be any length and can contain any words or characters. If NOTE is the first word of a paragraph, all of the sentences in the paragraph are treated as notes.


OPEN. Input-output. Makes one or more data files ready for reading or writing. Input files are named after the word INPUT; output files after OUTPUT. Such actions as checking and creating beginning-of-file labels are done automatically when a file is opened. Opening does not make input records available for processing; a READ statement is required for that. A file must be open before a READ or WRITE statement can act on it. (See READ, WRITE, CLOSE.)

(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.)


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).


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.
$\square$

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.


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).


Types of conditions that can be tested in IF statements.
Relation test. Consists of two "operands" with relational words or symbols written between them. Compares the values of the operands to see if their relation is the same as specified.

Relations that can be specified: EQUAL TO [=], NOT EQUAL TO [NOT =], GREATER [ > ] , NOT GREATER [NOT >], LESS [ <], NOT LESS [NOT <].

The operands may be data items, literals, and arithmetic expressions. A data item may be compared with another data item, with a literal, or with an arithmetic expression. An arithmetic expression may also be compared with a literal, or with another arithmetic expression. Two literals may not be compared.

When an arithmetic expression is written in a conditional statement, the value of the expression is calculated first, before the comparison is made.


Sign test. Consists of the name of a data item, or an axithmetic expression, followed by the specification of a sign condition. Determines whether the number, or the value of the arithmetic expression, matches the specified condition.

Sign conditions that can be specified: POSTTIVE, NOT POSITIVE, NEGATIVE, NOT NEGATIVE, ZERO, NOT ZERO. The value zero is considered to be neither positive nor negative. (For a zero item in storage, the sign, if any, is ignored.)


Note: A sign test is another way of stating a relation test for numbers. since "positive" means greater than zero, "negative" means less than zero, and "zero" is, of course, equal to the Iiteral "0".

Condition-name test. Consists of a name called a "condition name", with no other words or symbols. Tests whether a data item has a specific value; however, neither the name of the data item nor the value are written in the IF statement.


The condition name represents the condition that exists when the data item has a specific value. It is defined in the Data division, where the condition name and the value may be found in a level-88 entry, following the description of the data item which is to be tested.

Here is how the condition name used in the above example might be defined:


Interpretation of chis example: A job-classification code is one of the data items in the records that are to be processed. The name COMPUTER-PROGRAMMER represents the code value 72. Therefore, the IF sentence above means, "If the record contains job code 72 , perform the salary-increase procedure."

Note: A condition-name test is another way of testing whether a data item is equal to a literal.

Overflow test. A special type of condition-name test. Determines whether the forms in a printer are at the end of a page. (The end of a page is recognized when the printer senses a hole punched in channel 12 of the carriage control tape.)

|  | 115 | P | AGE-I | IS-F | ULL, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | $!$ |  | WRITE | E OVE | ERDU | E-ACC | COUN |  |  |  |  | A |  | TER |  |  |  |  |  |  |
|  | 1 |  | SKIPP | PING | -TO- | NEXT | -PAG |  | ; |  |  |  |  |  |  |  |  |  |  |  |
|  | 1 |  | OTHER | RWIS | E, W | R ITE | OVE |  | D | E | A | C |  | UNT |  |  | IN | E |  |  |
|  | 1 |  | AFTE\| | $R \quad S P A$ | ACIN | G-2-L | L INE | S | S |  |  |  |  |  |  |  |  |  |  |  |

The form-overflow condition name is defined in an APPLY entry in the Environment division. Here is how the name used in the above example might be defined:


Class test. Consists of the name of a data item followed by the specification of a class. Examines the item to see if the data it contains belongs to the specified class. Classes that can be specified: ALPHABETIC, NOT ALPHABETIC, NUMERIC, NOT NUMERIC.


An item is

- alphabetic if it consists only of letters, and possibly spaces. not alphabetic if it contains any digits or special characters. numeric if it consists only of digits, and possibly an operational sign.
- not numeric if it contains any spaces, letters, or special characters.

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.

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


Example.


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).


Example.

| 1 |  |  | IIF R | RECOVE | ERY | 15 | NO | OT G |  | EAT | E | $T$ |  | HAN |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| 2 |  |  | ISCRA | AP-POI | INT, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  | 1 | MOVE | ITE | $\mathrm{M}-\mathrm{N}$ | NUM | MBER |  | TO |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  | , |  | SCRA | P-I |  | EM, |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  | $1]$ | ADD | RECO | VER |  | TO |  | CRA | P |  |  |  |  |  |  |  |  |  |
| 6 |  |  | OTHE | RW I SE | E, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  | , | WRITE | E co | NSE | R | VATI |  | $\mathrm{N}-\mathrm{R}$ |  | COR | R | . |  |  |  |  |  |  |
| 8 |  |  | ADD | 1 TO | TRA | NSA | C. $T$ | TION |  | COU | NT | T. |  |  |  |  |  |  |  |  |

RECOVERY is compared with SCRAP-POINT. If RECOVERY is equal to or less than SCRAP-POINT, the conditional description is true, so the statements on lines 3-5 are acted on, and then control jumps to the next sentence (line 8). If RECOVERY is greater than SCRAP-POINT, the conditional description is false, in which case control jumps to lines 6-7, and then passes to the next sentence (line 8).

## $1$

Case Study

CASE STUDY. Customer billing program.

Input. Billing record.

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


| ACCOUNT HISTORY |  |  | LAST YEAR |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| YEAR | YEAR |  |  |  |  |  |
| OPENED | LAST <br> ACTIVE | HIGHEST <br> BALANCE | MONTHS <br> ACTIVE | MONTHS <br> OVER <br> 90 | TOTAL <br> PURCHASES | TOTAL <br> RETURNS |



| billing DATE | NUMBEROFTRANSACTIONS | CURRENT BALANCE | PURCHASES |  | PAYMENTS |  | CREDITS |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  |  | NUMBER | AMOUNT | NUMBER | AmOUNT | NUMBER | AMOUNT |


|  |  | COLLECTION HISTORY |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| RETURNS |  | OVERDUE BALANCES |  |  |  | LAST PAYMENT |  | $\begin{gathered} \text { DUNNING } \\ \text { CODE } \end{gathered}$ |
| NUMBER | AMOUNT | $\begin{aligned} & 30 \\ & \text { DAY } \end{aligned}$ | $\begin{aligned} & 60 \\ & \text { DAY } \end{aligned}$ | $\begin{aligned} & 90 \\ & \text { DAY } \end{aligned}$ | $\begin{aligned} & 120 \\ & \text { DAY } \end{aligned}$ | DATE | AMOUNT |  |

## Output. Monthly statement.




Identification division.

| IDENITIFIC | CATI | ON D | 1 V IS | ION. |  |  |  | 1 |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| I |  |  |  |  |  |  |  |  |  |  |  | 1 |  |  |
| PROG,RAM-I | ID. |  |  |  |  |  |  |  | 1 |  |  |  |  |  |
| ' BILL | LING | '. |  |  |  |  |  |  |  |  | , | 1 |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  | , |  |  |  |
| AUTHiOR. | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| IPAUL | MEL | ICHAR | R. |  |  |  |  |  |  |  | 1 |  | - |  |
| ! |  |  |  |  |  |  |  |  |  |  |  |  | + |  |
| DATE1-WRI | TTEN |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 'SEPT\|E | EMBE | R, 196 | 65. |  |  |  |  |  |  |  |  |  |  |  |
| ! |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| REMAIRKS. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Istat | EMEN | TS AR | RE N | OTS | ENT | TO 1 | NACT | 1 VE | ACcou | UNTS |  | OTHER | $R$ PR | OGRAM |
| IHAND | LES | ACCOU | UNTS | WIT | H NE | GATI | VE IC | CRED | (T) | B ALA | NCES |  |  |  |
| IFOUR | LIN | ES AR | RE P | RINT | ED O | N EA | CH S | TATE | MENT | , U'N | DER | CARR | IAGE | TAPE |
| 'CONT | ROL. | LIN | E 1 | IS P | R I NT | ED A | FTER | A S | K I P | TO C | HANN | EL 1 | , LI | NE 2 |
| IAFTER | $R$ A | SKIP\| | TO | CHAN | NEL | 2. L | I NES | 3 A | ND 4 | ARE | SIN | GLE- | SPAC | ED. |

## CASE STUDY (continued)

Environment division.

| ENV IRRONM | AENT | Divis | SIION |  | 1 |  |  |  |  | 1 |  | 1 |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
| ! ! | - 1 |  |  |  | 1 |  |  |  |  |  | - | 11 |  |  |  |
| CONFI, GUR | ATIO | N SEC | CTIO | N. |  |  |  |  | 1 | , | . | $\square$ |  |  |  |
| - | 11 |  |  |  |  |  |  |  |  | 7 | 1 | 1 |  |  |  |
| SOURICE-C | OMPU | TER. |  |  |  |  | 1 |  |  | - | $T$ | $\square$ |  |  |  |
| IIBM- | -360 | 650. |  |  |  |  |  |  | [1] | 1 | - | 1 |  |  |  |
| 1 |  |  |  |  |  |  |  |  | $\square$ | T | 1 | 1 |  |  |  |
| OBJEICT-C | OMPU | TER. |  |  |  |  |  |  |  | 1 | 1 | 1 |  |  |  |
| IIBM- | -360 | F30. |  |  |  |  |  |  |  | T | , | $\square$ |  |  |  |
| ! | - |  |  |  | , |  |  |  |  | 1. | , | - |  |  |  |
| InPUIT-OU | TTPUT | SEC | TION |  |  |  |  |  |  | 1 |  | $\square$ |  |  |  |
| 1 | - |  |  | 1 | , |  | 1 |  |  | - | $\square$ |  |  |  |  |
| FILEI-CON | NTROL |  |  |  |  |  |  |  |  | 11 | 1 | 1. |  |  |  |
| ISELE | CT B | ILLIN | NG-F | ILE, | As | IGN | To ' ${ }^{\text {c }}$ | B I LLF | FILE | - UT | ILIT | Y. |  |  |  |
| ISELE | CT C | UStom | MER- | BILL- | -FIL | E, A | SSIGN | N TO | 'cus | STBI | LL' | 1 |  |  |  |
| 1 | UNIT | -REC | ORD | 1403 | UNI | T. |  |  |  |  |  |  |  |  |  |

## Data division.

|  |  |  |  |  |  |  |  |  |  | T |  | $\square$ |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 1 |  |  |  |  |  |  |  |  | , | 1 |  |  |  | 11 |  |  |
| FILES SEC |  | CTIO | N. |  |  |  |  |  | 1 | $\square$ | 1 |  |  |  |  |  |  |
|  | ! |  |  |  |  |  |  |  | 1: | 1 | 1 |  |  |  | ! |  |  |
| FD | B 12 | LIN |  | F ILE |  |  |  |  | 1 |  |  |  |  |  |  | , |  |
|  | 'BLO | CK | CON | NTAI | NS 5 | RECO | ORDS |  |  |  |  |  |  |  |  |  |  |
|  | 'LAB | EL | REC | CORDS | S ARE | E STA | Anda | RD |  |  |  |  |  |  |  |  |  |
|  | IDAT |  | CO | ORD | IS BI | ILLIN | NG-R | ECOR | D. |  | $!$ |  |  |  |  | 1 |  |
|  | ! |  |  |  |  | - |  |  | T |  | 11 | - |  |  | . |  |  |
| 0.1 | : $\mathrm{B}_{1}$ | LIN |  | RECO | RD. |  |  |  |  |  | 1 | $\square$ |  |  |  |  |  |
|  | 102 | AC | COU | UNT- | - DENT | TIFIC | CAT | ON. | , |  | - : | 1 |  |  |  |  |  |
|  | ! | 03 |  | TYPE | -OF-A | Accol | UNT |  |  | PICTU | URE | $x$. |  |  | : |  |  |
|  | $!$ | 03 |  | ACCO | UNT-N | NUMB | ER. |  | 1 |  | $\square$ | 1/ |  |  |  | ! |  |
| T | 1 |  |  | 04 | STORE | E-NUM | MBER |  | 1 | PICT | URE | 999. |  |  |  |  |  |
| 1 | 1 |  |  | 04 | F I LE | - NUM | BER |  | ! | PICT | URE | 9(4). |  |  |  |  |  |
|  | 11 | 03 |  | B ILL | ING-C | CYCLE | E |  |  | PICTU | URE | 99. |  |  | 1 | , |  |
|  | 1 | 03 |  | CUST | OMER | - NAM | E |  |  | PICTU | URE | $x(22)$ | $)$ |  | - | , |  |
|  | $!$ | 03 |  | STRE | ET-AD | DDRE | SS |  |  | P ICTU | URE | $x(22)$ | 1. |  | - | 1 |  |
|  | 1 | 03 |  | C I TY | -stat | TE |  |  |  | PICTU | URE | $\mathrm{x}(22)$ |  |  | 1 |  |  |
|  | 102 | CR |  | IT-S | TATUS | S. |  |  |  |  |  |  |  |  |  |  |  |
|  | 1 | 013 |  | RATI | NG-CO | ODE |  |  |  | PICTI | URE | $x$. |  |  |  |  |  |
|  | 1 | 03 |  | PURC | HASE | -LIM | 1 T |  |  | P ICTU | URE | $9(4)$ | . | - |  |  |  |


|  | 102 | Accol | UNT-H | HISTIO | ORY. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 1 | 03 | YEAR | -OPEN | NED |  |  |  |  | PICTU | URE |  | 99. |  |  |  |  |  |  |
|  | I | 03 Y | YEAR | -LAS $T$ | T-AC | TIVE |  |  |  | PICTU | URE |  | 99. |  |  |  |  |  |  |
|  | 1 | 03 | HIGHE | EST-B | BALA | NCE |  |  |  | PICTU | URE |  | ( 4 ) | V99. |  |  |  |  |  |
|  | 102 | LAST- | -YEAR | R. ${ }^{\text {a }}$ | - 1 | , |  |  |  | T11 |  |  |  |  |  |  |  |  |  |
|  | 1 | 03 | MONTH | HS-AC | CTIV | E |  |  |  | PICTU | URE |  | 99. |  |  |  |  |  |  |
|  | ! | 03 | MONTH | HS-OV | VER- | 90 |  |  |  | PICTU | URE |  | 99. |  |  |  |  |  |  |
|  | ! | 03 | TOTAL | L-PUR | RCHA | SES |  |  |  | PICTU | URE |  | (5) | V99. |  |  |  |  |  |
|  | 1 | 03 | TOTAL | L-RE | TURN | S |  |  |  | PICTU | URE |  | (5) | V99. |  |  |  |  |  |
|  | 102 | THIS | -YEAR | R-TO | -DAT | E. |  |  |  | - | , |  |  | 111 |  |  |  |  |  |
|  | 1 | 03 | MONTH | HS-A | CTIV | E |  |  |  | PICTL | URE |  | 99. |  |  |  |  |  |  |
|  | ! | 03 | MONTH | HS-ON | VER- | 90 | T | 1 |  | PICTU | URE |  | 99. |  |  |  |  |  |  |
|  |  | 03 | TOTAL | L-PUR | RCHA | SES | 1 | T |  | PICTL | URE |  | (5i) | V99. |  |  |  |  | 1 |
|  | 1 | 03 | TOTAL | L-RE | TURN | S 1 |  |  |  | PICTU | URE |  | (5) | V99. |  |  |  |  |  |
|  | 102 | LAST |  |  | - |  | 1.1 | I |  |  |  |  |  | V 1 |  |  |  |  |  |
|  | 12 | 03 | NUMB | ER-OF | F-TR | ANSA | CTIO | NS |  | PICTU | URE |  | 99. |  |  |  |  |  |  |
|  | $!$ | 03 | BALAN | NCE-F | FORW | ARD |  |  |  | PICTU | URE |  | (4) | V99. |  |  |  |  |  |
|  | 102 | THIS | -MONT | TH. |  |  | 1 |  |  |  | 11 |  |  |  |  |  |  |  |  |
|  | 1 | 03 | BILLI | ING-D | d ATE |  |  |  |  | PICTU | URE |  | 9 (6) |  |  |  |  |  |  |
|  | 1 | 03 | NUMBE | ER-OF | F-TR | AINSA | CTIO | NS |  | PICTU | URE |  | 99. |  |  |  |  |  | 1 |
|  | 1 | 03 | CURRE | ENTI-I | BALA | NCE |  | 1 |  | PICTU | URE |  | 9 (4) | V99. |  |  |  |  |  |

(Continued on next page)

Data division (continued).


Data division (continued).

|  | cus | TonER | -812 | L-FI |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 号 ${ }^{\text {c }}$ | aplw | 6 mo | PE F |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | $L^{1} A B$ | EL RE | CORD | S AR | E | om 1 | tte | D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | DAT | REC | ORDS | ARE | B | 1 | -LI | ne- |  | $B_{1}$ | LL | in | NE | 2 |  | B |  |  | INE | E-3 |  |  |  |
|  | : | BILL | -LIN | E-4. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | ! |  | T1 | T 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 61 | - 12 | LIN | E-1. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | ${ }^{0} 2$ | F1L | ER |  |  |  |  |  |  |  | Pict | tur | RE |  | (4) |  |  |  |  |  |  |  |  |
|  | -2 | OLD- | BALA | Ance |  |  |  |  |  |  | PIC | TUR |  |  | \$, | \$ | S. |  |  |  |  |  |  |
|  | -2 | FILL |  |  |  |  |  |  |  |  | PIC | TUR |  |  | (7) |  |  |  |  |  |  |  |  |
|  | ¢2 | purch | HASE |  |  |  |  |  |  |  | PIC | TUR |  |  | \&, | , | \$. | \$ ${ }^{\text {d }}$ |  |  |  |  |  |
|  | -02 | FL4 | ER |  |  |  |  |  |  |  | PIC | TUR |  |  | (4) |  |  |  |  |  |  |  |  |
|  | b2. | Park | Ents |  |  |  |  |  |  |  | Pic | TUR |  |  | \$, 5 | s 5 | s. | s\% |  |  |  |  |  |
|  | -12 | FILL | ER |  |  |  |  |  |  |  | Pic | TUR |  |  | (4) |  |  |  |  |  |  |  |  |
|  | 02 | Cred | , its |  |  |  |  |  |  |  | pic | TUR |  |  | \$. ${ }^{\text {S }}$ | , ${ }^{\text {d }}$ | \$. | s\$ |  |  |  |  |  |
|  | 102 | FILL | ER |  |  |  |  |  |  |  | pIc | Tur |  |  | (7) |  |  |  |  |  |  |  |  |
|  | 102 | accou | UNT- | - Numb | ER |  |  |  |  |  | PIC | TUR |  |  | 998 | 89 | 99 |  |  |  |  |  |  |
|  | ${ }^{1}$ | F1L | ER |  |  |  |  |  |  |  | PIC | TUR | RE | x | (5) |  |  |  |  |  |  |  |  |
|  | -2 | Bill | ING- | date |  |  |  |  |  |  | pic | TUR |  |  | 9 BO | 9 | B9 |  |  |  |  |  |  |
|  | 02 | FIL닌 | ER |  |  |  |  |  |  |  | P, 10 | tur |  |  | (50) |  |  |  |  |  |  |  |  |



```
Procedure division.
```



|  | WRITE | E BIL | LL-L | -1NE |  | 1 AF | TER |  | SKIP |  | TO- |  | ARIR |  |  | CHA | ANNEL | -1 |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | 1 |  |  |  |  |  |  |  | - |  |  |  | 1 | ! |  |  |  |  |  |  |  |  |
|  | INE1-2-PR | ROCED | DURE | . |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | M MOVE | SPAC | CES | TO | B 12 | - LL-L | LIN | NE- | -2. |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | MOVE | CURR | RENT | T-bal |  | ANCE | TO | 0 A | AMOU |  | T-D | DUE | E. |  |  |  |  |  |  |  |  |  |
|  | MOVE | Cus | TOME | R-NA |  | ME TO | O $N$ | NAM | ME. |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | MRITE | E B IL | LL-L | INE |  | 2 AF | TER | R S | SIKIP | P- | TO- | C C A | ARR | IAG |  | CHA | NNEL | -2. |  |  |  |  |
|  | ! |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | INE:3-PR | ROCED | DURE | E. |  |  | T 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | MOVE | SPAC | CES | TO | B IL | 1 L L - | LIN | NE- | -3. |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | MOVE | STRE | EET- | ADDP |  | SS | TO | AD | D DRE | ES | 5. |  |  |  |  |  |  |  |  |  |  |  |
|  | - MRITE | E BIL | LL-L | INE | -3 | 3 AF | TER | R S | SINO | GL | E-S | spa | ACE |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LINE ${ }^{1}$ |  | ROCED | DURE |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | MOVE | c ITY | TY-st | tate |  | TOC | C ITY | Y. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | II MMRITE | E B IL | LLL-L | INE |  | 4 AF | TER | R | SUNC | GL | E-S |  | ACE |  |  |  |  |  |  |  |  |  |
|  | 11] |  |  |  |  |  |  |  |  |  |  |  | T |  |  |  |  |  |  |  |  |  |
| GET-INEXT- |  | -RECO | ORD. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | -GOTO | - REA | AD-A | AND-C |  | HECK | -RE | ECO | ORD |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 11 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| END-OF-RUN. |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | íclose | E BIL | LLIN | NG-F |  | LE, |  | STO | OMER | R- | -B, 1 |  | - FII | LE. |  |  |  |  |  |  |  |  |
|  | TSTOP | RUN |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

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? $\qquad$
- Did this course, in general, meet your needs?

Yes
No

- Did an IBM employee serve as your advisor?
- Did you receive a completion certificate?

- Did you find the material:

Easy to read and understand?

Organized for convenient use?


Well illustrated?

- Did you feel that any particular topics should be $\square$
$\square$ added or emphasized?
- Did you feel that any particular topics should not

$\square$ have been included?
- If you found any technical errors, please list them below, giving form number, page, and frame number.


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

POSTAGE WILL BE PAID BY....

IBMCorporation
1701 North St.
Endicott, N.Y. 13760

Attention: DP Education Development, Dept. 617

Fold

## 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? $\qquad$
- Did this course, in general, meet your needs?

Yes No

- Did an IBM employee serve as your advisor?
- Did you receive a completion certificate?

- Did you find the material:

Easy to read and understand?

Organized for convenient use? $\square$

Well illustrated? $\square$

- Did you feel that any particular topics should be added or emphasized?
- Did you feel that any particular topics should not $\square$ $\square$ have been included?
- If you found any technical errors, please list them below, giving form number, page, and frame number.

