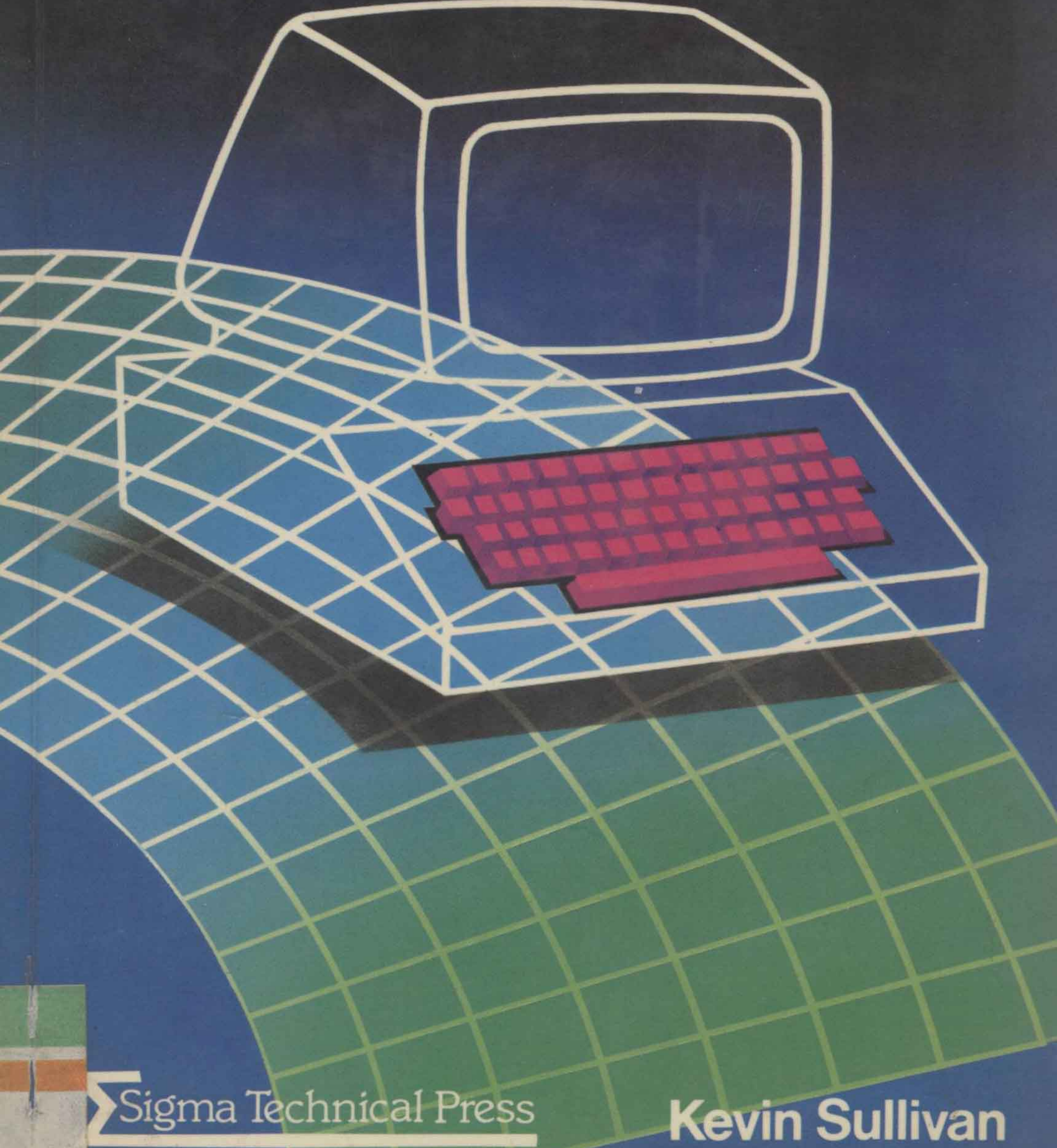
# PRACTICAL COBOL FOR MICROCOMPUTERS



Σ Sigma Technical Press

Kevin Sullivan

# Practical COBOL for Microcomputers

Kevin Sullivan

Sigma Technical Press

ISBN: 0 905104 60 9

**Published by:**

**Sigma Technical Press**
5 Alton Road
Wilmslow
Cheshire SK9 5DY
United Kingdom

Word processed on a Tandy TRS-80 Model III, using the Word-4-Word Wordprocessing package.

**Typesetting & Production by:-**

Designed Publications Ltd
8-10 Trafford Road
Alderley Edge
Cheshire.

**Distributed by:**

John Wiley and Sons Ltd
Baffins Lane
Chichester Sussex PO19 1UD
United Kingdom

Printed and bound in Great Britain by
J. W. Arrowsmith Ltd., Bristol

# PREFACE

This book is intended to be an introduction to COBOL, the most widely used business programming language in the world. COBOL is also one of the most standardised languages, so that the programs contained in the book will run on all COBOL systems with little or no modification. The programs were all developed and tested on a Tandy TRS-80 Model III microcomputer, using RMCobol, so in no sense is this book just a collection of mainframe programs adapted for a micro. RMCobol (marketed by Ryan-McFarland Inc) is one of the most widely used implementations of COBOL on micro, mini and mainframe computers, and is a completely standard version of the language. (For those of you concerned with standards, RMCobol is a GSA certified implementation of the ANSI X3.23 74 standard). So, if you have not already invested in a COBOL system, RMCobol (plus this book, of course!) is well worth considering.

Unlike the majority of books on COBOL, this one introduces the language through interactive screen-based programs, which start with some very simple examples and gradually develop to complete file handling systems.

All aspects of the language are covered, from basic screen handling techniques to indexed file methods. A background to different filing techniques and data structures is given, together with a section on structured programming with COBOL.

With the increasing use of micro-computers in business the gradual movement of COBOL from main-frame computers to micros has become inevitable. This book acts as an introduction to micro-computer based COBOL, and is ideal for the new student of COBOL who wants an example-oriented book. Since all of the examples are explained on a line by line basis, the book can be used without the need for any 'hands on experience', although this is desirable. The book itself forms an ideal 'first read' for anyone who wants to know more about the language before committing themself to one of the more advanced texts.

My thanks are due to Chris Lund of WLL Computer Services for reading the preliminary manuscript and for suggesting several technical changes; also, to Daf Sullivan for patiently reading and correcting the early drafts.

Kevin Sullivan, 1983.

# CONTENTS

# *Acknowledgements*

RMCobol is a trade mark of Ryan MacFarland Inc

TRS-80 is a trade mark of Tandy Corporation Inc.

CP/M is a trade mark of Digital Research Inc.

Appendix D is reproduced by permission of WLL Computer Services Ltd.

Appendices E and F are reproduced from the RMCobol manual by permission of Ryan-McFarland Inc.

Word-4-Word is a trade mark of Premier Publications Ltd.

# Chapter 1

# Introduction to COBOL

COBOL is a computer programming language that is widely used in the business environment. It is one of the earliest of the computer languages, dating back to the 1950's when it first became available on computer systems.

Most computer languages have one or more strong points and usually many weaknesses. COBOL was designed to be a fast efficient filing system for computers. As such it is not particularly good at complicated mathematical routines or at string handling. It does however produce excellent filing systems, which means that COBOL is ideally suited as a Data Processing Language and it is because of this that it is so widespread in the business/administrative environments. In addition COBOL has a very strong point in that it is a standard language. This means that about 95% of any COBOL system will be the same as any other. The differences will be minor ones. This has the great advantage that a programmer can go from one COBOL system to another with little or no problems. Languages like BASIC suffer from the fact that each manufacturer's version of the language can be very different from another.

The language itself reflects its age; it is a compiled language, this means that writing a program in it requires three quite distinct operations. Firstly, one has to write the initial program or source code using a program called the editor, this is then written to disc and the compiler program is loaded.

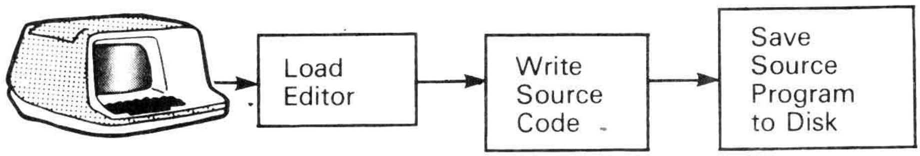
Secondly, the source program is then compiled, using the compiler and a separate program or object code is produced. This object code is then stored to disc and can be run when required. For the third and final stage, most micro-computer systems use another program called the 'runtime' program, which has to be loaded before the object code can be run.

Mainframe or Mini computers have the executable code produced directly from the compiler so this can be run directly.

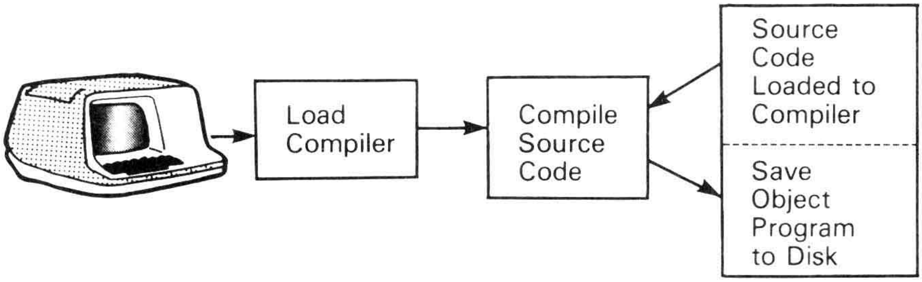
The sequence of events can be seen in Figure 1.1.

COBOL was originally designed to be run on large 'mainframe' computer systems, without the use of VDU's (visual display units). Therefore, its structure reflects this, with the programmer having to specify the type of system in use. The need for the system specification can be seen if we look at the way in which a COBOL program is divided.

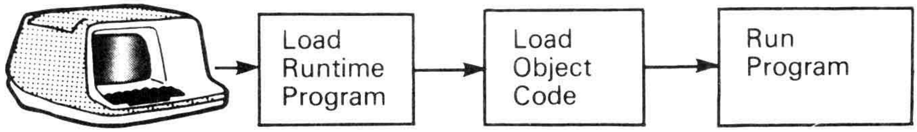
**Fig 1.1: Writing, Compiling and Running a COBOL Program**



**Stage 1: Write the Source Program**



**Stage 2: Compile the Source Program**



**Stage 3: Run the Object Program**

## The Four Divisions

Any COBOL program is divided into four parts, or divisions. These each have a particular function to perform and each must conform to a particular layout or format. The four divisions are as follows:

(1) IDENTIFICATION DIVISION
(2) ENVIRONMENT DIVISION
(3) DATA DIVISION
(4) PROCEDURE DIVISION.

We will now look at each division in turn

### The Identification Division

This is the division that gives information about the particular program. It consists of two essential parts which can be seen in the first sample program, PROGN1. Here the identification division consists of the phrases IDENTIFICATION DIVISION. followed by the phrase PROGRAM-ID. and the program name. The program name is any particular name that is allowed by the computer system being used.

```
000100 IDENTIFICATION DIVISION.
000110 PROGRAM-ID.    PROGN1.
000160 ENVIRONMENT DIVISION.
000170 CONFIGURATION SECTION.
000180 SOURCE-COMPUTER.   TRS-80.
000190 OBJECT-COMPUTER.   TRS-80.
000200 DATA DIVISION.
000210 WORKING-STORAGE SECTION.
000220 77 FIRST-MESSAGE  PIC X(54) VALUE "WELCOME TO THIS
PROGRAM".
000230 PROCEDURE DIVISION.
000240 START-N1.
000250DISPLAY " " LINE 1 ERASE.
000260DISPLAY FIRST-MESSAGE LINE 8.
000270 STOP-N1.
000280     STOP RUN.
```

The phrases mentioned above are essential, including all full stops, hyphens and spaces (one of the problems with an 'old' computer language, is that it is very particular about these). In addition to the essential information there are a number of phrases that can be included at the programmer's discretion. These are as follows:

AUTHOR. comment-entry.
INSTALLATION. comment-entry.
DATE-WRITTEN. comment-entry.
SECURITY. comment-entry.

All of the above details are reasonably self explanatory and a number of variations to the Identification division of PROGN1 can be seen in PROGN2 and PROGN3:

```
000100 IDENTIFICATION DIVISION.
000110 PROGRAM-ID.    PROGN2.
000120 AUTHOR.   KNS.
000130 INSTALLATION.   MY PLACE OF WORK.
000140 DATE-WRITTEN.   19/10/99.
000150 SECURITY.  SEE MAIN PASSWORD LISTS.
000160 ENVIRONMENT DIVISION.
000170 CONFIGURATION SECTION.
000180 SOURCE-COMPUTER.    TRS-80.
000190 OBJECT-COMPUTER.    TRS-80.
000200 DATA DIVISION.
000210 WORKING-STORAGE SECTION.
000220 77  FIRST-MESSAGE  PIC X(54) VALUE "WELCOME TO THIS
PROGRAM".
000230 PROCEDURE DIVISION.
000240 START-N1.
000250     DISPLAY " " LINE 1 ERASE.
000260     DISPLAY FIRST-MESSAGE LINE 8.
000270 STOP-N1.
000280     STOP RUN.
```

```
000110 IDENTIFICATION DIVISION.
000120 PROGRAM-ID.    PROGN3.
000130 AUTHOR.    THE AUTHOR.
000140 DATE-WRITTEN.   29/10/99.
000150 ENVIRONMENT DIVISION.
000160 CONFIGURATION SECTION.
000170 SOURCE-COMPUTER.    TRS-80.
000180 OBJECT-COMPUTER.    TRS-80.
000190 DATA DIVISION.
000200 WORKING-STORAGE SECTION.
000210 77   FIRST-MESSAGE    PIC X(54) VALUE "WELCOME TO
ANOTHER PROGRAM".
```

```
000220 PROCEDURE DIVISION.
000230 START-N1.
000240     DISPLAY " " LINE 1 ERASE.
000250     DISPLAY FIRST-MESSAGE LINE 8.
000260 STOP-N1.
000270     STOP RUN.
```

The purpose of the full stop at the end of each statement is to terminate that particular statement. It is possible to continue statements onto following lines, but for the time being we will restrict ourselves to statements that fit onto one line only.

## A Word on Columns

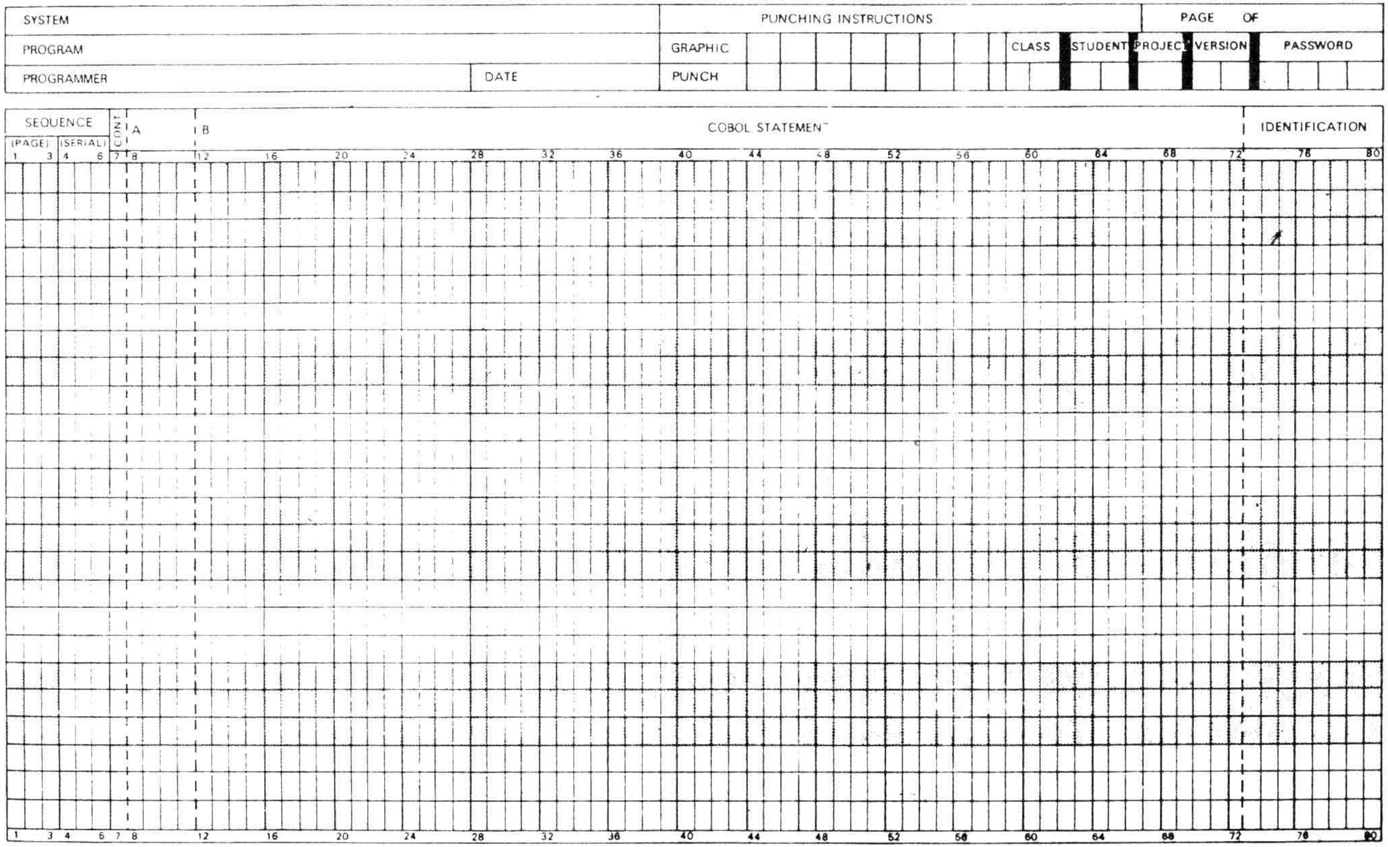
Having studied the sample programs shown so far it should be apparent that there is a particular layout to the programs. This layout is another inheritance from the old 'punched card' days and is present because the structure of the COBOL language was determined at a time when punched cards forced a particular layout on the computer user. The first six columns are used for line numbers only. The next column is used to mark comment lines. By placing a '*' in this column the compiler will ignore any entry on the same line. The main use for this is as an aid for program documentation. The following three columns are called the A margin and are used for Division and Section headers together with various data level entries. Most of the program lines are written in the next 60 columns called the B margin. The final 7 columns can be used as program identification remarks. This is a second 'comment' field and is also ignored by the compiler. Different manufacturers decided on a standardised system for these layouts or fields and these are adhered to even though relatively few systems still use punched card entry.

The various columns are shown in Figure 1.2.

| COLUMNS | DESCRIPTION |
|---|---|
| 1-6 | Sequence numbers for COBOL programs. |
| 7 | Primarily used for comments. Indicated by the presence of an * in the column. |
| 8-11 | The A margin. |
| 12-72 | The B margin. All lines not started in margin A start here. |
| 73-80 | Further comment columns. |

**Figure 1.2**

COBOL Coding Form



SYSTEM | PUNCHING INSTRUCTIONS | PAGE OF

PROGRAM | GRAPHIC | CLASS | STUDENT | PROJECT | VERSION | PASSWORD

PROGRAMMER | DATE | PUNCH

SEQUENCE (PAGE) (SERIAL) | CONT | A | B | COBOL STATEMENT | IDENTIFICATION

1 3 4 6 7 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80

## The Environment Division

The ENVIRONMENT division contains two sections. The first is called the CONFIGURATION SECTION and is that part of the program which can be used to record which computer was used to write the program and which computer was used to run the program. These are called the source and object computers respectively.

The second part of the ENVIRONMENT DIVISION is called the INPUT-OUTPUT SECTION and is used to describe and declare any files that are used in the program. We will look at the file structure in more detail later on. Briefly the information contained in this section covers the type of file (sequential, relative, indexed) and the various names that can be used to reference the file, both inside and outside the program.

In the main-frame world this section of the program would be used to tell the computer which disk drive to use for the file, or which printer to use for the output, and so on.

With micro-computers the disk used to store the file will usually be the first available, or the drive will be selected by the file name, (dependent upon the operating system used).

An example of an ENVIRONMENT DIVISION is given below,

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.
TRS-80.
OBJECT-COMPUTER.
TRS-80.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT DETAILS-1 ASSIGN TO RANDOM "DET/ONE"
ORGANIZATION IS INDEXED
ACCESS IS RANDOM
RECORD KEY IS NAME-ONE.
```

This example will be discussed in more detail in the chapter on indexed file handling.

## The Data Division.

This division is the one in which all the items of data to be used in the program are described. As before there are two sections to this division.

The first is called the FILE SECTION and it is this that describes the data structure of the file or files to be used. The second part is called the WORKING-STORAGE SECTION and this deals with any items of data that will be used in the program itself, during execution. The exact structure of the items in the data division is rather involved and will be covered in the chapter on data.

**The Procedure Division.**

It is this division that most people will think of as the program 'proper'. All the commands that instruct the program to do one thing or another are in this part. The PROCEDURE DIVISION is divided into a series of 'paragraph' names that are used as references. In many of the example programs the first paragraph is called START-N1 and is concerned with clearing the screen and possibly displaying any messages. The full structure of the PROCEDURE DIVISION will be dealt with separately.

# Chapter 2

# COBOL and Data

Because of its inherent data processing features, COBOL makes a rather big thing of how you describe the data that you intend to process. So, we'll spend quite a bit of time explaining how to do this.

## More about the Data Division.

The Data Division is the one in which all the data that will be used throughout the program is declared. This applies from the most sophisticated file descriptions down to small single characters. In this section of the book, the part of the data division called the WORKING-STORAGE SECTION will be discussed. The part of the division that deals with files will only be mentioned briefly and will be covered in greater detail in later parts of the book.

Data is described in the data division by using a level number. The simplest type of data is the type that refers to a single item, perhaps a name, a message, a number and so on. This type of data is called elementary data and has a level number of 77. If a data item is subdivided then this is called a group data item. Data items with this level number cannot be subdivided. At the time of writing there is talk of Level 77 data items being phased out. The feeling is that all data is related in some way and therefore is more correctly represented by a group data item.

All data items have to have a PICTURE clause specifying the length or format of the data. The three types of data are alphabetic, alphanumeric and numeric. Alphabetic data types are represented by an A in the picture clause, Alphanumeric data types by an X, and Numeric data by a 9.

The length of the data type is important. There are two ways of describing any data items:

(1) by representing each character or digit by its corresponding letter. For example, a five digit number could be represented by 99999. A two letter word would be represented by AA. An address of 10 characters by

XXXXXXXXXX. Clearly this somewhat simple approach is satisfactory for simple cases but would rapidly become unwieldy.

(2) The data type is indicated by a preceding letter and the length of the data item by a number in brackets. For example, a 10 digit number would be represented by 9(10). A 30 character line by X(30) and so on. This is the commonest method and will be used throughout the book.

We are now in a position to look at some examples of data items:-

```
77      FIRST-NAME          PICTURE A(20).
77      ADDRESS-ONE         PICTURE X(30).
77      TOTAL-NUMBER        PICTURE 9(10).
```

In each of the above cases the level number is the first entry in the line, this is followed by the name of the data item, and this in turn is followed by the picture clause. FIRST-NAME must consist of alphabetic characters of maximum length 20 ADDRESS-ONE must be alphanumeric of length 30 and TOTAL-NUMBER must be numeric of length 10. The picture clause can be abbreviated to PIC and it will be used as such in all of the following programs.

Before going any further, you may be wondering why the name of each data item is hyphenated. This is to avoid any possible confusion (on the part of the compiler) with COBOL reserved words. For instance, one example of a reserved word is DISLAY, which, if used as a data item, would be likely to cause a compilation error. A completely safe alternative would be to use DIS-PLAY, in which case only you might be confused! But, the point is that you might just forget which are and are not reserved words so, since none of them ever contain a hyphen, just hyphenate every data item and you will eliminate one possible problem!

To continue with the story, Level 77 data items are the simplest in that they cannot be further subdivided. More often we would want our data to represent some form that has various subheadings and other subdivisions. This can be achieved by using level numbers such as 01, 02, 03, 04 where the data item 04 is a subdivision of 03 and 03 is a subdivision of 02 which is a subdivision of 01. The following is an example of a data item CLIENT-RECORD which has three identical subdivisions.

```
01     CLIENT-RECORD.
       02      SUR-NAME          PIC X(20).
       02      FIRST-NAME        PIC X(20).
       02      JOB-TYPE          PIC X(15).
```