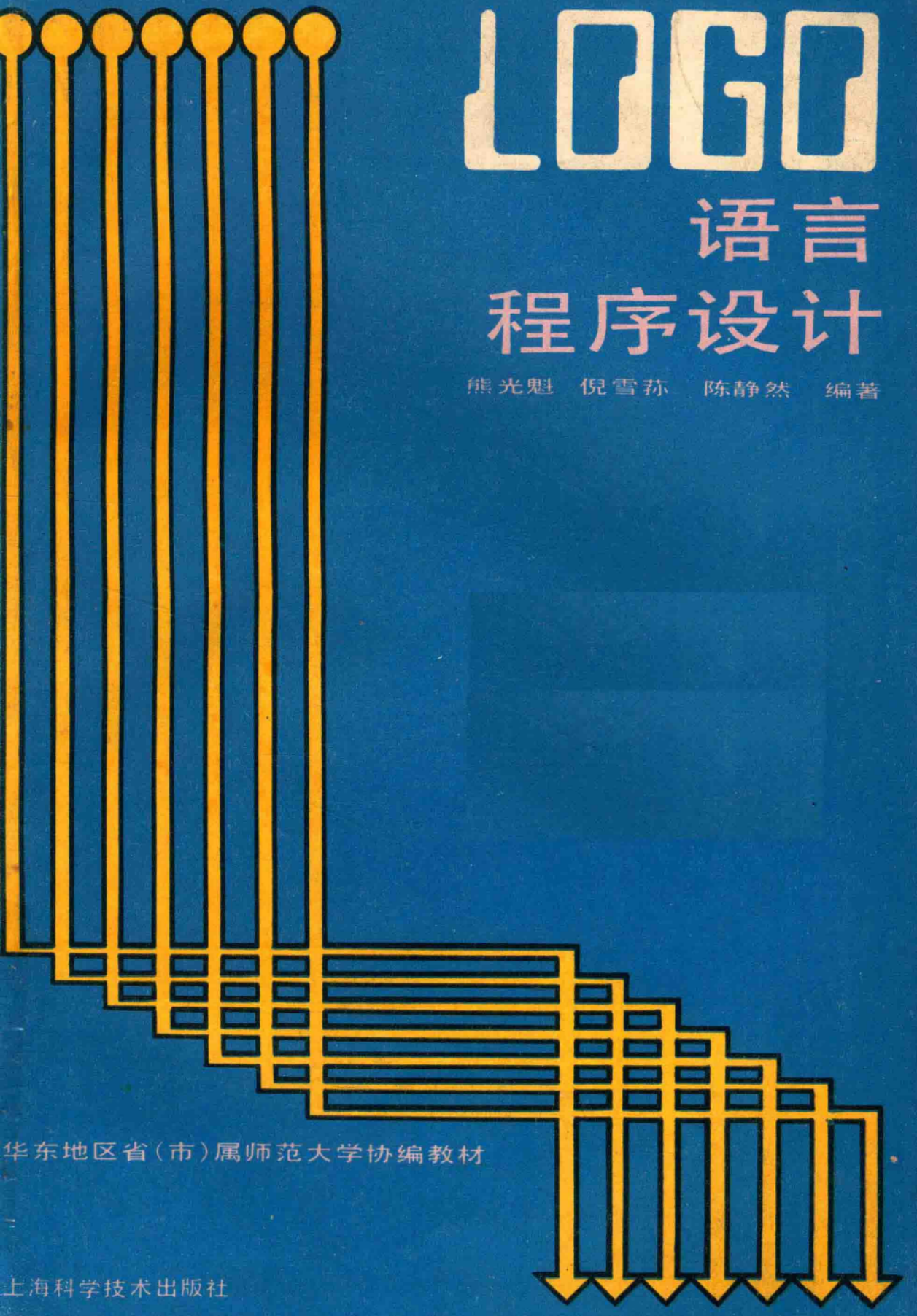
LOGO
语言程序设计

熊光魁　倪雪荪　陈静然　编著



华东地区省（市）属师范大学协编教材

上海科学技术出版社

LOGO语言程序设计

熊光魁　倪雪荪　陈静然　编著

上海科学技术出版社

责任编辑　周玉刚

LOGO 语言程序设计

熊光魁　倪雪荪　陈静然　编著

上海科学技术出版社出版发行
（上海瑞金二路 450 号）

浙江师范大学印刷厂印刷

开本787×1092 1/16 印张19.25　字数485,000

1988年3月第1版　1988年3月第1次印刷

印数：1—5,250

ISBN　7—5323—0690—9/TP.10

定价：4.75元

前　言

LOGO语言启迪开发人们的智力，具有很强的绘图能力及良好的词表处理功能，是国外青少年中极为普及的一门计算机程序设计语言。随着国内中小学计算机教学的发展，LOGO语言必将继BASIC语言之后迅速在我国青少年中得到广泛普及。

华东地区六省一市地方师范大学计算机专业协作组为适应计算机教育事业的蓬勃发展，为满足青少年学习LOGO语言的迫切需要，决定编写此书。本书为师范院校本、专科学生的计算机选修课教材，亦可为中、小学计算机教师，青、少年活动中心及少年宫计算机辅导员的教学参考书，若对内容作适当的舍取，也可作为中学生的LOGO语言学习教材。

本书以介绍APPLE LOGO语言为主，兼顾TERRAPIN LOGO及PC LOGO，内容深入浅出，例题和习题丰富，讲究逻辑性，科学性。本书集国内外LOGO教材之优点，在内容的安排和处理上，力求有新的特色。第一，本书在充分展现LOGO语言绘图能力的同时，也注意突出LOGO语言在数值计算方面所能发挥的作用；第二，在详尽地阐述LOGO词表处理功能的同时，注意归纳和总结LOGO语言的编程规律；第三，本书着意介绍了LOGO语言使用中的递归概念，通过大量的实例分析，阐述了LOGO语言在处理人工智能问题方面的应用；第四，本书还详细地介绍了LOGO语言对汇编程序的调用，意在为给读者今后对LOGO语言作进一步的开发提供帮助；第五，本书在最后的附录中还特意收入了张世正副教授的最新科研成果——中文LOGO语言的有关内容，它将为制作优秀的教学软件提供一条有效途径。

本书由江西师范大学熊光魁、浙江师范大学倪雪芬、山东师范大学陈静然编写，熊光魁为主编。上海师范大学张世正审校，并编写了第六章及第十章。本书还得到了上海大学朱鸿鹗教授的全面指导，在此一并致谢。

编　者

一九八七年十月

目　录

第一章 总 论

LOGO是一种计算机程序设计语言。

本章将简要介绍LOGO语言产生的历史背景，论述它的主要特点以及它与其它的计算机语言的不同之处。

§1.1 有关LOGO语言的一些历史知识

LOGO的开发始于60年代末，主要是由美国麻省理工学院人工智能实验室设计，属于微机语言的后起之秀。最初的LOGO版本是由Wallace Feurzeig, Daniel Bobrow和Seymour Papert在研究LISP语言的基础上，专为青少年学生和电脑初学者而设计的编程语言，其主导思想是试图使计算机编程成为教育青少年学生的有力工具。

每一种语言的产生都有着自己的时代背景。六十年代发展起来的计算机辅助教学的实践表明，如果使用只能接受一些死板程式的微电脑进行计算机辅助教学，在某种程度上是束缚了学生的想象和创造力。而人们所需要的是通过计算机辅助教学，培养和训练他们的思维、发展他们的智力和创造才能，正是在这种社会要求之下，开发了LOGO语言。因此可以说，LOGO的产生和发展既植根于计算机科学，特别是人工智能的研究，也植根于青少年的智力教育理论。

LOGO语言创始之后，在麻省理工学院的人工智能实验室和英格兰爱丁堡大学的人工智能系得到了发展，在1970年，海龟（Turtle）作图引入LOGO，这使得LOGO摆脱了一般计算机程序设计离不开数字计算的老格式以及作图离不开坐标系的老框框，使之易于为青少年和初学者所接受，某些费解的数学概念通过图解变得有趣和有启发性。通过对LOGO的学习，不但可以使学生掌握有关计算机的各种概念，获得应用计算机解决问题的能力，而且使计算机能帮助他们进行思考，训练他们的逻辑思维和抽象思维能力。

LOGO语言是一种便于自学的语言，它并不是强制性地向学生灌输知识，而是使他们在掌握了为数极少的通俗的LOGO命令之后，在发现中学习，在探求中求知，在学习计算机设计的同时，学习如何思考。

回顾LOGO的发展史，在开始的相当一段时间内，由于只能在大型机上运行，所以限制了它的普及。1979年MIT LOGO小组开始在微机上开发LOGO，于1981年完成了最早的微机LOGO版本（TI LOGO），它是在德克萨斯仪器公司的TI99/4上实现的。1982年在美国科学基金会和Apple计算机公司的资助下，由H·Abelson主持完成了APPLE LOGO，此外，TERRAPIN LOGO也是运行于Apple微机上的LOGO版本。以后，相继出现了IBM PC LOGO、DEC Professional 350 LOGO、TRS—80 LOGO、Laser—3000 LOGO，日本的MZ LOGO等。LOGO在微机上的实现使它得到了迅速的普及，作为一种初学者的计算机入门语

言，作为计算机教育语言，LOGO的影响日益扩大，随着微机技术的发展和应用领域的不断拓广，LOGO也在继续开拓自己的应用领域。

“LOGO”一词是由希腊文“思考”或“符号”、“文字”一词转来，英语中为标识语之意。设计者之所以为之取名LOGO，一方面表明了它有处理文字、概念、推理的功能，另一方面，也表明它的启发性的教育哲学的思想。

§1.2 LOGO语言的特点

七十年代以来，出现了软件发展中具有里程碑意义的结构程序设计的概念和方法。LOGO语言努力设计成能方便于这些概念和方法的运用，因此与其他的计算机高级语言相比，有其明显的特点。

1.2.1 模块化的程序结构

程序的模块化是指把一个大的程序分割成若干较小的彼此相互独立的模块。这些模块有各自的入口和出口，这些模块中的任意一个若被替换可以不影响程序中的其它模块。

程序设计的这种模块化结构，使我们能把注意力集中到全局结构、模块的划分和模块之间的联接上，而具体到每一模块的编制则成为局部性的工作，可以单独地进行编写和调试。许多典型的模块可以当作程序设计中的有用的工具使用。

LOGO语言中的过程（procedure）的定义方法和调用方式使它自然地构成这样的模块。LOGO语言中的过程其格式如下：

TO 过程名　　输入参数
过程体（命令序列）
END

其中过程体中的命令序列可以是LOGO的原始命令，也可以是用户定义的命令（用户自定义的过程）。

应用LOGO语言进行工作的关键就是要把我们想要解决的问题分割成若干部分，并将它们转化为彼此相互独立而又能有机地联系在一起的若干个LOGO过程。不难理解，这种工作方式在很大程度上符合人脑的逻辑思维方式，所以LOGO语言易学、易懂、易于掌握。

一般来说在现代高级程序语言中，除了BASIC之外，都有类似于LOGO过程的结构形式，但是LOGO过程更有其独自的特点。

在FORTRAN或PASCAL等语言中，类似的子程序和过程的编写和调试需要一定的运行环境，也就是说，必须为之编写相应的宣称，变量说明等等程序部分，而LOGO语言中的过程的编写和调试则完全可以独立进行。LOGO过程一经定义之后，就可以象原始命令一样的使用，LOGO过程之间，过程与原始命令之间是相当平等的。

LOGO过程模块的调用和过程之间的联接非常简单，只要引用过程名即可，不需要诸如转移或转子的概念，不存在改变分调运行环境再联调的问题，保证了程序的动态运行和其静态结构的良好的一致性。

根据上述的LOGO过程的特点，当一个LOGO过程被定义之后，就立即成为LOGO命令家族中的一个新成员，因此，LOGO语言用定义过程的办法提供了扩充命令集的简单、有

效的手段．初学时用户只需要记住最少的定义与规则，就能自己去定义新的词汇或命令．

LOGO 语言的这种扩充方法不仅支持自底而上的程序设计方式，而且也支持结构化程序设计所主张的自顶而下的设计方法．LOGO 过程定义中的过程名后面的输入参数在执行时可以用 LOGO 的命令序列（过程或命令）做实际参数值，也就是说，LOGO 允许把程序作为过程的输入或函数值．这样一来，这些用作输入参数的过程可以在最后的阶段编写、调试，不妨碍总过程的调试，便于采用自顶而下，逐步求精的设计方法．

1.2.2 递归性

LOGO语言具有递归的功能．LOGO过程允许递归调用，即在定义一个过程时允许在过程体中引用该过程本身．在计算机高级语言中，FORTRAN 和 BASIC 不允许递归调用，其它许多程序语言（包括 PASCAL，C，LISP 及 APL）也允许递归．然而在 LOGO 中，抽象的递归概念变得形象化，易于理解．

LOGO过程的递归调用表述了这样一种思维方法，为了解决一个复杂的未知的问题，把它化简为简单的同类的问题．如果最简单的情况可以给出解答；或者是把复杂的问题进行分解，其中一小部分可以解决．这时，可以让计算机完成一步步的反复化简，或是进行重复的过程调用，直到未知的问题得到解决．

利用递归过程能给出功能强而结构简单的高水平的程序．著名的 HANIO 塔问题就是一例，这种类似的问题如果用非递归程序求解往往就相当繁杂，而利用递归则使程序易写、易读、易懂．我们知道，如果计算机语言中有了递归功能，程序最终的实现过程就会变得复杂得多，一般来说，递归程序的运行时间效率低于相应的非递归程序，为了改善递归过程的运行效率，除了对于非尾部递归必须按常规递归过程处理之外，LOGO能自动识别尾部递归过程，并把它化为叠代（循环）处理，因此在一定程度上减少了递归造成的时间效率损失．

1.2.3 LOGO语言的形象直观性

LOGO语言，作为一种初学者的计算机语言，它的形象直观性能引起学习者强烈的兴趣．

LOGO引进海龟作图，通过运用“前进”、“后退”、“右转”、“左转”等通俗的命令指挥海龟运动，海龟运动的迹线就构成了千变万化的图形．与其它的语言不同，LOGO以一种独特的方式处理作图．它以海龟的当前位置和方向为参照进行相对运动，因此，在许多场合之下，不需要引进坐标系，避免了计算绝对坐标的麻烦，使丰富多彩的图形产生得自然、有趣，而且给予你充分发挥想象、进行试验的余地．当然 LOGO 也不排除常规的作图处理，提供了以屏幕中心为原点的直角坐标系，及一组有关的位置函数．

通过海龟作图，可以讲解几乎所有的程序设计的概念、方法，从基本的原始命令到过程的定义和调用，变量与数据，条件语句及循环，直至结构性的程序设计方法，递归算法等等．随着 LOGO 在微型机上的实现，LOGO 在计算机辅助教学领域中占有很大的阵地，甚至有人把它誉为计算机教学中的国王．

许多复杂图形的产生归结为十分简单的 LOGO 程序，许多难度甚大的智能问题通过LOGO语言能得到清晰的解答．青少年在浓厚的兴趣和轻松的环境中获得知识、受到训练，通过试验和自引导方式学会科学的逻辑思维．海龟作图为LOGO带来了极大的成功．

和其它的计算机语言相比，学习LOGO所必备的数学和计算机识知最少．同时，LOGO

中的变量无需预先说明和指定类型，使用简单、灵活。LOGO程序的书写格式也较之其它语言简单，便于掌握。所以把 LOGO 作为学习计算机语言的入门课程是合适的，它将为进一步的学习打下良好的基础。

§1.3 LOGO 与人工智能

LOGO语言的另一个特点是具有表处理功能。

LOGO的表处理功能，为我们提供了一个 LISP 风格的、更易于学习和掌握的表处理语言。正因为如此，LOGO语言适于编写知识库系统的软件，使得 LOGO 在人工智能的领域里有用武之地。

当电子计算机问世之后，利用计算机来摸拟人类的智能行为、设计和制造类似于人脑的智能机成为人工智能的主要研究内容。多年来的研究表明，作为人类思维的物质基础——大脑，当它进行有目的的活动时，表现出多层次的结构，这是由于人脑思维的复杂性所决定，不断地从一个层次跳跃到另一个层次，控制是很自由的。考察下面这个"智能测试"的例子。

从下列的英文字母中，找出哪一个是最具有特殊性的：

Z X T L V Y

面对这样一个问题，种种想法会立即出现在我们的脑海。例如，字母"Y"可被挑出，因为"Y"是其中唯一的一个元音字母，此外，如果建立起英文字母 A, B, C, ..., Z 与数字 1，2，3，…，26 的对应关系，"Y"是上述 6 个字母中唯一的一个与奇数相对应的。但是，如果从另外的一个观点出发——不考虑它的字母属性，而是考虑它的几何形状，那么字母"Z"应被选中，这是因为"Z"是其中唯一的一个由三条线段构成的，或者说它具有两个交点，而其它的都只有一个交点。然而，对于熟悉汉语拼音的小学生，从拼音角度考虑的话，这个问题的答案可能是"V"，过是因为唯独"V"不能作为一个拼音汉字的首字母，而其他五个字母都可以。

由此可见不同的人会有不同的回答，对于同一个人，考虑的时间越长，出现的判断就越多，使得过六个字母中的每一个都可能成为这个问题的答案。

现在设想为计算机编一个程序，让它来摸拟人脑求得这个问题的解答。在着手编写程序之前，必须解决如下的问题：为什么人们会从考虑"字母属性"跳到去考虑"几何形状"；而且为什么会想到"字母属性"，"几何形状"，"拼音汉字首字母"，或者其它的一些方面，也就是说，每一个可能想到的判断标准是怎样出现的。这些问题的研究就属于人工智能的范畴。

为了回答某一个问题，人脑的这些联想和判断是以人脑所具有的有关这个问题的知识与经验为基础，然后进行思维、推理，以寻求问题的答案，这是属于因果关系的过程。此外，非因果关系过程和随机性过程也是相当起作用的。

为计算机编程当然不能只写一种判断，而是要包括所有可能的判断标准。这所有的可能性不是那么容易确定的，因此，必须首先建立起有关这个问题的"知识库"，提供尽量完善的知识环境，然后，还要使计算机能摸拟人的推理的思维过程，由计算机系统自行处理这些有关的知识，最后得到问题的最佳解答。总之，要求程序本身具备探索解决问题的方法。

根据上面的分析可知，为计算机提供知识的表示，知识的获取，知识的处理功能是必须的，也就是能对知识进行层次性的描述、控制和理解．这些也就是当代智能计算机的研究课题之一．

LOGO 语言的数据类型包括数、词（WORD）和表（LIST），其中的表是表元素的有序集合．表元素可以是数、词，也可以是表．这样，就使得表能够表达层次关系，进而表达种种复杂的对象和结构——现实世界、知识结构以及自然语言等等．LOGO 的表可以描述句子、多维空间上的坐标、一般集合、树结构、乃至网络和几何图形．因此，利用 LOGO 中的表，可进行文字和语言分析，描述层次关系，存储知识，并能进行知识的更新和检索，便于处理层次性的结构，编写“知识库”等等．

LOGO 语言允许把 LOGO 程序视为表，因此，LOGO 程序也就可以作为过程的参数或函数过程的值，这就使得可以编制产生 LOGO 程序的程序．因此 LOGO 可以用来建立概念，进行推理，并能够从“知识库”中导出新的知识，编写专家系统等等．

人们一直在追求程序语言的非过程化——用不着告诉计算机如何求解一个具体问题，而是用一种语言提出问题，机器自动进行判断和推理，最后给出问题的解答．LOGO 朝着语言的非过程化前进了一步．

以上简要地论述了 LOGO 语言的特点及与人工智能的关系，建议读者在学习完本书之后再回过头来阅读本章，从而对 LOGO 语言有一个总体的深入的了解．

第二章　LOGO 启动与海龟作图

本书以由 Apple 计算机公司销售，由 LOGO 计算机系统公司开发的 LOGO 1.5版本为蓝本（即通称的 APPLE LOGO），全面介绍 LOGO 语言的程序设计。现在先介绍启动 LOGO 的一些预备知识。

§2.1　LOGO 的启动

2.1.1　系统配置

运行 LOGO 语言，至少需要下列系统配置：

一台具有 48K 内存的 APPLE Ⅱ 计算机及一个插于 0 号槽口的 16K RAM 板；

插于 6 号槽口的磁盘控制板及一个磁盘驱动器；

一台彩色或黑白监视器；

一张 LOGO 系统盘；

如果有一台打印机，即可打印 LOGO 过程清单和图形；

如果有一张空白磁盘片可用来存放用户实用程序。

2.1.2　系统开工

这里所介绍的是启动 LOGO 所必须的操作，有关 APPLE Ⅱ 微机的详细操作，请参照 APPLE Ⅱ 操作手册。

首先将 APPLE Ⅱ 微机及监视器电源线插头插到电源插座上。在接通开关之前，将 LOGO 系统盘水平插入磁盘驱动器内，标记面朝上，然后关好磁盘驱动器门。

接通监视器和计算机的电源开关，磁盘驱动器红灯闪亮，LOGO 系统程序被调入计算机内存。不久，磁盘驱动器上红灯灭，屏幕上出现如下信息：

PRESS THE RETURN KEY TO BEGIN

与此同时，计算机在屏幕上提示用户插入自己软盘。如不换盘插入，可直接按回车键（RETURN）。磁盘驱动器红灯又闪亮，LOGO 语言系统完全调入内存，此时屏幕上显示有关版本等信息。最后屏幕显示：

WELCOME TO LOGO

?

“?”是 LOGO 系统的提示符，它表明 LOGO 系统已准备就绪，进入 LOGO 的命令模式，等待接受命令。“?”号后面紧跟一个发亮的小方块，称为光标。光标可在屏幕上移动，其位置表明可键入字符的当前位置。

上述的 LOGO 启动方式称为冷启动，

如果在引导 LOGO 之前，计算机已经显示了提示符，这时你插入 LOGO 系统盘，键入

*PR*6*

并按回车键，同样也能调入 LOGO。这种启动方式称之为热启动。

2.1.3 键盘

在介绍LOGO语言之前，还须简要说明 APPLE Ⅱ 键盘以及某些特殊键的使用。

键盘如图2.1所示，与英文打字机的键盘类似。

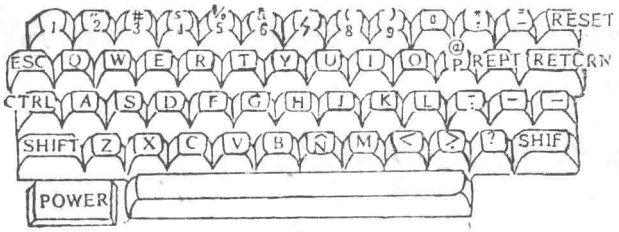


图2.1

下面介绍在 LOGO 命令模式下从键盘输入（键入）信息以及在键入信息而纠正错误时使用的特殊键。

(1) RETURN 键

说明　回车键，它将键入的命令送入计算机内存。

(2) 空格键

说明　该键打印一个不可见字符 —— 空格，它是 LOGO 的分隔符，在学习 LOGO 语言中，要特别注意空格键的确切功能。

(3) REPT 键

说明　重复键，按下一个字符的同时按下该键时，重复一串该字符。

(4) CTRL—RESET 键

说明　该键为复位键，它中断 LOGO 并返回 Apple 监控程序，一般情况下不可使用此键！

(5) SHIFT 键

说明　该键为转换键，用它实现双字符键的两种字符间的转换。在 LOGO 下，方括号“〔”和“〕”分别用 SHIFT—N 和 SHIFT—M 或 SHIFT—U 实现。

键入 LOGO 命令时，往往会出现漏打，多打，打错字符或是字符位置前后颠倒等错误。LOGO 提供了灵活的行编辑功能以便纠正打印错误，或修改整行命令。

(6) 左箭键 ←

说明　该键删除光标左边的字符，并且使光标左移一格。

(7) 右箭键 →

说明　该键使光标右移一字符位，不删除任何字符。

将 (6)、(7) 两个键与 REPT 键连用将会给你带来方便。下面是常用的控制复合键。

(8) CTRL—G

说明　该键能中断 LOGO 程序的运行.

(9) CTRL—Q

说明　LOGO 中，除了用空格作为分隔符外，还有以下的分隔符：〔,〕,=,<,>,+,−,*,/ 它们都有各自的特定意义．但是，如果将它们作为一个普通字符使用时，则在键入该字符之前要先按 CTRL—Q，例如，当把 A[B 作为一个字使用时，要键入 A, CTRL—Q, SHIFT—N, B.

(10) CTRL—Y

说明　该键的作用是重新显示先前键入的一行命令，并使光标位于行末端．此时，如果按回车键，则再次执行该命令.

(11) CTRL—A

说明　该键将光标移至当前显示行的行首.

(12) CTRL—E

说明　该键将光标移至当前显示行的行尾.

(13) CTRL—D

说明　该键删除光标位置上的字符.

上述的行编辑功能键能使你方便地移动光标所在位置，删除命令行上的任何字符；当你需要插入字符时，只须将光标定位后进行插入，光标及其右边的所有字符相应左移一格.

(14) CTRL—W

说明　该键停止程序的执行．当键入其它任何一个字符时可恢复执行.

(15) CTRL—Z

说明　该键使 LOGO 暂停.

§2.2　基本海龟作图命令

LOGO 语言可以用来指挥一个称为海龟 (Turtle) 的小小的三角图形在屏幕上移动，海龟移动所留下的轨迹便形成图形，海龟作图是 LOGO 语言的主要特色之一.

在 LOGO 的提示符“？”之下，键入任一个海龟作图命令后，便进入绘图模式．例如，键入

?CS

并按回车键，提示符“？”和光标便位于屏幕的左下角，而其正中出现一个尖头朝上的白色三角形，它能根据给定的海龟作图命令运动．下面逐个介绍全部海龟作图命令.

FORWARD 命令

格式：*FORWARD* (数字)

该命令带有一个数字的输入，其缩写为 FD，它使海龟沿当前方向按指定的输入前进．例如：

?FORWARD 50

海龟将沿当前指向前进50海龟步 (图2.2)．注意，命令 FORWARD 与参数50之间要用空格分隔开.

图2.2

海龟步的长度不同于日常使用的长度单位，它的实际长度取决于显示屏的尺寸——屏幕宽度为280海龟步，屏幕高度为240海龟步。

BACK 命令

格式：*BACK*（数字）

该命令也带有一个数字的输入，其缩写为BK，它使海龟按指定的参数后退，指向不变。例如：

?BACK 30

该命令使海龟后退30步（图2.3）

图2.3

RIGHT 命令

格式：*RIGHT*（数字）

该命令带有一数字的输入，其缩写为RT，它使海龟的指向按给定的参数右转(顺时针).例如：

?CS

?FD 15

?RT 40

?FD 20

此例中，海龟前进15步，右转40度后又前进20步，如图2.4所示。

图2.4

LEFT 命令

格式：*LEFT*（数字）

该命令也带有一数字的输入，其缩写为LT，它使海龟的指向按给定的参数左转（逆时针）。例如：

?CS

?FD 15

?RT 45

?FD 15

?BK 30

?LT 135

?FD 50

此例中，海龟按命令移动和转向，其经过的路径如图2.5所示。

图2.5

HOME 命令

格式：*HOME*

该命令不带输入，它使海龟回到屏幕中心，指向为正上方，我们称为起始位置，但该命令不清屏，海龟留下的轨迹依然存在。例如，当屏幕的显示如图2.5时，键入

?HOME

屏幕显示就成为图2.6所示的样子。

图2.6

CLEAN 命令

格式：*CLEAN*

该命令不带输入，它清除作图屏幕，但海龟的状态不变(包括它的位置和指向)。例如，

当屏幕的显示如图 2.5 时，键入

?*CLEAN*

屏幕显示如图 2.7 所示.

CLEARSCREEN 命令

格式：*CLEARSCREEN*

图2.7

图2.8

该命令不带输入，其缩写为 CS，它清除作图屏幕，使海龟回到起始位置，如图 2.8 所示.

最后，我们着重指出，凡是带有输入的命令，输入与命令之间都需要用空格分隔开.

§2.3 重复命令及作图

让我们首先应用前面学过的海龟作图命令画一个简单的几何图形：正方形. 于是，键入以下命令：

?*CS*
?*FD 50*
?*RT 90*
?*FD 50*
?*RT 90*

?*FD 50*
?*RT 90*
?*FD 50*
?*RT 90*

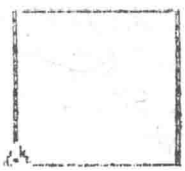


图2.9

执行各命令行，画出一个边长为50的正方形，如图 2.9 所示.

仔细观察上述作图程序不难发现，用来画出一个正方形的命令实际上只有两条：FD 50 和 RT 90，只是重复使用了它们.显然,这种重复性的工作应该让计算机去做. 为此， LOGO 提供了 REPEAT 重复命令.

REPEAT 命令

格式：*REPEAT* 〈重复执行的次数〉〔重复执行的内容〕

该命令要求两个输入，即重复次数和重复执行的具体内容. 重复执行的内容一定要用方括号括起来（即为表，详见第五章），否则计算机会给出错误信息而不予执行. 重复的内容可以是一条命令，也可以是一组命令序列,这时,命令与命令之间必须用空格分隔开. 例如：

?*REPEAT 4[FD 50 RT 90]*

将绘出一个正方形.

?*CS*
?*REPEAT 2[FD 35 RT 90 FD 70 RT 90]*

画出一个高为35，长为90的矩形，如图2.10所示.

图2.10

2.3.1 海龟总行程定理

如果我们键入

?CS

?REPEAT 5[FD 50 RT 72]

海龟将画出一个边长为 50 的正五边形，如图 2.11 所示．若再键入

?CS REPEAT 3[FD 60 RT 120]

海龟将画出一个边长为60的三角形，如图2.12所示．

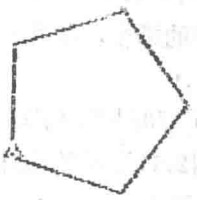


图2.11

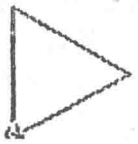


图2.12

仔细观察上述画正五边形，正三角形以及正方形的命令不难发现，它们属于同一型式，重复次数与转角的乘积总是 360 度．于是，我们有下表：

形 状	重 复 数	转角(度)	乘积(度)
三角形	3	120	360
四边形	4	90	
五边形	5	72	
六边形	6	60	
七边形	7	51.43	
八边形	8	45	
九边形	9	40	
十边形	10	36	

我们看到，不论海龟走怎样的路径，若要回到起始点，且指向不变，则必须旋转 360 度，或者 360 度的倍数，而与它画的边数无关．这就是所谓的海龟总行程定理．其所画图形的边数等于 360 除以转角．

我们知道正多边形的边数越多，它就越接近于圆．例如：

?CS REPEAT 36[FD 8 RT 10]

所画出的图形就很象一个圆（图2.13）．

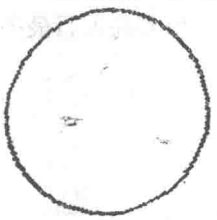


图2.13

2.3.2 重复命令的嵌套

重复命令 REPEAT 允许嵌套，也就是说在一个重复命令的执行内容中，允许包括另外的重复命令．例如：

?CS REPEAT 5[REPEAT 3[FD 40 RT 120] RT 72]

画出了五个正三角形，它们有一个公共的顶点，如图2.14所示．

?CS

?REPEAT 4[REPEAT 2[FD 40 RT 90 FD 20 RT 90] RT 90]

画出一个风车，如图2.15所示．若将上面的命令中最后一个RT 命令的参数改为 36，同时，将外层方括号前面的重复次数 4 改为10；

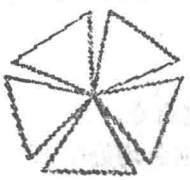


图2.14

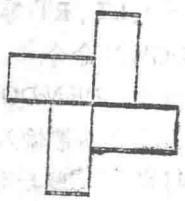


图2.15