IBM

SYSTEM/360 COBOL
COBOL Program Fundamentals
Reference Handbook

Programmed Instruction Course

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 C28-6516-3).

ACKNOWLEDGEMENT

The following information is reprinted from COBOL-61 EXTENDED, published by the conference on Data Systems Languages (CODASYL), and printed by the U. S. Government Printing Office.

This publication is based on the COBOL System developed in 1959 by a committee composed of government users and computer manufacturers. The organizations participating in the original development were:

Air Materiel Command,
United States Air Force
Bureau of Standards,
Department of Commerce
David Taylor Model Basin,
Bureau of Ships, U.S. Navy
Electronic Data Processing Division, Minneapolis-Honeywell Regulator Company
Burroughs Corporation
International Business Machines Corporation
Radio Corporation of America
Sylvania Electric Products, Inc.
Univac Division of Sperry-Rand Corporation

In addition to the organizations listed above, the following organizations participated in the work of the Maintenance Group:

Allstate Insurance Company
Bendix Corporation, Computer Division
Control Data Corporation
DuPont Company
General Electric Company
General Motors Corporation
Lockheed Aircraft Corporation
National Cash Register Company
Philco Corporation
Royal McBee Corporation
Standard Oil Company (N.J.)
United States Steel Corporation

This manual is the result of contributions made by all of the above-mentioned organizations. no warranty, express or implied, is made by any contributor or by the 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.

It is reasonable to assume that a number of improvements and additions will be made to COBOL. Every effort will be made to insure that the improvements and corrections will be made in an orderly fashion, with due recognition of existing users' investments in programming. However, this protection can be positively assured only by individual implementors.

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

TABLE OF CONTENTS

Introduction

Language Elements

Program Structure and Contents

Identification Division

Environment Division

Data Division

Procedure Division

Case Study

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

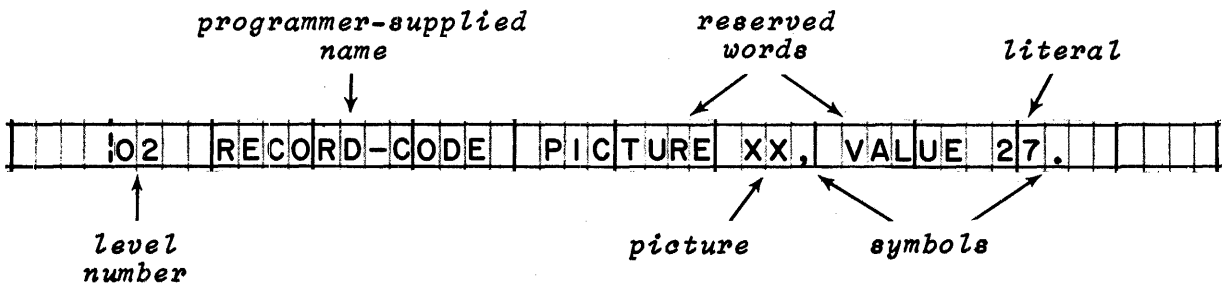
The COBOL language is made up of these elements:

- 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, ZERO 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
ACCESS
ACTUAL
ADD
ADVANCING
AFTER
ALL
ALPHABETIC
ALTER
ALTERNATE
AND
APPLY
ARE
AREA
AREAS
ASCENDING
ASSIGN
AT
AUTHOR

BEFORE
BEGINNING
BLANK
BLOCK
BY

CALL
CF
CH
CHANGED
CHARACTERS
CHECKING
CLOCK-UNITS
CLOSE
COBOL
CODE
COLUMN
COMMA
COMPUTATIONAL
COMPUTATIONAL-1
COMPUTATIONAL-2
COMPUTATIONAL-3
COMPUTE
CONFIGURATION
CONSOLE
CONTAINS
CONTROL
CONTROLS
COPY
CORRESPONDING
CREATING
CYCLES

DATA
DATE-COMPILED
DATE-WRITTEN
DE
DECIMAL-POINT
DECLARATIVES
DEPENDING
DESCENDING
DETAIL
DIRECT
DIRECT-ACCESS
DISPLAY
DISPLAY-ST
DIVIDE
DIVISION

ELSE
END
ENDING
ENTER
ENTRY
ENVIRONMENT
EQUAL
ERROR
EVERY
EXAMINE
EXHIBIT
EXIT

FD
FILE
FILES
FILE-CONTROL
FILE-LIMIT
FILLER
FINAL
FIRST
FOOTING
FOR
FORM-OVERFLOW
FROM

GENERATE
GIVING
GO
GREATER
GROUP

HEADING
HIGH-VALUE
HIGH-VALUES
HOLD

IBM-360
IDENTIFICATION
IF
IN
INCLUDE
INDEXED
INDICATE
INITIATE
INPUT
INPUT-OUTPUT
INSTALLATION
INTO
INVALID
I-O
I-O-CONTROL
IS

JUSTIFIED

KEY

LABEL
LABELS
LAST
LEADING
LESS
LINE
LINE-COUNTER
LINES
LINKAGE
LOCK
LOW-VALUE
LOW-VALUES

MODE
MORE-LABELS
MOVE
MULTIPLY

NAMED
NEGATIVE
NEXT
NO
NOT
NOTE
NUMERIC

OBJECT-COMPUTER
OCCURS
OF
OH
OMITTED
ON
OPEN
OR
ORGANIZATION
OTHERWISE
OUTPUT
OV
OVERFLOW
PAGE
PAGE-COUNTER
PERFORM
PF
PH
PICTURE
PLUS
POSITIVE
PRINT-SWITCH
PROCEDURE
PROCEED
PROCESS
PROCESSING
PROGRAM-ID
PROTECTION

QUOTE
QUOTES

RANDOM
RD
READ
READY
RECORD
RECORDING
RECORDS
REDEFINES
REEL
RELATIVE
RELEASE
REMARKS
REPLACING
REPORT
REPORTING
REPORTS
RERUN
RESERVE
RESET
RESTRICTED
RETURN
REVERSED
REWIND
REWRITE
RF
RH
RIGHT
ROUNDED
RUN

SA
SAME
SD
SEARCH
SECTION
SECURITY
SELECT
SENTENCE
SEQUENTIAL
SIZE
SORT
SOURCE
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
STANDARD
STOP
SUBTRACT
SUM
SYMBOLIC
SYSIN
SYSOUT
SYSPUNCH

TALLY
TALLYING
TERMINATE
THAN
THEN
THRU
TIMES
TO
TRACE
TRACK-AREA
TRACKS
TRANSFORM
TRY
TYPE

UNIT
UNIT-RECORD
UNITS
UNTIL
UPON
USAGE
USE
USING
UTILITY

VALUE
VARYING

WHEN
WITH
WORKING-STORAGE
WRITE
WRITE-ONLY

ZERO
ZEROES
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.

```
ADD REGULAR-WAGES, OVERTIME, GIVING GROSS-EARNINGS.
```

programmer-supplied names

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

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.

```
    MULTIPLY SALES BY .0525 GIVING BONUS.
                       ^
                       |
                  numeric literal
```

Rules governing non-numeric 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.

```
77  MESSAGE-4    PICTURE A(12)
    VALUE 'OUT OF STOCK'.
          ^
          |
     non-numeric literal
```

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.

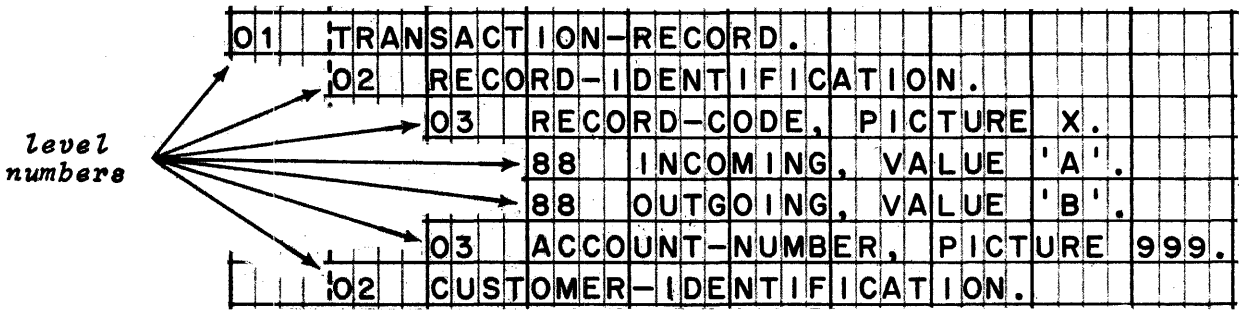
Level numbers 01 through 49 are used to designate the levels of data items that form records. Level number 01 is always assigned to the record as a whole, while 02 through 49 are assigned to items that are parts of records.

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

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.

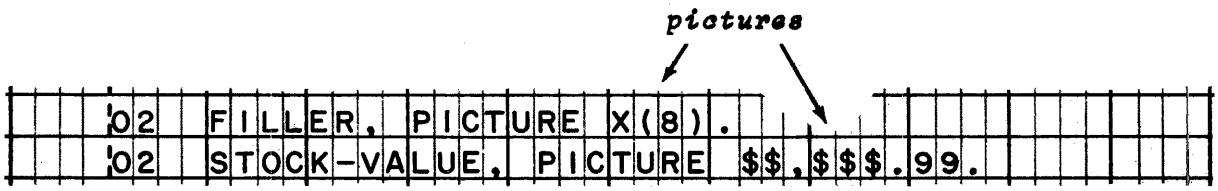
Each picture is a string of characters, and may be from 1 to 30 characters long. Pictures are composed of one or more of the characters listed below. (Each character or pair of characters has a symbolic meaning; the meanings of some of these characters are explained later in this book.)

A B CR DB E K P S V X Z 0 9 + - , . $ * ( )

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

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.

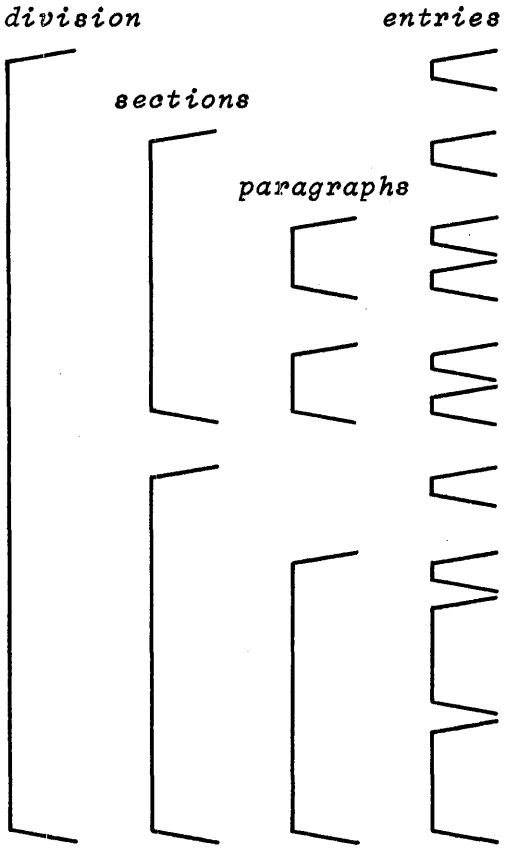
Entries. An entry can be defined as a series of two or more language elements, the last of which is a period. (However, the programmer cannot arbitrarily string together a bunch of elements, and call them an entry. The sequence of elements in each entry is dictated by precise format rules.)

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.




```
ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER.
    IBM-360 G40.

OBJECT-COMPUTER.
    IBM-360 E30.

INPUT-OUTPUT SECTION.

FILE-CONTROL.
    SELECT OLD-MASTER-FILE
        ASSIGN TO 'INV0001'
        UTILITY 2400 UNITS.
    SELECT NEW-MASTER-FILE
        ASSIGN TO 'INV0002'
        UTILITY 2400 UNITS.
```

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

IDENTIFICATION DIVISION ENTRIES

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 (external name. 8 characters - a letter followed by up to 7 seven letters and/or digits.)

The division may also contain up to six additional paragraphs. Fixed names are provided for these paragraphs(AUTHOR, INSTALLATION, DATE-WRITTEN, 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.

```
IDENTIFICATION DIVISION.

PROGRAM-ID.
    'PURCHASE'.

AUTHOR.
    PAUL MELICHAR.

INSTALLATION.
    PURCHASING RECORDS DEPARTMENT.

DATE-WRITTEN.
    ORIGINAL PROGRAM WRITTEN JULY,1962.
    THIS IS REVISION 4, WRITTEN SEPTEMBER,1965.

REMARKS.
    OUTPUT OF THIS PROGRAM IS A REPORT OF
    ALL PURCHASES FOR THE PREVIOUS MONTH.
    THE REPORT IS REQUIRED BY THE 5TH DAY
    OF EACH MONTH, AND IS DISTRIBUTED TO
    ALL PURCHASING REPRESENTATIVES. INPUT
    IS THE PURCHASING RECORDS CARD FILE,
    SORTED BY PURCHASE DATE WITHIN COMMODITY.
```

Environment Division

ENVIRONMENT DIVISION ENTRIES

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.

```
ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER.
    IBM-360 F30.

OBJECT-COMPUTER.
    IBM-360 F30.

INPUT-OUTPUT SECTION.

FILE-CONTROL.
    SELECT PURCHASING-FILE
        ASSIGN TO 'PURMAS' UNIT-RECORD 2540 UNIT.
    SELECT PURCHASE-REPORT-FILE
        ASSIGN TO 'REPO27' UNIT-RECORD 1403 UNIT.

I-O-CONTROL.
    APPLY LAST-LINE TO FORM-OVERFLOW
        ON PURCHASE-REPORT-FILE.
```

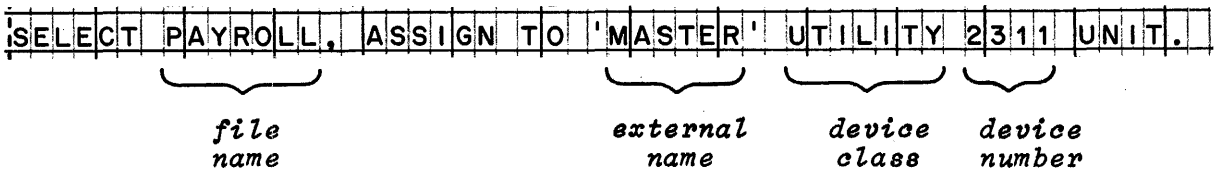
(Continued on next page)

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.

Data Division

DATA DIVISION ENTRIES

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.

```
DATA DIVISION.

FILE SECTION.

FD  PURCHASING-FILE
    RECORDING MODE IS F
    LABEL RECORDS ARE OMITTED
    DATA RECORD IS PURCHASING-RECORD.

01  PURCHASING-RECORD.
    02  COMMODITY.
        03  NUMBER                 PICTURE 9(12).
        03  DESCRIPTION            PICTURE X(30).
    02  PURCHASE.
        03  NUMBER                 PICTURE 9(8).
        03  DATE                   PICTURE 9(6).
    02  UNITS-PURCHASED            PICTURE 9(6).
    02  UNIT-COST                  PICTURE 9(4)V99.
    02  TOTAL-COST                 PICTURE 9(6)V99.
    02  CARD-CODE                  PICTURE X(4).
```

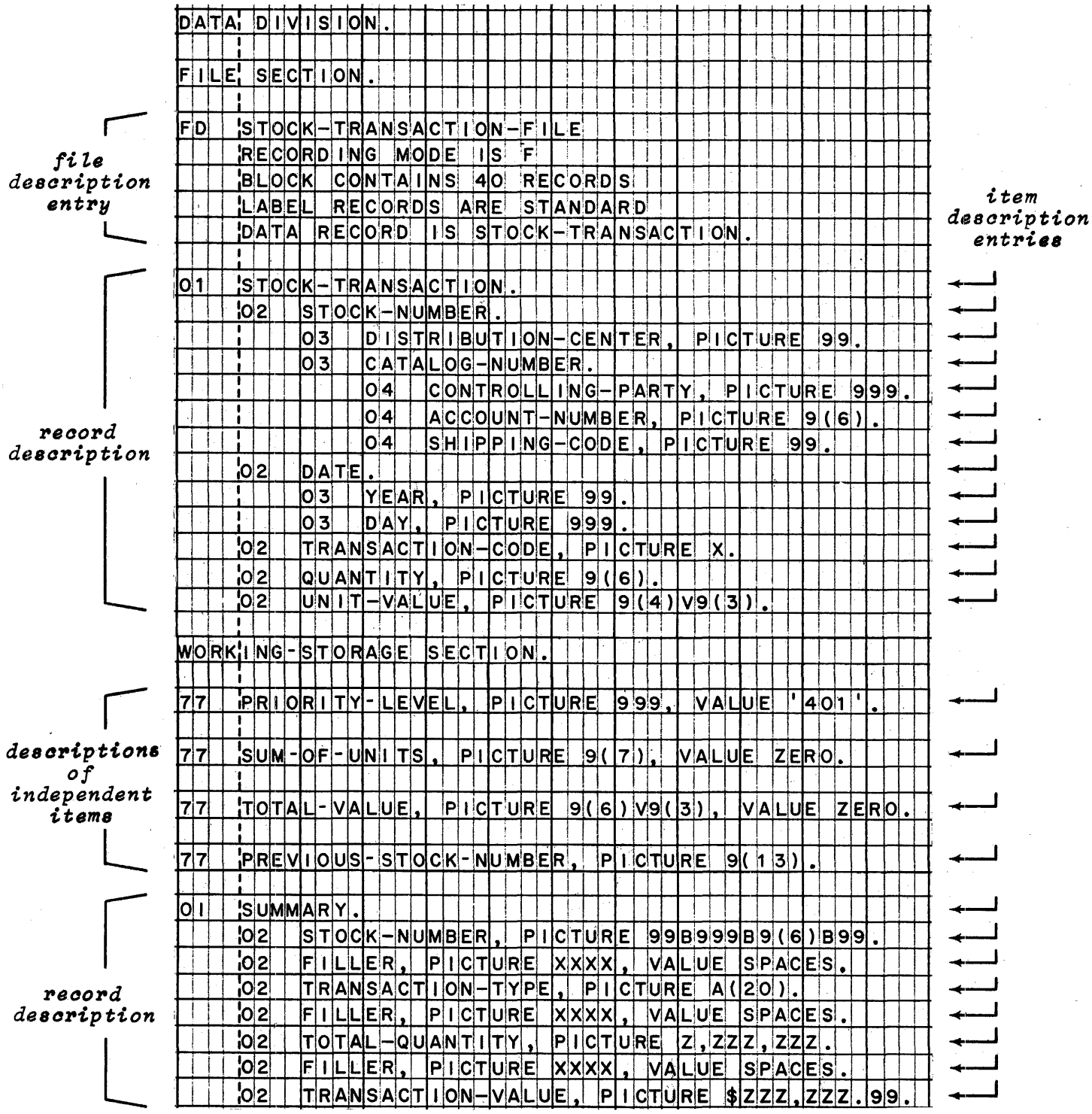
(Continued on next page)

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)

Another sample Data division, with entry types identified.




```
DATA DIVISION.

FILE SECTION.

FD  STOCK-TRANSACTION-FILE
    RECORDING MODE IS F
    BLOCK CONTAINS 40 RECORDS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS STOCK-TRANSACTION.

01  STOCK-TRANSACTION.
    02  STOCK-NUMBER.
        03  DISTRIBUTION-CENTER, PICTURE 99.
        03  CATALOG-NUMBER.
            04  CONTROLLING-PARTY, PICTURE 999.
            04  ACCOUNT-NUMBER, PICTURE 9(6).
            04  SHIPPING-CODE, PICTURE 99.
    02  DATE.
        03  YEAR, PICTURE 99.
        03  DAY, PICTURE 999.
    02  TRANSACTION-CODE, PICTURE X.
    02  QUANTITY, PICTURE 9(6).
    02  UNIT-VALUE, PICTURE 9(4)V9(3).

WORKING-STORAGE SECTION.

77  PRIORITY-LEVEL, PICTURE 999, VALUE '401'.

77  SUM-OF-UNITS, PICTURE 9(7), VALUE ZERO.

77  TOTAL-VALUE, PICTURE 9(6)V9(3), VALUE ZERO.

77  PREVIOUS-STOCK-NUMBER, PICTURE 9(13).

01  SUMMARY.
    02  STOCK-NUMBER, PICTURE 99B999B9(6)B99.
    02  FILLER, PICTURE XXXX, VALUE SPACES.
    02  TRANSACTION-TYPE, PICTURE A(20).
    02  FILLER, PICTURE XXXX, VALUE SPACES.
    02  TOTAL-QUANTITY, PICTURE Z,ZZZ,ZZZ.
    02  FILLER, PICTURE XXXX, VALUE SPACES.
    02  TRANSACTION-VALUE, PICTURE $ZZZ,ZZZ.99.
```

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

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.

```
FD  PRODUCT-USAGE-FILE
    RECORDING MODE IS V
    BLOCK CONTAINS 18 RECORDS
    RECORD CONTAINS 52 TO 306 CHARACTERS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS PRODUCT-USAGE-RECORD.
```

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 FD, which stands for File Description.

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

(Continued on next page)

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)

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 numbers 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

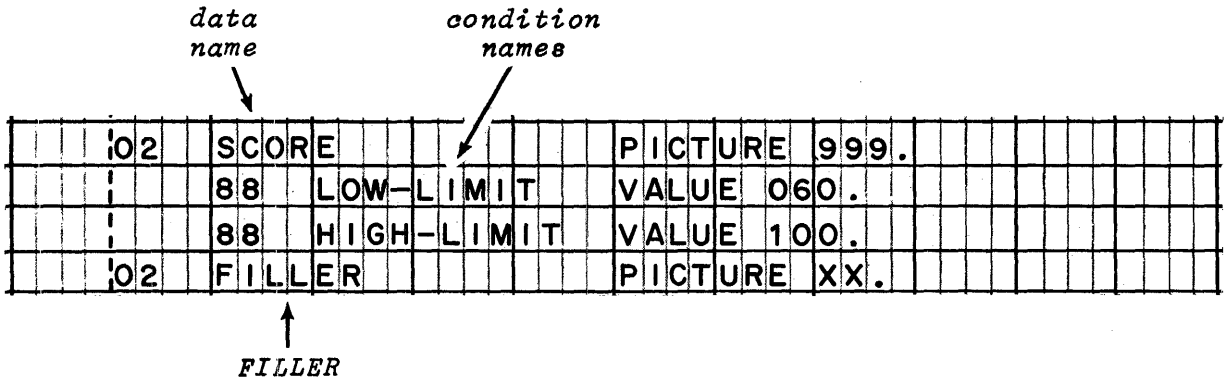
(Continued on next page)

ITEM DESCRIPTION ENTRIES (continued)

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.



(Continued on next page)

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

2 Key factors on how to tell what the usage of a group item or elementary item.
1). The first is that the usage specified for a group item applies to all of the items in the group.
2. The second is that when no usage at all is specified for an elementary item or for a group it is part of, the usage is assumed to be Display.

(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 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 is 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	$ZZ,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 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.)

(Continued on next page)

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.)

```
77  DISCOUNT, USAGE COMPUTATIONAL,
    PICTURE SV99, VALUE .02.
```

VALUE clause

A different use of the VALUE clause is found in level 88 (condition-name) 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.)

```
    03  SUBCRIPTION-BASIS, PICTURE X.
    88  REGULAR, VALUE '1'.
```

VALUE clause

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.

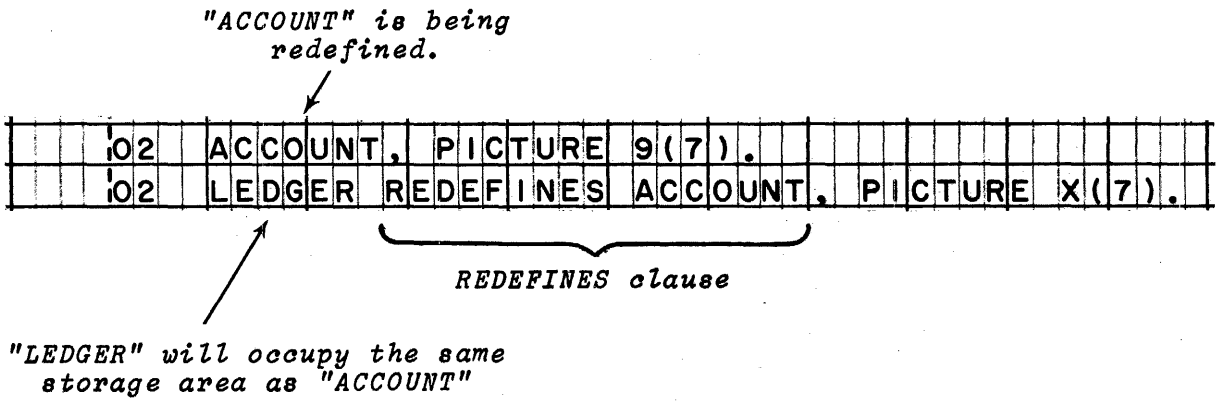
```
    02  SCORE, PICTURE S999
        COMPUTATIONAL-3, OCCURS 50 TIMES.
```

OCCURS clause

(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.



Procedure Division

PROCEDURE DIVISION ENTRIES

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.

```
PROCEDURE DIVISION.

BEGINNING-OF-JOB.
    OPEN INPUT PURCHASING-FILE.
    OPEN OUTPUT PURCHASE-REPORT-FILE.
    MOVE SPACES TO PURCHASE-REPORT-LINE.
    READ PURCHASING-FILE;
        AT END, GO TO END-OF-JOB.

DETAIL-PROCESSING.
    MOVE NUMBER OF COMMODITY TO OLD-NUMBER,
        COMMODITY-NUMBER.
    MOVE DESCRIPTION OF COMMODITY TO
        COMMODITY-NAME.
    MOVE DATE OF PURCHASE TO PURCHASE-DATE.
    MOVE UNITS-PURCHASED TO QUANTITY.
    MOVE UNIT-COST TO COST-PER-UNIT.
    MOVE TOTAL-COST TO PURCHASE-COST.
    ADD UNITS-PURCHASED TO QUANTITY-TOTAL.
    ADD TOTAL-COST TO PURCHASE-COST-TOTAL.
    IF LAST-LINE, WRITE PURCHASE-REPORT-LINE
        AFTER ADVANCING 0; ELSE WRITE
        PURCHASE-REPORT-LINE AFTER ADVANCING 1.
```

(Continued on next page)

Sample Procedure division (continued).

```
READ-NEXT-CARD.
    READ PURCHASING-FILE;
         AT END, GO TO END-OF-JOB.
    IF NUMBER OF COMMODITY IS NOT EQUAL TO
         OLD-NUMBER, PERFORM TOTAL-ROUTINE.
    GO TO DETAIL-PROCESSING.

TOTAL-ROUTINE.
    MOVE SPACES TO PURCHASE-REPORT-LINE.
    MOVE QUANTITY-TOTAL TO QUANTITY.
    MOVE PURCHASE-COST-TOTAL TO PURCHASE-COST.
    MOVE ZEROS TO QUANTITY-TOTAL,
         PURCHASE-COST-TOTAL.
    WRITE PURCHASE-REPORT-LINE AFTER ADVANCING 1.

END-OF-JOB.
    PERFORM TOTAL-ROUTINE.
    CLOSE PURCHASING-FILE, PURCHASE-REPORT-FILE.
    DISPLAY 'PURCHASE REPORT FINISHED'
         UPON CONSOLE.
    STOP RUN.
```

PROCEDURES

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.

```
INITIALIZE. MOVE ZEROS TO CONTROL-TOTAL.
```

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

```
INSERT-NEW-RECORD.
    MOVE CARD-RECORD TO OUTPUT-RECORD.
    WRITE OUTPUT-RECORD.
    READ CARD-FILE;
        AT END, GO TO FINISH-RUN.
    GO TO TEST-RECORD-CODE.
```

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

```
    MOVE NAME TO EMPLOYEE, MOVE
    SOCIAL-SECURITY-NUMBER TO ACCOUNT,
    ADD 1 TO NUMBER OF TRANSACTIONS.
```

- 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.

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)

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).

```
    DISPLAY 'RECORD NUMBER ' ITEM-NUMBER
        ' IS OUT OF SEQUENCE' UPON CONSOLE.
```

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.)

```
    DIVIDE NUMBER-OF-STUDENTS INTO
        SUM-OF-ALL-GRADES, GIVING
        AVERAGE-GRADE.
```

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.
```

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.

```
    IF DEMAND IS GREATER THAN STOCK-ON-HAND,
        ADD DEMAND TO BACK-ORDERS,
        MOVE BACK-ORDERED TO ACTION-CODE;
    OTHERWISE,
        SUBTRACT DEMAND FROM STOCK-ON-HAND,
        MOVE ORDER-FILLED TO ACTION-CODE.
```

(Continued on next page)

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.

```
    MOVE CURRENT-BALANCE TO AMOUNT-DUE IN
        CUSTOMER-BILL.
```

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.)

```
    MULTIPLY QUANTITY BY UNIT-COST,
        GIVING EXTENDED-COST.
```

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.

```
    NOTE -- THIS ROUTINE COMPUTES THE 'Q*9'
        FUDGE FACTOR, USED IN DETERMINING
        REORDER POINTS FOR WIDGITS.
```

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.)

```
    OPEN INPUT OLD-MASTER-FILE,
        TRANSACTION-FILE;
        OUTPUT NEW-MASTER-FILE.
```

(Continued on next page)

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.
```

TEST-CONDITIONS

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.

```
IF PAYMENT IS LESS THAN PREVIOUS-BALANCE +
    CURRENT-AMOUNT-DUE - DISCOUNT,
        GO TO PARTIAL-PAYMENT.
```

Sign test. Consists of the name of a data item, or an arithmetic 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: POSITIVE, 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.)

```
IF DIFFERENCE IS POSITIVE,
    DIVIDE DIFFERENCE INTO ACCUMULATED-TOTAL.
```

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 literal "0".

(Continued on next page)

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.

```
IF COMPUTER-PROGRAMMER,
   PERFORM SALARY-INCREASE.
```

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:

```
02  JOB-CLASSIFICATION-CODE, PICTURE 99.
    88 COMPUTER-PROGRAMMER, VALUE 72.
```

Interpretation of this 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.

(Continued on next page)

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.)

```
IF PAGE-IS-FULL,
    WRITE OVERDUE-ACCOUNT-LINE AFTER
    SKIPPING-TO-NEXT-PAGE;
    OTHERWISE, WRITE OVERDUE-ACCOUNT-LINE
    AFTER SPACING-2-LINES.
```

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:

```
APPLY PAGE-IS-FULL TO FORM-OVERFLOW
    ON OVERDUE-ACCOUNT-LISTING.
```

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.

```
IF ACTIVITY-RATING IS ALPHABETIC,
    GO TO HIGH-ACTIVITY-ANALYSIS.
```

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.

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.

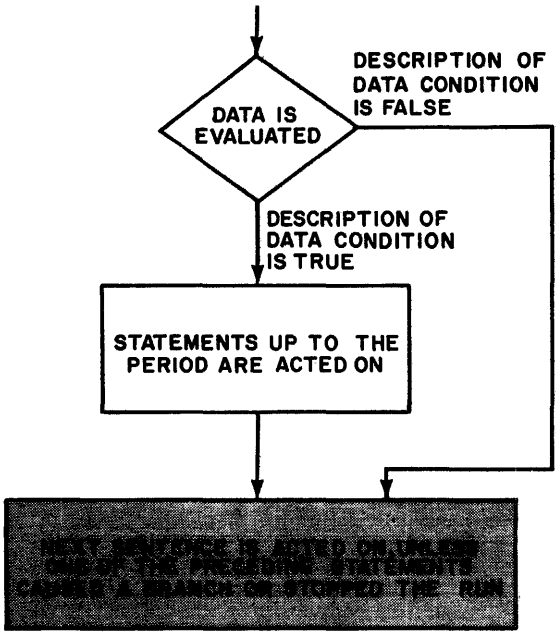
- a PERFORM statement gives control temporarily to another procedure.

- 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 through an IF sentence that does not contain ELSE or OTHERWISE.



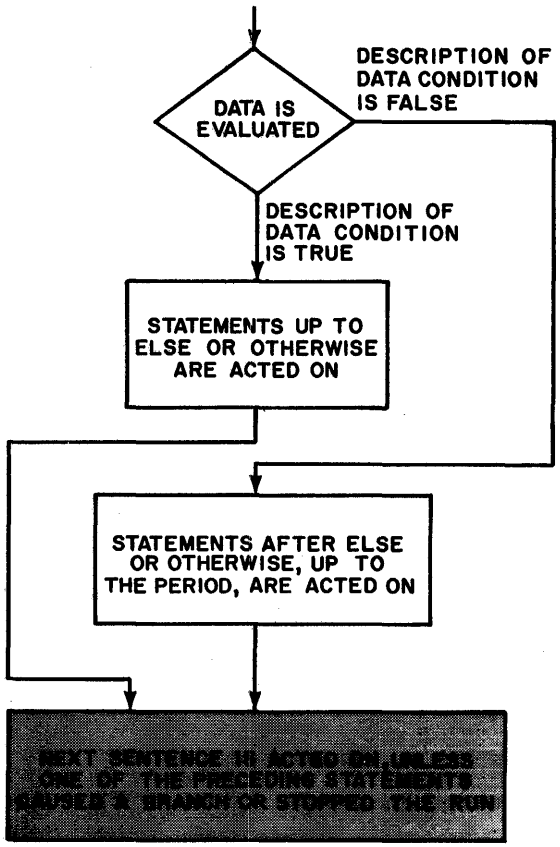
Example.

```
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)

Flow of control through an IF sentence that contains ELSE or OTHERWISE.



Example.

```
1    IF RECOVERY IS NOT GREATER THAN
2    SCRAP-POINT,
3        MOVE ITEM-NUMBER TO
4            SCRAP-ITEM,
5        ADD RECOVERY TO SCRAP;
6    OTHERWISE,
7        WRITE CONSERVATION-RECORD.
8    ADD 1 TO TRANSACTION-COUNT.
```

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).

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						

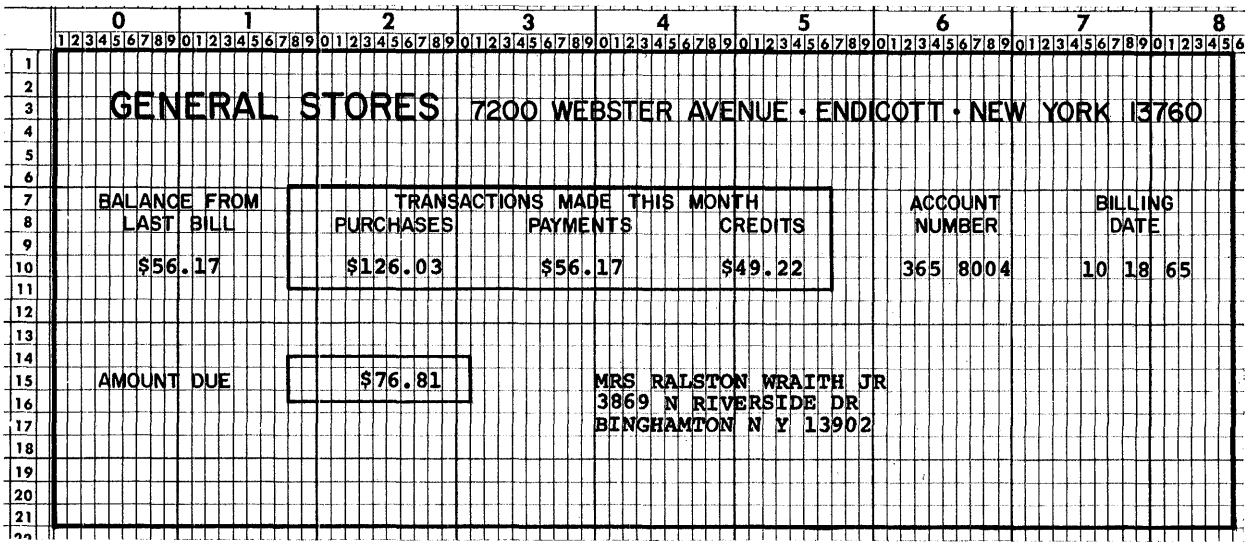
ACCOUNT HISTORY			LAST YEAR			
YEAR OPENED	YEAR LAST ACTIVE	HIGHEST BALANCE	MONTHS ACTIVE	MONTHS OVER 90	TOTAL PURCHASES	TOTAL RETURNS

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.



(Continued on next page)

CASE STUDY (continued)

Identification division.

```
IDENTIFICATION DIVISION.

PROGRAM-ID.
    'BILLING'

AUTHOR.
    PAUL MELICHAR.

DATE-WRITTEN.
    SEPTEMBER,1965.

REMARKS.
    STATEMENTS ARE NOT SENT TO INACTIVE ACCOUNTS. ANOTHER PROGRAM
    HANDLES ACCOUNTS WITH NEGATIVE (CREDIT) BALANCES.
    FOUR LINES ARE PRINTED ON EACH STATEMENT, UNDER CARRIAGE TAPE
    CONTROL. LINE 1 IS PRINTED AFTER A SKIP TO CHANNEL 1, LINE 2
    AFTER A SKIP TO CHANNEL 2. LINES 3 AND 4 ARE SINGLE-SPACED.
```

(Continued on next page)

CASE STUDY (continued)

Environment division.

```
ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER.
    IBM-360 G50.

OBJECT-COMPUTER.
    IBM-360 F30.

INPUT-OUTPUT SECTION.

FILE-CONTROL.
    SELECT BILLING-FILE, ASSIGN TO 'BILLFILE' UTILITY.
    SELECT CUSTOMER-BILL-FILE, ASSIGN TO 'CUSTBILL'
        UNIT-RECORD 1403 UNIT.
```

(Continued on next page)

CASE STUDY (continued)

Data division.

```
DATA DIVISION.

FILE SECTION.

FD  BILLING-FILE
    BLOCK CONTAINS 5 RECORDS
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS BILLING-RECORD.

01  BILLING-RECORD.
    02  ACCOUNT-IDENTIFICATION.
        03  TYPE-OF-ACCOUNT                PICTURE X.
        03  ACCOUNT-NUMBER.
            04  STORE-NUMBER               PICTURE 999.
            04  FILE-NUMBER                PICTURE 9(4).
        03  BILLING-CYCLE                  PICTURE 99.
        03  CUSTOMER-NAME                  PICTURE X(22).
        03  STREET-ADDRESS                 PICTURE X(22).
        03  CITY-STATE                     PICTURE X(22).
    02  CREDIT-STATUS.
        03  RATING-CODE                    PICTURE X.
        03  PURCHASE-LIMIT                 PICTURE 9(4).
```

```
    02  ACCOUNT-HISTORY.
        03  YEAR-OPENED                    PICTURE 99.
        03  YEAR-LAST-ACTIVE               PICTURE 99.
        03  HIGHEST-BALANCE                PICTURE 9(4)V99.
    02  LAST-YEAR.
        03  MONTHS-ACTIVE                  PICTURE 99.
        03  MONTHS-OVER-90                 PICTURE 99.
        03  TOTAL-PURCHASES                PICTURE 9(5)V99.
        03  TOTAL-RETURNS                  PICTURE 9(5)V99.
    02  THIS-YEAR-TO-DATE.
        03  MONTHS-ACTIVE                  PICTURE 99.
        03  MONTHS-OVER-90                 PICTURE 99.
        03  TOTAL-PURCHASES                PICTURE 9(5)V99.
        03  TOTAL-RETURNS                  PICTURE 9(5)V99.
    02  LAST-MONTH.
        03  NUMBER-OF-TRANSACTIONS         PICTURE 99.
        03  BALANCE-FORWARD                PICTURE 9(4)V99.
    02  THIS-MONTH.
        03  BILLING-DATE                   PICTURE 9(6).
        03  NUMBER-OF-TRANSACTIONS         PICTURE 99.
        03  CURRENT-BALANCE                PICTURE 9(4)V99.
```

(Continued on next page)

CASE STUDY (continued)

Data division (continued).

```
    03  PURCHASES.
        04  NUMBER                    PICTURE 99.
        04  AMOUNT                    PICTURE 9(4)V99.
    03  PAYMENTS.
        04  NUMBER                    PICTURE 99.
        04  AMOUNT                    PICTURE 9(4)V99.
    03  CREDITS.
        04  NUMBER                    PICTURE 99.
        04  AMOUNT                    PICTURE 9(4)V99.
    03  RETURNS.
        04  NUMBER                    PICTURE 99.
        04  AMOUNT                    PICTURE 9(4)V99.
02  COLLECTION-HISTORY.
    03  OVERDUE-BALANCES.
        04  30-DAY                    PICTURE 9(4)V99.
        04  60-DAY                    PICTURE 9(4)V99.
        04  90-DAY                    PICTURE 9(4)V99.
        04  120-DAY                   PICTURE 9(4)V99.
    03  LAST-PAYMENT.
        04  DATE                      PICTURE 9(6).
        04  AMOUNT                    PICTURE 9(4)V99.
    03  DUNNING-CODE                  PICTURE X.
```

(Continued on next page)

CASE STUDY (continued)

Data division (continued).

```
FD  CUSTOMER-BILL-FILE
    RECORDING MODE F
    LABEL RECORDS ARE OMITTED
    DATA RECORDS ARE BILL-LINE-1, BILL-LINE-2, BILL-LINE-3,
        BILL-LINE-4.

01  BILL-LINE-1.
    02  FILLER                     PICTURE X(4).
    02  OLD-BALANCE                PICTURE $$,$$$.99.
    02  FILLER                     PICTURE X(7).
    02  PURCHASES                  PICTURE $$,$$$.$$.
    02  FILLER                     PICTURE X(4).
    02  PAYMENTS                   PICTURE $$,$$$.$$.
    02  FILLER                     PICTURE X(4).
    02  CREDITS                    PICTURE $$,$$$.$$.
    02  FILLER                     PICTURE X(7).
    02  ACCOUNT-NUMBER             PICTURE 999B9999.
    02  FILLER                     PICTURE X(5).
    02  BILLING-DATE               PICTURE 99B99B99.
    02  FILLER                     PICTURE X(50).
```

```
01  BILL-LINE-2.
    02  FILLER                     PICTURE X(20).
    02  AMOUNT-DUE                 PICTURE $$,$$$.99.
    02  FILLER                     PICTURE X(11).
    02  NAME                       PICTURE X(22).
    02  FILLER                     PICTURE X(71).

01  BILL-LINE-3.
    02  FILLER                     PICTURE X(40).
    02  ADDRESS                    PICTURE X(22).
    02  FILLER                     PICTURE X(71).

01  BILL-LINE-4.
    02  FILLER                     PICTURE X(40).
    02  CITY                       PICTURE X(22).
    02  FILLER                     PICTURE X(71).

WORKING-STORAGE SECTION.

77  SKIP-TO-CARRIAGE-CHANNEL-1     PICTURE X   VALUE '1'.
77  SKIP-TO-CARRIAGE-CHANNEL-2     PICTURE X   VALUE '2'.
77  SINGLE-SPACE                   PICTURE X   VALUE SPACE.
```

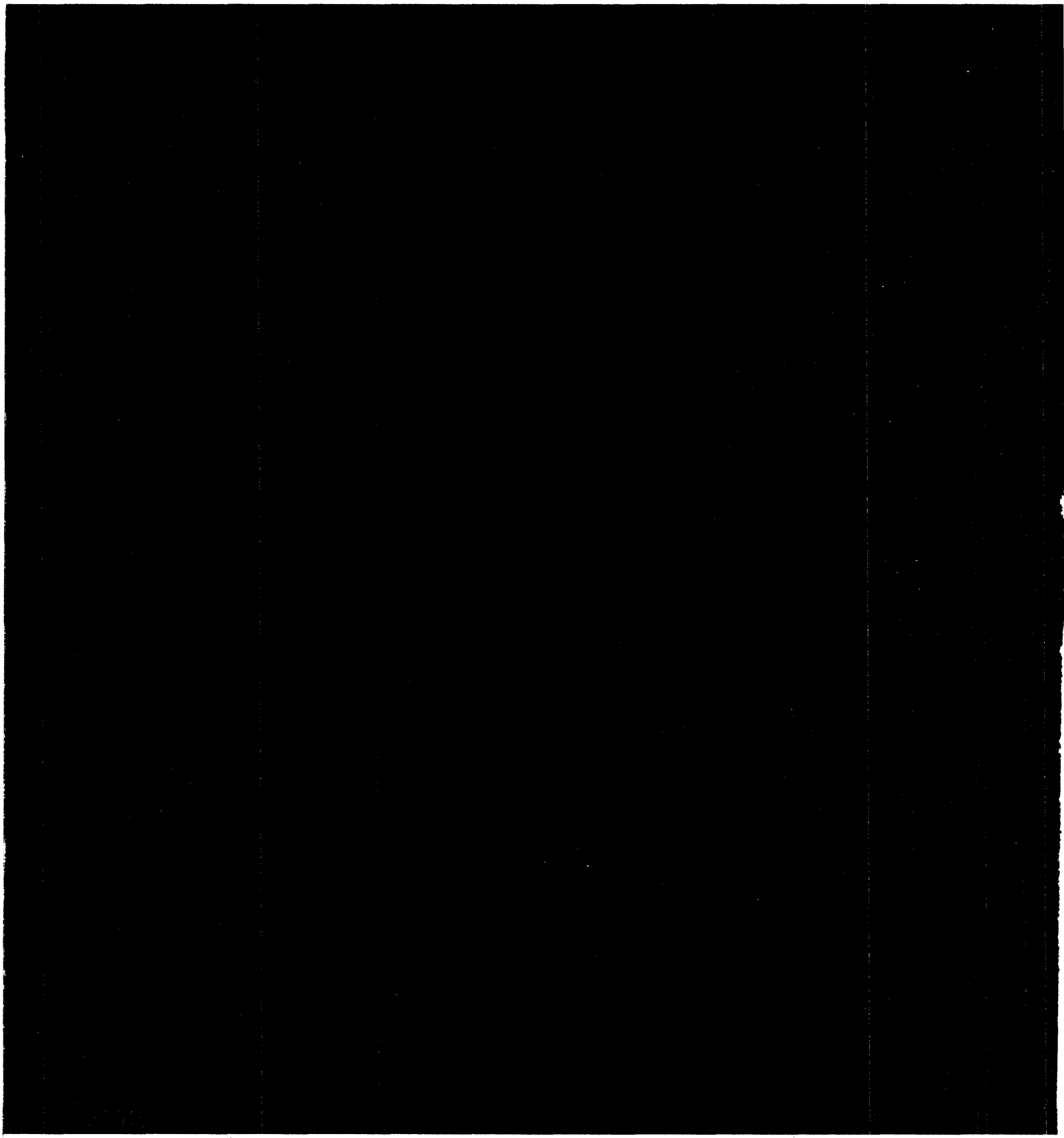
(Continued on next page)

CASE STUDY (continued)

Procedure division.

```
PROCEDURE DIVISION.

START-PROCESSING.
    OPEN INPUT BILLING-FILE, OUTPUT CUSTOMER-BILL-FILE.

READ-AND-CHECK-RECORD.
    READ BILLING-FILE; AT END, GO TO END-OF-RUN.
    IF NUMBER-OF-TRANSACTIONS IN THIS-MONTH IS EQUAL TO ZERO,
    OR CURRENT-BALANCE IS NEGATIVE, GO TO READ-AND-CHECK-RECORD.

LINE-1-PROCEDURE.
    MOVE SPACES TO BILL-LINE-1.
    MOVE BALANCE-FORWARD TO OLD-BALANCE.
    MOVE AMOUNT OF PURCHASES IN THIS-MONTH
        TO PURCHASES IN BILL-LINE-1.
    MOVE AMOUNT OF PAYMENTS IN THIS-MONTH
        TO PAYMENTS IN BILL-LINE-1.
    MOVE AMOUNT OF CREDITS IN THIS-MONTH
        TO CREDITS IN BILL-LINE-1.
    MOVE ACCOUNT-NUMBER OF BILLING-RECORD
        TO ACCOUNT-NUMBER IN BILL-LINE-1.
    MOVE BILLING-DATE OF THIS-MONTH
        TO BILLING-DATE IN BILL-LINE-1.
```

```
    WRITE BILL-LINE-1 AFTER SKIP-TO-CARRIAGE-CHANNEL-1.

LINE-2-PROCEDURE.
    MOVE SPACES TO BILL-LINE-2.
    MOVE CURRENT-BALANCE TO AMOUNT-DUE.
    MOVE CUSTOMER-NAME TO NAME.
    WRITE BILL-LINE-2 AFTER SKIP-TO-CARRIAGE-CHANNEL-2.

LINE-3-PROCEDURE.
    MOVE SPACES TO BILL-LINE-3.
    MOVE STREET-ADDRESS TO ADDRESS.
    WRITE BILL-LINE-3 AFTER SINGLE-SPACE.

LINE-4-PROCEDURE.
    MOVE CITY-STATE TO CITY.
    WRITE BILL-LINE-4 AFTER SINGLE-SPACE.

GET-NEXT-RECORD.
    GO TO READ-AND-CHECK-RECORD.

END-OF-RUN.
    CLOSE BILLING-FILE, CUSTOMER-BILL-FILE.
    STOP RUN.
```



International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York