Mastering COBOL

COBOL 从入门到精通

〔美〕 Carol Baroudi 著

邱仲潘 等译

- 当代程序员最清晰、最丰富和最根本的COBOL资源
- 鲜为人知的对付传统代码问题的诀窍

SYBEX

電子工業出版社
Publishing House of Electronics Industry
URL:http://www.phei.com.cn

Mastering™ COBOL

COBOL从入门到精通

〔美〕 Carol Baroudi 著

邱仲潘 等译

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 提 要

面对2000年及传统代码的维护和改进等问题，当代程序员无不感到巨大的压力。本书的出现可以大大缓解这种压力，它为程序员提供了解决这些问题的具体办法和一些从未公开发表的诀窍。

全书分三大部分，共28章，第一部分介绍COBOL语言的背景、要素、结构和扩展功能；第二部分讨论了传统代码和传统系统问题；第三部分讲述了COBOL的现代编程问题，介绍了结构化COBOL，面向对象COBOL，图形用户接口，与其它语言集成，以及使用COBOL 2000等问题。

图书在版编目（CIP）数据

COBOL从入门到精通/（美）巴路弟（Baroudi. C.）著；邱仲潘等译—北京：电子工业出版社，2000.1
书名原文：Mastering COBOL
ISBN 7-5053-5361-6

Ⅰ. C… Ⅱ. ①巴… ②邱… Ⅲ. COBOL语言 Ⅳ. TP312

中国版本图书馆CIP数据核字（1999）第76457号

书　　名：**COBOL从入门到精通**
著 作 者：［美］Carol Baroudi
译　　者：邱仲潘　等
责任编辑：张勤
印 刷 者：北京天竺颖华印刷厂
装 订 者：三河金马印装有限公司
出版发行：电子工业出版社　URL:http://www.phei.com.cn
　　北京市海淀区万寿路173信箱　邮编：100036
　　北京市海淀区翠微东里甲2号　邮编：100036
经　　销：各地新华书店
开　　本：787×1092 1/16　印张：35.75　字数：900千字
版　　次：2000年1月第1版　2000年1月第1次印刷
书　　号：ISBN 7-5053-5361-6 / TP·2688
定　　价：55.00元

版权贸易合同登记号　图字：01-1999-0932

致　　谢

创作这样的一本书是对个人和职业的各方面的挑战。没有众多人的热情帮助，本书是不可能面世的。

首先，感谢Fujitsu Software公司的Ron Langer，他从始至终支持着这本书。

感谢Nick Hubacker对本书的关照，没有他便没有本书。

Maureen Barry、Judith Grimes、Glenn Dent、Ed Arranga和Richard Fulmer尽责尽职地检查了所有软件。

编辑Marilyn Smith将一堆草稿变成了内容连贯、主题明晰的整体。我深深地尊敬和感谢她，她是我见过的最好的编辑。

在本书创作过程中，由于创作的压力而在我的生活中产生了许多变化，婚姻失败、职业改变、两次车祸，感谢诸多最好的朋友的支持，使本书得以继续，这些热忱的朋友是Jonathan Weinert、Eleanor Topping、Debbie Carson、Arnold Reinhold和John Garison。

感谢Waterside产品部门人们的支持，特别要提到Matt Wagner和Maureen Maloney。还要感谢Sybex的各位同仁，组稿编辑Krista Reid-McLaughlin，作稿编辑Emily Wolman，生产协调员Catherine Morris和Blythe Woolston，电子排版专家Nila Nichols、Cyndy Johnsen和Kate Kaminski和图形艺术家Tony Jonick和Ginger Warner。

感谢Don Schricker的鼓励，Artur Reimann的参与，感谢Artur、Danny Miller和Danny Truxaw认真的技术编辑。

译 者 序

写书难，译书也不易。费尽九牛二虎之力终于耕完最后一个标点符号，还是留了许多麻烦给编辑、录入排版人员和校对人员。当这本书终于与你见面时，真是一种深深的缘份。是对同一个主题的兴趣让我们走到了一起，是日新月异的信息技术让我们走到了一起，我很珍惜这种缘份，每下一笔都慎之又慎，但一个个小bug却仍然在我稍一松懈时钻进了笔端，使我每次面对自己的作品时，不由得心跳加快、呼吸紧张。希望读者不吝指教，让我能以更高的警惕性抵御这些bug。

在本书翻译过程中，得到了周阳生、刘文红、邹能东、彭振庆、黄志坚、李耀平、江文清等同志的大力帮助，刘文琼、刘云昌、刘昌和、严明英等同志完成了本书的录入与校对工作，在此深表感谢。

译者

1999年5月

序

COBOL 2000的思考

在COBOL编程语言诞生40周年之际，Baroudi女士的《COBOL从入门到精通》一书出版了。在COBOL推出时，没有人料到，它会有如此长的寿命。这既有好处也有坏处。COBOL、FORTRAN和Algol被称为第三代编程语言（3GL）——它改变了整个计算机行业。

Grace Hopper和其他智者一起，提出了第三代编程语言的概念。COBOL出现之前，计算机应用程序通常是由汇编语言（Autocoder、Easycoder等）写成的。每种汇编语言都针对一种计算机模型。因此，开始编写程序前，要先确定执行该应用程序的具体计算机模型，根本不可能进行平台间移植。Carol Baroudi在本书第1章详细地介绍了COBOL的起源。了解这些背景知识意义重大，因为它显示了植根于COBOL语言之中的信念。COBOL的由来与COBOL的现状具有同样重要的意义。

COBOL和其他第三代语言使我们可以为通用机器开发应用程序。我们可在高层生成应用程序（编写指令），再将高级指令编译成实际机器指令，然后在特定机器上执行这个应用程序。这些听起来很简单，但它需要大量的协调工作。

要为普通计算机编写COBOL程序，目的是要让它在两台机器上执行效果完全一样。这就要求该语言的规则（语法和词法）准确，并使歧义性减到最小。

“建立语言，自有来者”。

建立COBOL和其他第三代编程语言后，计算机厂家即可根据其硬件生成COBOL编译器。并且他们也确实这样做了。

COBOL迅速成为最著名的编程语言，不仅因为它使编程人员能编写通用商用应用程序，而且（也许更重要的是）COBOL语言是自成文档的。COBOL语言的创始人认为（CODASYL COBOL委员会），编程人员是商业人员，而不是工程师或科学家，这些编程人员更熟悉商业，是商业专家而不是计算机专家。COBOL提供了巨大的技术资源，满足了迅速扩展的IT产业需求。大多数其他语言（例如FORTRAN）更多地针对工程与科学社区的编程需求，而COBOL则是寻常百姓的编程语言。正是这一点，使得商业计算机使用激增。

但COBOL也有美中不足之处。由于COBOL生成的应用程序易于阅读和破译，因此也易于维护和管理。在COBOL之前，复杂程序系统推倒重来要比升级更合算。而COBOL的成功则使应用程序能具有更长的寿命。那么COBOL有什么不足呢？

70年代初期，IT预算狂升，难以控制。公司想控制这种IT技术的应用成本，因此寻求减少应用程序开发成本的途径。改进程序维护方式，从而延长商用应用程序的寿命，这是控制成本的关键方案之一。70年代的IT产业采用了两种补救措施。COBOL成为所有商用应用程序的首选。这时发生了一场结构化革命。如果原应用程序用结构化格式设计，则设计应用程序、编写代码和测试系统将更加有效。将程序功能分解成各个模块可以将高维护功能与其他功能分开，从而大大简化系统维护工作。

COBOL的清晰性与结构化设计概念的组合，使旧系统的维护更方便，更经济。但这种成功地尝试也有意外的后果。据行业统计，大多数IT管理员希望系统寿命为6年～11年。谁能料到，由于结构化的COBOL的有效性和易于维护，许多旧的应用程序至今仍在运行，以致于遇上了“千年虫”问题，这些程序是60年代和70年代编写的，当时根本无法预料会有跨世纪的程序。

保持旧程序运行的一个最严重的后果是，程序最初生成时包含了对所在时段和环境的假设及相关的特殊性。“千年虫”问题则是其中最著名的问题。系统设计员和编程人员根本没料到系统会进入本世纪末，从1999年～2000年。有人指责COBOL有“千年虫”问题，的确如此，假设COBOL不是这么好用，不是这么成功，则COBOL已进入历史，不会被人再指责了。COBOL实在太好了，只能怪设计系统和数据库的人没有足够的远见。

Carol的书在应用程序开发史上最重要的时刻推出。该书是在新世纪到来之际COBOL编程人员的重要资源。COBOL应用程序的开发仍在继续，COBOL的光辉依然闪耀。还没有任何其他编程语言能像COBOL一样高度清晰、可维护、与其他编程语言交互并适用于几乎所有操作系统（包括大型机、微机）和数据库管理系统。COBOL几乎是个万能编程语言。谁敢说这些特性在应用程序开发中不重要呢?

谨以此献给COBOL 2000！

Jerome Garfunkel于纽约市

前　言

COBOL编程人员急需解决传统代码的维护和改进，使《COBOL从入门到精通》一书应运而生。在多年的信息系统上，COBOL提供了坚实的基石，却很少引起注意。如今，这个基石经再次修复并重新焕发生机。随着企业中现代技术的渗透，COBOL必定成为新开发工具中必不可少的部分。本书使有关的编程人员可以探索传统系统，将企业带入新世纪。

美国当前的传统代码超过5000亿行。随着2000年的到来，有关人士估计，美国需要50万～70万个熟练的大型机编程人员（加上现有的50万）才能解决美国目前面临的编程危机。而英国则需要30万个大型机编程人员。这些编程人员仅仅利用标准语言手册是无法解决工作中面临的问题的。

本书的合作者都深爱COBOL，这种编程语言在当前的软件开发人员培训中倍受歧视。我们从人们谈论的COBOL笑话体会到COBOL问题的严重性。故事是这样的：

> 一个COBOL编程人员在遇到2000年问题时，决定把自己速冻起来，到下个世纪再唤醒。由于机构调整，他的身体被忽略了很长时间。等他终于苏醒时，他见到的是急切、热情而陌生的人群。由于找不到熟人和亲人，他便问，“我是谁？今夕何年？我怎么会在这里？”他们说，“你好，这里是华盛顿，9999年，我们知道你了解COBOL，总算等到你了。”

本书希望弥补这个空白，让天才的编程人员学习COBOL、传统代码环境及其现代应用。

本书的读者对象

本书适用于需要维护传统代码、提供2000年方案或开发现代COBOL应用程序的编程人员。与其他有关基础COBOL编程的图书不同的是，本书着重介绍修改现有代码中文档资料未说明的问题。

对于学过其他语言的编程人员，本书简化了COBOL开发环境的简述。从其他软件开发环境向COBOL开发环境的过渡，通常是比较困难的。对于COBOL编程有经验的用户，本书提供了该语言的新东西，介绍了当前软件开发所需的概念，如面向对象编程和图形用户接口（GUI）开发。

本书不是编程者的入门教材，阅读本书，需要读者有COBOL或其他语言的编程经验。即使是不熟悉COBOL和传统环境，但最好也应编写过或修改过程序。

本书内容

本书由三个部分和四个附录组成。第一部分介绍COBOL语言的细节，我们用1985修订的COBOL标准作为本书的基础，大多数传统代码都是基于这个标准。第1章介绍COBOL的背景及其在商业环境中的重要性。第2～11章介绍COBOL程序的元素和COBOL语言的结构。第一部分最后的第12章介绍各种COBOL版本中提供的COBOL扩展功能。

第二部分介绍COBOL传统代码。第13章说明，为什么维护传统代码是重要的问题，并介绍传统系统的常见软硬件环境。第14～16章详细介绍四个主要的编程领域：CICS（客户信息控制系统）、SQL（结构化查询语言）、IMS（信息管理系统）和JCL（作业控制语言）系统。第17章和第18章介绍传统系统的编译、连接、测试和调试；第19章介绍2000年问题的方案；第20章介绍如何处理欧元问题，使程序能支持欧元。第21章介绍处理常见COBOL程序数据挑战的准则。

第三部分介绍COBOL的现代编程问题。其中，第22章介绍结构化的COBOL编程；第23章介绍COBOL的下一个标准（COBOL 2000）；第24章介绍COBOL面向对象编程；如果要建立COBOL与其他语言之间的通信，则第25章提供了内部数据类型表示的细节；第26章提供了将COBOL应用程序开发从大型机转移到PC环境的方法；第27章介绍了如何建立COBOL程序的图形用户接口，以及使用本书光盘所带软件的练习。本部分的最后一章，第28章介绍COBOL与C、COBOL与JAVA的混合语言编程。

本书的附录提供了其他材料和参考资料：

- 附录A介绍未来的64位COBOL。
- 附录B介绍COBOL保留字的快速参考。
- 附录C介绍ASCII和EBCDIC字符集。
- 附录D介绍本书选配光盘中的Fujitsu COBOL4。

本书中分布着许多提示、说明和警告，它们在文中以特殊的楷体示出，其中包含了一些有价值的补充信息。

本书选配光盘

本书选配光盘包含Fujitsu COBOL4的Windows版编译器。这个一流的编译器包括了所有COBOL 2000的特性，如面向对象的COBOL语言元素。使用Fujitsu编译器可以编写新代码或测试本书中的样本代码。

关于安装并使用Fujitsu COBOL4的信息，见附录D。本书的第27章还介绍了用Fujitsu COBOL4所带的组件PowerCOBOL建立GUI应用程序和自定义控件的步骤。

本书的约定

本书的程序和代码采用等宽字体。COBOL语言语法采用直接注释系统，直接注释系统包括如下元素：

- 引用格式中的大写字母是COBOL单字，在COBOL中有特定含义，只能在指定上下文中使用。保留字加下划线时，是格式中所必需的，不加下划线时则是可选的，对元素的含义没有影响，可加可不加。加上可选保留字通常能使语句更易读（附录B列出了COBOL保留字）。
- 小写字母泛指非标准环境定义的COBOL元素，所选名称用于表示其含义。在说明COBOL项目时，小写字母后面通常加上一个整数（例如identifier-1）。整数表示该元素出现的次序。
- 中括号（[ ]）中的部分是元素中的可选部分。如果中括号内有竖条，则可以选择其中一项内容。

- 大括号（{}）中的部分是必需的元素。如果大括号内有竖条，则可以选择其中一项内容。
- 选择标识符（|）包括一组元素，其中要有一个或几个元素，但任何元素都不能重复。
- 省略号（…）表示格式中前面的元素可以重复。前面的元素可以是一个元素（小写字母单词引用的内容）或放在大括号、中括号中的整组元素。
- 格式中出现的特殊字符（+ - > < = > = < =）是必需的，即使不加下划线，也具有COBOL单字的状态.
- 标点符号（逗号和分号）放在空格后面，可以用在格式中使用空格处，对格式没有影响。标点符号（句号）放在空格后面具有必需字的状态。

第2章将详细介绍语法规则，包括例子和样本程序。

本书作者

本书是集体创作的的结果，是Internet、电话和人类的共同成果。我们到处寻找COBOL各个方面的专家，因而得到了各方面的建议，使各位专家的高见尽现本书之中。关于各位专家及其贡献，见本书最后的“关于贡献者”。

2000年以后

据美国政府备忘录称，有些政府机构到2012年还会抱怨2000年问题。我们见到了这份备忘录，但没有验证其来源，因此不知是否确有其事。

我们相信，COBOL和COBOL编程人员的需求不会很快消失。在遇到2000年问题前，我们尚未发现COBOL的缺陷。但我们知道，2000年问题是巨大的，其影响也是深远的。掌握COBOL，会帮助读者长期立于不败之地。特别地，如果能掌握COBOL并用于企业的实际应用中。我们相信，本书能提供读者所需要的一切。

欢迎指教，请发e-mail到cobol@baroudi.com，可以查阅本书在Sybex Web站点www.sybex.com上的页面，也可以查阅Carol Baroudi的Web站点www.baroudi.com，其中有本书的更新信息及各位贡献者的详细信息，包括e-mail地址和站点链接。

第一部分 COBOL语言

第1章 COBOL简介

- COBOL的目的
- COBOL版本与可移植性
- COBOL的演变
- COBOL标准发展
- 当前COBOL应用

本章要介绍COBOL并浏览本书其他部分的内容。COBOL不仅是一种编程语言，而且是地球上几乎每个国家的公用或专用基础结构中的集成组件。当今世界中很少有人的生活没有受到COBOL程序的影响。这些程序服务于政府、银行、运输系统、制造业及批发系统的日常工作中。COBOL并不引人注目，但它却很强大，它是信息系统的砖和瓦。

COBOL及其重要性

COBOL是Common Business-Oriented Language（公用面向商业的语言）的缩写。COBOL自60年代初开始就广泛应用于计算机应用领域（商业和其他领域）。COBOL是采用英语语法的高级语言，以其可读性、可维护性和可移植性受到商业单位和政府部门的青睐。COBOL不断地演变并吸收计算机技术的进展。与许多当代编程语言不同的是，COBOL针对商业世界的使用，它是真正商用应用程序开发的首选语言。如今，COBOL是企业的解决方案。

如果想了解核心商业的大型信息系统或开发新的商用应用程序，则必须知道COBOL。好在COBOL很容易理解、很直观也很容易编写。此外，COBOL丰富多彩、富于挑战，可以从最复杂的系统中遇到，让人百看不厌。

标准、版本与可移植性

如果只有很少的计算环境中编程经验，则不会了解将代码从一种环境移植到另一种环境时具有的问题，不会知道语言标准、特定厂家版本和移植性之类的问题。但如果要掌握COBOL，就必须对这些问题有所了解。

随着时间的推移，公用应用领域中的技术发展需要标准化才能有效地使用。计算机软件和编程语言更是如此。

COBOL语言的定义是科技人员几十年工作的结果，他们致力于制定在特定平台上实现COBOL时可用的标准。标准委员会由各有关单位的代表、各个COBOL编译器厂家的代表和用户社区的成员组成。

当某个组织实现COBOL标准时，用该组织的COBOL编写而成的程序可以和任何其他采用同一标准的组织的COBOL编写而成的程序一样工作。只要代码元素严格按照标准，代码就可以在不同环境间移植。

尽管COBOL标准对语法作了严格定义，但早期版本通常并未考虑特定的平台特性和运行环境。后来的版本解决了这个问题，例如，它们考虑了不同的数据格式，如压缩十进制格式和二进制格式。这样，标准随计算机行业软硬件的改变而不断地演变。

在实际中，大多数COBOL厂家在实现标准之后都进一步提供扩展特性，以吸引开发人员。有时，这些特性被证明为非常有用，并被吸收到下一版标准中。记住，非标准特性的版本是没有统一定义的，移植到另一环境时不一定可行。

本书第一部分几乎全部介绍当前COBOL标准语言定义，即修订COBOL-85标准。我们希望使其中介绍的COBOL基础和原理尽量一般化和平台独立。但实际上COBOL有许多版本，针对不同机器和操作系统。具体演示采用大型机上的IBM COBOL和PC机上的Fujitsu与Micro Focus COBOL。

第三部分将介绍下一版标准准备加入的特性，包括面向对象编程元素、可移植的标准算法和其他开发新应用程序的工具。我们将包括这些特性的新标准称为COBOL 2000，因为标准委员会希望在2000年推出这个新标准。

COBOL简史

要了解COBOL的全部影响，就要知道，COBOL是专业人员为普通程序员设计的。这种语言不是针对某类机器的，各类机器上的用户都能随心所欲地工作。专家小组在过去的40年里建立、维护并扩展了这个充满活力的语言。而行业领头羊的投资、支持和指导也是COBOL取得成功的重要原因。

早期COBOL

50年代有一种潮流，就是建立专门解决商业问题的编程语言。这种潮流与许多编程人员和计算机权威的观点相反，该潮流的推崇者认为机器语言不是计算机唯一理解的语言，而是适合各种任务的最佳语言。这种潮流的成功改变了编程人员和计算机权威的观点。

成功的商业编程语言范例是1954～1958年之间Sperry-Rand的开发并修订的FLOW-MATIC。许多大公司和美国空军都采用了FLOW-MATIC。FLOW-MATIC易于编写程序和调试，因为它采用了英语式语法（如ADD和MOVE），数据名可以较长和带含义（如STATE-TAXES和TOTAL-PAY）。

1959年，高级语言的潜力得到充分证明，计算机用户和学术机构与制造商联合定义了商业化商业语言的目标。当年四月在宾西法尼亚大学召开了一次会议，提出要有一种所有计

算机上均能使用的通用商业语言。这个小组向美国国防部寻求资助，得到了国防部的同意。

说明： 有趣的是，人们经常以为美国国防部支持COBOL作为军队标准化计算的途径。虽然这不完全属实，但五角大楼的确很快发现了这个工作的意义，并投入了大量的精力。

1959年5年，五角大楼召集了一次会议，暂时把他们的项目称为公用商用语言（CBL）。小组确定这种语言的主要要求如下：简单英语动词，易用性优先于功能，数据与过程相分离。这样，COBOL的语法中就借用了段、句、词的概念。也许COBOL程序并不像创始人预料的那样可读（显然比现代的编程语言更冗长），但这种把“神秘和高深”赶出程序的做法值得称赞。

COBOL创始人致力于扩大潜在的编程队伍，使新程序的开发要求商业知识而不要求对特定计算机的过细了解。为此，小组认为这种语言应能在多种计算机系统间移植。

这样就形成了三个委员会：分别针对该语言的短期、中期和长期开发。实际上，短期委员会理由充足地提出了这种语言，根本毋庸置疑。委员会成员指派一个管理小组监视这种语言的开发，这个小组就是著名的CODASYL。

到1959年夏天，这个委员会主要根据FLOW-MATIC词法及其三个部分（过程、数据描述和环境）开发了一种语言。这些部分现已成为COBOL的四个部。这个委员会还借用了IBM的COMMERCIAL TRANSLATOR，特别是其PICTURE从句和组项目（组成01、02等表示的层）。这个委员会将该语言命名为COBOL，表示公用面向商业语言。

1959年12月，第一个COBOL规范的最后草案完成。尽管从此以后COBOL在标准化过程中不断修订，但当时的许多技术决策一直影响至今。

作为委员会建立和维护的语言，COBOL的演变主要通过争论达成，从而形成某种折衷。例如，构造数字公式时是否使用符号或词的争论就产生了用COMPUTE作为表示等式的更加符号化方式，并认为ADD、SUBTRACT、MULTIPLY和DIVIDE更易读。

将“折衷语言”与不可移植的专用语言相比，付出的代价是值得的——COBOL得以迅速普及。1960年，美国国防部宣布，所有购买的计算机都要包括COBOL编译器，私营公司迅速跟随其后。

尽管COBOL最初只要求这个标准能维持几年，但COBOL一直是大型机上最普及的语言，其用途还在其他平台上不断扩展。COBOL如此普及的原因之一是，国防部要求用COBOL，避免多种语言的争夺。国防部的参与使他们可以要求COBOL包括商业数据处理最有用的元素。COBOL普及的另一个原因是它要求在所有计算机上产生一致性结果。这个特性通过了政府接收测试的验证，使公司和政府有信心在COBOL中大量投资。

COBOL的硬件无关的特性在该语言规范的演变中起着巨大的推动作用。这个规范强调标准输入/输出方法，因为这些操作是商业计算机的骨干。由于输入/输出设备的特性和计算机技术的日新月异，COBOL需要相应扩展和改变。COBOL自1960年首次出现以来，进行了许多次修改。大多数修改都是通过严格标准化过程的论坛实现的。

COBOL标准的发展

1960年，美国计算机与商业设备制造商协会（CBEMA）成立了一个委员会，叫做美国计算机与信息处理国家标准委员会，简称X3。X3委员会的分会X3.4编程语言分会建立了X3.4.4工作组“处理器规范与COBOL标准小组”。

X3.4.4（后更名为X3J4）小组负责建立COBOL标准。这个小组的第一次会议于1963年召开，由计算机制造商和用户的代表参加。他们确定了小组的目标是根据COBOL的CODASYL标准（1959年该委员会产生的最初COBOL标准）定义COBOL国家标准。

1968年，美国标准协会（USASI）批准了COBOL分会开发的标准，发表号为X3.23-1968。这个文档定义了COBOL包括内核（Nucleus）和下列八个功能模块：

- Table Handling（表格处理）
- Sequential Access（顺序访问）
- Random Access（随机访问）
- Random Processing（随机处理）
- Sort（排序）
- Report Writer（报表写入）
- Segmentation（分段）
- Library（库）

每个模块分成最多三层，高层提供更多功能，低层提供高层的子集。COBOL的基本版本应包括内核、表格处理和顺序访问模块的低层功能。完全版本的COBOL应包括所有模块的高层功能。美国标准协会（USASI）于1966年由ASI更名为USASI，1969年更名为美国国家标准协会（ANSI）。这个ANSI标准即著名的COBOL-68。

1974年对标准进行了修订，8个功能处理模块扩充为11个：

- Table Handling（表格处理）
- Sequential I/O（顺序I/O）
- Relative I/O（相对I/O）
- Indexed I/O（索引I/O）
- Sort-Merge（排序/合并）
- Report Writer（报表写入）
- Segmentation（分段）
- Library（库）
- Debug（调试）
- Inter-Program Communication（程序间通信）
- Communication（通信）

每个模块包括两层或三层。在9个模块中，最低层为空集。每个低层都是高层的子集。对于COBOL-68，基本版本应包括内核、表格处理和顺序I/O模块的最低层。

ANSI COBOL的最新版本于1985年发布，本书采用了这个版本（我们将注明COBOL-85对COBOL-74作出补充和改变的地方）。1985模块与COBOL-74中相同，只是没有单独的表格处理模块，这个功能加进了内核模块。下一个标准COBOL 2000将消除模块和层的概念。

CODASYL COBOL委员会（CCC）长时间与X3J4并列存在。CCC负责所有语言发展，从而产生各种版本的“发展月刊”（JOD）。X3J4根据JOD定义标准。这种运行方式一直持续到COBOL-85标准完成。之后两个委员会最终合二为一，X3J4最近更名为J4，负责处理COBOL语言的发展标准化。

尽管COBOL是由美国的小组用英语作为语法基础进行定义和标准化的，但商业数据处理的国际社区对这种语言及其在其他环境中的使用兴趣浓厚。ANSI COBOL委员会与国际组织密切合作，建立的COBOL-68标准符合COBOL的ISO（国际标准化组织）推荐标准。同样，后来的两个ANSI COBOL标准COBOL-74和COBOL-85也被ISO批准和采用。国际工作小组从ISO角度协调并控制这个发展过程。

COBOL与结构化编程

50年代，编程人员需要很高的编程技巧。GO TO语句是汇编跳转指令的改进，在当时使用非常普及。但编程人员要维护、修改和改进这种程序相当困难。

说明：GO TO语句通常跳到程序代码中另一部分。这种跳转很难跟踪，让人看不出代码中如何转移到某个分支或转移后要执行哪块代码。

尽管编程人员可以使用流程图和编写GO TO语句，但这个方法很快便显示出弱点。人们发现，有些程序的平均寿命通常在七年以上。事实上，有些系统运行更长时间，可以工作30多年。编程人员必须设法生成更可靠、更易维护的系统。这种可维护性变得比软件创作的具体技巧更重要。

由于对软件期望的改变，60年代，人们提出了聚合软件结构理论，从而导致了模块化（或结构化）编程的运动。一开始，人们强调得更多的是模块内代码的形式。后来，开发人员认识到，系统的总体模块布局更加重要。

所有程序并非生而平等。分析问题的方法有好有坏。许多人开始寻找好的模块结构应有的特征。而功能分解（即把程序分解成最小功能）产生了整洁利索的模块。

结构化编程揭示了程序设计有好有坏，从而引出了评估模块设计的两大问题：

- 模块是否聚合，模块内部是否很好地相关联？
- 模块是否耦合，参数与模块功能是否相关？

COBOL ANSI标准的改变反映了结构化编程的潮流。GO TO语句最终在ANSI COBOL-85标准中从常用COBOL编程做法中消失。这个标准引入了范围限制符、Case结构（动词EVALUATE）和PERFORM语句。1989年补充指定了内部函数。

当代软件开发不仅要求可靠、代码可复用，而且使可复用代码成为新系统和重新设计传统系统的关键。面向对象编程的思想将在第三部分介绍，它是开发可复用代码和图形用户接口（GUI）的方法。新的COBOL标准预计在2000年推出，新版标准将提供面向对象编程结构和其他现代编程结构。

COBOL工作环境

COBOL语言经历了40年的风雨沉浮，已经存在近千亿条COBOL语句。所以一定要了解COBOL操作的环境。

从发展看，COBOL应考虑其创建的编程环境。出现个人计算机之前，各大硬件制造商（例如DEC、EDG、Unisys和IBM等公司）都有某种COBOL系统。

多年来，IBM一直是计算机行业先驱。IBM成功地在全世界100多个国家销售其大型计算

机。只要有IBM机器的地方，就有COBOL。IBM COBOL应用程序在政府和公司系统中随处可见。在IBM环境中，系统围绕COBOL成长、与COBOL混用、直到与COBOL密不可分。

如今，到IBM商店购买COBOL应用程序时，你会发现，它与JCL、CICS、IMS、SQL等混在一起。由于许多环境中必须了解这些组件才能使用COBOL，本书第二部分将介绍这些组件。

在计算机科学与软件工程世界中，COBOL备受指责。有些人要用户单纯地使用Unix和Dos命令。有些没有用过COBOL的人也对COBOL持很坏的印象。但COBOL是个无所不在的商业开发语言，其现代版本能解决当前商业问题。事实上，一流COBOL开发人员正在编写使用COBOL的Web应用程序。本书第三部分将介绍如何开发新的COBOL应用程序。

第2章 COBOL程序概述

- 一个简单的COBOL程序
- COBOL的语法说明
- COBOL单字与直接数
- 源程序格式
- COPY与REPLACE语句

本章首先概述COBOL程序的结构和元素，还介绍COBOL语言语法和源程序格式，最后再介绍如何用COPY和REPLACE语句在COBOL程序中插入文本。

要了解这些概念，我们首先介绍一个简单的COBOL程序，体会一下这种语言的总体结构。

一个简单的COBOL程序

COBOL程序与其他语言写成的程序不同。一旦熟悉了COBOL程序的结构，你就会发现，COBOL程序比其他类型的程序更易读。当然，具体的程序可读性取决于编程人员对程序中使用的各个元素名称的选择。但COBOL的基本结构和性质有利于人们编写简单易读的程序。

要介绍COBOL程序的结构，可以从一个简单例子开始。清单2.1是计算贷款每月偿还额的COBOL程序。图2.1是运行清单2.1计算汽车贷款的结果。

清单2.1 MORTGAGE程序

```
000010 IDENTIFICATION DIVISION.
000020  PROGRAM-ID.  MORTGAGE.
000030  AUTHOR.  Mastering-COBOL.
000040*
000050 ENVIRONMENT DIVISION.
000060*
000070 DATA DIVISION.
000080 WORKING-STORAGE SECTION.
000090 77 AMOUNT         PICTURE 9(8)V99.
000100 77 INTEREST       PICTURE 999V999.
000110 77 MONTHS         PICTURE 9(5).
000120 77 M-INTEREST     PICTURE 999V9999.
000130 77 PAYMENT        PICTURE ZZZ,ZZ9.99.
000140*
000150 PROCEDURE DIVISION.
000160*
```