电子计算机软件

结构COBOL程序设计

商业实用程序53例详解

陈常仁　聂彦美译

谭浩强审校

湖南科学技术出版社

电子计算机丛书

结构COBOL程序设计

商业实用程序30例详解

电子计算机软件

结构COBOL程序设计

商业实用程序53例详解

陈常仁　聂彦美译

谭浩强审校

湖南科学技术出版社

电子计算机软件

结构COBOL程序设计

商业实用程序53例详解

陈常仁　聂彦美　译

责任编辑：周翰宗

*

湖南科学技术出版社出版发行

（长沙市展览馆路8号）

湖南省新华书店经销　湖南省新华印刷二厂印刷

*

1990年1月第1版第1次印刷

开本：787×1092毫米　1/16　印张：24.5　字数：613,000

印数：1——3,300

ISBN 7—5357—0634—7

TP·24　定价：9.00元

地科89—39

译 者 的 话

本书译自美国Harper & Row出版公司一九八三年出版的《Structured COBOL》一书，作者为纽约大学的Henry Mullish教授。

结构程序设计是一种现代科学的程序设计技巧。本书结合程序实例由浅入深地介绍了COBOL语言和用结构程序设计的原则进行COBOL程序设计的方法。

该书内容广泛，既包括了COBOL入门的知识，又包括了一定深度的内容，例如相当深度的表处理和报表程序的编写方法。在一些书中都忽略了的问题，如读取负数和 DATA RECORD子句的实际用途等，在本书中都可找到满意的答案。

由于该书起点不高，且以教科书的要求编写，所以适于初学者；又由于书中介绍了结构COBOL编程方法并有大量实用程序范例，所以又可作为学过COBOL的人的参考书。书中共有52个程序实例，大部分都可以作为结构COBOL编程模式查阅参考，许多子程序都可直接抄用。

在本书出版前承蒙对外经济贸易部计算中心付全铎校阅，在此表示衷心感谢。

限于本人水平，译文中肯定有不少缺点错误，敬请读者批评指正。

陈常仁　聂彦美　1985.8

前　言

按照美国全国标准化协会的要求，需要将下述“致读者”完整地转载如下。

任何组织如有兴趣均可随意地全部或部分转载该COBOL标准和详细说明，也可用该文本的思想作为说明手册的基础或其它用途。然而，所有这些组织对于任何这样的版本必须全部转载如下几段致读者作为其序言的一部分。（任何组织引用该文件的部分段落，例如在书评中引用时，需要提及COBOL致读者的来源，但不需要对它加引号。）

- COBOL是一种工业语言，它不是哪一个公司或哪一个公司集团的私有物，也不是哪个组织或组织集团的私有物。
- 不论是CODASYL（数据系统语言会议）程序设计委员会还是对COBOL作出了贡献的人都没有对该程序设计系统和语言的精确性与功能作任何明确的或含蓄的保证，当然也不会在这方面承担任何责任。
- 作者和拥有版权（即本书中引用的拥有版权的材料的版权）者已经过特别认可，即在其COBOL说明书中部分地或全部地引用该材料。上述授权还可以扩大到翻印和在程序设计手册或类似的版本中使用COBOL说明。

目　录

序　言

COBOL是一种工业语言。它不是任何公司或公司集团及任何组织或组织集团的私有物。本书是根据美国国家标准学会推荐的1974COBOL标准。该标准已被美国政府批准采用，同样已为国际标准化组织所公认。**国际标准化组织的成员包括如下国家。

澳大利亚	日本	意大利	南非	罗马尼亚	巴西	比利时	美国
新西兰	瑞典	加拿大	英国	墨西哥	德国	菲律宾	瑞士
南朝鲜	捷克	匈牙利	荷兰	土耳其	波兰	南斯拉夫	法国

由此可见，COBOL74标准得到广泛的国际性的承认。这个标准实际是过去的ANSI COBOL—68的修订版，该版本仍在世界上广泛使用着。本书试图说明两个标准在实质上的区别。

自从一九六0年COBOL第一次被设计定型后，其间经过系统的应用和发展。虽然今天COBOL的一般形式与一九六0年的标准是相同的，但为丰富完善其功能，增加了许多内容，使它作为一种主要的商用计算机语言以及在数据处理的领域中变得更为有效。不仅COBOL语言本身有所发展，在其它一些计算机上，特别是在小型计算机上（甚至在某些个人计算机上）COBOL的应用变得日益有效。并且在编写COBOL程序的风格上，亦经历了重要的进展，这种风格一般称为“结构程序设计”。

结构程序设计的极大成功应归功于一位名叫Edsger·W·Dijkstra的荷兰人，他是荷兰的因霍文大学的教授。早在六十年代中期，他和他的计算机科学小组在研究构成一个好的程序（不管采用什么样的语言）应该采用什么样的结构和不应该采用什么样的结构以及应采用什么样的设计风格方面得出了基本的结论。在那个时期以前，不管是用于商业方面的语言还是用于科技方面的语言，基本上视计算机编程为“随心所欲”的工作。当时对一个程序的主要评价标准是看其能否使用，而不管其方法如何。这种传统的编程手法常常导致编出的程序十分难读。一旦程序写好后，其逻辑极为隐蔽，甚至连程序员本人都很难再次走通它，更不要说为了维护的目的要别人去修改这些程序了。在传统的编程方法下，程序员编的程序反映了他个人的风格。然而，在结构编程方式中，这种风格一般就有所减弱，而取代它的是一种更通用、更具有一致性的程序结构。使用这种编程结构，即使原来对程序陌生的人，不管他是否意识到为解决某具体问题而编程的意图，都能读懂程序。在艺术上或独创性的写作方面个人风格，甚至非常特色风格都是受到鼓励的，然而在结构编程方式中，就须将这种个性的自我表现置于次要的地位。事实上，结构编程常常被称为“非个性化”的编程。

无疑，初学结构COBOL要比初学传统的非结构COBOL在细节上须多加注意。这也许就是现在只有百分之廿五的程序是用结构风格编写的原因。虽然许多程序员在口头上都欣然承认并肯定结构编程的有效性与合理性，但编写程序时他们仍沿用原已熟悉的编程风格而“言行不一”。一种坏的习惯是不易丢掉的。

然而，由于学习结构COBOL多花的力气最终还是能由结构编程的优越性得到补偿。结构程序易写，易修改。而且一旦程序运行，大多数另外的程序员修改起来也不困难。

如果你至今还可能根本没有接触到任何一种编程风格来编写程序，这倒是一件好事。也许你应当首先着手了解某些一般概念，然后学习一个小COBOL程序以便能够写出一个更好，更优美的结构COBOL程序。这些内容包括在第四章中。

COBOL是COmmon Business Oriented Language（面向商业的通用语言）的缩写。使用“通用”一词是具有许多令人信服的理由的。COBOL是经周密计划的深思熟虑的集体成果。从五十年代末，美国政府各部门，商业界代表及计算机厂家集中力量，共同努力，设计一种旨在专门用于商业、金融领域数据处理的语言。COBOL是一种通用语

**译者注：即国际标准化组织宣布的ISO COBOL—78。

，这是由于随着时间的推移，它已经成为今天使用的主要语言，最终会成为商业界的唯一语言。在当今世界上编写的所有程序中，大约有百分之八十是用COBOL写成的。事实上，COBOL在商业同行中已成为通用的语言。由于COBOL的普及，以及商业界已经依靠它去处理他们每日的巨大的工作量这种不可改变的事实，使掌握COBOL显得日益重要。至于结构COBOL就更重要了。此外，近年来对结构COBOL的需求不断增加，迫使教育机构也转向这种优美的程序设计风格。

结构COBOL在商业界编程方式的无比优越性导致更多的开明公司积极地鼓励结构编程风格。许多公司声称不再雇用使用那种“随心所欲”编程方式的程序员。

本书中例举的绝大部分程序都在CYBER170/720计算机上运行过，其编译程序是ANSI COBOL—74。其中大部分还能在配有ANSI␣COBOL—68的机器上运行，以便使那些仅配有ANSI␣COBOL—68的计算机也能运行本书内所列出的程序。

从已取得的教学经验看，该书可适用于二或四年制学院作为教科书，特别适用于第一次接触计算机语言的学生。本书中各章后的复习题和练习题能为教师布置家庭作业提供方便。还应提请注意的是近年来国内一些高中已决定将COBOL列为他们的新课程。可以断定，今后几年，这种趋势会继续下去，因为COBOL被确认为一种“面包及黄油”语言。

诚然，任何一本书都不可能包括COBOL的全部内容，而本书则以足够的篇幅来阐明COBOL的编程到底是怎么回事，并给读者提供一些今后在其它资料方面深入探讨的基础知识。由美国国家标准化协会和各厂家出版的手册中给出该语言的专门的版本提供了详细的资料，但是这些手册并不能作为学习该语言基本概念的行之有效的工具。尽管为了使COBOL成为一个面向问题的语言而不是面向机器的语言做出了相当的努力，遗憾的是并不是所有制造厂都能够坚持统一的标准，以致于在某一特定计算机上能够正常运行的COBOL程序如不加修改，在另一个计算机上就不一定能正常运行，即使是一个简单的小程序也是这样。一般地说，COBOL所应用的对象是那些工业，保险公司，政府部门和一般事务问题。

在阅读本书的时候，你会清楚地看到，这儿遵循一系列严格的原则，即：

1. 假设读者以前不具备计算机编程和事务处理的知识。
2. 书中有意识地避免使用专门术语。在某一章内讲述问题时，其中须用的术语应在前面出现过。
3. 本书采用的手法是不拘格式的，能使读者容易接受而不是望而生畏。
4. 读者最早可在第一章的中间学到如何写一个实际的COBOL程序。
5. 书内所有53个程序例子在各方面都是完整的，其程序清单原封不动地打印出来、程序实际运行时使用的数据及程序的输出也全部印出来并加以解释，所有内容无一遗漏。
6. 为了引起读者注意，各章中的要点在每章的开始即列出。
7. 每章后备有复习题，以便于消化、吸收书中内容。答案附于书后。
8. 根据各章的内容，每章还另备有练习题、为提供教师一些教学之需还另附补充材料。
9. 该书内容广泛。既包括COBOL的入门知识，又包括了一些一般书中都忽略的具有一定深度的内容。例如相当深度的表处理和报表程序的编写方法。
10. 书中所有的程序都是原封不动的并且可以直接应用于商业部门。
11. 最后一点是本书自始至终遵循结构编程的风格，以期读者能掌握这种风格。

致读者：

我要深深感谢我的老同学，伊利诺斯州的欧文·包姆巴，他无私地为本书贡献了他的聪明才智。只要感到手稿中有不足之处，他总是提出看法，打印程序清单并做具体修改。他的思想和建议都是合理的，并且大部分已被采纳。同时还要感谢苏珊·钱尼，她在百忙中抽出时间阅读了手稿，并提出不少对初学者有益的修改建议。最后还要感谢纽约大学的高才生，他们帮助我最后定稿。感谢大家。

还应指出的是，COBOL之父格雷斯·默里霍珀教授在UNIVAC计算机上第一个使用COBOL编译程序，这对COBOL的诞生和发展具有指导意义。

热烈欢迎COBOL世界的到来！

第一章 COBOL程序设计的基本概念

在本章中，我们将学习下述重要概念：

- 计算机程序
- CRT(终端显示器)
- 计算机的速度
- 计算机听从于程序指令
- 作为高级语言之一的COBOL
- 为何COBOL程序有时看起来显得冗长
- 机器语言
- 编译程序
- 诊断信息
- 逻辑错误
- 源卡片组
- 目标卡片组
- 穿孔卡片及其起源
- 赫尔曼·何勒内斯博士
- 字符编码
- COBOL程序的结构
- 第七列的用途
- 标识部
- A区和B区
- PROGRAM-ID(程序标识)名
- 设备部
- 数据部
- 过程部
- 程序的控制流程
- DISPLAY(显示)语句
- COBOL程序的层次特点

1.1 COBOL 编译程序

为解决某特定问题的计算机指令序列称为程序。对于COBOL的程序，这种指令(语句)过去用穿卡机穿成卡片。如今，将程序语句直接键入终端显示器日益普遍。终端显示器也叫CRT(即Cathode Ray Tube的词头)。从我们的观点来看，这两种方法是相同的，因为二者都是将穿孔的或键入的信息传送给计算机。计算机内部执行指令时很象我们一行一行地读书。当然，不同的是计算机是能快速依次地执行每条指令，能轻易而举地沿着指令序列飞驰。它既不知道为什么要这样做，也不懂得所做的任何部分，纯粹是一种机械的顺序动作。计算机毫无想象力，全无幽默、判断、怜悯、同情和感觉，仅仅是以飞快的速度毫厘不差，毫无知觉地执行人们通过程序指示它干的事，事实上，计算机有时出错的根本原因之一就是它只能按人们告诉它作的去做，而不能按人们的愿望去做。计算机程序一经启动运行，它就忠实地遵从顺序的每条指令。

虽然计算机毫无思维和智力，也没有任何动机和情感，但它却是人类所创造的、空前的惊心动魄的机器之一，能改变人类社会原来的生活方式。计算机是为了开拓而制造的。过去，社会上有大批人从事那种冗长乏味、单调易错的事务工作。计算机问世后，出色地承当了这些工作，将这些人从中解放出来，从这个意义上讲，它确实是功勋卓著，意义重大。

通常，在我们交谈的时候，我们可以使用相当含糊的语言。漏掉了个别字或词对交谈也没有多大的妨碍。但向计算机发出的所有命令均须准确而无二义性，措词必须严格地遵循计算机的规定。如果在该有句点的地方省略了句点，或者在不该有句点的地方加上了句点，就会

造成严重后果。有时候，这些规则往往是人为规定的，但人们必须严格地遵循它。

COBOL语言是第一个商用高级语言。之所以称为高级语言，是因为COBOL程序的词、短语和语法与通常书写的英语文章没有根本区别。因此，任何人只要懂英文，即使他从未接触过程序设计或商业的实践也会发现COBOL相对来说容易学习。

一般说来，COBOL的缺点是程序有些冗长，有人认为这是缺点，这是无疑的。但实际上，COBOL正是为了使哪些不熟悉该语言的人（包括经理和管理人员）也能读程序并能较好地理解程序表达的思想才这样设计的。由于COBOL程序是用通行的类似于英文的词汇写成的，所以它具有自说明的功能，这也正是COBOL的主要优点。

计算机并不懂得高级语言，只懂得"机器语言（0和1两个数码组成）"。机器语言极端乏味并且非常容易出错，对我们来说极感困难，但对计算机而言，机器语言却是理想的。0和1表示那些微小开关的通和断。

一个COBOL语句可以由计算机翻译成很多条机器（二进制）语言的指令。因为用机器语言编写程序非常单调并且要费很多时间，而且写成之后既不容易读懂也不容易修改，所以很少用它来编程序。这是很自然的。然而，编写和修改一个COBOL程序却不用费这样大的力气。由于COBOL程序如此容易懂，而且修改起来极为方便，因此美国有关部门规定，政府管理部门必须购买带有COBOL编译系统的计算机。

任何一个能运行COBOL程序的计算机必须具备一个特殊的复杂程序，通常称为编译程序。该编译程序的功用是逐次扫描COBOL程序的语句并把它转换成相应的机器指令。一条COBOL语句很可能翻译成20条甚至更多的二进制的机器指令。一旦转换成了机器指令形式，计算机就可以用近似电子的速度遵循这些指令执行。

现在，可能有人要问，COBOL是如何作这种转换的呢？一旦写成了一个COBOL程序，组成程序的语句都必须转换成计算机能处理的指令的二进制形式。编译程序由一系列极其复杂的机器指令子程序组成，这些子程序逐一检查COBOL程序的每条语句。如果经检查确认语句是合法的，就将它们转换成以后执行的二进制机器指令形式。只有最后执行机器指令才会产生结果。设计编译程序的主要目的是检查程序的每条语句语法是否正确，关键字是否拼错，以及确定特殊的保留字是否用得合适。如果发现某个COBOL语句非法会怎样呢？在这种情况下，就打印出错误信息和提示出错的原因。这与医生检查一个病人的健康情况一样。如果病人通过一定条件的所有检查，就可以确定是否健康。当然，这种比喻不可能很确切。编译程序容许轻微的错误，这时程序可以通过，同时打出警告性的信息；如果错误是致命性的，程序将立即终止运行。有时尽管一个COBOL程序的语句都符合语法规则，但该程序运行后并不一定能产生预期的结果。换句话说，就是语句的安排可能存在着逻辑错误。这样的逻辑错误在日常生活中却是常见的，相对说孩子更容易产生。例如，如果告诉一个孩子要穿衣服，还要洗个澡。这句话本身是正确的，但孩子如果按照说的顺序去做就会使你啼笑皆非。

可惜，迄今为止的所有编译程序还没有一个能纠正逻辑错误。纠正这种逻辑错误以及由编译程序查出的其它语法错误是程序员的责任。程序员最后能得到一份打印的编译清单及计算结果。还应当指出的是，有时一个错误就可能引起"诊断错误信息"的泛滥。一旦发生了这种出乎意料的事情，并不一定需要大量的改动。因为只要改掉这一个错误就可以消除所有的后续的、连锁的错误信息。有时，程序员经过相当大的努力，编译后还是返回许多"诊断错误信息"，这时就可能使人气馁了。在这种情况下，不要急燥为难，谁都会有错误，你亦不例外，随着实践经验的增长，错误就会迅速减少，并且改正错误也越来越变得轻易而举。

一旦COBOL语言的程序翻译成相应的机器语言程序后，就可以通过将它穿孔成卡片或写入磁盘或磁带上建立拷贝。如果再次运行该程序，比如说用新的数据重新运行一次，直接执行这种机器指令的二进制代码程序可以节约宝贵的计算机时间，因为这样无须再进行编译了。二进制代码程序是现成的可直接执行的程序。为了区分这两种程序，人们将原来用COBOL 语言写成的程序称为源程序，而编译好的形式称为目标程序。

1.2 穿孔卡片

当今世界上大多数计算机都接收80列的标准穿孔卡片。然而，现在有一种强烈的倾向，即人们对数据输入的兴趣从穿孔卡片转向显示终端。显示终端与电视机极为相似，只是它多了一个类似于电传打字机或穿卡机的键盘。其不同之处是已穿孔的卡片不能再次穿孔。不管你用的是哪种输入设备，了解一下穿孔卡片的知识还是有益的。

尽管穿孔卡片从三十年代起即已广泛地应用，实际它起源于美国革命的后期。当时，一位名叫杰克夸的法国纺织工已用木制的卡控制他的织布机。

一八八〇年，美国政府为进行人口统计需要处理大量数据。美国人口统计局为此招聘了一位名叫赫尔曼·何勒内斯的统计学专家解决制表问题。就在一八八七年，何勒内斯发明了一种机器能读的穿孔卡片，取得了极大的成功。早期的何勒内斯卡片长三英吋，宽五英吋。现在的卡片尺寸是$7^{3}/_{8}\times3^{1}/_{4}$英吋。这种八十列的何勒内斯(Hollerith)卡片从左到右顺序地分为八十个垂直的列。所有列又分为十二个水平行。在行和列的交叉处就是一个可以穿孔的位置。这样，一张卡片上最多可以穿960个孔。卡片的每列可以代表一个字符的信息，如代表一个数字，一个字母或一个特殊符号(注意，为了记录一个数字，仅需在一列上穿一个孔即可，而一个字母需要每列穿二个孔，一些特殊字符每列需穿三个孔)。

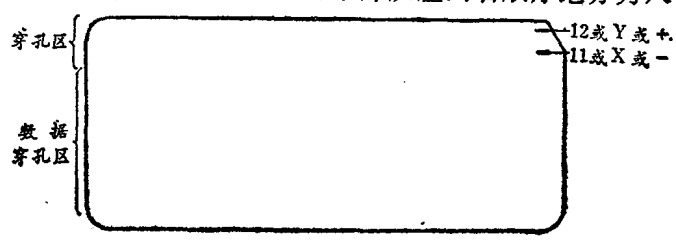

图1—1

一般，具体某一列形式的孔代表某个特定字符。这就要有一种统一的约定，否则，一张卡片上穿的信息在一种计算机上是一种意义，而在另一类计算机上可能就是另一种意义。这种卡片如图1—2。

图1—2的穿孔卡片上包含了一些代码信息。如果不提前提供穿孔编码，就不可能将卡片上的信息译码。下面是Hollerith卡片信息编码。它读作：

LETTER (字母)	HOLES (孔)	LETTER (字母)	HOLES (孔)	LETTER (字母)	HOLES (孔)	SYMBOL (字符)	HOLES (孔)	SYMBOL (字符)	HOLES (孔)
A	12 1	J	11 1	S	0 2	*	11 4 8	$	11 3 8
B	12 2	K	11 2	T	0 3	!	11 2 8	(	0 4 8
C	12 3	L	11 3	U	0 4	?	0 7 8	)	12 4 8
D	12 4	M	11 4	V	0 5	·	12 3 8		
E	12 5	N	11 5	W	0 6	>	0 6 8		
F	12 6	O	11 6	X	0 7	<	12 4 8		
G	12 7	P	11 7	Y	0 8				
H	12 8	Q	11 8	Z	0 9				
I	12 9	R	11 9						

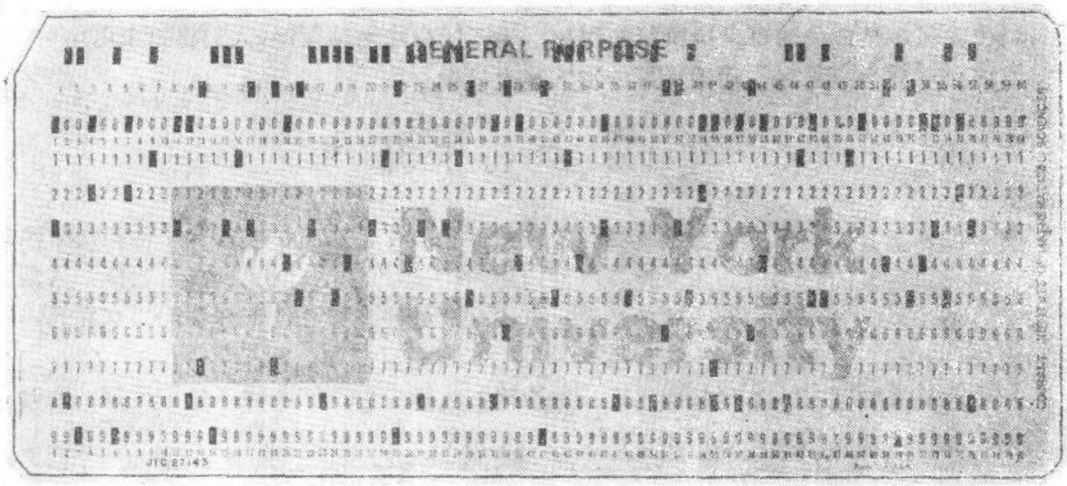

图1—2

数1代表在第一行穿孔，2代表在第二行穿孔等等。

按照这些信息编码，我们现在就可以翻译穿孔卡片。例如，图1—2的卡片。译码时必须检查卡片的每一列。相应翻译出的信息倒着打印如下。读作：

THIS IS A TYPICAL PUNCHED CARD CAN YOU READ THE HOLES? YOU HAVE 10 MINUTES

现在有的读卡机每分钟能读入12000张80列的卡片！近年来，IBM已经推出了 96 列的卡片，这种卡片的尺寸只有目前流行的80列卡片的1/3。图1—3是这种卡片的例子：

顺便指出，卡片上穿孔的信息越多，卡片就越轻，你明白这是什么原因吗？

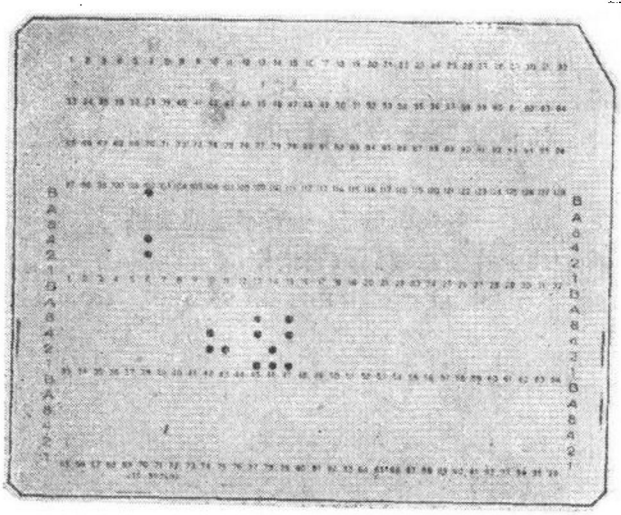

图1—3

1.3 COBOL程序的结构

每个COBOL程序分为4个主要部分，它们是

a.IDENTIFICATION DIVISION.	标识部
b.ENVIRONMENT DIVISION.	环境部
c.DATA DIVISION.	数据部
d.PROCEDURE DIVISION.	过程部

这些部的顺序必须按照上述那样予以熟记，或用“IN Every Darned Program”这几个词来帮助记忆。这几个英文词的中文意思是“在每一个编写的程序段”。其第一个字母(即I、E、D、P)恰好与四个部的第一个字母相同。

当在一个标准的80列穿孔卡片上将程序穿孔时，必须遵守一定的规则。前面提到的四个部的每一个部，必须各自用一张卡片，而且穿每张卡片时并不是随便在哪儿穿都行，它们必须从“A区”开始。所谓“A区”即从第八列开始到第十一列(包括第八列和第十一列)间的区域。(见图1-4)

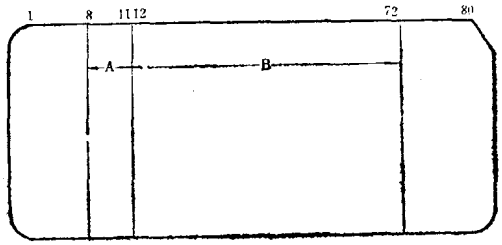

图1—4

随后就要说到“B区”，B区是从十二列开始一直到七十二列之间的部分(包括十二列和七十二列)。第七十三到列八十列不作程序内容处理，可以用来记顺序号、注释、或其它类似的用途。这些规定也适用于在将程序键入显示终端的情形。这时“列”这个术语称为“位置”。

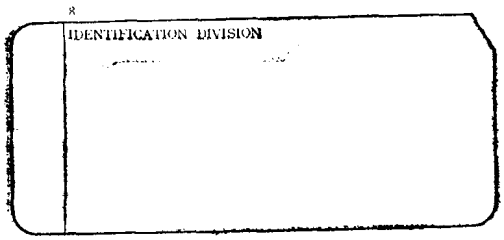

图1—5

需要的话，第一列到第六列可以用作附加的顺序号区域。虽然这属于任选的，但在程序很长时特别有用（当然一到六列可为空格）。第七列可用作注译行标志位置或续行标志位置（随后在以后适当的章节中将具体加以讨论）。

标识部

如图1—5所示，从A区开始（第八列）穿以IDENTIFICATION DIVISION标示标识部虽然标识部完全可以从八、九、十、或十一列开始，但为了一致起见，希望读者也象书中一样都从第八列开始。在两个词中间至少有一个空格。在COBOL中，任何允许一个空格的地方同样也允许任意数量的空格。IDENTIFICATION DIVISION. 部头在任何计算机系统中的COBOL程序中都是最前面的一项。部头及后面即要描述的各项都必须以句点结束。接着，下一行必须是“PROGRAM—ID.”，它也是A区开始的项，其后跟一个句点。附带指出，PROGRAM一词与ID之间必须用连字符“-”连在一起。在句点之后至少跟有一个空格。接着就是程序员任选的程序名，程序名之后同样以句点结束。PROGRAM-ID. 段实际上是标识部中唯一一个必须出现的段。

PROGRAM-ID后面的程序名的构成规则是：

1.该名字可以全部由字母，或全部由数字，或由字母和数字混合而构成。

2.名字最长允许30个字符，但某些编译程序可能只承认开始的八个字符有效，而且这样

的编译程序占多数。

3.可以使用连字符“-”(即减号),但绝不能出现在程序名的开始或结尾。

4.不允许出现任何特殊字符，例如逗号、句点等。

5.不允许嵌入空格。

还应当指出，并不是所有的编译程序都是类似这样处理PROGRAM-ID名的。许多IBM公司的机器只承认程序名的前八个字符；控制数据公司（CDC）的计算机只承认前七个；而宝来公司的机器仅承认前六个字符。IBM公司将嵌入的连字符处理为零。

6.选定的程序名（用户字）必须不是COBOL保留字。在COBOL中，有几百个单词的表，这些单词保留作为特别的用途。这些词除非正确地用到需要用它的程序中，其余地方都要避免使用。如DATA，COMPUTE，INPUT，ADD，MULTIPLY，EXIT，READ，WRITE等词在COBOL中各有其特定的用途，它们就是保留字的一些例子。本书封底附有保留字的表。由于太多，不可能全部记住，使用用户字时，有几种方法可采用：

a.需要时就查阅该保留字表。

b.故意在这些关键字前放一个X或其它字母，或放个数字以与关键字相区别。

c.保证记住一些常用的保留字。通过实践避免用错保留字。

下面一些例子就是标识部的一些正确示范。

```
a.IDENTIFICATION DIVISION.        b.IDENTIFICATION DIVISION.
  PROGRAM-ID, ARREARS               PROGRAM-ID.FINES.
c.IDENTIFICATION DIVISION.        d.IDENTIFICATION DIVISION.
  PROGRAM-ID.BILLS-PAID.            PROGRAM-ID.MSTR-FILE.
```

环境部

第二个部是ENVIRONMENT DIVISION（环境部或设备部)。之所以称为环境部或设备部，是因为它说明运行该程序的计算机环境或设备，也就是说，它规定程序运行的环境。设备部有一个节允许指定源计算机——在它上面编译COBOL程序；同时可指定目标计算机——在它上面运行COBOL目标程序。环境部的这个节称为CONFIGURATION SECTION.（配置节)。假设源计算机和目标计算机都是IBM370/145，则配置节可以全部略去，因为IBM的编译程序默认是IBM370。包括一个完整的配置节的形式如下：

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  IBM-370-145.
OBJECT-COMPUTER.  IBM-370-145.
```

如果使用的是IBM360就假定它的规格是45，其说明将是：IBM-360-45.

在Cyber计算机（本书例举的绝大部分程序所运行的计算机）上，该项形式是：

```
SOURCE-COMPUTER.  CYBER-170-720.
OBJECT-COMPUTER.  CYBER-170-720.
```

数据部

处理问题都必须从数据开始。实际上，程序设计可以定义为处理数据的技术。在COBOL程序中用到的各类数据都在DATA DIVISION.（数据部）中予以规定和描述。随后，我们将对其重点讨论。现在，我们仅涉及从A区开始其上穿有DATA DIVISION.的一张卡片，其它内容后面再详细讨论。

过程部

最后，我们要讨论四个部的最后一个部——PROCEDURE DIVISION。(过程部) 了。这是一个“执行部”，它通常是四个部中最复杂的一个部。在这个部中必须规定解决特定问题的精确的处理过程。目前，我们仅涉及只包括一个简单的 DISPLAY（显示）语句的过程部实例。这儿 DISPLAY 语句的作用仅是简单地将单引号引起来的全部内容都打印出来。注意，某些计算机，例如由数据控制公司生产的机器需要用双引号表示常量。但是按标准规定常量要用单引号(即撇号)引起来，因此，该书中全部采用单引号。比如要打印出“HI THERE”，我们可以使用COBOL语句这样写：

DISPLAY 'HI THERE'。

这种语句通常从B区（卡片的第十二列开始)。开始穿孔。实际上，所有的语句都要在B区内穿孔。下面是DISPLAY语句的一般形式。

DISPLAY {数据名—1 / 常量—1} [数据名—2 / 常量—2]…

COBOL合法的符号标记法

以上是用在COBOL中的说明语句结构形式的一般格式。这就是，如果一个词是大写并且下面划线，(例如DISPLAY一词) 则该词就是一个必写的关键字（只要使用到这种特性，这个字就必不可少)。如果一个词是大写的但没有在下面划线，则该词就是任选的。根据程序员的选择，可以写上，也可以不写。但不管是下划线还是非下划线的词，只要是大写词，都不能拼错。小写的词（译者都译为中文)，例如data-name-1（数据名-1）和literal-1（常量-1）用以代表COBOL的一般数据名和常量。

如果某项用花括号（{ }）括起来，如上面的数据名—1和常量—1，表示垂直列出的两个以上的项必选其一。在方括号（[ ]）中括起来垂直列出的各项是任选的。

一个省略号（…）由三个句点组成，根据程序员的选择可以从这儿起将该位置前面的项重复任意次。

在下面的程序中，STOP RUN（停止）语句的功能是终止程序的运行。注意，这条语句由两个独立的词组成，它们中间不能用连字符连起来。很多初学者常常将它们用连字号连起来，导致程序不能正确运行。当执行了STOP RUN之后，就立即结束程序的运行。STOP RUN也是B区的项。

作为一个可运行程序的实例，仅是为了说明问题，除了让计算机打印出短语“HI THERE”外，没有任何其它用处。下面是这个程序和它的输出。注意，即使在这个简单的程序中，四个部的部头卡片一张也不能少。ANSI COBOL-74规定，过程部头后必须从一个段名开始。这儿我们使用的段名（必须从A区开始）是HERE-WE-GO。

```
 IDENTIFICATION   DIVISION.
 PROGRAM-ID.      PROGI.
*
 ENVIRONMENT      DIVISION.
*
 DATA             DIVISION.
*
 PROCEDURE        DIVISION.
 HERE-WE-GO.
     DISPLAY 'HI  THERE'.
     STOP RUN.
```