徐元宙　朱三元　陈娜芬　吴伟雄　译

COBOL
语言基础

上海科学技术出版社

COBOL 语言基础

徐元宙　朱三元
陈娜芬　吴伟雄　译

上海科学技术出版社

COBOL 语言基础

徐元宙　朱三元
陈娜芬　吴伟雄　译

上海科学技术出版社出版
（上海瑞金二路 450 号）

新华书店上海发行所发行　上海市印十二厂印刷
开本 787×1092 1/16　印张 10.75　字数 252,000
1984 年 5 月第 1 版　1984 年 5 月第 1 次印刷
印数：1—24,400

统一书号：13119·1129　　定价：(科五) 1.45 元

译 者 序

计算机程序设计语言是人们与电子计算机进行通讯的媒介，也是人们使用计算机必不可少的工具之一．近二十多年来，随着计算机本身的不断发展，以及计算机应用领域的不断拓广，各种程序设计语言如雨后春笋般地不断涌现，至今已不下数百种．COBOL 是其中之一．随着计算机大量应用于政府机关、公用事业和各种事务数据处理领域，迫切要求有一种主要用于数据处理的程序设计语言，COBOL 正是处在这种条件下产生的．据有关报道，在国外，COBOL 的使用范围，居各种语言之首位．有的说占 70%，有的说占 80%，不管是多少，说明 COBOL 已经使用得十分普遍、广泛，是应用计算机的各类人员必须了解的一种程序设计语言，这一点是毋容置疑的．并且，国外的主要类型机器上都已配有 COBOL 编译程序，供用户使用．

在我国，使用 ALGOL 60 语言已有一定历史，大家对这种语言已非常熟悉，但学习 COBOL 的书籍却不是很多．近来，许多单位、部门都在大力推广普及计算机的应用，并且，把计算机应用于科学管理、事务数据处理领域的要求日益迫切，不少同志要求尽早得到自学 COBOL 的书籍．正是根据这方面的要求，我们翻译了这本书．想通过这本书对读者掌握 COBOL 的基本知识有所帮助．

COBOL 本身仍在不断发展，因此，任何有关这方面的书籍都带有一定的局限性．本书的内容主要依据美国国家标准化协会规定的 COBOL 语言结构进行叙述的，1974 年这个协会制订的 COBOL 标准文本，在 1978 年由国际标准化组织(ISO)定为国际标准 COBOL 文本；1976 年 CODASYL 组织也规定了一个标准 COBOL 文本，这些文本对原来的文本都在不同程度上作了增、删、修订．为了尊重原著，我们没有对原著作这方面的任何更动．在这里，只是告诉读者，在使用 COBOL 时，应根据所使用计算机提供的 COBOL 文本，不要抓住已了解的知识去指摘机器．但是，本书所提供的内容，确实是 COBOL 中最基本、最核心的内容，作为了解 COBOL 的基本知识，已经足够了．关于那些有更动的部分，可以进一步参阅有关的文本和书籍．

本书是一本适合于自学 COBOL 的书籍．同时，也可作为 COBOL 讲座或课程的参考书．但作为一本教科书，略有不足，主要是 COBOL 中许多规定的含义从文本上不容易理解，必须要有大量实例加以说明，本书已提供了不少实例，但对某些内容来说，例子仍嫌不够．

由于时间仓促，加上译者水平有限，错误不足之处在所难免，敬请读者不吝赐教，批评指正．

在翻译本书的过程中，曾得到 1932 所资料室、南京大学有关同志的大力协助，在此一并致谢．

译 者

目　录

第二部分 表 处 理

第三部分 排 序

第四部分 大容量存贮器

第一部分 COBOL 语 言

1. 引 论

1.1 什么是 COBOL?

要想确定什么是 COBOL，首先要确定它是怎样产生和为什么产生的. 如果能用一个单词来说明其存在原因的话，这个单词就是"通讯"(communication). 商业、教育和政府机关中所有从事特定业务的部门，早就有改善通讯的要求，而这些机关通常没有共同的语言。随着这些机构日益复杂，必需引进新的工具——计算机——以便满足各自的需要. 这种设备的引进，虽然有重要的价值，但在通讯方面也添加了一些新的障碍. 因为，不仅管理人员要了解技术员、分析人员要了解会计员和统计员或者领会他们的各种意图有困难，而且，要所有的人了解计算机也有困难，或者更明确地说，要把计算机技术应用到他们的各自不同的专业，并使所得到的结果对有关的少数人以外的人也广泛有用，是有困难的.

数字计算机，顾名思义，是接受数字信息并产生数字结果. 这些结果对于多数人来说，不作注释几乎是没有意义的. 然而，COBOL 致力于定义一系列的规则和过程，借助于它，就能用一种语言——效仿英语——来提出问题和设计解法，且为所有涉及到一种特殊的数据处理应用的人所接受. 的确，COBOL(COmmon Business Oriented Language)使得人们接近于有一种能按自身需要进行定制的计算机语言，因为它允许使用英语，并以一般人都易于理解的形式来标识许多程序成份.

"字"("word")是 COBOL 中基本的语言成份. 如同在英语中那样，COBOL 也含有许多种类的字，用以构成种种有意义的思想. "有意义"是这种语言的关键字义，也是任何一种语言的关键字义. 人们在交谈中，一个字所以有意义，是因为在某一时刻这个字已经定义好，并且已经与一个特定的概念或对象相联系. 在 COBOL 中一个字所以有意义，或者是因为程序员已把它与一个特定量或一组特定量相联系，赋予了它一定的含义；或者是它本来就是语言的一个固有的或内部的成份. COBOL 程序将使用许多类型的字. 它要使用程序员建立的名词，用以命名程序施行运算的各种数据成分；它要使用 COBOL 系统提供的动词，用以指明处理数据的方式；还要使用某些选用字，以改善语言的易读性或完整一个句子的意思. 例如，下面的句子就是使用这几种字的例子：

SUBTRACT DEDUCTIONS FROM GROSS-PAY GIVING NET-PAY.

(从工资中减去扣除费给出净工资)

这里，DEDUCTIONS, GROSS-PAY 和 NET-PAY 都是由程序员用来命名各特定量的名词. SUBTRACT 是 COBOL 提供的动词，用来指定特定的算术运算. FROM 和 GIVING 是 COBOL 中用来完整该句子意思的字. 象 IS、ARE 和 ON 这样一些字，通常用在语句中改善易读性，因而也可以省略. 本手册将解释这各种字的使用以及它们可取的形式.

简言之，COBOL 语言是为使用英语来陈述计算问题及其解法提供一种方法的系统的一部分．它由用来定义和建立程序的英语字和符号的一个基本集合所组成．并且，还允许用户按自己的命令定义和命名数据，而不必考虑计算机的特性．它构成了一种在很大程度上不依赖于计算机结构和型号的有效的语言．

1.2 历史和背景

1959 年 5 月 28、29 日，在五角大楼召开了一次会议，目的是研究为商业数据处理中电子计算机的程序设计建立一种公共语言的愿望和可行性．来自用户、政府机关、计算机制造商及其他有关团体的代表们出席了会议．在会上，几乎一致认为，这个设想在当时是必要的，并且也是可行的．并且一致同意建立三个委员会或特别工作队的想法．它们是短期(short range)、中期(intermediate range)和长期(long range)委员会，各自在适当时机开展工作．短期委员会由六个制造商和三个政府代表组成．这个委员会在 1959 年 6 月 23 日举行了首次会议，当时决定委员会的任务是四个总的方面，并成立下述工作组：

- 数据描述
- 过程语句
- 使用调查
- 用法和经验

前二个组多次开会并且提出了供全委会考虑的建议，全委会为向执行委员会提出报告，在 8 月 24 日～25 日开了会．另外二个组的工作成果的资料，后来用于进一步完善 COBOL 的开发．给执行委员会的报告是在 1959 年 9 月提出的，这个报告指出，短期委员会认为，它已经准备了一个框架，根据这个框架，就可以建立一个有效的公共商业语言．执行委员会认为这个报告有不完善之处，因而需要补充．并进一步要求由短期委员会去完成这个任务，并于 1959 年 12 月左右要公布这个系统．同时，还要求短期委员会在 12 月以后继续工作，以监督系统的实现．这两个要求都被批准了．

短期委员会在 1959 年 9 月 18 日至 11 月 7 日之间举行了几次会议，在解决问题和完善语言的工作中稳步前进．并采用了 COBOL(COmmon Business Oriented Language) 这个名字．

在 1959 年 11 月 16 日至 20 日这个星期内，短期委员会审查并批准了 COBOL 系统。向执行委员会的报告的最后编纂和首次分发是在 1959 年 12 月 17 日完成的．

在数据系统语言协会(COnference on DAta SYstems Language，简称 CODASYL)的执行委员会接受了这个报告以后，于 1960 年 4 月由政府出版局发表：

> “COBOL——呈数据系统语言协会的一份报告，包括用于数字电子计算机程序设计的面向商业的公共语言(COBOL)原始的说明书．”

由 CODASYL 团体的执行委员会成立了一个维护委员会 (Maintenance Committee)，负责收集并审查人们推荐的修改意见并加以修改，以保持 COBOL 的先进性．这个维护委员会由用户和制造商所组成．它研究并同意了对 COBOL 的若干修改意见．

为了把注意力集中于发表一个经修改和校正过的“COBOL-1960”，执行委员会成立了一个特别任务组 (Special Task Group)，这个任务组的成果就是 1961 年中期发表的 COBOL-1961 手册．在 1963 年中期，维护委员会提出了扩充的 COBOL-1961 版本．

参加维护委员会或特别任务组的组织有：

美国空军航空材料司令部 (Air Material Command, United States Air Force)

全国保险公司(Allstate Insurance Company)

本迪克斯公司(Bendix Corporation, computer Division)

波音公司(The Boeing Company)

伯罗兹公司(Burroughs Corporation)

曼哈顿银行(Chase Manhattan Bank)

控制数据公司(Control Data Corporation)

美国海军船舶局台维特泰勒船模试验池(David TaylorModel Basin, Bureau of Ships, U. S. Navy)

杜帮公司(Dupont Company)

通用电气公司(General Electric Company)

通用汽车公司(General Motors Corporation)

国际商业计算机公司(International Business Machines Corporation)

洛克希德飞机公司(Lockheed Aircraft Corporation)

明尼阿波利斯-杭尼韦耳调整器公司 (Minneapolis-Honeywell Regulator Company)

国家现金出纳公司(National Cash Register Company)

欧文·伊利诺斯有限公司(Owens-Illinois Incorporated)

飞歌公司(Philco Corporation)

美国无线电公司(Radio Corporation of America)

皇家麦克比公司(Royal Mc Bee Corporation)

空间技术实验室有限公司(Space Technology Laboratories, Incorporated)

南方铁路公司(Southern Railway Company)

标准石油公司(Standard Oil Company)(N. J.)

塞尔瓦尼亚电气产品有限公司(Sylvania Electric Products, Inc.)

斯伯里-兰德公司通用电子计算机部(Univac Division of Sperry Rand Incorporated)

美国钢铁公司(United States Steel Corporation)

西屋电气公司(Westinghouse Electric Corporation)

1964 年正月，为了成为实业团体并且扩大业务范围，COBOL 维护委员会作了改组. 经过这次改组，设立了三个分会：语言分会、评价分会和出版分会. 所有这三个分会都向 COBOL 委员会报告工作.

语言分会接办维护委员会的业务，即 COBOL 的维护和进一步发展.

出版分会负责提供正式的 COBOL 刊物，并就"COBOL 信息公报"的内容和美国标准化协会(USASI)联系. COBOL 信息公报是发向 COBOL 团体的有关 COBOL 资料的汇集.

评价分会的任务是对编译程序的实现和对用户所作的调查结果进行分析和评价. 这个分会向 COBOL 委员会提供有关 COBOL 使用的情况.

1965 年 11 月，COBOL 委员会又提出了 COBOL-1965 版本.

1961 年，中期委员会的一部分和长期委员会合并为开发委员会 (Development Committee). 这个委员会由一个系统组和一个语言结构组所组成. 虽然这两个组只是偶

而在一起开会，但人们已得出结论，把这两个组分为两个各自的委员会更为有益. 因此，CODASYL 执行委员会在 1965 年 4 月批准改组这两个组，并给它们如下指示：

(a) CODASYL 系统委员会(CSC)的任务是开发一个数据系统语言，这种语言使用通用的、容易理解的语言作为中间表示；独立于计算机；并且作为用于所有的数字计算机系统的公共语言，以及作为人们通讯的一种媒介.

(b) CODASYL 语言结构委员会(CLSL)的任务是开发一个说明数据处理系统的统一结构，重点放在所要获得的数据处理结果的性质方面，而不是放在所需要的各个步骤的顺序方面. 建立这种语言，主要是便于从事各种数据处理问题的人使用，而不是从事计算机程序设计的人使用；并且，要求在处理数据的机器上容易被实现. 1965 年 7 月，CLSL 应其自身的要求而解散.

1960 年，在商业设备制造商协会(Business Equipment Manufactures' Association，简称 BEMA)的发起下，成立了美国标准化协会(ASA)的计算机和信息处理分部委员会 ASAX3(American Standards Association Sectional Committee X3). ASAX3 依次设立了 X3.4 分会，从事于公共程序设计语言标准化方面的工作. 此后，为处理机设计各种说明和为 COBOL 的标准化，又成立了 X3.4.4 工作组，该组的首要任务是提出美国 COBOL 标准.

1961 年 12 月 17 日，X3.4.4 工作组的一次组织工作会议，对参加开发美国 COBOL 标准可能感兴趣的制造商和用户发出了邀请. X3.4.4 工作组的第一次会议于 1963 年元月 15 日、16 日在纽约市召开. 1966 年 8 月 ASA 改为美国标准化协会(USASI).

自 1963 年元月的第一次会议以后，又召开过多次会议，且在 1966 年 8 月 23 日，X3.4.4 工作组为将要提出的 USASI 标准 COBOL 审定了内容和格式. 这个推荐的标准是以 CODASYL 的 COBOL-1965 版本为基础，它相当于那个版本所定义的整个 COBOL 语言的一个真子集.

1.3 使用 COBOL 的获益

下面是使用 COBOL 的一部分优点：

·具有相容性

把在某一台计算机上运行的程序，经少量修改，放在另一台完全不同结构和型号的计算机上运行，这种可能性是已经证明了的事实. 这样，改编程序所需要的时间，比用一个面向设备的语言重新编码所需要的时间，大大减少了.

·扩大了通讯范围

由于 COBOL 使用类似英语的语句，因此，书写的程序容易为所有了解应用的人所理解. 这样，程序设计的成果、知识和以前被隔绝的许多专家的成果都能用它进行交流了.

·更快的程序设计

由于程序员从使用面向设备的语言时非常繁琐的书写工作中解放了出来，所以对新的应用或改变了的应用编制程序的时间减少了，这样就可以把注意力从详细编码的程序设计工作，移到去确定和分析种种问题.

·提高了程序的正确性

在广泛的用户以及大量系统经验的基础上，建立了使用 COBOL 的标准化约定，发展了

有效的编码技术. 因为这些技术是自动地引入到程序中去的，所以提高了编码的效率. 这种按照用户的要求调试好的编码段，使程序出错的可能性减小，并且简化了设计.

·缩短了训练时间

同使用面向设备的语言相比，只需用相当少的时间来训练新的程序设计人员去编写工作程序. 采用 COBOL，就不必拥有大量受过训练的程序员. 只需对少数经过挑选要培养为高度熟练、细心的程序员，教授更复杂、费时的计算机编码技术.

·降低了程序设计的成本

由于能很快地针对一个问题编出程序，并且，程序的大部分更正确，所以就降低了程序设计的成本. 重编程序的费用也降低了，因为对某个系统编制的程序，经修改而不需要全部重新编码，就可在另外的系统上正确运行.

这些只是使用 COBOL 所得益处的一部分. 随着用户对 COBOL 的进一步熟悉，该系统许多其他优点会进一步显示出来.

1.4 COBOL 和计算机

数字计算机仅接受并执行以特定方式表示的指令的固定集合. 虽然指令的个数和复杂程度可随计算机的不同而变化，但是某些一般概念是相同的. 每台计算机至少都具有实现下述功能的一些指令：

数据传送

算术运算

判别控制

输入/输出操作

其中每一种功能都可以由许多指令来实现，并且，它们说明书的格式，随着当今市场上各个计算机结构和型号的繁复种类而各不相同. 与迎合各别计算机指令特性的各种面向设备(面向一台计算机或面向一个计算机系列)的语言不同，COBOL 着眼于功能. 这就是说，它根据要解决的问题，而不是根据需要进行程序设计的计算机，以英语的形式，为处理上述这些功能提供了方便.

当然，COBOL 语言并不是自由式的英语. 它具有英语的风格，但只限于风格上的相似. 与人脑不同，计算机不能接受表达上的细小差别，不能分辨声调的变化，也不能识别为强调所作手势的含义. 因此，计算机能接受什么和不能接受什么，这是由计算机内在的限制所制约，根据这种内在的限制，COBOL 语言就在某种意义明确的界限之内起作用，这些界限，是由非常明确的过程规则所设定. 然而，对于任何一种语言，不论它是为计算机，还是为人类而设计的，都包含各种结构和使用规则. 这些规则提高了作为会话和通讯工具的这个语言的使用价值. 程序员必须使自己熟悉 COBOL 中基本的规则和过程，才能精通问题的确定、分析和解法.

1.5 COBOL 系统

COBOL 系统实际上有两个主要成份：程序和程序处理器(编译程序).

当我们谈到程序时，实质上指下面两个程序：

· 源程序

程序员作出的一组体与语句,且实现下述功能:

——描述数据.

——利用处理数据的一组过程步骤,命令计算机求取问题的逻辑解.

——标识程序.

——描述要使用的设备.

·目标程序

计算机能接受的一组编码指令和数据,用来实现源程序表示的逻辑,并对要处理的数据进行存贮分配.

为了把面向问题的 COBOL 源程序,翻译成面向机器的、用特定计算机能接受的编码形式书写的目标程序,就需要提供一个 COBOL 程序的处理器(编译程序). 这种编译程序是一个程序,分析 COBOL 程序的字和字符,并产生一个用特定计算机所能接受的编码写成的新的程序. 对于单个 COBOL 语句,编译程序可能翻译成许多条机器指令. 该编译程序与源程序一起进入计算机,然后得到能运行的目标程序,以求取问题的解.

在翻译源程序为目标程序时,编译程序实现如下功能:

·译码

对用源语言书写的各别字符或字符组的预定含义加以确定,并把它们转换为计算机可以接受的指令和数据.

·转换

数字信息从一种数基转换成另一种数基(即从十进制转换成二进制),以及在需要时,从定点表示转换为浮点表示,或者相反.

·选择

可以从程序库中选出所需要的例行程序.

·生成

所需要的例行子程序由各种参数和源程序中指定的程序纲要 (skeletal coding) 生成.

·分配

为各种程序成分分配实际的内存存贮单元.

·装配

把例行子程序(提供的、选取的或生成的)组合成一个程序.

·记录

记录有关目标程序的详细信息,并可以打印.

1.6 COBOL 的模块和级

为了更有效地实现并使用已经设计出来的 COBOL, COBOL 语言已修订成包括九个功能处理模块. 新的结构注重于功能处理的概念,同时,在每个模块中仍保留原来的部分结构(标识、环境、数据和过程). 这些模块是:

·核心

·表处理

·顺序存取

·随机存取

- 排序
- 报表输出
- 分段
- 库

每个功能处理模块分成二个或更多的级. 这些级提供了每一个功能处理模块中的层次. 在所有情况下,该模块中的一个较低级构成较高级的真子集. 这种层次模块结构,在图1-1中加以说明,它使用户能够定制他自己的编译程序,以便得到他的特殊应用所仅仅需要的功能.

功能处理模块							
核心	表处理	顺序存取	随机存取	排序	报表输出	分段	库
2 级	3 级	2 级	2 级	2 级	2 级	2 级	2 级
	2 级		1 级	1 级	1 级	1 级	1 级
1 级	1 级	1 级	空	空	空	空	空

附注: 能够实现的最小 COBOL 系统是核心(1 级)加表处理(1 级)、顺序存取(1 级).

图 1-1 COBOL 语言层级

1.7 本手册的范围

本手册依据 USASI 所定义的语言结构,描述应用于磁带、卡片和打印机文件(通常是顺序地读写的文件)的核心、顺序存取、分段和库等模块(表处理、排序、大容量存贮和随机存取、报表输出以及语汇表,都在本系列的其他手册中加以叙述).

本手册讨论的都面向最高级(第二级),因而提供了最充分的实现 COBOL 语言的手段。

2. 程序组织

2.1 概述

为了解决一个问题,一个源程序必须包括四个基本成分:

- 加工数据的描述
- 建立一组处理数据的过程或操作
- 使用设备的描述
- 标记或标识一个特定应用的信息

本章讨论源程序的这些成分,并且指出它们之间的相互关系. 本章还提供一个应用实例的英语叙述方式的讨论,来说明一个 COBOL 程序的结构.

2.2 描述数据

程序员必须对目标程序所要操作的每一个数据成分加以描述,其中必须包括下列信息:

- 一个名字,程序用该名字来识别一个特定的数据.

• 数据的类型(即它是字母、数值还是字母数值?).

• 以字符个数计算的数据长度.

• 诸如货币符号、逗号、十进小数点之类的特殊符号的位置.

• 这个数据在记录中,和要进行运算的其它数据的相对位置.

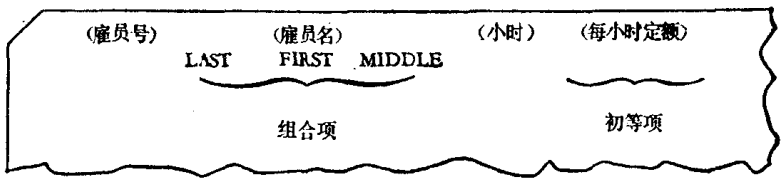
当程序员提供了这些信息,就等于指出在目标程序运行时,将有某个数据成分被加工,并且基于程序员所表示的要求,编译程序将提供存贮空间. 例如,假定程序员希望对雇员的薪金这类数据进行加工. 首先,他应给该数据起一个名字,比如 SALARY. 然后程序员应说明这个域是数值的(如果它包含数字以外,还包含字符,那么它就是字母数值的),说明该域的最大范围,并指出货币符号与十进小数点的位置. 程序员还应说明它与雇员的记录中其它数据的关系. 编译程序根据这些信息,将为该数据留出一个存贮区域,且当目标程序运行时,每个雇员的薪金将被放在这些准备好的单元中,由程序中的指令进行加工.

为了供给这些信息, COBOL 语言提供了特定的规则和过程. 本手册的第三章详细地讨论它们.

下面几节描述 COBOL 程序的数据的逻辑分组和结构.

2.2.1 项与组合项

数据的最小单位是初等项. 这是一种在程序中被调用时不再进一步分为更小单位的数据. 在一个雇员每周工作时间的卡片上,"每小时定额"这个域就是初等项的一个例子;它不再分为更小的单位.



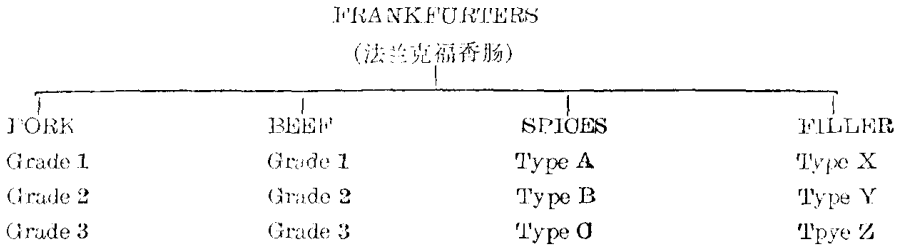
在同一张每周工作时间卡片上,有一个叫"雇员名"(EMPLOYEE-NAME)的域. 但这个数据分成更小的单位 LAST、FIRST、MIDDLE. 在这种情形下,我们不再处理单个数据单位,而是处理整个名叫 EMPLOYEE-NAME 的数据组合项,它的各个成分是初等项.

当构造数据时,也可能有组合项的组合项. 这由下面的例子来说明:

假定 XYZ 肉类公司 (MEAT COMPANY) 持有一张生产法兰克福香肠和腊肠所需各种原料的存货清单. 和这些原料有关的数据可按如下方式构造:

SAUSAGES
(腊肠)

PORK	BEEF	SPICES	FILLER
(猪肉)	(牛肉)	(香料)	(填料)
Grade 1(1 级)	Grade 1(1 级)	Type A(A 类)	Type X(X 类)
Grade 2(2 级)	Grade 2(2 级)	Type B(B 类)	Type Y(Y 类)
Grade 3(3 级)	Grade 3(3 级)	Type C(C 类)	Type Z(Z 类)



GRADE 和 TYPE 都是初等项，并且它们每一个都指定特定原料的一个单个量. 但是 PORK、BEEF、SPICES 和 FILLER 都是包含几个相关项的组合项. SAUSAGES 和 FRANKFURTERS 也都是组合项，但都是组合项的组合项，而不是初等项的组合项.

重复一下，初等项是最小的数据单位，并且不再细分. 组合项是由几个初等项或组合项组合而成的较大的数据单位.

2.2.2 记录

相关的初等项和组合项结合起来形成记录. 在工作时间卡片的例子中，例如每个雇员的那张卡片可以构成单个的或逻辑的记录. 虽然记录的格式保持相同，但是显然各初等项的值将随着雇员的改变而改变. 这对于 XYZ 肉类公司的例子同样成立. 在这个例子中，初等项和组合项的整个系列可以组合成一个叫做 INVENTORY-ON-HAND (现有存货) 的逻辑记录.

需要加工的信息以整个记录的形式读入计算机. 不可以只读入一个记录的某一部分. 同样地，只有整个记录才可以输出；即在某些诸如磁带或穿孔卡片的外部介质上作有效的输出.

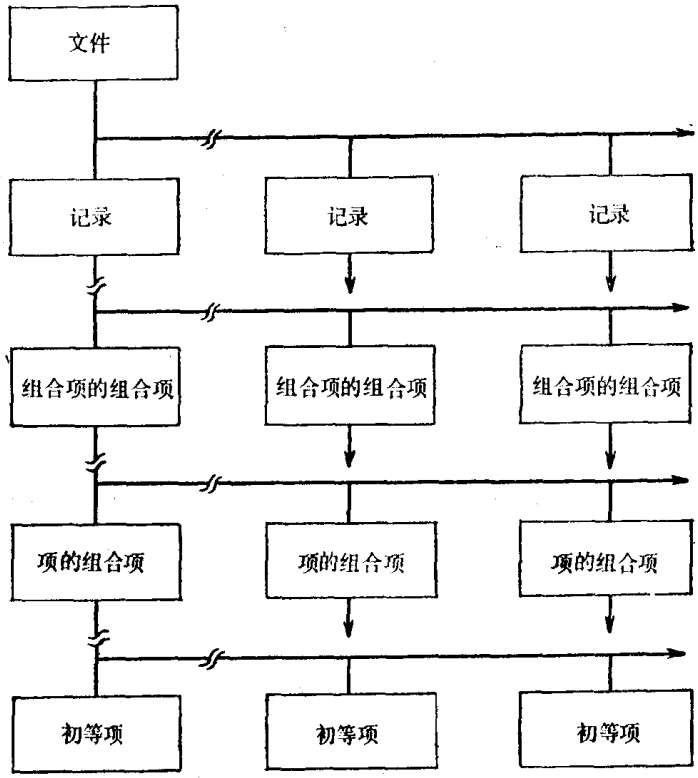
2.2.3 文件

文件是记录的一个集合. 在一个特定的文件中，并不是所有的记录均需具有相同的格式. 当为了加工而读入一个记录时，它就代替了该文件的前一个记录. 在某个时刻，该程序如果需要一个特定的文件包含多于一个记录的话，那么在读入第二个记录之前，必须将第一个记录传送到中间工作区去. 做这件事的精确方法在后面几章讨论.

在 COBOL 中使用"文件"这个字，是为了模仿在任何办公室中使用的文件柜. 例如，每个抽屉相当于一个记录，其中包括了各种类型的有关数据：文件柜也可以用某种方式标记，以指示其内容. 类似地，一个数据处理文件常常有一标号记录，以提供标识. 通常有一个标号记录出现在文件的开头，而在文件的末尾出现另一个标号记录. 这些标号记录用作检验手段，以保证正当的数据被读入，以及指明文件已经结束. 磁带文件的长度和磁带卷之间没有直接的关系. 一个磁带文件可以存放在许多卷磁带上，或者一卷磁带可以存放许多文件. 其它的文件诸如打印文件和卡片文件，有它们自己独特的特性.

2.2.4 小结

下面的图说明了 COBOL 中描述数据的层次结构.



每个文件由一组记录组成. 它们可以是一种类型的记录，或者若干种类型的记录. 每种类型的记录通常被描述成一组初等项；但是，一个记录自身，如果它不再进一步细分，那么就可看作一个初等项.

将两个或多个初等项组合起来通常是方便的. 这些组合项，还可以进一步组合成组合项的组合项，如此等等.

2.3 叙述解题过程

一旦数据已描述好并且组织好，程序员就可专心于为解他的问题设计所需的各种操作或过程，此时，可使用四个主要的功能：

- 数据传送
- 算术运算
- 抉择操作
- 输入/输出

另外，程序员在他的命令中要有控制功能，使他能改变程序运行的顺序，或者引起某些程序段的重复.

在为解答问题而建立各个过程时，程序员不必知道有关他使用的特定计算机的内部细节. 他可以完全致力于求解问题的逻辑.

各种功能起如下的作用：

- 输入/输出功能允许同磁带或穿孔卡片等等外部介质上的信息进行交换，以此获得要处理的特定记录，或者把最后记录交付输出.
- 抉择操作的能力允许数据进行比较，以此决定要执行的几种可能操作中的某一个.