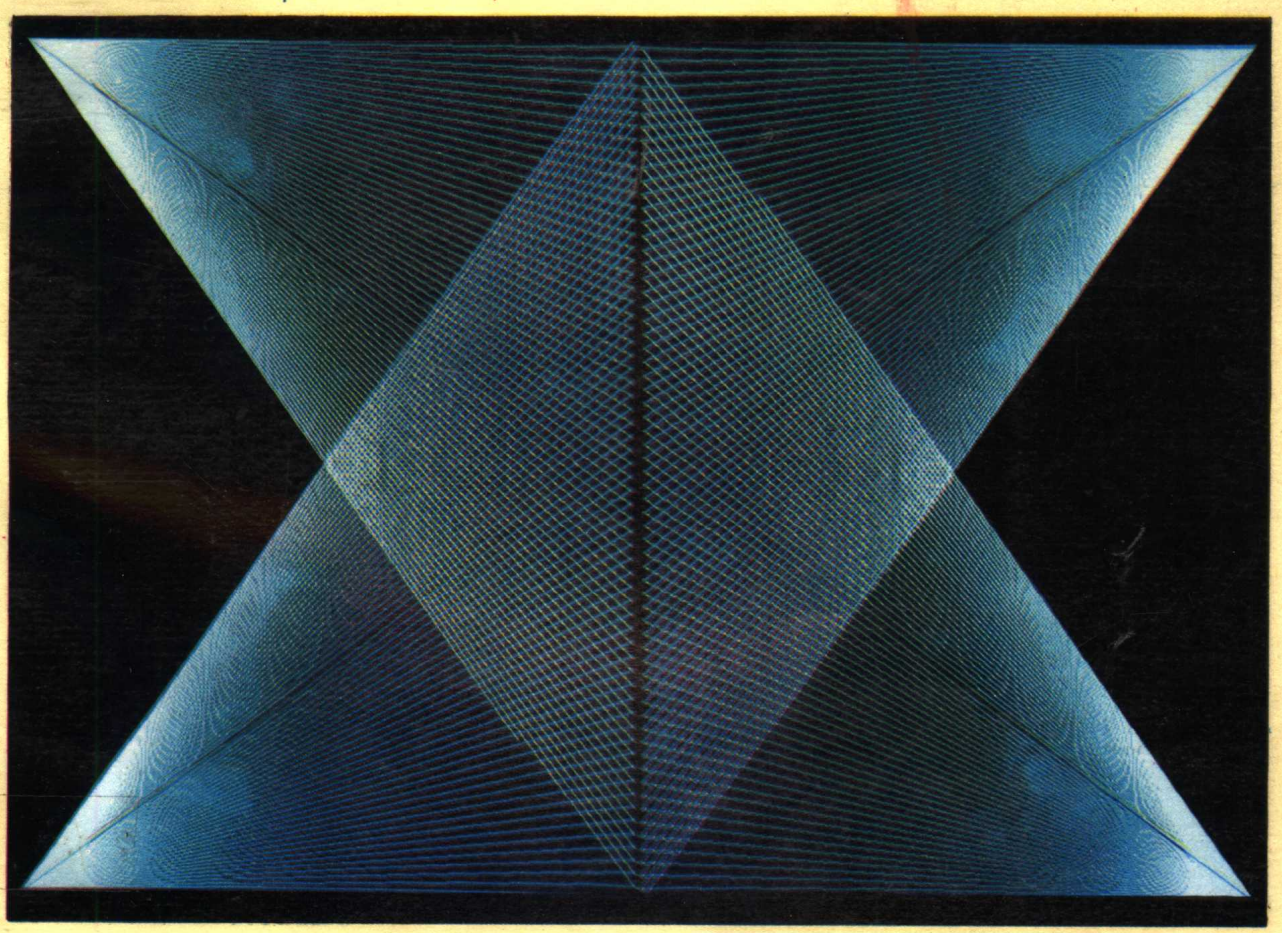
微机常用程序设计语言（一）

Level Ⅱ COBOL　MS-COBOL

COBOL 语言与结构化程序设计

主　编　赵锡腾

COBOL 语言与结构化程序设计
COBOL 语言与结构化程序设计
COBOL 语言与结构化程序设计



吉林大学出版社

微机常用程序设计语言(一)

Level Ⅱ COBOL MS-COBOL

COBOL 语言与结构化程序设计

主　审　唐建邦

主　编　赵锡腾

副主编　辛国平　杨　刚　崔福臣

参　编　姜素琴　李　宁　刘　涛　刘　鹤

赵彦苗　马玉亭　姚伟义　谭家祥

李洪军　李爱国

吉林大学出版社

内容简介

本书系统地介绍了在 IBM-PC 系列微机上广泛使用的 Level I COBOL 和 MS-COBOL 语言的全部功能，包括屏幕节和 ACCEPT/DISPLAY 语句扩充功能等。着重阐述了文件处理和表处理，注意把传授知识和培养结构化程序设计能力结合在一起。在编写过程中力求由浅入深，简明易懂、突出实用。在重点内容上都给出了完整的实例程序，并提供了上机操作所需资料，可供读者在实际工作中参考。

本书可作为计算机应用专业和各类程序员培训班 COBOL 语言课的教材，也可作为软件人员及其他有关人员自学 COBOL 语言的参考书。

COBOL 语言与结构化程序设计

赵锡腾　主编

唐建邦　主审

责任编辑：陈维钧　　封面设计：张沭沉

吉林大学出版社出版
（长春市东中华路 29 号）
吉林省新华书店发行
长春大学印刷厂印刷

开本：787×1092 毫米　1/16
印张：20
字数：495　千字
1992 年 4 月第 1 版
1992 年 4 月第 1 次印刷
印数：　1－6 100 册

ISBN　7-5601-1245-5/TP・21　　定价：9.00 元

前　言

COBOL 语言自 60 年代初问世以来，在计算机应用领域中得到了极其广泛的应用。成为国际上最流行的高级程序设计语言之一。已有许多统计资料表明，它在应用范围和占用计算机运行时间方面都远远高于其它高级语言，居于世界范围之首。

COBOL 语言之所以能得到如此成就，主要是由于它最适合于数据处理；具有很强的数据描述和处理功能、强有力的文件处理能力和独特的表处理能力。又加上通用性强，标准化程度高。这些都是其它高级语言无法比拟的。

本书选择 IBM-PC 系列微机上目前广泛使用的 Level Ⅱ COBOL 和 MS-COBOL 两个版本为文本，以微机为对象，以键盘输入数据、磁盘文件为主。在介绍 COBOL 语言的基础知识的同时，着重阐述了文件处理和表处理的内容，把传授知识和培养结构化程序设计能力有意识的结合在一起，以培养编写程序和分析程序的能力。在编写过程中力求由浅入深、简明易懂、突出实用。在重点内容上都给出了完整的典型的实例程序，如顺序文件、索引文件和相对文件，表的定义、赋值、引用，表与文件的相互传送和转换等都有实例程序。还介绍了 MS-COBOL 语言的屏幕节和 ACCET/DISPLAY 语句的扩充功能，程序调试功能等。最后以银行活期储蓄业务为例介绍结构化程序设计方法。在内容上即考虑了教学上的需要，也考虑了从事 COBOL 语言编程工作的软件人员的需要，对一些难度较大而实际上又需要的内容亦尽量做了阐述。书中带“＊”标志的章节系属于此类，初学者可暂不学习，待基础知识掌握以后，再回头学这些内容更有益处。

本书分十三章和五个附录。由赵锡腾主编，辛国平、杨刚、崔福臣副主编，参加编写的还有姜素琴、李宁、刘涛、刘鹤、赵彦苗、马玉亭、姚伟义、谭家祥、李洪军、李爱国，经过中国农业银行信息部主任唐建邦审定。在编写和出版过程中，得到了中国农业银行信息部的鼓励和支持。在此表示衷心的感谢。

COBOL 语言和其它语言一样，随着计算机科学技术的发展和应用领域的不断扩大，也在不断的发展和进一步完善。COBOL 语言的最新国际标准已于 1985 年修改通过，它更适合于结构化程序设计的需要，以及与数据库技术的结合。因此必将进一步推动 COBOL 语言的发展。

本书可以作为大、中专计算机应用专业和各类程序员培训班 COBOL 语言课程的教材，亦可作为软件人员和其他有关人员自学和参考。

由于计算机科学发展很快，以及编著者水平和经验有限，书中难免有不妥和错误之处，恳请有关专家和广大读者批评指正。

编著者

1991.8 月

目　录

第一章　COBOL 语言综述

§1　COBOL 语言的发展简史及特点

COBOL 一词是英语 Common Business Oriented Language 的缩写，直译为通用商业语言。它主要用于解决商业和企业管理等领域的大量数据处理问题，又称为企业管理语言、数据处理语言等。

COBOL 语言的研制始于 50 年代末，当时计算机的生产已进入第二代，计算机应用领域开始从数值计算扩大到数据处理。商业和企业事务管理中需要进行处理的数据量急剧增加，迫切需要有适用于这个领域的程序设计语言，以提高编制程序的质量和速度。COBOL 语言就是在这种社会背景下应运而生的。

1959 年 5 月，美国五角大楼召开了一个旨在讨论建立一种通用商业语言的可能性的会议，研究了开发 COBOL 语言的有关问题。在美国国防部的协助下，会后成立了数据系统语言协会 CODASYL(Conference on Data System Language)的常设机构来研究这种语言。1959 年 12 月，CODASYL 提出了第一个 COBOL 语言的版本。经过修改后于 1960 年 4 月由美国政府正式发表，称为 COBOL-60。其后，COBOL 得到了迅速发展和不断完善。二十多年来，CODASYL 制定许多 COBOL 文本，如：COBOL-61，COBOL-65，COBOL-68，COBOL-69，COBOL-73，COBOL-76 等。

为了使 COBOL 成为通用的数据处理语言，美国国家标准化协会 ANSI（American National Standard Institute）和国际标准化组织 ISO（International Standard Organization）对 COBOL 进行了标准化工作。1968 年 8 月，ANSI 通过了美国国家标准文本——ANSI COBOL X3. 23-1968，简称 ANSI COBOL-68；1972 年 ISO 通过了国际标准 COBOL 文本——ISO COBOL-72。它实际上就是 ANSI COBOL-68。1974 年，ANSI 对 COBOL-68 作了修改扩充，发表了 ANSI COBOL-74 文本。1978 年 ISO 宣布 ANSI COBOL-74 文本作为 ISO COBOL-78 文本。国际标准文本允许用英语、俄语和法语加以说明。ISO COBOL-78 版本是经过近 20 年的发展而产生的，为各国所接受，是目前流行最广、使用时间最长的一个版本。70 年代末以后，随着结构化程序设计方法的出现和数据库系统的发展，COBOL 语言的新版本经过讨论制定，于 1985 年通过了新标准。

COBOL 语言标准版本的实现依赖于具体型号计算机上的 COBOL 编译程序，而每种编译程序所实现的 COBOL 功能又是以标准 COBOL 文本为依据，是标准 COBOL 版本的子集，并做适当的扩充。

COBOL 是目前世界上应用最广泛的一种程序设计语言。据不完全统计，除军事部门外，大约有 60%～70%的应用程序是用 COBOL 语言编写的。在金融系统中更是这样，国外国内各银行目前使用的软件绝大多数都是用 COBOL 语言编制的。

COBOL 语言之所以能得到如此广泛而持久的应用，是由于它有以下几个主要特点：

(1) COBOL 语言面向文件，是一种典型的按文件系统方式进行数据处理的语言。它把要处理的对象(如档案、帐册等)按一定方式组织成数据文件，存放在磁盘、磁带等外部介质上。数据是按记录组织的。数据描述层次分明，便于修改、检索、排序等各种加工处理，极适合于现代各种管理领域的数据处理方式。

(2) COBOL 语言接近于自然语言——英语，便于阅读理解。它的描述性语句接近于英语的主表结构。它的操作性语句接近于英语的命令语句。例如，把 A 和 B 两个数据项的内容相加，结果放在 B 中，其语句为：

```
ADD  A  TO  B
```

可见，它的语义最易懂。对比较熟悉英语的人来说，用 COBOL 编写的程序有成文自明的效果。

(3) 通用性和可移植性好。COBOL 程序中，对运行环境、作为程序处理对象的数据以及实际的操作过程这三种内容分别在三个部分单独描述，而且通过定义助忆名方法，使操作处理过程中的描述不直接涉及到所使用的外部设备名字。这样以来，程序对计算机硬设备的依赖性很小。因此为某一种机器编写的 COBOL 程序，只需作少量关于设备描述的修改，即可移植到其它机器上运行。可见 COBOL 语言是一种高度独立于计算机系统的语言，它的通用性和可移植性很强。

(4) 功能模块化。在标准 COBOL 文本中把各种不同的功能划分成模块，并对每种功能模块的特性、语义和限制等都作了严格规定，模块本身又进一步分成一、二、三级，并且逐级包含，即高一级模块包含低一级模块的全部功能。编译系统的设计者可以根据需要和系统本身的规模选用全部或部分模块的功能。ISO COBOL-78 有 12 个功能模块(见下节所述)，而 Level Ⅱ COBOL 根据微机的特点选择了除报表模块外的全部功能模块，并且都是二级。选择模块的多少，不影响语言的基本结构。系统设计者还可在适当时候，加选另一些模块，扩充系统的功能。可见模块化为系统设计、扩充、移植等带来了极大方便。

§2 COBOL 语言的功能模块

COBOL 语言是由一个核心和多个功能模块所组成的，随着它的发展，功能模块不断增加，ANSI COBOL-68 只有一个核心和 7 个功能模块，到了 ANSI COBOL-74 就增加到 12 个功能模块。每个模块又分为一、二或三级，级别越高，功能越强。有些模块允许是空级(即不选用)。

表 1-1 ISO COBOL-78 功能模块

核心	表处理	顺序 I-O	相对 I-O	索引 I-O	排序合并	报表编制	程序分块	程序间通信	通信	调试	库
二级	三级	二级	二级	二级	二级	二级	二级	二级	二级	二级	二级
	二级		一级	一级	一级	一级	一级	一级	一级	一级	一级
一级	一级	一级	空	空	空	空	空	空	空	空	空

各个厂家的编译系统都是 ISO COBOL-78 的子集。最小功能配置是由核心一级、表处理一级和顺序存取一级三个模块组成，最大配置是由所有功能模块的最高级组成。

L I COBOL 语言选取了除“报表编制”模块外的其它所有功能模块，而且取二级功能。并且某些模块还进行了功能扩充。

MS-COBOL 语言大部分功能模块都实现了二级，对“程序间通信”、“排序与合并”、“调试”等功能模块进行了扩充。

§3 COBOL 语言的基本元素

构成 COBOL 语言的基本元素有 COBOL 字符、COBOL 字和常量三类。

(一)COBOL 字符集

COBOL 字符是组成 COBOL 程序的最小单位，它通常包含下述 51 个字符：

0～9	10 个数字
A～Z	26 个大写英文字母
+	加号
−	减号或连字符
*	乘号或星号
/	除号或斜线
=	等号
,	逗号或特殊规定下的小数点
.	句号或小数点
;	分号
”	双引号(有些机器允许使用单引号)
(	左圆括号
)	右圆括号
<	小于号
>	大于号
␣	空格
$	货币符号

L II COBOL 和 MS-COBOL 语言还允许使用小写英文字母来代替大写英文字母。

(二)COBOL 字

一个 COBOL 字是一个不超过 30 个字符的字符串，它的组成规则如下：

(1) COBOL 字仅由字母 A～Z，数字 0～9 和连字符“−”这三类字符组成，并且必须以字母开头，最多不超过 30 个字符。

(2) 连字符只能出现在字的中间，不允许出现在开头或结尾，相连多个连字符只保留一个。

(3) 一个 COBOL 字中至少要有一个字母。

(4) COBOL 字的两端要用分隔符定界。

分隔符包括标点符号、空格、关系运算符和算术运算符。多个空格和一个空格等效。

COBOL 字按其性质可分为保留字和用户字两种：

1. 保留字

保留字是在COBOL语言中有确定的语法含义，不能再作为用户定义字由程序员来定义。保留字是由英语单词或缩写来构成，它的语法含义一般是和英语单词的语义一致，以便易学易记。保留字又可分为必写字(下边划横线)、选择字(下边未划横线)和表意常量。

2. 用户字

由程序员根据需要而自行定义的COBOL字，按用途可将用户字分为以下几类：

(1) 数据名：

用于对一个数据项命名，而数据项是COBOL程序进行存贮、传送和加工等操作的基本数据单元。

(2) 条件名：

即条件的名字，用以表示条件变量当前取值情况。

(3) 过程名：

指过程部中由程序员自行定义的节名和段名。过程名的组成规则基本上与COBOL字相同，但它允许只用数字组成，此时只有当两个过程名的数字组成完全相同时，才能说两个过程名是相同的，而两个过程名的值相等，并不能说明过程名相同，如0001和001是两个过程名，虽然值相等，但组成的数字个数不等。

(4) 文件名：

用于对程序中使用的数据文件命名。

(5) 助忆名：

助忆名是与某种外部设备相联系的名字，这种联系在环境部的“专用名段”中建立。以后在过程部中就可以直接用助忆名来表示相对应的设备。

(6) 层号和块号：

层号是用来说明数据项之间的层次关系的，有01～49，77，66，88等两位数字。

块号是在采用覆盖结构时，用来说明过程部中的节属于哪块。块号最多由两位数字组成。

(三)常量

常量又称常字、直接量，它在程序中用来表示一个固定数据，在程序运行过程中保持不变，它不需要进行定义，可直接引用。COBOL语言中的常量有以下三种：

1. 数值常量

是表示数值大小的常量，它由数字0～9及＋、－符号、小数点等13个字符按下列规则组成：

(1) 至少一个数字，至多18个数字；

(2) 至多只能有一个符号，它若出现必须是第一个字符，若不出现，就认为是正数；

(3) 至多只能含有一个小数点，它可以出现在除最右位置外的任何位置上，若不出现小数点，就认为该常量是整数。

数值常量的值就是组成它的字符所表示的代数值。

2. 非数值常量

是指两端用引号定界的字符串，字符串内可以包括除引号以外的任意ASCII码字符。非数值常量的长度是引号之间的字符个数，最大允许长度为120，最短长度为1。非数值常量可以用它的编码与其它非数值数据项进行比较，但它不能参加算术运算。例如，“25.5”表示四

个字符的字符串，并不表示数值 25.5。

3. 表意常量

表意常量又称象征常量、赋形常量。它是用保留字命名的一种常量，它们的名称指出了它们的值，当源程序中出现了表意常量时，编译程序自动把它们替换成对应的常数值。COBOL 中的表意常量有以下六种：

(1) 表示数值为零或一至多个“0”字符：

ZERO

ZEROS

ZEROES

(2) 表示一个或多个空格字符：

SPACE

SPACES

(3) 表示一个或多个以十六进制“FF”表示的最高值代码：

HIGH-VALUE

HIGH-VALUES

(4) 表示一个或多个以十六进制“00”表示的最小值代码：

LOW-VALUE

LOW-VALUES

(5) 表示一个或多个双引号字符(有些机器为单引号)，此引号不能作为非数值常量的定界符：

QUOTE

QUOTES

(6) 表示一个或多个特定的字符：

ALL 常量

其中常字只能是非数值常量或除 ALL 外的表意常量，而不能是数值常量。

例如：ALL “*”

ALL “ABC”

ALL ZEROS

都是合法的，ALL “*”表示一列“*”，ALL “ABC”表示用 ABCABC…填充整个数据项，而 ALL ZEROS 的 ALL 显然是多余的。

对表意常量作如下说明：

(1) 表意常量的单数形式和复数形式等价，引用时可以任选。

(2) 表意常量和非数值常量不同，它不能在引号内，不能由用户自定义，而且它表示一个由系统规定了的确定值。

(3) 表意常量与其它数据项发生关系时，例如传送表意常量到某一数据项时，该表意常量表示的长度就等于其数据项的长度。

(4) 表意常量与其它数据项不发生关系时，它只表示一个字符，在这种情况下，ALL 不能使用。例如 DISPLAY ZEROS 只显示一个零。而 DISPLAY ALL “A”在语法上是错误的。

使用表意常量的举例见表 1-2。

表 1-2

语　句	数据 A 的字节数	执行语句后 A 中的值
MOVE ZERO TO A	5	00000
MOVE ZEROS TO A	1	0
MOVE SPACE TO A	5	␣␣␣␣␣
MOVE SPACES TO A	1	␣
MOVE HIGH-VALOE TO A	4	FFFFFFFF(十六进制) 即每个字节均为二进制的 11111111
MOVE LOW-VALVE TO A	2	0000(十六进制) 即每个字节均为二进制的 00000000
MOVE QUOTE TO A	4	' ' ' '
MOVE ALL "*" TO A	6	* * * * * *
MOVE ALL "AB" TO A	5	ABABA

§4　COBOL 程序结构

用 COBOL 语言编写的程序叫 COBOL 源程序，每个 COBOL 源程序顺序地由下列四个部分组成：

标识部：IDENTIFICATION　DIVISION.

环境部：ENVIRONMENT　DIVISION.

数据部：DATA　DIVISION.

过程部：PROCEDURE　DIVISION.

每个部由上述部标题(部首)开始，到下一个部标题或程序结尾为止。部标题在程序中必须单独占一行，一个部可由若干节组成，亦可直接由段组成。

一个节由节标题开始，到下个节标题或下个部标题或程序结尾结束，节标题由节名跟空格和 SECTION 及紧跟着的句点和空格组成。节标题也必须单独占一行。一个节由若干段组成，有些节也可不包含段。

一个段由段标题开始，到下一个段标题或节、部标题或程序结尾。段标题由段名及紧跟的句点和空格组成。过程部中的段由若干可执行的句子组成，其它部的段由若干描述体组成。句子由语句构成。语句中又可包含附加功能的子句或短语。描述体由子名或短语组成。

标识部、环境部和数据部都有固定的节或段。只有过程部的节和段由程序员自定义。

COBOL 源程序结构示意图见图 1-1。

COBOL 源程序举例：

```
IDENTIFICATION　DIVISION.
PROGRAM-ID.　EX1.
ENVIRONMENT　DIVISION.
```

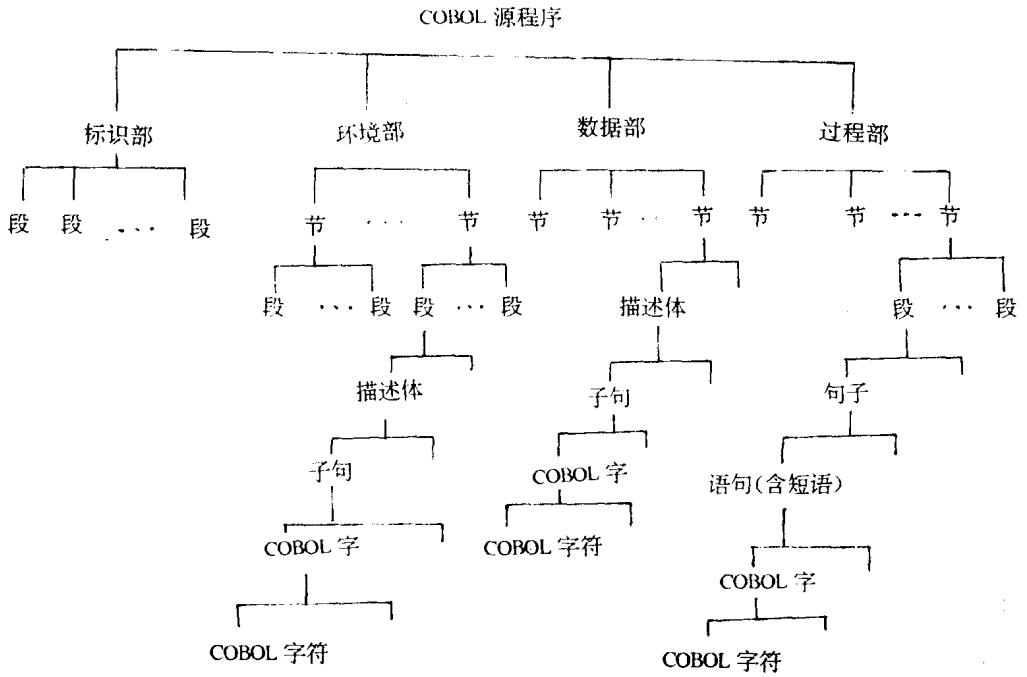


图 1-1 源程序结构图

```
DATA  DIVISION.
WORKING-STORAGE  SECTION.
77 BJ  PIC  9999.
77 LL  PIC  9V99.
77 BLH  PIC  9(5)V99.
PROCEDURE  DIVISION.
    ACCEPT  BJ.
    ACCEPT  LL.
    COMPUTE  BLH = BJ * (1 + LL).
    DISPLAY  BJ, BLH.
    STOP  RUN.
```

这是一个求本利和的 COBOL 源程序，第一行是标识部的标题，第二行是程序名段，其中 EX1 是自定义的程序名，第三行是环境部的标题，第四行是数据部标题，第五行是数据部中的一个节名，叫工作单元节，第六～八行是三个描述子句，分别描述三个数据项，BJ 是本金，LL 是利率，BJH 是本利和。第九行是过程部标题，第十～十三是输入本金、利率和计算本利和的语句，第十四行是显示本金和本利和语句，最后一行是程序结束语句。

§5 COBOL 源程序的书写格式

COBOL 语言开始时采用 80 列卡片作为输入介质，为了与其相一致，COBOL 源程序按每行 80 列的格式书写在程序纸上，并按这种书写格式输入计算机，并进行编译和运行。这种书

写格式称为常规标准格式。近年来，出现了一种终端标准格式，它是标准格式的简略格式，便于在计算机终端上输入源程序，它比常规标准格式规矩，取消了行号区和注释区，使用的空格减少了。但这种格式在编译前要用转换格式程序转换成常规标准格式，然后再进行编译和运行。终端标准格式，不同计算机有不同的规定，要参照使用手册进行。下面我们介绍标准格式(见图 1-2)。

1 6	7	8 11	12 72	73 80
		IDEN	TIFICATION DIVISION.	
		PROG	RAM-ID. EX.	
		ENVI	RONMENT DIVISION.	
		DATA	DIVISION.	
		WORK	ING-STORAGE.	
		77	BJ PIC 9(4)V99.	
		77	LL PIC 9V99.	
		77	BLH PIC 9(5)V99.	
		PROC	EDURE DIVISION.	
			ACCEPT BJ.	
			ACCEPT LL.	
			COMPUTE BLE = BJ * (1 + LL).	
			DISPLAY BJ, BLH.	
			STOP RUN.	

图 1-2 COBOL 源程序书写格式

COBOL 源程序书写格式规定如下：

1. 行号区：占程序纸的 1～6 列。用于书写页号行号，其中 1～3 列写页号，4～6 列写行号。页行号的使用主要是为了便于程序员对源程序进行修改。对编译无意义。

2. 指示区：占程序纸的第 7 列，可写如下一些符号：

(1) 空白：正常方式处理源程序的行。

(2) 连字符"-"：表示本行是上行的继续。COBOL 书写格式规定，语句或子句允许从 B 区开始一直写到第 72 列，如果没写完，允许从下一行的 B 区接着写。如果前一行的最后一个词的最后一个非空格字符与后一行的第一个非空格字符是属于同一个 COBOL 字，则在后一行的第 7 列上要写上连字符"-"。

(3) 星号"*"：第 7 列为"*"时，表示该行为注释行，在编译时注释行不进行编译，在列清单时原样列出。

(4) 斜线"/"：第 7 列为"/"的行仍是注释行，与"*"不同点是，在打印源程序清单时，自动进行换页，从下页开头打印这行。

(5) 调试符"D"：第 7 列为"D"时表示该行为调试行，这行在调试时，即在环境部 SOURCE-COMPUTER 段带有 WITH DEBUGGING MODE 子句时，才被编译运行，否则视为注释行。

3. A 区：占程序纸的第 8～11 列，部标题、节名、段名、文件描述符 FD、层指示号(01～49,77,66,88)等都从 A 区开始书写。

4. B 区：占程序纸第 12～72 列，此区又叫正文区，语句、子句等都从 B 区开始书写。

5. 注释区：占程序纸第 73～80 列，此区只能书写一些简略的说明，对程序的执行不起作用。只是为了阅读程序写注释，通常是空着不用的。

§6 数据、数据项和数据名

数据是 COBOL 语言处理的最基本单位。在 COBOL 语言中数据包括常量和数据项，常量在前面已经介绍了，是具有固定值的可以直接引用的量。而数据项类似其它语言的变量，是存放在程序运行中不断变化的量。数据项的名字叫数据名，用数据名来代表数据项，以实现对数据项的引用。数据名由程序员定义，但必须遵守前面介绍过的 COBOL 字的组成规则。为了便于阅读理解，通常根据数据项内容的含义来定义数据名，如年、月、日的数据项可以用英语单词来定名为 YEAR、MONTH、DAYY（DAY 为保留字，可以再加一个“Y”字母以示区别）。对于掌握汉字的我们，用汉语拼音来定义数据名更方便一些。如年、月、日用 NIAN、YUE、RI 更宜于我们阅读理解。

数据项之间通常是相互联系的，而且往往体现出从属关系。例如，一个职工的工资清单可以由职工编号、姓名、收入、扣除和实发工资等数据项组成。而收入项又包括基本工资、工龄工资、副食补贴、交通费、洗理费、加班费等，扣除项包括房租费、托儿费、水电费、病事假等，加班费又包括时间和款数，病事假包括天数和款数。可见工资清单各数据项之间，有的是并列关系，有的是从属关系。如编号、姓名、收入、扣除和实发是同一层次的并列关系。收入与下属项基本工资，工龄工资……等是从属关系。见图 1-3。在事务处理的数据结构中大量存在这种关系。COBOL 语言为了描述这种具有层次结构的数据，在数据名前赋予一个层号，根据不同层次，赋予不同的层号，通过层号描述数据之间的从属和并列关系。相同层号数据项为并列关系，不同层号为从属关系。

工资清单(GZQD)

编号 BH	姓名 XM	收入(SR)							扣除(KC)					实发工资 SFGZ	签字 QZ
		基本工资	工龄工资	副会补助	交通费	洗理费	加班费(JBF)		房租费	托儿费	水电费	病事假(BSJ)			
		JBGZ	GLGZ	FSBZ	JTF	XLF	时数 SS	款数 JKS	FZF	TEF	SDF	天数 TS	款数 KKS		

图 1-3 数据结构举例

层号使用的具体规则如下：

(1) 层号用两位数字表示，从 01 开始到 49 为止。最高层号为 01。如上例中，工资清单(GZQD)应赋予层号 01。

(2) 在层次结构中，具有并列关系的数据应赋予相同的层号，如上例中的编号、姓名、收入、扣除，实发工资等应具有相同的层号。

(3) 具有从属关系的数据名，其层号必须是递增的，即下层数据项应具有更大的层号，但不一定连续，如，基本工资，工龄工资等从属于收入数据项，因此它们的层号应大于收入数据项。