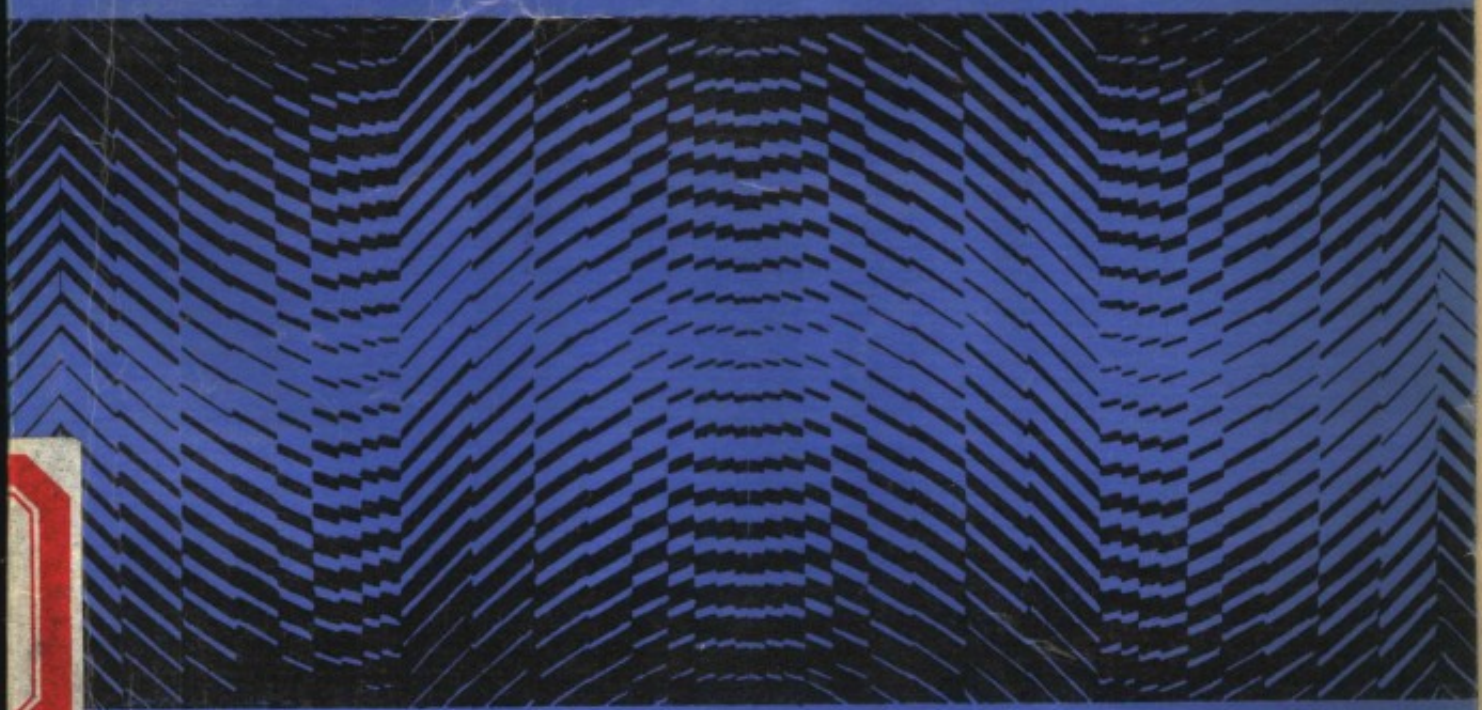
FORTH

语言教程

张怀宁　等编著



机械工业出版社

ISBN 7-111-01605-X/TP·94

科技新书目：204-006
定价：18.60元

FORTH 语言教程

张怀宁等编著

机 械 工 业 出 版 社

FORTH 是一种新颖独到的计算机语言和软件工具。它适用于实时控制、事务管理、人工智能、机器人、科学计算等各个领域。

本书以在 IBM 和长城个人计算机上运行的 FORTH 83 系统为蓝本，一步一步地讲解如何使用 FORTH 语言和进行 FORTH 程序设计。通过对系统程序的说明，详细讲解 FORTH 语言内部的工作机理是本书的重点。

本书适于高等学校作为教材使用，也可供从事计算机工作的科技人员参考。本书分为基础篇、中级篇和高级篇三部分，可满足不同水平读者的需要。需要与本书配套的磁盘的同志请与昆明工学院教材科李红宇同志联系。

FORTH 语言教程

张怀宁等编著

*

责任编辑：曲彩云

*

机械工业出版社出版(北京阜成门外百万庄南街一号)

(北京市书刊出版业营业许可证出字第 117 号)

兵器工业出版社印刷厂印刷

新华书店北京发行所发行·新华书店经售

*

开本 787×1092 1/16·印张 22 3/4 ·字数 542 千字

1989 年 12 月北京第一版·1989 年 12 月北京第一次印刷

印数 0,001—2,190 ·定价：18.60 元

*

ISBN 7-111-01605-X/TP·94

前　言

FORTH 语言是一种富有强大生命力和巨大潜在能力的计算机高级语言，又是一种与已流行的计算机语言很不相同的软件开发工具。FORTH 语言问世之初，除发明者之外只有很少几个人理解它。FORTH 语言的社会实践效果显示了它强大的生命力和多方面的适应性，使人们对它刮目相待，开始理解它并进而热爱它。现在 FORTH 语言在计算机世界占有特殊的地位，广泛应用于过程控制、数据库、人工智能和科学计算等各个领域。

FORTH 开始传入我国是 1980 年以后的事。那时 FORTH 在我国好像是一颗小小的雪粒，经过几年的翻滚运动，现已发展到了一定的体积。雪球愈大，翻滚它就需要更大的气力，也就需要有更多的人来参加。现在我国社会需要一本讲解如何在 IBM 公司和长城公司的个人计算机上运行 FORTH 语言的书，这本书不但要符合 FORTH 语言的最新标准 Forth-83 标准，要教会读者如何进行 FORTH 程序设计，而且还应当讲解清楚 FORTH 语言内部的工作机理。本书就是为满足这种需要而编写的。

本书以在 8086 机器上运行的 FORTH 83（简称 F83）2.1.0 版本为蓝本，有系统地、由浅入深地介绍 F83 系统的各个方面。FORTH 83 2.1.0 版本的作者是美国的两位专业 FORTH 程序员 Henry Laxen 和 Michael Perry，他们两人从一开始就积极参加 FORTH 兴趣小组的活动，并参加制定了 Forth-79 和 Forth-83 标准。尽管本书中所举的系统程序大多来源于这个版本，但透过这些系统程序所显示的 FORTH 语言的工作机理对任何一个具体的 F83 系统都是适用的。

本书分为基础篇、中级篇及高级篇三部分。基础篇介绍 FORTH 语言的基本概念及基本 FORTH 程序设计；中级篇是本书的主干，FORTH 语言的内部结构及工作机制主要放在这一部分中介绍。初学者在掌握基础篇的内容之后可先学习中级篇中不带星号的章节，然后再学习中级篇的其余内容。高级篇由几个相对独立的专题组成，可供各方面的读者进一步研究 FORTH 时学习参考。

本书由昆明工学院计算机及自动化系张怀宁副教授、吕杨讲师编写。其中，第 9、12、18、22、24 和 25 章由吕杨编写，其余各章由张怀宁编写。张怀宁主持全书的编写工作并审阅了全部书稿。在本书编写过程中，屈维德教授，林文兰、缪尔康副教授，杨宁同志给予了很多支持和鼓励，在此谨致衷心的谢意。

由于我们水平不高，编写时间仓促，错误或不当之处在所难免，殷切期望广大读者批评指正。

编者

一九八八年四月

目　录

第一部分　基 础 篇

第 1 章　FORTH 语言引导

1.1　FORTH 语言的发展概况及特点

1968 年，美国计算机科学家查理斯·穆（Charles H. Moore）发明并实现了 FORTH 语言（在 IBM1130 和 Univac 1108 计算机上）。作为一个专业的程序设计者，查理斯·穆使用过很多种程序设计语言，但没有哪个语言令他满意，于是就自己潜心开发一些特殊的程序设计工具，以求创造一种沟通人与计算机的自然而又简单的人机交流工具。经过 10 年的探索，他的努力最终结晶成为 FORTH 语言。1970 年，FORTH 语言定型成为我们今天知道的样子，当时查理斯·穆在美国国家射电天文台（NRAO）工作。FORTH 语言最早的应用是控制位于凯特峰（Kitt Peak）的巨大的天文望远镜和分析所收集到的数据。随着射电望远镜的出口，FORTH 语言传播到了许多国家。由于 FORTH 在天文台的巨大成功，1974 年国际天文协会正式接收 FORTH 语言作为标准的程序设计语言。

1973 年，查理斯·穆及他的一些同事离开国家射电天文台，成立了 FORTH 公司（FORTH, Inc.），这是 FORTH 软件商品化的开端。

从 1968 年到 1978 年是 FORTH 语言最初的 10 年。在这期间，无论是 FORTH 语言本身的进化发展还是它的传播都较缓慢。其主要原因，一是由于没有讲授 FORTH 语言的教科书；二是由于 FORTH 软件价格昂贵。

个人计算机工业的蓬勃发展，给 FORTH 语言的发展和传播创造了良好的条件。1978 年，10 来个热爱 FORTH 语言的系统程序设计者成立了一个非赢利的 FORTH 兴趣小组（Fig），其目的是鼓励和推进 FORTH 语言在个人计算机上的应用。到 1979 年初，Fig 小组的成员成功地在 8080、6800、6502 和 PDP-11 等计算机上实现了 FORTH 语言。由于他们实现的 FORTH 质量好而价格又非常便宜，于是由他们实现和公布的 FORTH 版本得到了很快的传播，这就是直到今天还在流行的 Fig-FORTH 的由来。Fig 在 1978 年的另一项重要工作是成立了 FORTH 标准小组（Forth Standards Team），FORTH 标准小组的第一个工作成果是 Forth-78 标准。但这个标准制定得不够好，因而从未流行过。FORTH 标准小组公布的第一个被广为采用的标准是 Forth-79 标准，很多软件商及 FORTH 公司在他们的产品中都采用了这一标准。

经过几年的实践，人们发现 Forth-79 标准存在着一些问题，其中最主要的有：很多定义的功能与系统所处的状态有关；循环控制的结构；当商为负时对所得余数的处理；逻辑值"真"的表示等。作为对上述问题的新的决定，FORTH 标准小组在 1983 年夏公布了一个新的标准，这就是 Forth-83 标准。FORTH 标准小组还决定在最近一些年中不再考虑修改这一标准，以便让它在 FORTH 世界站稳脚根，得到广泛地承认和采用。由于不

少软件公司在为IBM公司的个人计算机及莫托洛拉公司的68000等机器上开发FORTH时都是采用的Forth-83标准，因而从1984年开始，符合这一新的FORTH标准的各种FORTH版本和软件开始广泛流行。

除FORTH公司以外，在美国还成立了不少生产和经营FORTH软件及系统的公司，其中最有名的是MVP公司（Mountain View Press, Inc.）、LMI公司（Laboratory Microsystems Incorporated）、Novix公司等。在欧洲，成立了欧洲FORTH使用者协会（EFUG），它出版了一个FORTH标准定义词表，这就是所谓的Forth-77标准。1980年以后，FORTH传入我国。经过一些大学和科研单位的努力，已经取得了一批最初的科研和应用成果。为了更好地开展对FORTH语言的研究与应用，促进FOTH软件的研究、开发、推广和交流工作，中国FORTH应用研究会（CFIG）已于1987年6月在北京成立。上海交通大学从1984年开始每两年举办一次国际FORTH研讨会。FORTH的发明人查理斯·穆出席了1986年的研讨会，在会上做了精彩的学术报告和表演了他的手提FORTH计算机。

在FORTH语言即将进入它的第三个10年的今天，当初只有十几个人的第一个Fig小组已经发展成为拥有众多会员的国际性学术团体，使用FORTH的人越来越多。现在FORTH广泛地应用于实时控制、人工智能、事务管理、科学计算、机器人、办公室自动化等各个领域。FORTH为什么会这么吸引人，为什么会在这么多的领域内大显身手，这是由于FORTH语言具有如下的特点和优越性:

FORTH语言具有可扩展性，或者说它是一种自定义功能的语言。使用FORTH语言进行程序设计就是在不断地扩充FORTH语言，以便建立起一个最切合需要的程序设计环境并由此达到既定的目的;

FORTH语言的运行速度比其它任何高级语言都快得多，可与汇编语言相比拟，在程序编制上它又象别的高级语言一样方便，而且它可以直接和系统中的各个硬部件打交道，因此FORTH语言特别适合于实时控制、实时管理和决策的需要;

FORTH语言是一种完全结构化的程序设计语言。一个FORTH程序是由若干个相对独立的模块（词）组成，每一个模块的入口和出口都是唯一的。一个程序一旦设计调试完毕并成为FORTH语言中的一员后，它又可以参与组建更高一级的模块（下一个程序）。因此FORTH程序在结构上象是一座"金字塔"：塔基是由FORTH语言提供的基本模块，塔身是由程序设计者按照任务的需要创造出来的分为若干级的一些模块，而位于塔尖的则是提供给用户使用的、让整个FORTH机器运行以达到既定目标的运行命令;

FORTH语言是一种新颖的高级语言，它以词典为核心，以堆栈为通道，在结构上非常简练。它集编辑程序、编译程序、汇编程序、解释程序、操作系统等于一身，使它成为一种可以最大限度地发挥和发展程序设计者创造能力的十分灵活的软件工具;

FORTH语言是一种短小精干的语言。由于FORTH语言的词可以编译为穿线编码(threaded code)，也即地址串，使得完成同一任务的FORTH程序的目的码比汇编语言程序的目的码还要短小。而支持某一具体工业应用程序的FORTH系统的体积也是很小的。因而FORTH语言也很适合智能化仪器仪表和机电一体化的需要。

此外，FORTH语言的易移植性，良好的人机对话性能也是它明显的优点。

查理斯·穆曾说过:"FORTH是一种能力非凡而又简单的计算机的语言。……在模块

化与交互性上它类似 LISP，在速度和控制上它类似汇编语言，但在简捷紧凑和多方面的适应性上没有任何一门别的语言和它相似。具有大量的词汇、简单的语法和内在的可扩充性，它与最成功的自然语言英语相类似。"。在记住 FORTH 上述种种优点的同时，我们应注意到，查理斯·穆还说过："然而，FORTH 是一个具有两种截然相反性质的放大器：(用它写程序) 写得不好是可能的。在好与坏两者的区别上，FORTH 大于其它的语言。"

1.2 如何学习 FORTH 语言

本书以讲透 FORTH 语言的工作原理为宗旨，以扩大 FORTH 语言的影响，促进 FORTH 的传播和应用为目的。

读者只要把本书的目录和讲授其它计算机语言的教科书比较一下，就会立即发现，学习 FORTH 和学习其它任何一门高级计算机语言有着近乎是"质"的区别：首先，FORTH 语言的内容非常丰富，如 1.1 所述。因此，与其说学习 FORTH 是学习一门计算机语言，不如说是学习一门综合性的软件课程；其次，FORTH 语言"几乎没有"语法，也没有传统计算机语言中的语句。因此，学习计算机高级语言主要是学习语句和语法这一点对于学习 FORTH 来说全然不成立。学习 FORTH 语言，主要是学习 FORTH 的内部结构。这是因为进行 FORTH 程序设计的过程就是扩展 FORTH 语言的过程，用 FORTH 语言解决一个实际的工程问题，往往要求建立一个具体的 FORTH 工程版本和体系。因此，必须对 FORTH 语言的内部结构和机理有较深入的了解。

依照循序渐进的原则，本教材从 FORTH 的"A、B、C"开始。全书在深度分成三个不同的层次：基础篇是第一层次；基础篇加上中级篇中不带星号的章节构成较深入的第二层次；中级篇的全部加上高级篇构成最深入的、专题性质的第三层次。这三个不同深度的层次各成系统，并配有大量例题，以帮助读者减少学习困难和符合不同水平读者的需求。为便于组织教学，全书设章较多，这样做也便于读者很快发现他所感兴趣的部分。本教程的基础篇和中级篇中不带星号的章节适宜作为普及、推广 FORTH 语言的学习班的教材，推荐学时为 50 学时；基础篇、中级篇的全部适宜作为大学本科生的教材，推荐学时为 60 学时；高级篇中的内容则适于计算机专业高年级学生和研究生使用。本书第 2～22 章配有习题，为了方便自学，在附录中给出了练习的参考解答。为了能正确理解书中的内容，及时纠正各种误解，上机练习是非常必要的。如果上机练习时间很多，则课堂讲授的内容和学时数都可减少。

FORTH 语言的基本"单位"是词，就这一点来说，学习 FORTH 有一点像学习一门新的外国语。学习外语离不开词典，学习 FORTH 同样也需要词典。本书每一章的后面有一个词表，该表由在本章中初次介绍的词及初次接触到的词（作者自己编写的例词不包括在词表中）组成。另外，在附录中有一张按词名的 ASCII 代码顺序排列而成的词名索引表，该表指出本书中的每一个词出现在分章词表中的页码，以及该词的定义组成出现在哪一页。

学习任何一科学都需要参考书。附录参考文献[1]，[2]，[3]，[4]都是容易找到的参考书；由中国 FORTH 应用研究会和北京多思软件公司主办的《多思通讯》经常登载介绍和应用 FORTH 的文章，传播有关 FORTH 的各种信息。相信上述参考书和期刊一定会给各位读者很大的帮助和启迪。

第 2 章　FORTH 语言初貌

2.1　导言

FORTH 语言是一种高级程序设计语言，在运用 FORTH 进行程序设计时，人们通过"定义"（Definition）和 FORTH 打交道。定义又称为"词"（Word），它也就是 FORTH 语言中的"命令"（Command）。定义、词和命令这三者是完全等价的概念，几乎在每一个场合，这三个术语都可以互换使用。

FORTH 语言是由一个一个的词构成的，FORTH 中的词相当于其它程序设计语言的程序。FORTH 程序设计就是根据问题要求，运用已有的词去建立一个新的定义。而这个新定义的名字，也就是今后使用者下达给 FORTH 计算机的运行命令。运行命令下达后，FORTH 计算机会按规定给出相应的响应，人机之间可以建立起良好的通讯联系。因此 FORTH 语言又是一种交互式语言。

FORTH 语言中所有的词都存放在 FORTH 计算机的"词典"（Dictionary）中。一个新词设计成功并成为词典中的一员后，就可以不断地被引用：直接运行它或用它组成别的新词。使用者可以给 FORTH 语言添加新词是 FORTH 语言可扩展性的一个方面。

本章的目的是展示 FORTH 语言的初貌，建立 FORTH 程序设计的概念。

以下是本书采用的部分约定：

约　定	意　　义
<CR>	代表按下回车键的动作。
小号字	在例中表示 FORTH 的输出或响应，以和使用者的键盘输入相区别。
冒号	在文字叙述中用来代表名为：　的 FORTH 定义。
分号	在文字叙述中用来代表名为；　的 FORTH 定义。

2.2　FORTH 中的定义

当你第一次使用 FORTH 语言时，按下回车键后在显示屏上就会看到一个"ok"：

```
<CR>    ok
```

"ok"在 FORTH 中是一个很重要的信息，它表示 FORTH 正在等待着你的命令。看到它后就可以从键盘上输入一个 FORTH 命令：

```
ok
FORTH 命令   <CR>   ok
```

若在显示屏上再次出现"ok"如上，则表明 FORTH 已成功地执行了你的命令，并且又在等待接收新的命令。所以"ok"相当于 FORTH 在说"我已准备好了"或者"执行成功，我又准备好了"。因此使用 FORTH 的人都很喜欢"ok"，并希望在每一次<CR>之后都能看到"ok"。

单纯输出文字信息的程序是很简单而又很有用的，下面就是一个这样的定义：

```
: HOPE-OK    ." ok FORTH!" ;
```

这是我们碰到的第一个 FORTH 定义，它向我们展示了一个 FORTH 定义的各个组成部分和先后顺序:

组成部分	作　　用
:	以后我们就叫它"冒号"。冒号告诉FORTH，一个新的定义开始了。记住，新定义总是由冒号开头。
HOPE-OK	定义名。每一个定义都必须有自己的名字，因为你以后还要用到它。定义名最好能表达出定义的目的。FORTH 中的名字最长可达 31 个字符。
定义内容	是定义的实体，规定定义的功能。定义 HOPE-OK 的功能是输出文字信息，定义中的 ." 告诉 FORTH 一段文字信息开始了，它必须与后面的文字信息隔开一个空格；文字信息是以符号 " 做为结束的标志。
;	以后我们就叫它"分号"。分号告诉FORTH一个定义结束了。记住，一个以冒号开头的新定义必须以分号结束。

尽管定义 HOPE-OK 如此简单，但并不是每个人在初次输入它时都获得了成功。这是因为在 FORTH 中，空格和定义是同等的重要，在输入 FORTH 程序时，词与词之间至少要被一个空格隔开（两个或两个以上空格也可以），否则 FORTH 就不知道一个词在何处结束，下一个词在何处开始。HOPE-OK 的定义行中，冒号、分号以及 ." 都是 FORTH 语言中已有的词，它们和定义行中的相邻成分之间必须有空格隔开。而在第一次输入 FORTH 定义时，很可能会漏掉某一个必不可少的空格，特别是在词 ." 和它后面文字信息之间的那个空格。

现在按照上面的格式从键盘上输入词 HOPE-OK 的定义:

```
: HOPE-OK ." ok FORTH!" ; <CR> ok
```

FORTH 响应 "ok" 表示输入成功，也即给 FORTH 增加了一个名为 HOPE-OK 的新词（注意，迄今为止我们还未运行过它）。下面运行该词，也即从键盘键入它的名字并按下回车键:

```
HOPE-OK<CR> ok FORTH! ok
```

在输出文字信息之后 FORTH 又给出了 "ok"，表明执行成功，并又准备好接收新的命令。

显然，被包含在 ." 和 " 之间的文字信息可以是任意的可印刷键盘字符。比如我们可以定义一个名为 STAR 的词，它的功能是打印出一个星号:

```
: STAR " *" ; <CR> ok
```

只要 STAR 是以冒号开头，分号结束，星号在 ." 和 " 之间，并且该有空格的地方至少有一个空格，则对 STAR 的输入就一定会成功。运行它:

```
STAR <CR> * ok
```

为了使输出的文字信息醒目，我们往往希望它们从下一行的第一个字符处开始出现。这时，就要用到一个名为 CR 的词，其功能是使在它后面的显示从下一行的第一个字符处开始。比如我们可以模仿 HOPE-OK 定义一个 WISH-OK 如下:

```
: WISH-OK CR ." ok FORTH!" ; <CR> ok
```

运行该定义:

```
WISH-OK<CR>
```

```
ok FORTH! ok
```

当一个定义太长在一行内容容纳不下时，或为了排列整齐便于阅读，可以把一个定义分成几行输入，在每一行的末尾按一次回车键。FORTH 并不会把换行用的<CR>当作定义结束的标志，它要一直等到分号出现时才认为当前的定义结束了。例如:

```
: SHU <CR>
    CR  STAR  CR  STAR  <CR>
    CR  STAR  CR  STAR  <CR>
    ;   <CR>  ok
```

运行 SHU :

```
SHU <CR>
*
*
*
* ok
```

上面我们初次接触了几个 FORTH 语言的定义: 冒号、分号、 ." 和 CR ，并运用它们设计了几个新的定义: HOPE-OK 、STAR 、WISH-OK 和 SHU 。在 FORTH 中，空格与定义处于同等重要的地位，当输入命令行（也即 FORTH 程序）时，该有空格的地方至少得有一个空格。从下一节起，凡要按回车键的地方不再用符号<CR>标出，请读者自己注意。

2.3 FORTH 中的词典

FORTH 语言中所有的词，不论是系统原来就有的还是使用者后来添加的，都存放在 FORTH 系统的词典中。词典是 FORTH 语言的核心，它使用计算机中一块特定的主存区域。在词典中不但妥善地存放着当前所有的词，并且还有余地容纳随时添加进来的新词。新增加的词总是位于词典的顶部（也即高地址端），随着新词的加入，词典不断地向高地址方向膨胀。图 2-1 显示了当前词典的状况:

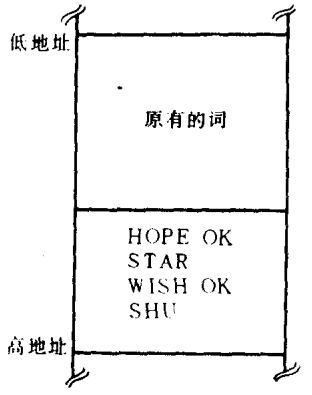


图 2-1 词典示意图

FORTH 只认识词典中已有的词。试键入

```
ttttt TTTTT ?
```

由于词典中没有 ttttt 这个词（在绝大多数场合，FORTH 视大写字符与小写字符为同一字符），于是 FORTH 显示刚键入的名字和一个问号，告诉你它不知道这个词。此时只需按下回车键，FORTH 就又响应"ok"，表明刚才发生的错误没有任何关系。

若键入一行词典中已有的词，例如:

```
CR  STAR  STAR  HOPR-OK  STAR STAR CR
```

（并按下回车键后）FORTH 就从左到右读入命令行: FORTH 依靠空格区分命令行中的每一个词，每区分出一个词后就查词典，查到后就执行它，执行成功后就又开始处理命令行中的下一个词，在执行完最后一个词后给出"ok"，报告整个命令行业已成功地执行完毕。上述命令行的执行结果如下:

```
* *ok FORTH!* *
 ok
```

若想给词典增加一个新词，则键入的命令行必须以冒号开头，例如:

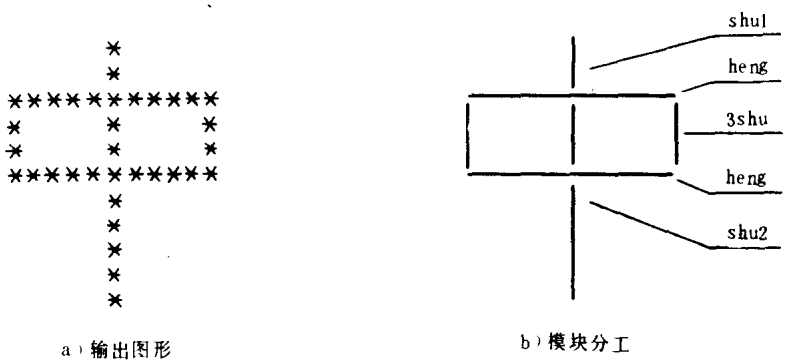
```
: 2STARS  STAR  STAR ;
```

由于命令行以冒号开头，按照约定，FORTH 得知一个新词来到了，新词的名字跟在冒号之后（起码要有一个空格隔开），于是 FORTH 就把新词名（此时是 2STARS）放入词典的顶部，并把 2STARS 之后的所有词编入词典，使它们成为 2STARS 的具体内容。以上过程称为"编译"，编译直到分号才告结束，此时 FORTH 给出 "ok" 表示编译成功。请注意在编译 2STARS 时并不打印出两个星号。能够清楚区分什么时候一个词被编译及什么时候一个词被执行是理解 FORTH 的关键之一。

2.4 模块化设计

FORTH 程序具有模块化的结构，一个定义（也即程序）的每一个词就是该定义的一个模块。一个新定义诞生后，它又可以成为组成另外一些更新定义的模块。

在设计 FORTH 程序时，程序员总是尽可能地使一个定义小而简单，因为只有这种类型的定义，才能成为有用的、可供其它很多定义采用的模块。采用模块化程序设计有助于提高程序设计的效率，有助于调试程序并改善程序的可读性。下面以设计输出由星号组成的"中"字为例，说明 FORTH 模块化程序设计的方法，参见图 2-2。



a）输出图形　　b）模块分工

图 2-2　词 ZHONG 的输出图形和组成模块

首先，需要定义一些基本的模块:

```
: STAR  ." *" ;          输出一个星号，前面已定义过。
: K3  ."    " ;          输出 3 个空格。
: K4  ."     " ;         输出 4 个空格。
: K8  ."         " ;     输出 8 个空格。
: K8SC  CR  K8  STAR ;   在下一行空 8 格后输出一个星号。
: 11STARS  ." * * * * * * * * * * *" ;
```

把它们编入词典（也即分别键入每一个定义及按回车键并看到"ok"）后就可以执行:

```
 ok
STAR * ok
K3     ok
K4      ok
K8          ok
K8SC
```

```
        * ok
11STARS * * * * * * * * * * * ok
```

以上面的模块为基础，就可以进一步构成图 2-2b 所示的各个模块:

```
: SHU1   K8SC  K8SC ; ok
: HENG   CR  K3  11STARS ; ok
: 3SHU   CR  K3  STAR  K4  STAR  K4  STAR
         CR  K3  STAR  K4  STAR  K4  STAR  ;   ok
: SHU2  K8SC  SHU1  SHU1 ;  ok
```

看一下它们的执行情况:

```
  ok
SHU1
          *
          *  ok
HENG
    * * * * * * * * * * *  ok
3SHU
    *       *       *
    *       *       *  ok
SHU2
          *
          *
          *
          *
          *  ok
 ok
```

最后得出 ZHONG 的定义如下:

```
: ZHONG  SHU1  HENG  3SHU  HENG  SHU2  CR  ;
```

把 ZHONG 编入词典后执行之:

```
   ok
ZHONG
          *
          *
    * * * * * * * * * * *
    *       *       *
    *       *       *
    * * * * * * * * * * *
          *
          *
          *
          *
          *
 ok
```

以 FORTH 语言固有的词（例如冒号、分号、CR 、 ." 等）为基础可以定义出一批新词（例如 STAR 、11STARS 等），利用以上两者又可构成更高一级的新词，如此反复直到完成预定的目的，这就是 FORTH 语言的程序设计过程。FORTH 语言程序具有内在的模块化形式，每一个模块可以单独设计、单独调试；每一个模块都有自己的名字，

因此便于使用。采用模块化方法设计出来的程序看起来像是一座建造得很巧妙的金字塔:塔基是FORTH语言固有的词和一些基本的模块，塔尖是使用者启动整个FORTH机器运转以达到预期目的的命令（例如本节中的词 ZHONG ）。

下面是本章所介绍的FORTH定义，做为示例的词不包括在其中:

词	功 能
:	按下述格式使用: : 定义名 定义内容 ; 它把一个新定义送入词典。
;	结束一个冒号定义。
."	按下述格式使用: ." 文字信息" 它把文字信息放入定义中，以后执行定义时文字信息原样印出。注意 ." 之后的第一个空格不属于文字信息；注意它必须和文字信息结束标志 " 配合使用。
CR	使选中的输出设备换行。

假若你刚完成的定义（比如名为ABC ）与词典中的一个词同名，则FORTH会告诉你该定义不是唯一的，但是没有关系。例如:

```
: ABC ••••••••• ;   ABC isn't unique  ok
```

从今以后但凡用到词ABC时，FORTH都是执行新的ABC。因此在上机练习时，你可以沿用原来的名字重新定义一个你所不满意的词。

练　习

从本章最后的词出发设计5个定义，它们分别打印大写字母 F、E、H、T和U。要求打印从第5列开始。定义 F 的执行情况如图所示。（注: 练习参考解答请见书末附录。参考解答是使用编辑程序编写的，如何使用编辑程序将在第5章中介绍。）

```
 ok
F
    * * * * * * *
    *
    *
    * * * * * * *
    *
    *
    *
    *
  ok
```