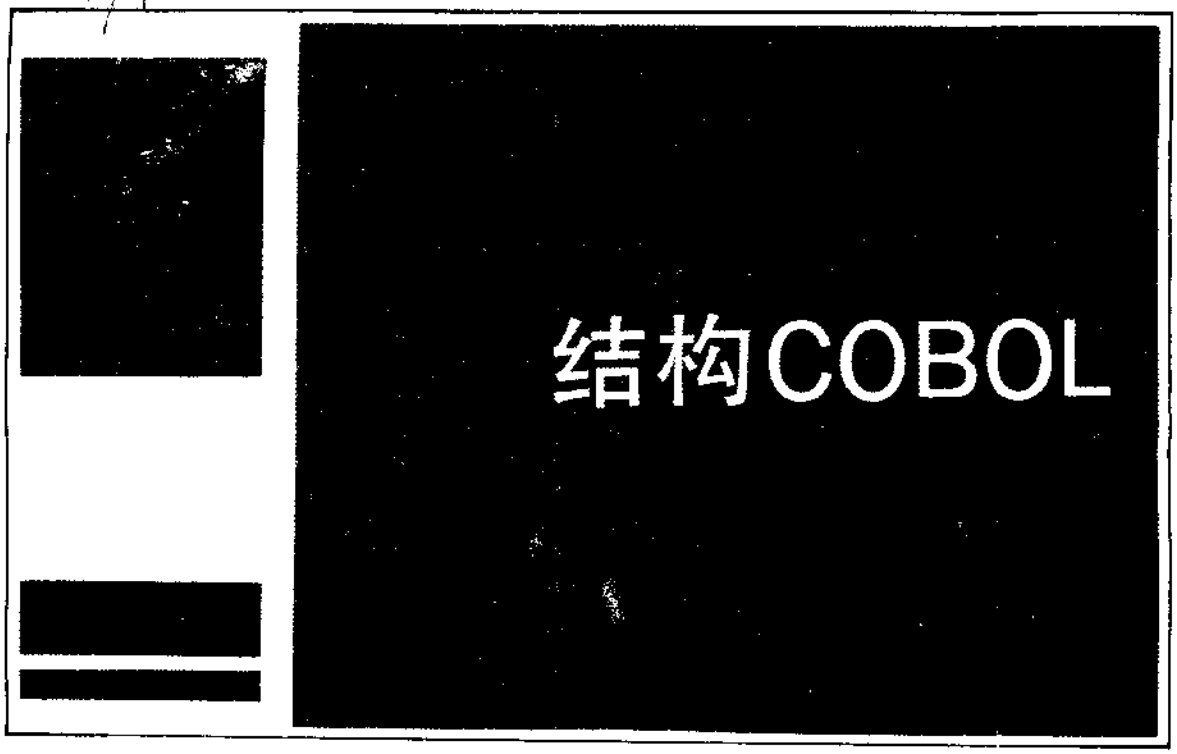
结构 COBOL



H.密莱希 著

张 靖 冯博琴 译
施宏宝 校

上海翻译出版公司

Structured COBOL: A Modern Approach
By Henry Mullish
Harper & Row, Publishers, Inc., N. Y., U. S. A., 198?

结构COBOL
张 靖 冯博琴 编译
上海翻译出版公司
（上海复兴中路597号）
新华书店上海发行所发行 浙江新华印刷厂印刷
开本 787×1092 1/16 印张24 字数58.4千字
1988年2月第1版 1988年2月第1次印刷
印数 1—6000
ISBN 7-80514-007-3
O·49
定价：8.10元

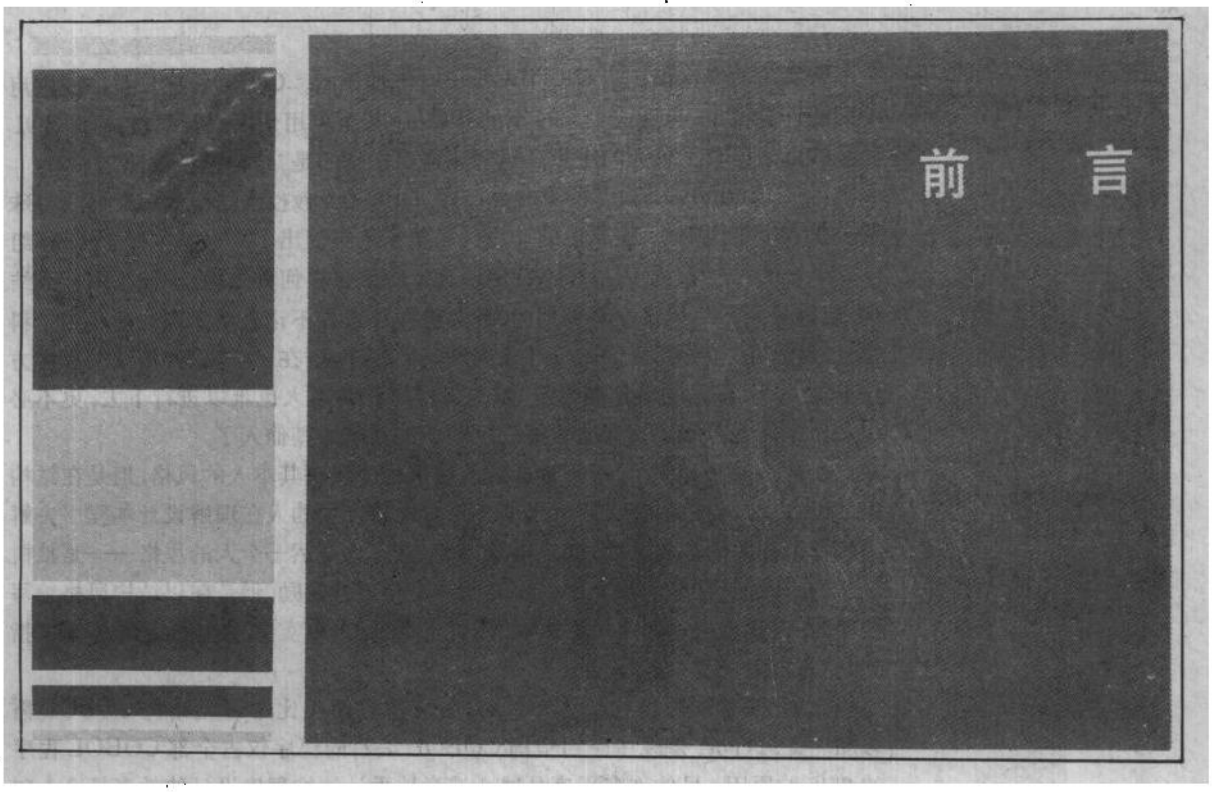
内 容 提 要

本书叙述的是1974年COBOL标准，并阐述了它与1968年标准之间的主要区别。全书自始至终以结构程序设计的风格讲述COBOL，分布在各章中共有这样完整的程序实例52个。

为了有助于读者更好地吸收这些材料，章前指出重点，每章结束有一定量的习题，答案附在书末。

本书可供企业负责人，从事数据处理的科技人员，管理、财会干部，有关大专院校师生学习和参考。

民威同志参加了部分章节的初译工作。



前　言

COBOL 是不属于任何公司或公司集团的，是任何组织或集团组织的一种工业语言。

本书是以美国国家标准协会建议的 1974 年 COBOL 标准为基础的。本标准已为美国政府采用，并为国际标准组织一致同意批准采用，世界各国标准协会联合会成员国如下：

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

显然，已获得国际公认。1974 年标准实际上是世界上还在广泛应用着的 1968 年标准的修正，由于两个标准所述材料相互有关，故本书还试图阐明两个标准之间的主要区别。

自从 1960 年 COBOL 首次设计和制定以来，已有了系统的发展和改进，虽然现今 COBOL 的一般版本和 1960 年的版本相同，但为了丰富起见，对语言已作了许多补充，因而使它作为商业和数字处理界的主要计算程序语言显得十分有效。

所发生的变化仅在于对 COBOL 语言不加限制。COBOL 已日益广泛应用于别的计算机上，特别应用于较小的计算机(甚至应用于家庭计算机)，COBOL 程序的风格已有了较大的进步。这种风格通常指的是"结构程序设计"。

荷兰 Eindhoven 大学的 Edsger. W. Dijkstra 教授因结构程序设计获得殊荣。在六十年代中期，他和他的计算机科学家小组提出了一个基本概念：怎样组成一个好的程序(不管应用什么特定语言)，应该采用何种结构，不该采用什么结构，应该采用什么风格。直到那时，计算机程序设计不论是用于商业还是用于科学上，还处在"按你愿意的做"，并且判断一个程序是"在工作吗"？传统的程序方法非常复杂，它的逻辑含糊不清，以致使程序作者本人也难以进行下去，更不必说从程序作好以来负责修改程序以反映新的需求的其他人了。

在通常的研究中，程序设计者编的通用程序反映其本人的风格，但是在结构程序设计中，风格更统一，以致于使一个对程序陌生的人在理解设计本程序去解决的那个问题的本质之前，知道所期待的是什么。虽然一个人的风格——是独树一帜的，特别是在艺术和文学创作方面经常会得到鼓励，但是在以结构风格编写程序时，自我表现对程序风格来说，应置于第二位。事实上，结构设计程序通常指的是"非自我表现"程序设计。

无疑，开始学习结构 COBOL 时，在细节方面应比学习无结构 COBOL 需多加注意，这也许是现在应用结构 COBOL 编写的程序仅占全部 COBOL 程序的 25%的原因。虽然许多程序员诚心诚意地承认结构程序设计的效率是令人向往的，但他们还是用自己已经学过的比较熟悉的风格去编写程序，坏习惯是很难改的，所以这些程序员对结构设计只是在口头上赞颂而已，问题就成为"要按我所说的那样做，别按我所做的那样做"。

但是，遵照结构程序设计的规则采取训练，最终会有收效的，这种程序容易编写而且是正确的。在程序设计后投入运行时，大多数程序员可以毫无困难地修改它们。

刚才讨论了程序设计风格的优点，因为可能你在以前还没有涉及到常规的风格或是结构风格的论题。你首先应逐渐熟悉些一般的概念，尔后学一点 COBOL 语言。这样，你就能编写出优美的 COBOL 结构化程序，这包含在第四章中。

COBOL 是 Common Business Oriented Language(面向商业通用语言)的字首组合词。COBOL 是一个仔细计划、反复思考和集体努力的成果。这种努力得到美国政府各部门、商业代表和计算机制造商的承认。1950 年后期他们曾共同努力设计过一种专门从事商业、数据处理和特别财政问题的语言，COBOL 作为一种通用语言是因为随着时间的流逝，它已成为今天使用的重要语言，最终可能成为商业上使用的唯一语言。现在世界上在已编好的程序中，据统计有 80%是用 COBOL 编写的。COBOL 在商业部门中已很通用。由于它的普遍和通用，商业界已逐渐依靠 COBOL 来处理它的日益膨胀的工作量，故而掌握该语言已日趋重要，并且最近对所谓的结构 COBOL 的需要已增加到使如此多的教学部门都转到这个更优美的程序设计风格方面来了。

在商业程序设计界，COBOL 的空前流行已导致一些比较开通的公司主动鼓励用结构风格编写程序，更重要的是，主动公开宣布不雇用用习惯的"按你愿

意做的"风格编写程序的程序员。

本书说明的大多数程序都已在应用 1974 编译程序的 CYBER 170/120 计算机上运行过。但已有足够的材料包括在 1968 的编译程序中，以便使仅仅能使用这种设备的读者可以运行所编的程序。

从实际课堂试验中已清楚地显示了本书在二年制或四年制的学院中使用所具有的优越性，特别是，如果 COBOL 是学生所接触的第一个计算机语言。大量选择题的练习使教师布置家庭作业比较容易。应指出，近年来许多高中已决定把 COBOL 教学包括在它们新开展的计算机科学课程中，看来，这种势头在未来几年内将继续下去，因为 COBOL 无疑地是一种最普通的语言。

没有一本书能包罗万象地把 COBOL 说清楚。本书为了更清楚地描述 COBOL 程序设计是什么，提出了足够的材料，并能使读者为今后和在深度上研究这项主题提供了基础。ANSI(美国国家标准协会)出版的手册和各制造商给出了有关语言特殊转换的细节，但是它们的格式化表示妨碍了它们成功地成为一个供人们可以藉以学习语言的初步知识的工具。遗憾的是，不是所有的制造商都遵守通用的标准，因此在一台计算机上工作的 COBOL 程序往往不得不作某些修改，而在另一台计算机上工作(即使是一台较小的机器)。与其说用了很大的努力使 COBOL 成为面向问题的语言，不如说是面向计算机的语言，自然，COBOL 所面向的是工业、保险公司、国家和一般商业方面的问题。

在你看这本书的过程中，可很清楚地看出它严格地按以下的原则处理：

1. 没有假设计算机、程序设计语言或企业概念的预备知识。

2. 正文自觉地避免了使用不必要的术语，合适地讨论了有关问题，提出了恰当的术语。

3. 采用的书写风格是日常使用的、非强制性的和"使读者欢迎的"。

4. 早在第一章的中间就向读者介绍了实际书写 COBOL 程序。

5. 给出的所有 52 个程序在各方面都是完整无缺的，打印了它们的全部程序和清单，显示并注释了程序运行时使用的实际数据，给出了全部输出。

6. 为了使读者事先可以预习，在讨论前先指出每章的重点。

7. 每章结束有一些复习题，书末有答案，一般都比较详细，以便读者能更好地吸收这些材料。

8. 每章中的练习是以本章的材料为基础。

9. 本书包括了 COBOL 的正规导论课，加上为高级课程准备的更为广泛的材料，包括象表格处理那样的问题，它是在有相当深度进行处理的人以及许多书中似乎完全忽视的有关报告打字机有争议的课题。另外还给出一些子程序的调用。试验表明，如果不超过在工业上服务的 COBOL 程序员的要求，掌握本书所包含的材料是足够的。

10. 所有的程序都具有独到之处，有许多可直接用于商业，如工作日程序和已消逝时间程序。

11. 本书自始至终贯彻了严格的结构程序设计原则。希望初学的读者能以此作为自己的风格。

鸣　谢

深切感谢 Illinois 州 Skokie 我的出色的学生——Irvin Poremba,他自愿把他的超群的天才无私地贡献给本书的编写。不论程序要用这样或那样的方法进一步改善,他总是提供建议,打好程序并且修改它们。他的想法和建议是那样的好,因此其中多数都能实现。同时还要感谢另一位出色的学生 Susan Chan,他在百忙之中无私地审阅了手稿并提出了新建议。最后还要感谢纽约大学的所有学生,他们用种种方式对本书最后定稿给予了帮助。

特别感谢 COBOL 的前辈 Grace Murry Hopper 博士,他做了 UNIVAC'S 上的第一个 COBOL 编译程序的工作,他始终不渝地通过对 COBOL 语言的各种演变进化进行了指导,帮助它成长。

欢迎到 COBOL 世界来!

HENRY MULLISH

目　录

第3章 介绍文件

第4章 重要的指令

第5章 准备一个业务程序

第6章 数据的输入、打印和编辑

第7章 基本程序

第8章 商业类型问题

第9章 使用附加技术的程序

第10章 表

基本 COBOL 程序设计概念

在本章中，我们要研究下列重要的概念：

- 计算机程序
- CRT
- 计算机的速度
- 计算机盲目服从程序指令
- 作为一种高级语言的 COBOL
- 为什么 COBOL 程序有时看上去冗长
- 二进制语言
- 编译程序
- 诊断信息
- 逻辑错误
- 源卡片
- 目标卡片
- 穿孔卡片和它的起源
- Herman Hollerith 博士
- 字符编码
- 一个 COBOL 程序的结构
- 列 7 所起的作用
- 标识部
- 区域 A 和区域 B
- PROGRAM-ID(程序标识)名
- 环境部
- 数据部
- 过程部
- 贯串程序的控制流程图
- 显示语句
- COBOL 程序的层次特性

1.1 COBOL 编译程序

每天早上我醒来
高兴地大声呼唤
COBOL 是一种语言
象是特地为我而制成!

解决一个重要问题的计算机指令序列被称为程序。在 COBOL 程序情况下，指令可由一个键控穿孔机打孔在一个卡片上。直接将指令输入到有时被称为 CRT(示波管)有时被称为 VDT(电视显示终端)的显视终端上，已变得越来越普及(和便宜)。从我们的观点来看，这两种方法是一样的，因为两者都传送打孔信号或打印信息到计算机，一旦计算机内指令被"执行"，也就是说，在很大程度上和你读这本书的方法一样，是一行接一行的。当然，主要区别在于计算机以罕见的高速度操作每个连贯的指令。实际上，大量地夸扬计算机的本领，恰好起源于这内在的特征。它毫无拘束地全速运行遍每个有顺序的指令，一点也不具备最微小的比如为什么这么做的理解力，也不了解它的运算结果的任何部分，计算机全然不知道它运行的结果，没有想象力、情绪、判断、同情和赞成，也没有感知。由于它没有灵魂和人类的敏感性，它仅能做那些借助于程序命令它所做的事，以惊人的速度和准确无误的精度执行。事实上，有关计算机最令人失望的事之一就是它只能永远准确无误地去做它被命令去做的事，而不能去做你想让它去做的事。一旦执行计算机程序，它就单一地按全部指令的顺序执行每一条指令。

尽管计算机只是一个还不理想、还不成熟的机器况且是没有类似于任何动机和情感的机器，但它却是人类建造的从未有过的最令人神往的机器之一。它肯定会改变社会的相互联系方式。计算机是为"被应用"而建造的，然而，字"被应用"有其积极的含意，这是因为计算机在做它出现以前人们为之忧愁的、冗长乏味的、单调的、易出错误的零碎事上是很杰出的。

通常，在我们平常的相互交谈时，所使用的言词可能很不精确。即使省略一些话也丝毫无损于交流。这与我们同计算机打交道形成鲜明的对比。当我们发命令给计算机时，每个命令都必须毫不含糊，措词必须严格地按照计算机所要求的方式，比如应该有句号的地方如果被你疏忽了，或者不应有的地方被你掺杂进一点，这都会产生不好的后果。有时这些规则可能显得非常武断——确实它们常是如此，——然而严格地遵照这些规则却是基本的要求。

COBOL 是迄今以来第一个高水平的商业用语言。COBOL 被尊为高水平的语言是因为它由字、短语组成，而且语法和英文书写没有一点不同。于是每一个具有英文知识的人将发觉 COBOL 很容易学——即使人们不具备任何初步的程序设计知识或者商业经验。

一般说来，有时加给 COBOL 的非难是：它太罗嗦。这点是不能否认的。但事实上，它的意图是明确的，使那些对语言生疏的人能读程序而且对它是在干什么有一个清晰的概念。由于 COBOL 通常是用和英语同样的语汇写成的。程序又自成文，这是 COBOL 语言的一个主要优点。

但是，当你知道 COBOL 指令甚至连计算都不懂的时候，你会感到非常吃惊。计算机能够明白的语言仅是被称为二进制的语言。这种语言全由“0”和“1”组成。二进制语言不但乏味、容易出错，而且人们掌握它也非常困难。可是对于计算机，二进制语言是理想的，把 0 和 1 看作微小的开关。它们要么是“开”，要么是“关”。

由于一条 COBOL 指令能产生大量的机器指令，写程序几乎不用二进制语言，这一点对读者会变得越来越明显，因为它太单调、费时，且修改和弄懂都很困难。另一方面，修改一个 COBOL 程序能大大地减轻负担。由于 COBOL 程序如此易懂，易于修改和更新，不带 COBOL 编译程序的计算机，美国政府机构通常是禁止购买的。

任一能接受 COBOL 作为程序设计语言的计算机，装备有被称为编译程序的一组特殊的高精尖程序，编译程序的用途是逐一地扫描 COBOL 指令，而且将它们转换成与它们相应的二进制指令，一个 COBOL 指令被转换成 20 个或更多的机器语言指令（二进制）是很可能的。一旦被转换成这种二进制形式，计算机就能以电子的速度或者至少极为接近于这一速度来处理指令。

从一种语言转换成另一种语言，这个概念对你们可能不太熟悉，特别地，假若你住在一个大都市内，譬如在纽约市，当你在一个小餐馆里订购膳食，常常听到一个人的订单被转为行话，这行话和通常的订单很少相干或者一点不相干。例如，一杯矿泉水被编码成数字“91”。一杯热的可口可乐被编成“51”，而一杯水被转换成“81”。一小杯可口可乐是“shot（小容量）”，而一大杯以“stretch（大容量）”代替。数字“210”意思为“某人没付款就离开了餐馆”。信不信由你。短语“看一下冰块”是一句“看一看那位刚进餐馆的漂亮姑娘”的行话。所有这些介绍可能相当吸引人。一位英国出生的作家对用于订单上的行话“一块烤英国松饼”提出抗议，因为它非别的意思，正是“打倒英国人”，或者更糟的意思：“一个烧焦的英国佬！”

把简单的订单转变为专用行话有三重道理：第一，它给餐馆的雇员提供一定的威望；第二，转变的帐单一般较短，而且减少了模棱两可；第三，它避免厨师听到重复订购声的可能性——一是由顾客发出的，而另一个是侍者传送的。

你可能对所有这些不得不用 COBOL 来完成感到怀疑。在某种意义上，当侍者从顾客那儿传递订单到厨师时，他的作用就好象一个编译程序，把顾客用普通英文给出的订单转变成行话译文，厨师明白且处理这些译文。

一个 COBOL 程序一旦写好，包含于该程序内的指令必须转变成计算机能处理的形式。该形式正如我们先前说起的，是这些指令二进制的表达形式。编译程序是由令人迷惑的复杂的程序序列组成的，它检查 COBOL 程序的每条指令，假若检查后发觉是正确的，将它们转变为供后面执行的二进制形式，仅当程序执行完之后，才能产生结果。编译程序被设计成按正确语法规定的句法和正确的关键字拼法去检查每个程序指令，且确保专用基字使用适当。

一旦发觉一条 COBOL 指令有毛病，那会发生什么情况呢？在这种情况下，一个诊断信息伴随试图解释该错误的大概原因，一起被打印出来。这使人们联想起，一位体格检查的医生用一整套测试方法，对病员进行体格检查的方式，

倘若一个病人在某一确定的限定范围内通过了所有的检查，显然，他(或者她)的健康状况是良好的。如此比拟并不牵强附会，因为编译程序用恰当的打印一些警告信息的方法允许极微小的违犯通过.而倘若错误属严重的一类，则立即停止程序运行.尽管程序可能含有句法规则正确的指令系列，指令可以某种方式堆放在一起.使其不能产生预期的结果。换句话说，在它们的逻辑安排上存在错误。这种逻辑错误在日常生活中.特别是对于儿童，常常易被发现。例如，假若一孩子被告知穿上漂亮的衣服并给人看看。指令本身无错，但照规定的顺序去执行这些指令，不一定取得预期的效果。

很遗憾，至今还没有出现如此高级的编译程序，以致于它能修改逻辑错误。这类错误以及由编译程序查出任一别的需要修改的错误则是程序设计员的责任。程序的打印列表和任一计算结果一起回送给程序员，这里要指出的是：很有可能一个错误产生一串诊断信息。这可能会令人失望，但不必为此而心烦意乱，因为改正这一错误必将清除所有的这些诊断信息。经努力后，程序和比指令还多的诊断信息一起被回送时，程序员有时会丧失信心。在这关头，怕难为情是无益的.任何人都会产生错误.为什么你就不会？对工作有了经验和成绩，错误的数量就会很快地减少，而且修改它们，也会逐渐地变成很容易了。

一旦 COBOL 语言由编译程序转换成与之等价的二进制语言程序，则获得一份穿孔卡片或写在不论是磁带或磁盘上的二进制程序复制品即成为可能。倘若程序顺序执行——可能具有新数据——运行此程序的二进制版本便能节省宝贵的计算机时间.因为它不必经过编译。二进制的程序是很容易执行的。

为了区别程序的这两种版本.一种用 COBOL 写成的原版称作源程序；反之，编译过的版本称为目标程序。

1.2 穿孔卡片

当今世界上绝大多数计算机采用标准的 80 列穿孔卡片。不过，现在有一个极大的趋势.即对穿孔卡片远不如对视频终端感兴趣。视频终端好象熟悉的电视机，除了它有一个类似于一打字机的键盘和键盘穿孔机之外——后者是用于穿孔卡片的，穿孔卡片被彻底抛弃这未必可能，不管怎样，穿孔卡片仍在被使用着，所以掌握一些穿孔卡片的知识是有益的。

尽管自 1930 年以来.穿孔卡片被广泛应用，但它起源于美国革命末期。当时有一位名叫贾柯德(Jacquard)的法国织布工用木制穿孔卡去控制织布机的操作。

在详尽研究编译 1880 年美国人口调查统计数据一事的准备工作中，有一位名叫霍曼·赫莱里斯(Dr. Herman Hollerith)的先生是统计学家，被人口调查局雇为特别代理人以帮助缓和制表问题。1887 年，赫莱里斯发展了他关于机器读穿孔卡片的理论，他获得了惊人的成功。早期的赫莱里斯卡片是 3×5 英吋的。如今它们是 $7^{3}/_{8}\times3^{1}/_{4}$ 英吋。图 1-1 显示一张典型卡片，上面穿有 0 到 9 的孔，还有 26 个英文字母及一些常用符号。