微型计算机

IBM PC的原理与应用

张福炎 蒋新儿 李滨宇 编著

南京大学出版社

封面设计：郝庆祥

统一书号：15336·001
定　　价：7.00 元

微型计算机IBM PC的原理与应用

南京大学出版社

微型计算机
IBM PC 的原理与应用

张福炎　蒋新儿　李滨宇　编著

南京大学出版社

1984·南京

内 容 简 介

本书较全面地介绍了目前国际上广泛使用的十六位微型计算机系统 IBM PC 的原理与应用。全书分八章论述。前四章重点讨论了 IBM PC 的系统结构与系统软件，内容包括 PC 的逻辑结构、MS—DOS 操作系统、BASIC 语言和 UCSD P 系统及 UCSD PASCAL 语言。后面四章侧重介绍面向应用的软件，如表格处理软件 VisiCalc、Multiplan和 SuperCalc，数据库管理系统 dBASE Ⅱ，集成软件 Lotus 1—2—3 以及 IBM PC 的实用图形软件。为便于读者理解和掌握有关内容，书中还有例题150多个。本书资料新颖，内容丰富，叙述清楚，适合于广大从事微型计算机科研、生产、教学和应用开发的科技人员参考，也可作为高等院校计算机及有关专业的教材和教学参考书。

微型计算机 IBM PC 的原理与应用

张福炎　蒋新儿　李滨宇　编著

责任编辑　丁　益

*

南京大学出版社出版

（南京大学校内）

江苏省国营练湖印刷厂印刷

封面　江苏省国营射阳印刷厂承印

江苏省新华书店发行　各地新华书店经销

*

开本：787×1092　1/16　印张：33.5　字数：858(千字)

1984年11月第1版　1986年4月第4次印刷

印数：70,001—90,000

统一书号： 15336·001　　**定价：** 7.00元

微型计算机
IBM PC 的原理与应用

张福炎　蒋新儿　李滨宇　编著

南京大学出版社

1984·南京

内 容 简 介

本书较全面地介绍了目前国际上广泛使用的十六位微型计算机系统 IBM PC 的原理与应用。全书分八章论述。前四章重点讨论了 IBM PC 的系统结构与系统软件，内容包括 PC 的逻辑结构、MS—DOS 操作系统、BASIC 语言和 UCSD P 系统及 UCSD PASCAL 语言。后面四章侧重介绍面向应用的软件，如表格处理软件 VisiCalc、Multiplan和 SuperCalc，数据库管理系统 dBASE Ⅱ，集成软件 Lotus 1—2—3 以及 IBM PC 的实用图形软件。为便于读者理解和掌握有关内容，书中还有例题150多个。本书资料新颖，内容丰富，叙述清楚，适合于广大从事微型计算机科研、生产、教学和应用开发的科技人员参考，也可作为高等院校计算机及有关专业的教材和教学参考书。

微型计算机 IBM PC 的原理与应用

张福炎　蒋新儿　李滨宇　编著

责任编辑　丁　益

*

南京大学出版社出版

（南京大学校内）

江苏省国营练湖印刷厂印刷

封面　江苏省国营射阳印刷厂承印

江苏省新华书店发行　各地新华书店经销

*

开本：787×1092　1/16　印张：33.5　字数：858(千字)

1984年11月第1版　1986年4月第4次印刷

印数：70,001—90,000

统一书号：15336·001　　**定价**：7.00元

前　言

七十年代初期，由电子计算技术和超大规模集成电路相结合而诞生的微型计算机，因其体积小、功耗低、工作可靠和价格便宜等优点，发展迅速，应用广泛，已成为当今世界新技术革命的主要标志之一。

微型计算机技术发展速度很快，产品更新尤为迅速。被誉为第二代个人计算机的IBM PC是美国国际商业机器公司(IBM)于1981年开发成功的微型计算机产品。该机以具有十六位运算处理能力的微处理器Intel 8088为核心，有多种类型的扩充件可供选用，以便加接各种外围设备。特别是IBM PC配备了极其丰富的系统软件和应用软件，例如各种操作系统和程序设计语言；数据库管理系统、表格处理软件、文字处理软件、通讯软件、财务会计软件、商用图形软件、教育软件、游戏软件等。因此，IBM PC在小型事务处理、办公室自动化、教育、通讯、控制、工程设计等许多领域都得到了广泛的应用，是目前国际市场上最畅销的微型计算机机种之一。

IBM PC及其兼容机在我国的推广应用比较迅速。为了配合国内微型计算机生产、科研、教学和应用开发等工作的开展，我们根据近年来从事IBM PC应用开发和教学工作的一些经验，结合有关资料编写了这本书。

全书共分八章，大体可分成两个部分。前四章主要介绍IBM PC的系统结构和系统软件。首先比较详细地剖析了PC机的硬件结构与工作原理，特别是软硬件接口的一些基本方法。然后介绍了IBM PC的主操作系统——MS-DOS(2.00版)的基本概念、操作方法以及有关的内部结构。接着通过实例扼要地叙述了PC机四种版本的BASIC语言(磁带BASIC、磁盘BASIC、高级BASIC和编译BASIC)的一些特点。对交互性强、可移植性好的UCSD P操作系统及其支持的UCSD PASCAL语言也作了概括的阐述。书的后半部分侧重介绍了使用相当广泛的几种面向应用的软件：例如被誉为“大众数据库”的dBASE Ⅱ关系式数据库管理系统；通用性极好的表格处理软件VisiCalc，Multiplan和SuperCalc；去年居美国软件销售量第一位的集成软件Lotus 1－2－3；以及越来越受到用户关注的图形软件。

本书前面的一些章节要求读者对微型计算机的工作原理和程序设计语言具有一定的基础知识，但有关应用软件部分并不要求读者具有任何专门的知识。对那些只希望了解和掌握某种应用软件的读者，可以跳过书中的一些章节而直接阅读感兴趣的内容。

本书在编写过程中力求做到概念清楚，通俗易懂。为了帮助读者掌握有关软件的使用方法和技巧，书中列举了大约150多个例子，其中绝大部分都已在IBM PC机器上经过调试和运行。当然，它们的解法并非最佳，程序也未必十分完善，但对于读者深入掌握书中的内容以及开发IBM PC的实际应用，也许会有一定的参考作用。

本书第一、四、八章由张福炎执笔。第五、六、七章由蒋新儿执笔。第二、三章由李滨宇执笔。全书由张福炎主持编写并最后修改定稿。

本书承南京工学院王能斌和朱静华、杨祥金、孙志挥、董逸生、徐宏炳等同志审阅，并

提出了许多宝贵的意见。在编写过程中，还得到了本系陈世福同志及江苏省微电脑应用协会林德清同志的大力支持。谭学厚、黄强、沈默君、潘晶、赵沛、符建峰等同志在程序调试、书稿抄写、校对等方面做了大量工作，陆西骋同志在本书校对、付印过程中也做了不少工作。编者在此谨向他们表示衷心的感谢。

由于编写时间仓促以及限于编者水平，书中错误和不妥之处在所难免，敬请读者不吝批评指正。

编　者　1984年8月

于南京大学计算机科学系

目　录

第一章　IBM PC 的硬件与系统结构

IBM PC是一种新型个人计算机，由美国IBM公司采用微型计算机系统中的较新技术设计而成。目前该机生产数量已达一百万台以上，成为国际上广泛使用的微型计算机系统之一。

本章介绍IBM PC和PC/XT的系统部件（中央处理器和存贮器）和常用的几种基本选件，如字符、图形显示器及其控制器，打印机及其控制器，异步通讯控制器和软盘及硬盘控制器等。本书并不准备详细描述它们的电子线路及有关的物理过程，而主要是从软件和应用开发的角度出发，讨论它们所呈现在程序设计方面的特性和逻辑功能，以及这些硬件成分相互之间的接口关系与规程，也就是说，主要是从体系结构的角度来介绍IBM PC。

第一节　概　　述

IBM个人计算机最小的硬件配置只需要三个部分，即键盘、显示器和一个安装了系统板（上面有CPU和存贮器）及一块选件板（显示控制器）的主机箱。这种最小配置仅能使用系统内部固化了的BASIC语言，一般适合于教学或开展简单的数据处理和控制方面的应用。为了扩大IBM PC的应用范围，它的存贮容量和输入输出功能，以至它的运算处理能力等，都需要作进一步的扩充，例如：

* 内存贮器容量　　系统板上可以扩充到64KB，PC/XT可以扩充到256KB。添加存贮器选件板之后，还可以进一步扩充。如果把系统中只读存贮器等的容量也计算在内，则系统的最大内存容量可达到1MB（1兆字节）。

* 外存贮器　　系统可用录音机作为外存贮器，但更常用的是在主机箱内安装两台5吋软磁盘驱动器，每台驱动器的存贮容量可达320KB（或360KB）。PC/XT则可安装一台温彻斯特硬磁盘机（容量在10MB以上）和一台软盘机。如果需要的话，还可购买扩充机箱再增加两台软盘机。

* 运算处理能力　　系统板上可以增加一个大规模集成电路芯片——协处理器8087，从而使运算处理的速度提高几十倍。

* 输入输出设备　　单色显示控制器插板上的并行打印机接口，可以连接一台打印机作为硬拷贝输出设备，通常使用的是每秒钟打印80个字符的点阵式打印机。为了具有显示彩色图形的功能，可在机箱内增加一块彩色图形选件板和一台彩色监视器。为了具有数据通讯能力，可增加同步或异步通讯控制板。这样，既能实现PC与其它计算机的通讯，还可以利用一个或几个标准的串行接口连接其它种类的外部设备，如绘图仪、打印机、图形数字化仪、汉字终端等等。如果需要使用操纵杆（Joystick）或电位控制器（Paddle）进行交互式图形显示和做游戏，则可配置游戏控制器选件板。

总之，IBM PC的硬件配置比较灵活，可以适应许多应用领域的不同要求。图1—1是系统硬件配置的一个简单概括。

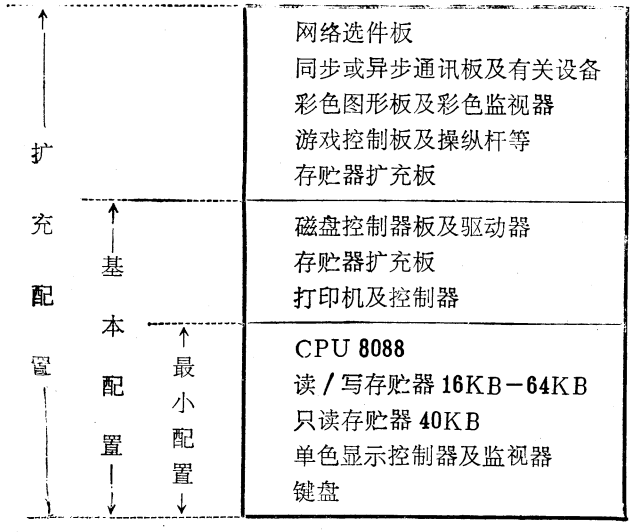


图 1—1　IBM PC 的硬件配置

第二节　系统的基本逻辑结构

从物理结构的角度来看，IBM PC 的所有运算处理、存贮、控制和输入输出接口电路等都集中在主机箱内的一块大底板（下称“系统板”）和各种选件板上。选件板由用户根据应用的需要插入系统板上的槽口（插座）内，它们与底板形成一个整体进行工作。下面就系统板的结构与功能、存贮器布局、输入输出通道及键盘等四个方面来介绍 IBM PC 的基本逻辑结构与工作原理。

1．系统板的结构与功能

系统板水平地安装在机箱内，按功能可以划分为五个部分：中央处理器，读/写存贮器，只读存贮器，输入输出控制以及输入输出通道。图 1—2 是它的示意图。

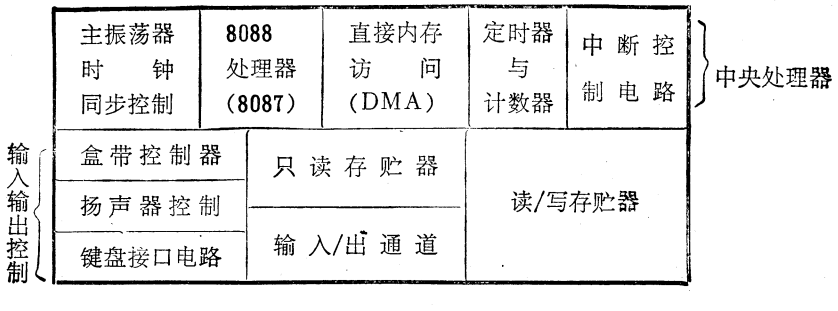


图 1—2　系统板的功能划分

中央处理器的核心部分是 Intel 8088 微处理器及有关电路。8088 微处理器可以处理 16 位的二进制数据。它有 20 根地址线，寻址能力达到 1 兆字节（1MB）。需要时还可以添加浮点运算处理器 Intel 8087 芯片，从而使数学运算速度大大提高。

支撑 8088 工作的辅助电路有：主振荡器及时钟信号发生器 8284A，四通道 20 位直接内存访问（DMA）控制器 8237A—5，三路 16 位定时器/计数器电路 8253—5，8 级中断排优控制器 8259A 等等。

主振荡器晶体的频率为 14.31818MHz，经过三分频之后得到 4.77MHz 的中央处理器时钟信号，即每个时钟信号的周期为 210ns。

四通道 DMA 控制器的作用是保证在不妨碍 CPU 操作的情况下，提供输入输出设备和内存贮器之间的高速数据传送。其中三个通道提供给输入输出总线使用，每一次数据传送需要 5 个时钟信号，即 1.05μs。第 4 个通道专门用于进行动态存贮器的刷新，每次刷新需要 4 个时钟脉冲即 840 ns。

三路定时器/计数器在系统中的作用分别为：1 号定时器用来定期地向第 4 个 DMA 通道请求一次假的输入输出传送，从而引起一次存贮器的读操作，以达到周期性地进行动态存贮器刷新的目的。0 号定时器被系统用作通用的计数器，它是实现“日时钟”（即显示时、分、秒）的基础。2 号定时器用来支持扬声器的声调发生器，以便发出各种需要的音调。

Intel 8259 中断控制器用来对输入的八个中断信号排出优先次序。其中优先级最高的 0 级中断信号来自系统板上的 0 号定时器，当它作为日时钟使用时，每秒钟产生 18.2 次中断；优先级次之的 1 级中断来自键盘控制电路，键盘每输入一次，就会引起一次 1 级中断。其它的六个中断信号都来自插在输入输出槽口的选件板上，只有在插入有关选件时，才会有相应的中断信号产生。下面是常用的基本选件所产生的中断信号的优先级：

* 同步通讯（SDLC）控制器中断（第 3 级）
* 异步通讯控制器中断（第 4 级）
* 硬磁盘中断（第 5 级）
* 软磁盘中断（第 6 级）
* 打印控制器中断（第 7 级）

系统板上的第 2 部分是只读存贮器（ROM）。在底板上可以插入 6 片 8KB 的只读存贮器芯片，总容量达 48KB。但一般只安装 40KB 的固件，其中包括盒式磁带版本的 BASIC 解释程序和一组叫做 BIOS 的基本输入输出子程序。BIOS 通常包含有下列内容：

* 加电后的硬件测试程序
* 系统配置（如存贮器大小，选件板的种类等）的分析程序
* 显示器、打印机、键盘、异步通讯控制和软磁盘等的驱动程序
* 日时钟控制程序
* 盒式磁带操作系统
* 软盘的引导装入程序

IBM PC 基本型系统板上的读/写存贮器最多只能安装 64 KB，而 PC/XT 却可以在系统板上安装 256 KB。这是因为前者用的是 16K×1 的 RAM 芯片，后者使用了 64K×1 的芯片。存贮器芯片的工作周期为 410 ns，访问时间为 250 ns。存贮器的每 8 位（一个字节）都附加有一位奇偶检测位。存贮器工作时，如发现有奇偶检测出错，将向 CPU 发出一个不可屏蔽的中断信号 NMI，然后由系统中相应的软件处理。

系统板上还包含有用于连接盒式磁带录音机、键盘和扬声器的输入输出控制电路。盒式录音机控制电路允许使用质量较好的普通录音机作为外存贮器，它们可以通过话筒口或辅助

输入口进行互连。在程序的作用下，计算机还能对录音机的起停进行控制，对数据读写的正确性进行循环码检查等等。数据的读写速率大约在1000到2000波特之间。

键盘控制电路用于连接键盘的串行接口。当收到了来自键盘的一个完整的扫描码之后，就向系统发出一次中断请求（中断优先级为1），相应的软件就会对该扫描码作出必须的处理。

为了具有音响输出的能力，系统板上还装有一个2¼吋的扬声器以及有关控制电路和驱动电路。控制电路能以三种不同的方式驱动扬声器发音：

① 程序直接控制某寄存器的特定位，其值的变化将给出一串脉冲波形，供扬声器发音；

② 2号定时器/计数器在程序的控制下，产生一定规格的波形送给扬声器；

③ 第②种方式里的定时/计数器的时钟输入信号，还可以进一步由程序控制的某寄存器位来进行调制，以便修改送到扬声器去的波形。

上述三种驱动方式可以同时执行，这样，输出的音响将会达到更好的效果。

2. 存贮器空间的布局

图1—3是IBM PC内存贮器空间的布局。可以看出，整个存贮空间的总容量为1兆字节，其中只读存贮器（ROM）位于内存空间的尾部，一般只安装40KB，但可使用ROM

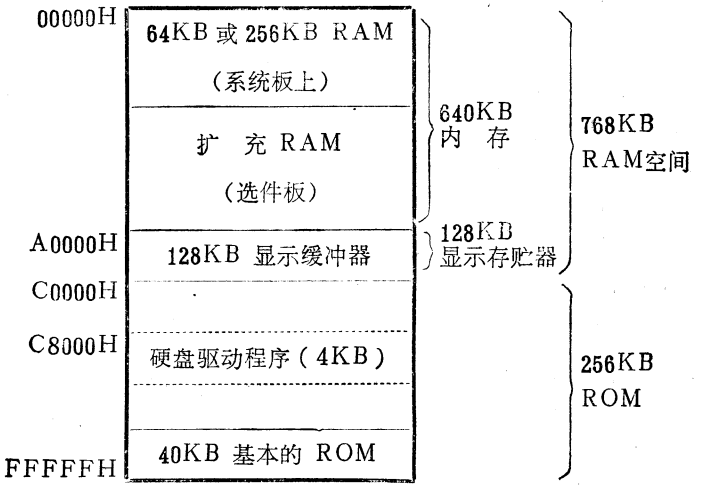


图1—3 存贮器空间的布局

选件板扩充到256KB。这些ROM扩充板中存放的内容可以作为新的外围设备的驱动程序或者是汉字字库，也可以存放完整的应用软件。随机读写存贮器从0号地址开始，可以最大扩充到640 KB。单色显示器和彩色显示器的显示缓冲器均位于第640KB—第768KB的空间范围内（即十六进制表示的地址A0000H—BFFFFH）。目前，单色显示器的缓冲器共4KB，地址为B0000H—B0FFFH；彩色图形显示器的缓冲器（即刷新存贮器）共16 KB，其地址范围为B8000H—BBFFFH。

3. 输入输出通道

输入输出通道也叫输入输出总线，它是8088微处理器总线的一个扩充。实际上，它仅仅是系统板上5个或8个对应插脚相互连接的62线插座以及一些附加的控制电路。它包括8位的双向数据总线，20位的地址总线，6根中断信号线，3根DMA控制线，4根电源线，以及其它各种控制线共62根。

IBM PC通道上输入输出设备的编址可达512个。例如，系统板上的有关芯片或控制电路的输入输出地址和常用选件板上的输入输出地址分别如表1-1和表1-2所示。

表1-1　输入输出地址分配表（之一）

芯片或电路名称	占用地址数	地址码(16进制)
DMA控制器(8237)	16	00—0F
DMA页面寄存器(74LS670)	4	80—83
中断控制器(8259)	2	20—21
定时器/计数器(8253)	4	40—43
并行接口(8255)	4	60—63
NMI屏蔽寄存器	1	A0

表1-2　输入输出地址分配表（之二）

选件板名称	占用地址数	地址码(16进制)
硬盘控制器	16	320—32F
软盘控制器	8	3F0—3F7
单色显示器/并行打印机	16	3B0—3BF
彩色图形显示器	16	3D0—3DF
异步通讯控制器	8	3F8—3FF
BSC同步通讯控制器	10	3A0—3A9
SDLC同步通讯控制器	13	380—38C
游戏控制器	16	200—20F

输入输出通道的62根信号线在插座上的排列次序及信号名称见图1—4，这些信号线可以按其功能分成下列五类：

（1）地址线A19—A0（20根）

地址线用来指出内存地址或输入输出地址，当用来指出输入输出地址时，A19－A10无效。地址线上的信号（地址码）可以由CPU生成，也可以由DMA控制电路生成，A19为最高位，A0为最低位，寻址空间可达$2^{20}=1MB$。

（2）数据线D7—D0（8根）

八根数据总线用来在中央处理器、存贮器和各种输入输出控制器之间传送数据，它们是双向总线，每次可以传送一个字节。

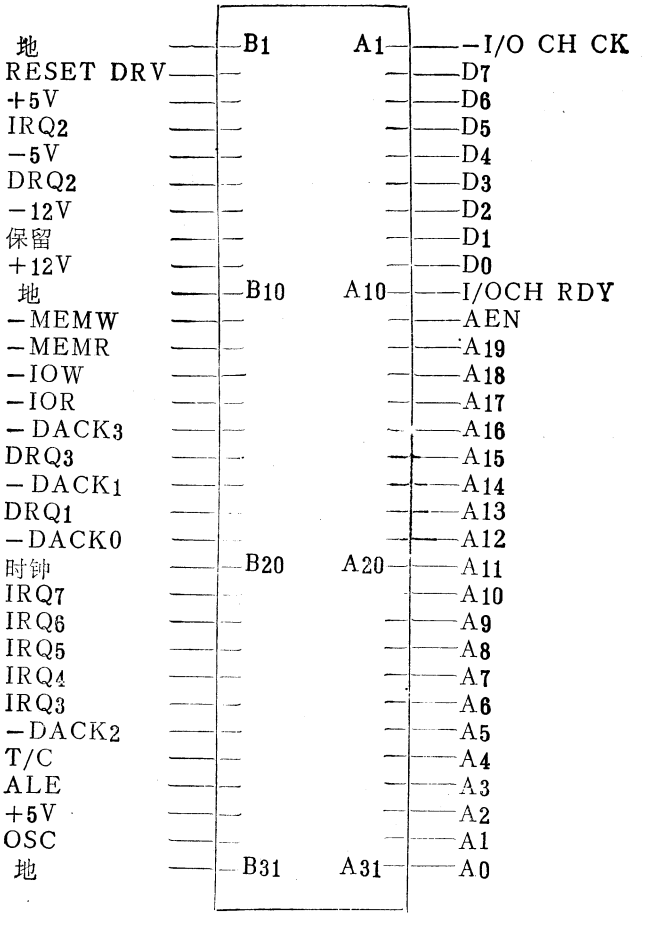


图 1—4　输入输出通道信号名称与位置

（3）控制线（21 根）

控制信号线中，6 根中断请求线（IRQ2—IRQ7）可作为输入输出控制器向中央处理器发出中断请求信号之用。3 根直接访内请求线 DRQ1—DRQ3 用作外部设备请求 DMA 服务之用。当某外部设备控制器需要直接与内存交换数据时（例如软磁盘的读或写操作），相应的请求线电平升高，直到对应的认可信号（-DACK0—-DACK3）之一发出后才能降下。-IOR 和-IOW 一般用来指出 CPU 正在执行的是输入指令还是输出指令，所以，数据总线上的数据是与输入输出控制器有关的。而-MEMR 和-MEMW 两个信号由 CPU 或 DMA 控制器发出，用来要求内存贮器进行数据的读出或写入。RESET DRV 是一个总清信号，它用来使系统初始化，通常在系统加电时或者利用键盘进行总清（Ctrl、Alt 和 Del 三键同时按下）时发出。AEN 信号用于 DMA 操作，当它为高电平时，所有地址线、数据线及-IOR、-IOW、-MEMR 和-MEMW 等等均受 DMA 控制器的控制，而不再受 CPU 控制。ALE 信号的作用是使 CPU 给出的地址码临时锁存起来，此信号也用于输入输出通道作为有效的 CPU 地址的指示（当与 AEN 配合时）。T/C 是一个脉冲信号，在进行 DMA 方式传送数据时，一旦达到了预定的字节传送个数，就会发出 T/C 信号。

（4）状态线（2根）

－I/O CH CK用来向CPU指出输入输出通道上的扩充存贮器或外设发现了奇偶错误，如果允许的话，它会向CPU发出一个不可屏蔽中断（NMI）。－I/O CH RDY是“就绪”信号，它的作用是使CPU与较慢速的输入输出控制器芯片或扩充存贮器芯片同步工作。正常情况下它应为高电平状态，当慢速存贮器芯片或输入输出控制芯片被CPU访问时，“就绪”信号降为低电平，于是引起CPU产生的存贮器读写周期降低到840ns，输入输出读写周期降低到1.05 μs。

（5）辅助线和电源线等（11根）

OSC是系统板提供给输入输出通道的主振荡器信号，周期为70ns（即频率为14·31818 MHz），占空系数为50%。CLK是系统的时钟信号，它由主振信号三分频获得，周期为210 ns（4.77MHz），占空系数为33%。此外，输入输出总线上还有地线及+5V、－5V、+12V和－12V等电源线以及一根备用线。

4. 键　盘

IBM PC的键盘是与主机箱分开的一个独立装置，它通过一根5芯的接口电缆与主机相连接（图1—5）。

按键采用电容技术。键盘中内藏的Intel 8048单片微型计算机用来执行键盘扫描功能。

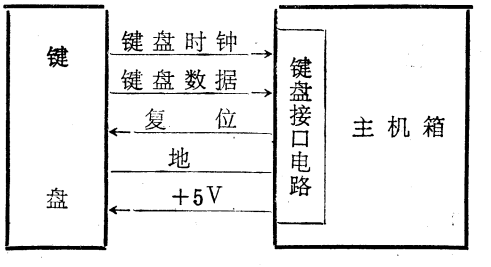


图1—5　键盘与主机箱的接口

整个键盘包含83个键，分成三组：中间是标准的打字机键盘，左面为十个功能键，右面为一个16键的小键盘。

键盘中的Intel 8048单片机可以完成多种功能。例如，加电时或系统需要时对键盘进行检测（ROM的循环码检测，随机存贮器测试，按键测试等），键盘扫描，消去重键，自动重发，扫描码的缓冲，以及与主机进行通讯等等。

由于从键盘送入主机的不是通常的ASCII码，而是键盘扫描码（即每个按键的位置码），然后再通过主机ROM中的键盘驱动程序（是BIOS程序的一部分）来定义其逻辑意义，因而具有很大的灵活性。

在BIOS键盘驱动程序的解释下，IBM PC的键盘除了提供有通常的输入ASCII字符（编码为20H—7FH的96个可打印字符及编码为00H—1FH的32个控制字符）的功能之外，它还具有如下功能：

* 直接向系统输入某字符的编码　IBM PC允许处理8位的全部256种组合的字

符编码，输入方法是先按下 Alt 键，然后再从右面的数字小键盘上打入相应字符的编码（十进制值 000—255）。

* 功能键　十个功能键的作用在不同软件系统中有不同的定义，它们在有关章节中再作介绍。使用功能键的优点是操作方便，节省键盘输入时间。

* 光标控制与编辑键　在 NumLock 键未按下时，数字小键盘上的↑（光标上移）、↓（光标下移）、←（光标左移）、→（光标右移）、Home（光标回屏幕左上角）、End（光标移到屏幕右下角）、PgUp（光标不动，屏幕画面向上翻滚一行）和 PgDw（光标不动，屏幕画面向下翻滚一行）等都起着光标控制键的作用。而 Ins（嵌入一个字符）和 Del（删除一个字符）起着编辑的作用，它们为用户提供了不少方便。

* 专用功能的实现　键盘上某些键的同时使用，可以实现一些专用功能，例如：

Ctrl+Alt+Del	使系统复位（相当于系统关电后再次加电一样）。
Ctrl+ScrLock	终止正在执行的程序。
Ctrl+NumLock	暂停系统的操作，直到按下任何一个键再继续下去。
PrtSc	打印屏幕上显示的内容。
Ctrl+PrtSc	使任何键盘输入及系统输出的内容，同时在屏幕上和打印机上输出，直到再一次发出这个“命令”为止。

关于 IBM PC 系统所使用的 8 位字符的定义、编码、键盘输入方法以及输出显示符号一览表，请参看附录 I 和附录 II。

第三节　中央处理器

IBM PC 的中央处理器是 Intel 8088。这是一种在 Intel 8080/8085 和 Intel 8086 基础上发展起来的准 16 位微处理器芯片。它既能处理 8 位数据，又能处理 16 位数据，直接和 Intel 8086 的软件及 8080/8085 的硬件与外围控制芯片兼容。在工艺上它采用 N 沟道耗尽型硅栅技术（HMOS），封装在 40 条引线的管壳中，时钟频率为 5MHz，电源为 5V。其主要特点为：

* 8 位的数据总线接口。
* 16 位的内部结构，具有十四个 16 位的寄存器。
* 20 条地址引线，直接寻址能力可达 1 兆字节。
* 24 种操作数寻址模式。
* 软件与 Intel 8086 兼容。
* 具有对字节、字和字组（Block）进行操作的能力。
* 既能执行 8 位又能执行 16 位的二进制或十进制算术运算，包括乘除法运算在内。
* 可配用 8155、8355 和 8755A 等外围芯片。

1. 寄存器结构与数据通路

Intel 8088 从功能上来说分成两大部分：总线接口单元 BIU（Bus Interface Unit）和执行单元 EU（Execution Unit），如图 1—6 所示。

BIU 负责与存贮器接口，即 8088CPU 与存贮器之间的信息传送都是由 BIU 进行的。具

体地说，即 BIU 负责从内存的指定区域取出指令，送到指令流队列中排队；在执行指令时所需的操作数，也由 BIU 从内存取出，传送给 EU 部分去处理。

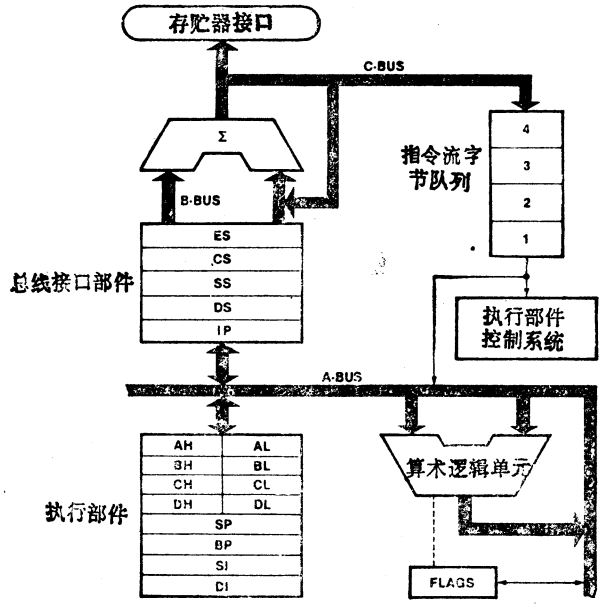


图 1—6　8088 的寄存器结构与数据通路

EU 部分负责指令的执行。这样，8088 的取指令部分与执行指令部分是分开的。一条指令正在执行的时候，就可以取出下一条（甚至多条）指令，在指令流队列中排队。由于这样，一条指令执行完了就可以立即执行下一条指令，减少了 CPU 为取指令而等待的时间，提高了整个运行速度。

执行单元 EU 由九个 16 位的寄存器及一个 16 位的算术逻辑单元（即加法器）所组成。这些寄存器的主要分工大致如下：

* AX 寄存器　这是累加器，用于字乘法，字除法，字输入输出，其中 AH 还用于字节乘法，字节除法，AL 用于字节乘法，字节除法，字节输入输出，字节翻译，十进制算术运算等。
* BX 寄存器　这是基数寄存器，也用来进行字节翻译操作。
* CX 寄存器　串操作，循环次数。其中 CL 用于可变次数的移位和循环。
* DX 寄存器　这是数据寄存器，还用来进行字的乘除法，间接的输入输出。
* SP 寄存器　堆栈指针。
* BP 寄存器　基数指针。
* SI 寄存器　源操作数的索引寄存器。
* DI 寄存器　目的操作数的索引寄存器。
* FLAGS 寄存器　状态标志寄存器，用来保存指令执行后的结果状态，共 9 位。

8088 与 8080/8085 是兼容的，所以它也能处理 8 位的数据。上述 AX、BX、CX、和 DX

四个寄存器可以作为八个8位寄存器使用（这时，我们使用符号AH、AL、BH、BL、CH、CL、DH和DL表示），相当于8080/8085中的A、HL、BC、和DE四个寄存器；SP寄存器与8080/8085中的SP相当；FLAGS寄存器与F寄存器相当，但它增加了4位，所以需要占用两个字节。

2．存贮器组织与地址的形成

Intel 8088有20条地址引出线，它的寻址能力为$2^{20}=1$兆字节。所以，在一个以8088为CPU的系统中，可以有多达1MB的内存贮器，这1MB的存贮器空间在逻辑上是一个线性的阵列，地址（十六进制表示）从00000到FFFFF。更进一步，我们还可以把它分成若干段，例如：程序代码段，数据段，堆栈段等等，每段最大可达64KB。段的划分（用它的起始地址来表示）由总线接口部件中的四个段寄存器CS、SS、DS、和ES来给出，这些寄存器虽然都是16位的，但由于系统规定它们的单位为2^4，所以实际上它们都代表着20位的地址码，不过最低4位的值始终是“0”而已。图1—7是存贮器的组织以及它与段寄存器之间的关系。

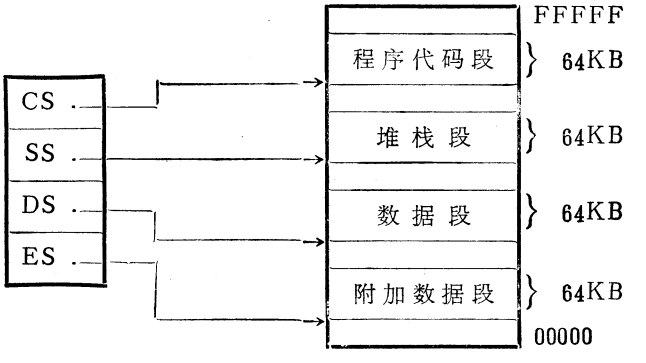


图1—7　存贮器组织与段寄存器的关系

8088的寄存器都是16位的，用来存放存贮器地址码的寄存器如指令指示器（IP），堆栈指示器（SP），基地址指示器（BP），两个变址器（SI和DI）等也不例外。那么怎样才能产生一个长度为20位的真正的物理地址呢？图1—8表示了物理地址的形成过程。

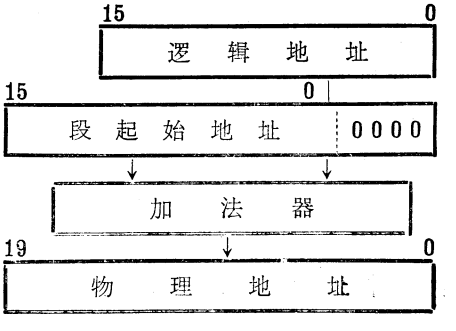


图1—8　物理地址的形成

从图中可以看出，每当访问内存需要形成一个 20 位的地址码（称为“物理地址”）的时候，CPU 就会根据操作的性质和要求选择某一个段寄存器，把它左移四位，再与一个 16 位的逻辑地址相加，其结果就是送出到外部地址总线上去的真正的物理地址。

例如，每当是取指令的时候，就会选择程序码的段存寄器 CS，加上指令地址指示器 IP 的内容之后，形成要读取的指令的物理地址；而当涉及到一个堆栈的操作时，CPU 就会自动选择堆栈段寄存器 SS，再加上堆栈指示器 SP 的内容后，得到真正的 20 位栈顶地址；当需要向内存读或写一个操作数时，CPU 所选择的段寄存器将是数据段寄存器 DS 或附加数据段寄存器 ES，再加上 16 位的偏移量（可能是指令中的直接地址，也可能是某寄存器的内容，或者是它们两者之和，详见下述。）后形成操作数的物理地址。操作数可以是一个字节，也可以是一个字，此时，物理地址指出的是该字的低位字节，加“1”后才是它的高位字节的地址。

上述 8088 内存的分段方法，既解决了把内存空间扩大到 1MB 的问题，使用起来也十分方便。假设有一个“任务”（也叫做一个“作业”），它所需要的数据空间、堆栈空间、以及程序代码的长度一共不超过 64KB，则可以在程序开始时令 CS、SS、DS 等寄存器内容相等，程序就能正常工作。早期在 8080/8085 微型机上开发的大部分程序就正是这种情况。

如果某个任务的数据区、堆栈区、及程序代码的长度分别各不超过 64KB，则需要在程序开始时分别给 DS、SS、CS 置值，此后在程序执行过程中就不必再考虑这些寄存器了。

如果程序中要用到的数据区超过 64KB，或者要求从两个（或多个）不同区域中去存取数据，则每次在存取不在同一段内的数据前，需要给数据段寄存器置适当的值才行。

这种分段方法对于程序的重定位特别方便。所谓“重定位”，就是要求同一程序代码（包括它所处理的数据）能装在内存的“任意”不同区域中运行，而不必修改代码本身。要做到这一点，对 8088 来说是比较方便的。除了在程序中使用相对转移指令等之外，只要在运行该程序时，首先根据程序、数据等的装入位置，向各有关段寄存器中置入相应的值就行了。图 1—9 是使用段寄存器使程序重定位的示意图。

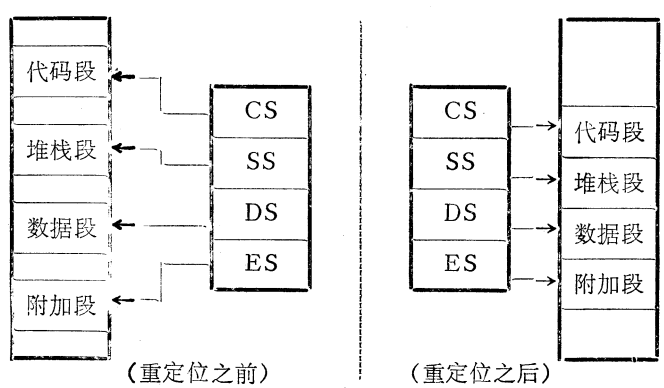


图 1—9　使用段寄存器使程序重定位

8088 的内存空间中，某些存贮单元是专供 CPU 特定操作而用的。例如，地址为FFFF0—FFFFF 的十六个存贮单元专门用于系统复位，其中应包括一条转移指令，当系统复位或系

统加电时，CPU 总是首先执行位于 FFFF0 中的指令，然后再转向系统的初始化子程序执行。而地址 00000—003FF 却是用于中断处理的（图 1—10）。

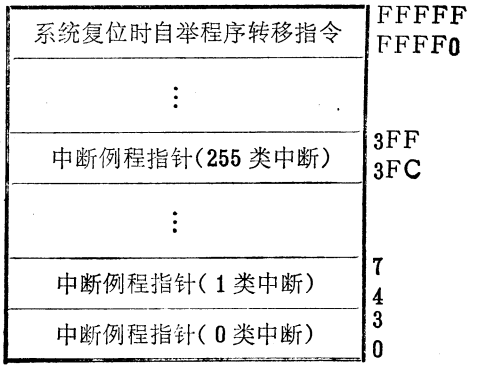


图 1—10　系统专用的存贮器位置

由于系统最多允许有 256 种不同的中断服务例行程序，因此，每种中断处理例程的段地址和段内偏移量（逻辑地址）就需要用四个字节来表示，它们依次存放在内存 00000—003FF 的 1024 个字节单元中。当发生某种中断需要处理时，CPU 先从相应单元中读出中断服务例程的段地址及段内偏移量，经相加后，得到真正的转移地址，然后才转向中断服务例程去执行。

3．8088 芯片硬件

8088 是四十条引线封装的大规模集成电路。由于引出线的数量受到了限制，所以不少引线脚的逻辑含义具有两种定义。参看图 1—11，下面对这些外引线的逻辑意义分组进行概括的介绍。

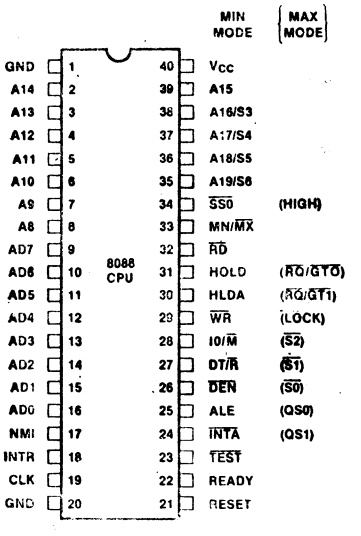


图 1—11　8088芯片的引线排列

* 地址/数据线（AD7—AD0）

这八根引线有双重作用。在CPU访问存贮器或外设时，它们先表示的是地址码的低8位。当外电路中的地址锁存器把这些地址锁存下来以后，它们就可以作为数据线来传送读/写的8位的数据字节了。

* 地址线（A15—A8）

这是单一用途的八根地址线，用来表示地址码的第15—第8位。

* 地址/状态线（A19/S6，A18/S5，A17/S4，A16/S3）

这四根线在CPU访问存贮器时它们表示地址码的最高4位。此后，它即用来表示状态信息。其中S6不使用，始终为“0”；S5表示FLAGS寄存器中的“中断允许位”的状态；S4和S3用来指出那一个段寄存器正在被使用，它们的编码如下：

A17/S4	A16/S3	意　　义
0	0	扩充数据段寄存器
0	1	堆栈段寄存器
1	0	程序代码段寄存器
1	1	数据段寄存器

* 中断信号线（INTR，NMI，$\overline{\text{INTA}}$*）

INTR是可屏蔽中断的请求信号，只有当FLAGS寄存器中的中断允许位为“1”时，CPU才会响应INTR而进入中断响应周期；NMI则是不可屏蔽中断的请求信号，它使得CPU在现行指令执行完毕后立即引起中断处理；$\overline{\text{INTA}}$是中断认可信号（信号名称上面带“——”或前面带有负号“—”者，表示该信号在低电平时起作用），它在中断响应周期内出现，用来读取中断控制器（例如8259A）中的一个字节，以便识别中断的具体类型，该字节乘以“4”后即可用来查表，以取得相应中断服务例程的入口地址。

* 芯片复位及状态线（RESET，READY，$\overline{\text{TEST}}$）

RESET信号使处理器立即结束现行操作，内部电路状态复位后，即转向执行在FFFF0处的指令；READY是当CPU访问较慢速的存贮器或I/O芯片时，后者返送回来的状态信号。若后者不能及时完成数据的读写操作，则把READY信号降为低电平，于是CPU会在T3周期之后插入等待周期（T_{WAIT}），直到READY信号恢复为高电平后才进入T_4周期，完成数据传送；$\overline{\text{TEST}}$信号是在CPU执行到“WAIT”指令时才起作用的。其时，若$\overline{\text{TEST}}$信号为高电平，则CPU就进入空转状态，否则，CPU将继续执行下一条指令。

* 总线控制及状态标志线（IO/$\overline{\text{M}}$*，DT/$\overline{\text{R}}$*，$\overline{\text{SSO}}$*，$\overline{\text{RD}}$，$\overline{\text{WR}}$*，ALE*，$\overline{\text{DEN}}$*，HOLD*，HLDA*）

IO/$\overline{\text{M}}$表示CPU是访问存贮器还是访问输入输出设备，DT/$\overline{\text{R}}$表示数据传送的方向（即CPU读还是写），$\overline{\text{SSO}}$与上面两个状态线构成表1-3所给出的八种组合，这样，系统就能完整地识别出总线的现行状态了。

$\overline{\text{RD}}$和$\overline{\text{WR}}$是芯片输出的选通信号，它们分别表示处理器正在执行存贮器或输入输出端口的读操作和写操作。

表 1-3　8088 的总线状态标志

IO/$\overline{M}$	DT/$\overline{R}$	$\overline{SSO}$	总线状态
0	0	0	中断认可状态
0	0	1	读输入输出端口
0	1	0	写输入输出端口
0	1	1	停止状态
1	0	0	读指令
1	0	1	读存贮器
1	1	0	写存贮器
1	1	1	无总线周期

ALE（地址锁存允许信号）表示允许把地址码锁存到 8282/8283 地址锁存器中去，通常它在 T_1 时产生。$\overline{DEN}$（数据允许信号）表示允许 8286/8287 发送器中的数据输出，它在每次访问内存或访问输入输出端口时产生，或中断响应期间的 T_2 中间到 T_4 中间产生。

HOLD（保持信号）是 CPU 之外的其它主设备要求占用总线进行直接内存访问时，向 CPU 发出的总线请求信号。当 CPU 识别出 HOLD 信号后，就输出 HLDA（保持认可）信号。同时，处理器使地址线、数据线和有关的一些控制信号线浮空，表示它让出了总线。一旦 CPU 检测出 HOLD 信号变低后，它也使 HLDA 变低，同时又再次占用了总线。

* 辅助线（CLK，VCC，GND）

四根辅助线中，CLK 是时钟线，用作处理器和总线控制器的基本同步信号；它是占空比为 33% 的不对称脉冲信号；VCC 是电源线（+5V ± 10%）；GND 是两根地线。

* 结构模式标志（MN/$\overline{MX}$）

因为 8088 仅有 40 条引线，所以引线的单一定义无法满足使用的要求，为此，MN/$\overline{MX}$（Minimum/Maximum）信号就用来表示 8088 是使用在最小结构组态（当MN/$\overline{MX}$ 接通 +5V 电源时）还是使用在最大结构组态（当MN/$\overline{MX}$ 接地时）。当使用在最大组态中时，前面介绍过的信号线中凡有“*”者均按下表另行定义。

表 1-4　8088 的最小组态与最大组态

引脚号	最小结构组态	最大结构组态
34	$\overline{SSO}$	（高电平）
31	HOLD	$\overline{RQ}/\overline{GT0}$
30	HLDA	$\overline{RQ}/\overline{GT1}$
29	$\overline{WR}$	$\overline{LOCK}$
28	IO/$\overline{M}$	$\overline{S}_2$
27	DT/$\overline{R}$	$\overline{S}_1$
26	$\overline{DEN}$	$\overline{S}_0$
25	ALE	QS_0
24	$\overline{INTA}$	QS_1

在最大结构组态中，8088 必须配用一个总线控制器芯片 8288，由它来解释状态码 $\overline{S}_0$、

$\overline{S_1}$ 和 $\overline{S_2}$ 所表示的总线状态，并产生 DT/$\overline{R}$、DEN、MCE/$\overline{PDEN}$、ALE 等控制信号，图 1—12 是 IBM PC 中总线控制器 8288 的使用方式，有关 8288的详细说明请查阅有关资料，这里不再详述。

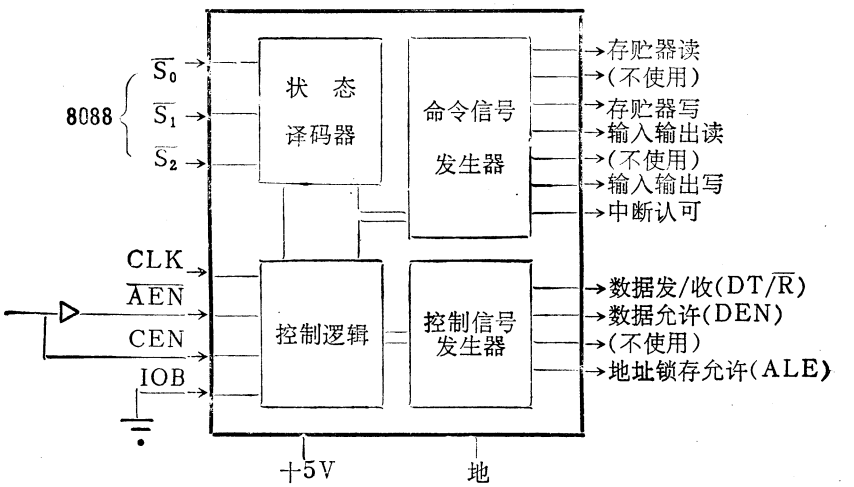


图 1—12 IBM PC 中总线控制器 8288 的使用

其中，S_0、S_1和S_2 的编码和定义如下：

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	处理器状态	8288命令
0	0	0	中断认可	$\overline{INTA}$
0	0	1	读 I/O 端口	$\overline{IORC}$
0	1	0	写 I/O端口	$\overline{AIOWC}$
0	1	1	停	(无)
1	0	0	读指令	$\overline{MRDC}$
1	0	1	读存贮器	$\overline{MRDC}$
1	1	0	写存贮器	$\overline{AMWC}$
1	1	1	无源状态	(无)

使用了 8288 之后，就使得 8088 的一些引脚起着其它的控制作用，它们是：

* $\overline{RQ/GT0}$ ，$\overline{RQ/GT1}$ （输入输出）

这些请求/同意信号，是由外部的主控设备用以促使处理器在现行总线周期结束时让出总线用的。

* $\overline{LOCK}$

此信号有效时，表示其它的总线主设备不能获得对系统总线的控制权。该信号由前缀指令 "LOCK" 使其有效，并一直保持到下一条指令执行完毕。

* QS1，QS0

它们提供了 8088 内部指令队列的状态以便外部进行追踪。队列状态的定义如下，它们在时钟信号 CLK 周期内有效。

QS_1	QS_0	特　　性
0	0	无操作
0	1	操作码第1字节出队
1	0	队列空
1	1	后继字节出队

4．指令格式与寻址方式

8088 的指令长度是可变的，它可以在一个字节到六个字节之间变化（图 1—13），这是由操作码、寻址方式以及操作数的长短等因素来决定的。

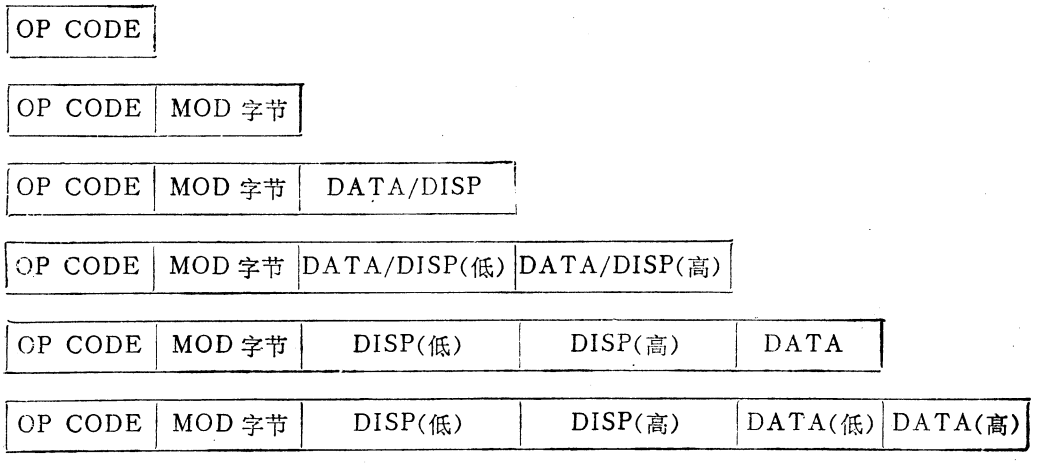
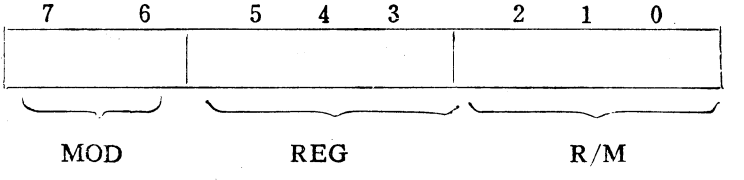


图 1—13　8088 不同长度的指令格式举例

8088 提供了多种多样的寻址方式，这主要是通过指令中的操作码（OP CODE）和寻址模式字节（MOD 字节）来决定的。MOD 字节由三部分组成，它的格式为：



其中，MOD 字段用来区分是存贮器寻址（操作数在存贮器中）还是寄存器寻址（操作数在寄存器之中），在存贮器寻址的情况下，还用来指出寻址模式字节后面有多少偏移量字节。具体规定如下：

* MOD＝00　存贮器寻址方式，但无偏移量字节（即偏移量 DISP＝0）。
* MOD＝01　存贮器寻址方式，使用一个字节的偏移量（即$-128 \leqslant DISP \leqslant 127$）。

* MOD＝10　存贮器寻址方式，使用两字节的偏移量（即　0≤DISP≤65535）。
* MOD＝11　寄存器寻址方式。此时，R/M用来表示寄存器，并与指令操作码中的最后一位（w 位）一起确定是 8 位还是 16 位的寄存器。

REG 字段用来选择本操作中使用哪一个寄存器，指令操作码中的 w 位用来确定是 8 位还是 16 位操作，下面是寄存器选择的具体规定。

表 1-5　8088 指令中寄存器的选择

REG 字段(或 MOD＝11 时 R/M)			w＝0	w＝1
0	0	0	AL	AX
0	0	1	CL	CX
0	1	0	DL	DX
0	1	1	BL	BX
1	0	0	AH	SP
1	0	1	CH	BP
1	1	0	DH	SI
1	1	1	BH	DI

寻址模式字节中的 R/M 字段用来规定逻辑地址（也叫“有效地址”）的形成方法，表 1-6 是 MOD＝00、01或 10 时 R/M 所规定的逻辑地址的各种形成方法。

需要注意的是 MOD＝00且 R/M＝110 的情况，这时，寻址模式字节后面仍需要有两个字节，它们就是逻辑地址本身，因此叫做“直接寻址”方式。

表 1-6　逻辑地址的形成

R/M			MOD＝00	MOD＝01或10
0	0	0	(BX)+(SI)	(BX)+(SI)+DISP
0	0	1	(BX)+(DI)	(BX)+(DI)+DISP
0	1	0	(BP)+(SI)	(BP)+(SI)+DISP
0	1	1	(BP)+(DI)	(BP)+(DI)+DISP
1	0	0	(SI)	(SI)+DISP
1	0	1	(DI)	(DI)+DISP
1	1	0	直接寻址	(BP)+DISP
1	1	1	(BX)	(BX)+DISP

最后，我们可以把 Intel 8088 的操作数寻址方式按程序设计的观点归纳成下面八种类型：

① 寄存器寻址　　操作数在指定的寄存器中
② 立即寻址　　操作数就在指令中
③ 直接寻址　　逻辑地址由指令直接给出

④ 相对寻址　　逻辑地址＝（IP）＋DISP

⑤ 寄存器间接寻址

$$\text{逻辑地址}=\begin{cases}(\text{SI})\\(\text{DI})\\(\text{BX})\\(\text{BP})\end{cases}$$

⑥ 基址寻址

$$\text{逻辑地址}=\begin{cases}(\text{BP})+\text{DISP}\\(\text{BX})+\text{DISP}\end{cases}$$

⑦ 索引寻址（变址）

$$\text{逻辑地址}=\begin{cases}(\text{SI})+\text{DISP}\\(\text{DI})+\text{DISP}\end{cases}$$

⑧ 基址索引寻址

$$\text{逻辑地址}=\begin{cases}(\text{BX})+(\text{SI})+\text{DISP}\\(\text{BX})+(\text{DI})+\text{DISP}\\(\text{BP})+(\text{SI})+\text{DISP}\\(\text{BP})+(\text{DI})+\text{DISP}\end{cases}$$

一般说来，②、④两种寻址模式主要由指令的操作码来确定，其它寻址模式则由指令中的第2字节——寻址模式字节来确定。另外，访问内存贮器所使用的真正的物理地址，还需要由逻辑地址加上有关的段寄存器内容后才能形成（见图1—8）。

5．指令系统

在介绍8088的指令系统以前，先说明标志寄存器FLAGS。它比8080/8085的F寄存器中的五个标志位又增加了四个标志位，因而使其长度增为一个字，各标志位的位置与含义如下：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
×	×	×	×	O	D	I	T	S	Z	×	A	×	P	×	C
				溢出	方向	中断允许	跟踪	符号	零		辅助进位		奇偶		进位

Intel 8088的指令系统从功能上可以分成数据传送、算术运算、逻辑运算、控制转移、串处理、处理器控制等六个组。下面分别进行简单的介绍。

（1）数据传送指令

14种数据传送指令可分为四类，它们在存贮器和寄存器之间、寄存器与输入输出端口之间进行各种传送操作。

① 通用数据传送指令

* MOV（传送字节或字）

  把一个字节或一个字的源操作数传送到目标操作数所在单元。

* PUSH（把字压入堆栈）

  堆栈指针SP减2后，把一个字的源操作数送入SP指出的现行栈顶。

* POP（把字弹出堆栈）

  把SP指出的栈顶中的一个字传送到目标操作数所在单元，然后SP加2。

* XCHG（交换字节或字）

  使源操作数与目标操作数相互交换。

* XLAT（字节翻译）

  设BX指向一张256个字节的表的起点，AL中是表的索引值，本指令执行后，AL中即为查表所得到的内容。XLAT经常用来把一种代码翻译成另一种代码。

② 输入输出数据传送指令

* IN（输入指令）

  把来自输入端口的一个字节或一个字送入AL或AX寄存器。

* OUT（输出指令）

  把AL或AX中的字节或字传送到输出端口。输出（或输入）端口地址由立即寻址方式指定时，其范围为0～255；若由寄存器DX来指定，其范围为0～65535。

③ 目标地址传送指令

* LEA（装入有效地址）

  本指令把源操作数的有效地址（而不是操作数）传送到目标操作数所在单元，后者应为一通用寄存器。

* LDS（把指针内容装入DS）

  把源操作数（必须是存贮器操作数）的前16位送入目标操作数（应为通用寄存器之一），后16位送入DS寄存器。

* LES（把指针内容装入ES）

  同LDS，但后16位操作数传送到ES中。

④ 标志传送指令

* LAHF（把标志装入AH）

  把FLAGS寄存器中的S、Z、A、P和C等五位标志（8080/8085的标志）送入AH寄存器的第7，6，4，2和0位。

* SAHF（把AH内容存入标志寄存器）

  同上，但传送方向相反。

* PUSHF（标志入栈）

  把FLAGS寄存器内容送入堆栈。

* POPF（标志出栈）

把栈顶内容送入 FLAGS 寄存器。

（2）算术运算指令

8088 可以对四种类型的数据进行运算：无符号二进制数，带符号二进制数，无符号的紧凑十进制数或非紧凑十进制数。二进制数可以是 8 位或 16 位长；十进制数则以字节为单位。算术指令的操作结果的某些性质可以用标志来反映，如进位标志，辅助进位标志，符号，零标志，奇偶标志，溢出标志等，然后可以由紧跟着的条件转移指令进行测试。

由于算术指令比较简单，如无特殊需要，仅列出指令名称和记忆符号，不再给出解释。

* ADD（加法）
* ADC（包括进位的加法）
* INC（增量）
* AAA（加法的 ASCII 调整）

用于改变 AL 寄存器中的内容，以得到有效的非紧凑型的十进制数。

* DAA（加法的十进制调整）

对 AL 中保存的两个紧凑型十进制数相加的结果作调整，使之成为有效的紧凑型十进制数。

* SUB（减法）
* SBB（包括借位的减法）
* DEC（减量）
* NEG（求反）

从零中减去目标操作数，结果返回目标操作数。

* CMP（比较）
* AAS（减法的 ASCII 调整）
* DAS（减法的十进制调整）
* MUL（无符号乘法）
* IMUL（整数乘法）

上面两条指令，若为字节运算，则相乘后的结果回送到 AH 和 AL；若为字运算，则相乘结果送入 DX 和 AX 寄存器中。

* AAM（乘法的 ASCII 调整）
* DIV（无符号除法）
* IDIV（整数除法）

上面两条除法指令，被除数在累加器之中。若为字节除法，则商送入 AL，余数送入 AH；若为字除法，则商送入 AX，余数保存在 DX 内。

* AAD（除法的 ASCII 调整）
* CBW（字节转换为字）

把 AL 中字节的符号扩展到 AH 中去。

* CWD（字转换为双字）

把 AX 中的字的符号扩展到 DX 中去。

（3）逻辑运算指令

逻辑运算指令可对字节或字进行常用的几种逻辑操作处理及各种移位处理。

* NOT（取反）
* AND（逻辑乘）
* OR（逻辑加）
* XOR（按位加）
* TEST（测试）

  两操作数作逻辑乘，仅修改标志位Z、P、S等，但不回送结果。
* SHL（逻辑左移）
* SAL（算术左移）

  这两条指令实际上是相同的。
* SHR（逻辑右移）
* SAR（算术右移）
* ROL（循环左移）
* ROR（循环右移）
* RCL（通过进位的循环左移）
* RCR（通过进位的循环右移）

  上面八种移位指令，其移位次数可达255次。移位一次是特殊情况，可由操作码直接指出，其它情况则由寄存器CL给出移位次数。

（4）串指令

这里所说的“串”，既可以是字节串（一组字节），也可以是字串（一组字），依据指令的操作码而定。

串指令有两类：五种基本的串操作指令和五条控制操作重复执行的前缀指令。

串操作时，下列寄存器及标志起着特定的作用，程序应根据操作的具体要求给它们预置初值。

SI寄存器　　源串变址用

DI寄存器　　目标串变址用

CX寄存器　　重复次数计数器

AL/AX　　扫描值

FLAGS中的D标志位　　0表示重复操作中SI和DI应自动增量；1表示应自动减量

FLAGS中的Z标志位　　用来控制扫描或比较操作的结束

下面是关于串操作的一些指令：

* MOVS（串传送操作）

  本指令可以传送字符串（汇编记忆符号为MOVSB），也可传送字串（用MOVSW表示），当它不和下面所说的重复指令连用时，一次操作只传送一个字节或字。
* CMPS（串比较操作）
* SCAS（串扫描操作）

  把AL或AX的内容与目标串作比较，比较结果反映在FLAGS有关标志位中。

* LODS（装入串）

  把源串中的元素（字节或字）逐一装入 AL 或 AX 中。

* STOS（保存串）

  是 LODS 的逆过程。这两条指令一般不与重复指令配合使用。

* REP/REPE/REPNE/REPZ/REPNZ（重复）

  这五种不同的记忆符实际上只对应着两条不同机器码的一字节前缀指令，它们用来控制随后的串操作指令不断重复执行，直到条件满足为止。

对于 MOVS、LODS 和 STOS 来说，操作结束的条件是 CX 中的计数值减为 0；而对于 CMPS 和 SCAS 指令来说，除了 CX＝0 作为结束条件之外，若 FLAGS 中的 Z 标志位满足了一定条件，则串操作也应终止而不再重复。图1—14 是使用重复指令时串操作的执行过程。

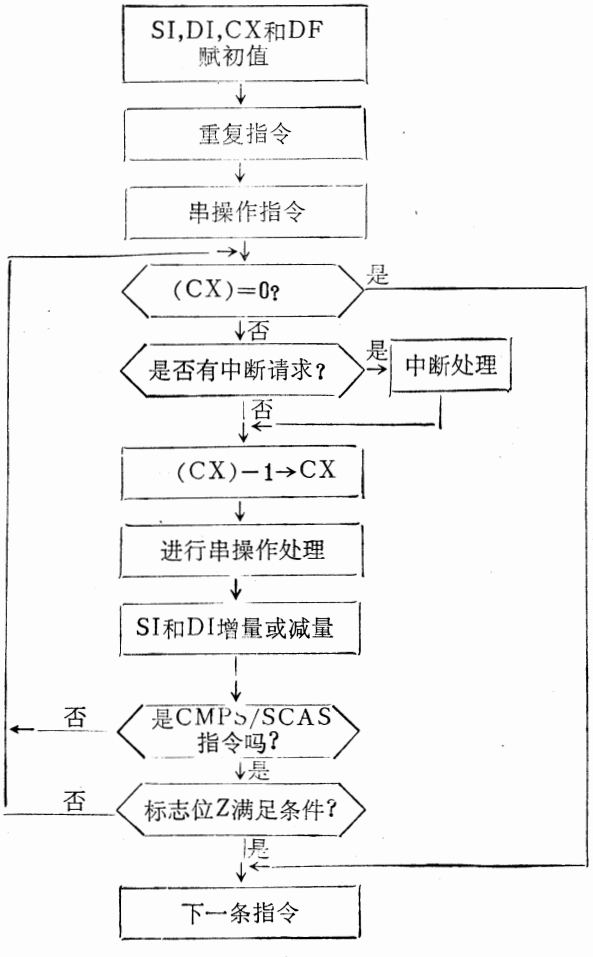


图 1—14　串操作指令的重复执行

（5）程序转移指令

8088 程序中指令的执行顺序是由代码段寄存器 CS 和指令指针 IP 来决定的。程序转移指令用来改变这两个寄存器的内容，从而改变了指令的正常执行顺序。当程序转移指令出现

时，指令队列中已不再是正常的指令。所以，总线接口部件 BIU 用新的 CS 值和 IP 值去取出指令，并直接把指令送给执行部件 EU，然后，从新的存贮区域中再取出指令填满队列。

转移指令可分成四组：无条件转移指令，条件转移指令，循环控制指令以及有关中断的指令。

① 无条件转移指令

* JMP（无条件转移）
* CALL（过程调用）
* RETURN（过程返回）

需要注意，JMP 和 CALL 指令可以有多种寻址方式，例如直接寻址，相对寻址，寄存器间接寻址等等。它们既能在段内进行转移，又能跳出本段而转向另一新段中去，具有很大的灵活性。

② 条件转移指令

条件转移指令根据指令执行时刻 FLAGS 中的状态来决定是否转移。所有的条件转移指令都是“短转移”，也就是说，目标必须在现行代码段内，并且在下一条指令第 1 个字节的 −128 到 +127 的距离之内。

条件转移指令共 18 条，它们是：

* JA/JNBE（不小于或不等于时转移）
* JAE/JNB（大于或等于转移）
* JB/JNAE（小于转移）
* JBE/JNA（小于或等于转移）
* JC（有进位时转移）
* JE/JZ（等于转移）
* JG/JNLE（大于转移）
* JGE/JNL（大于或等于转移）
* JL/JNGE（小于转移）
* JLE/JNG（小于或等于转移）
* JNC（无进位时转移）
* JNE/JNZ（不等于时转移）
* JNO（不溢出时转移）
* JNP/JP（奇偶性为奇数时转移）
* JNS（符号位为“0”时转移）
* JO（溢出转移）
* JP/JPE（奇偶性为偶数时转移）
* JS（符号位为“1”时转移）

注意，前四条转移指令测试的是无符号整数运算的结果(标志 C 和 Z)，而 JG、JGE、JL 和 JLE 四条指令测试的是带符号整数的运算结果(标志 S、O 和 Z)，它们的用法是不一样的。

③ 循环控制指令

循环控制指令用来控制一个程序段的重复执行。这些指令用 CX 寄存器作计数器，转移目标只能在离自身 −128 到 +127 的范围内，即它们都是短转移指令。

* LOOP（CX 不为零时循环）
* LOOPE/LOOPZ（CX 不为零且标志 Z=1 时循环）
* LOOPNE/LOOPNZ（CX 不为零且标志 Z=0 时循环）
* JCXZ（若 CX 为零则转移）

④ 中断指令

中断指令允许程序在需要时去启动中断服务例行程序，它的作用与硬件所启动的中断是类似的，不过它并不引起处理器硬件去执行一个中断响应总线周期，所以有人往往称它们为“软中断”。

* INT（中断指令）

本指令需指出一个中断类型号码（0～255），它的作用是去启动指定类型的中断服务程序。其执行过程大致为：先把 FLAGS 寄存器推入堆栈；再把 CS 寄存器

表 1-7　IBM PC 中断向量表

中断类型号	性质	名称
0	故障与调试中断	零作为除数
1		单步操作
2		不可屏蔽中断
3		断点中断
4		溢出中断
5		屏幕打印中断
6		（不用）
7		（不用）
8	外部中断	日时钟中断
9		键盘中断
A		（不用）
B		同步通讯中断
C		异步通讯中断
D		硬盘中断
E		软盘中断
F		打印中断
10	软中断	显示器驱动程序
11		设备检测程序
12		存贮器容量判断程序
13		软盘驱动程序
14		通讯驱动程序
15		盒式带驱动程序
16		键盘驱动程序
17		打印机驱动程序
18		磁带BASIC
19		引导程序
1A		日时钟程序

内容入栈；然后把 IP 内容入栈；把中断类型号乘 4 后作为地址码，从内存取得新的CS 值和 IP 值；按新的 CS 和 IP 内容跳转到中断服务例程去执行。

* INTO（溢出中断）
* IRET（中断返回）

最后，顺便把 IBM PC 常用的中断服务程序名称及其对应的中断类型号码（中断向量）列出于表 1-7 供读者参考。

（6）处理器控制指令

这组指令允许程序去控制 8088 与外部事件同步，并对 FLAGS 寄存器进行修改。

* HLT（处理器暂停，直到出现中断或复位信号才继续执行）
* WAIT（当芯片引线$\overline{\text{TEST}}$为高电平时使 CPU 进入等待状态）
* ESC（转换到外处理器）
* LOCK（封锁总线）
* NOP（空操作）
* STC（置进位标志位）
* CLC（清进位标志位）
* CMC（进位标志取反）
* STD（置方向标志位）
* CLD（清除方向标志位）
* STI（置中断允许位）
* CLI（清中断允许位）

关于 8088 的指令系统就扼要介绍到这里，详细的说明请参阅有关资料。

第四节　单色显示控制器

单色显示器（俗称“黑白显示器”）在系统中通常用作控制台字符显示器，它的主要性能指标为：

* 每屏可显示 25 行×80 列字符。
* 每个字符块的大小为 9 ×14 点。
* 字符块中的字符由 7×9 点组成。
* 能够显示（或处理）8 位的 256 种不同编码的字符输出。
* 每个输出字符均有各自的显示属性，如加亮、闪烁　反视频（即白底黑字）、下横线等。

单色显示器的控制器与并行打印机控制器合在一块选件板上。本节只介绍显示控制器的原理，打印机控制器将在第六节中另行介绍。

1．结构与工作原理

图 1—15 是单色显示控制器的逻辑框图。从图中可以看出，整个控制逻辑是围绕着 Motorola 6845 CRT 控制器芯片设计的。6845 芯片的主要功能是形成字符缓冲器以及属性

存贮器的地址码，并且同步地产生水平扫描、垂直扫描等信号並送到视频控制逻辑去。字符缓冲器及属性存贮器各 2K 字节，分别用来存放 80×25＝2000 个待显示的字符码及其对应的属性。它们既可被 CPU 访问，以便 CPU 送入需要显示的字符代码，又可以被 6845CRTC 所访问，逐一地读出其中的字符码和属性，再把它们转换成显示器所需要的视频信号。

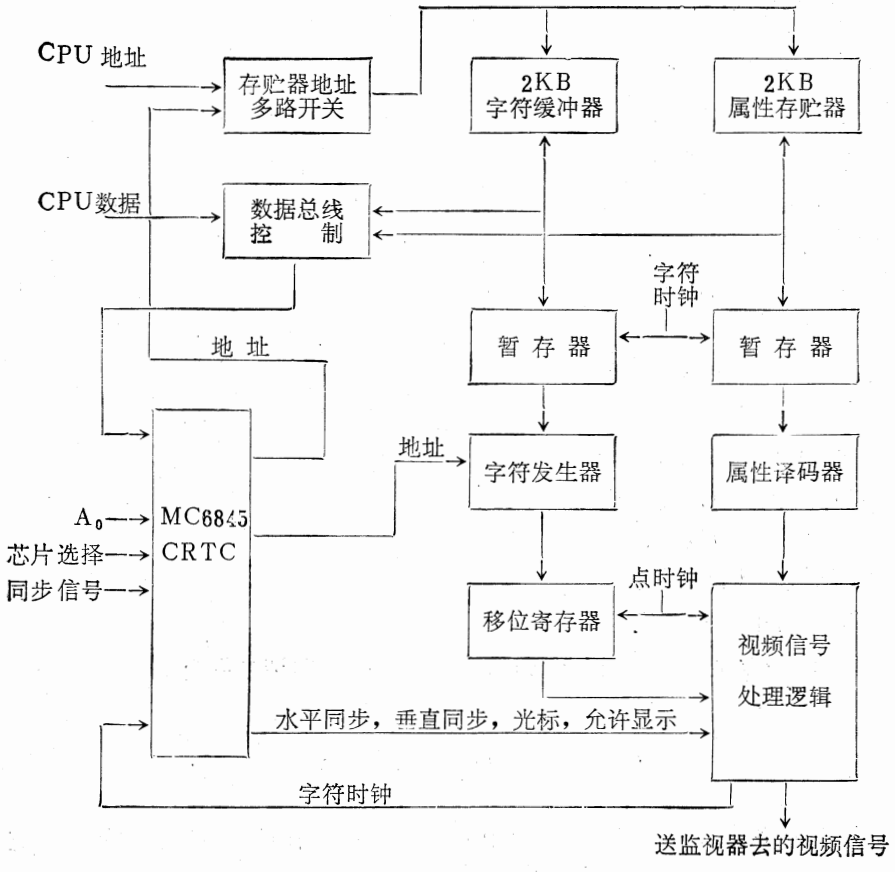


图 1—15　单色显示控制器框图

显示控制器的工作过程大致如下：

（1）CPU 在显示器水平和垂直回扫期间，把欲显示的字符码及其属性送入字符缓冲器和属性存贮器；

（2）6845 CRTC 芯片以每秒 50 帧的频率，一面产生行同步、帧同步等信号送到视频信号处理逻辑，一面产生地址码去读出缓冲器中的字符码及其属性；

（3）以缓冲器中读出的字符代码和 6845 提供的扫描线序号为地址，读出字符发生器中的字形信息，字符的属性则进行一定的译码处理；

（4）字形信息读出后送入移位寄存器，然后逐位移出到视频控制逻辑中，与有关的属性、光标等组合处理后，形成显示器所需要的视频输出信号和亮度信号。

需要指出，字符发生器是一个容量为 8KB 的只读存贮器，其中保存有三套不同体的字

符集（共256种不同的字符）点阵模式。字体的选择可以通过选件板上的跨接器来进行。显示字符的形状及编码可参见附录Ⅰ。256种字符大体可分为如下几类：

* 16个专用的游戏符号
* 15个用于文字处理编辑操作的符号
* 96个常用ASCII字符
* 48个外语字符
* 48个商用图形符号
* 16个常用希腊字母
* 15个常用科学符号

显示控制器与通道的接口比较简单。其中，地址总线和数据总线用来传送地址码和数据；MEMW、MEMR、IOW和IOR用来指出是访问字符缓冲器（及属性存贮器）还是访问6845内部的寄存器，是写入还是读出；RESET信号用来复位；I/O READY用来使CPU和显示控制器之间同步，因为显示控制器的操作主要是用“字符时钟信号”由（16.257MHz分频而得到的1.8 MHz的信号）来定时的，所以CPU在访问显示器时必须插入等待状态；I/O CLOCK信号用于CPU访问6845芯片时作为时钟信号。

显示控制器与CRT显示器（又称“监视器”）的连接采用直接驱动方式（图1—16）。控制器通过一个9芯的D形插座与监视器相连接，其中6脚为亮度信号，用来控制字符的

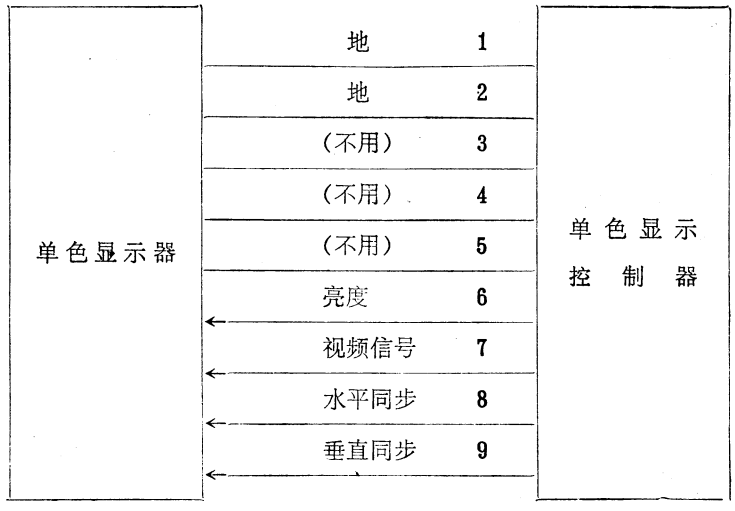


图1—16 显示控制器与显示器的接口

“加亮”；7脚为视频输出信号，带宽为16.27 MHz；8脚为水平同步信号，频率为18.432 KHz；9脚为垂直同步信号（帧同步信号），每秒50帧，每帧350线，每线720点。IBM PC原配的单色显示器使用长余辉（P39）显示管，显示质量较好。

2. 显示器的程序设计

单色显示器的程序设计需要考虑的问题有两个方面。一方面是如何把输出显示的字符及其属性组织好送入字符缓冲/字符属性存贮器；另一方面是如何设置或改变控制器的工作参数，以得到合适的显示效果。由于单色显示器在系统中仅仅作为字符显示器使用，因此程序设计比较简单。

单色显示器共25行，每行80列，因此字符/属性存贮器各为2K字节，它们均位于系统的内存贮器空间，其布局如下图：

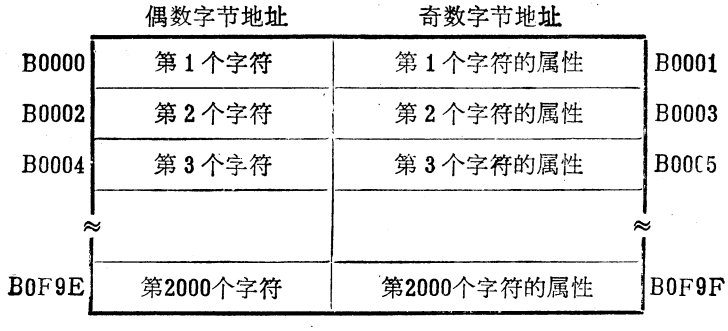
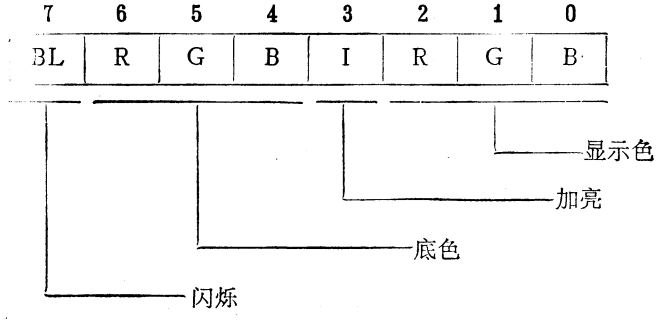


图1—17 字符缓冲/属性存贮器的布局

属性字节的格式与定义如下：



其中，底色和显示色实际上只能有两种：黑色（000）或白色（ 11），但组合起来可以有下列含义：

底 色 (RGB)	显 示 色 (RGB)	显 示 效 果
0 0 0	0 0 0	不显示
0 0 0	0 0 1	加下横线
0 0 0	1 1 1	黑底白字
1 1 1	0 0 0	白底黑字(反视频)

从程序设计的角度来看，字符输出显示的过程，实际上就是把字符的代码及属性送入缓冲器和属性存贮器的过程，这是由 ROM 中 BIOS 的显示器驱动程序来完成的。在此过程中，显示器驱动程序还须作如下处理：

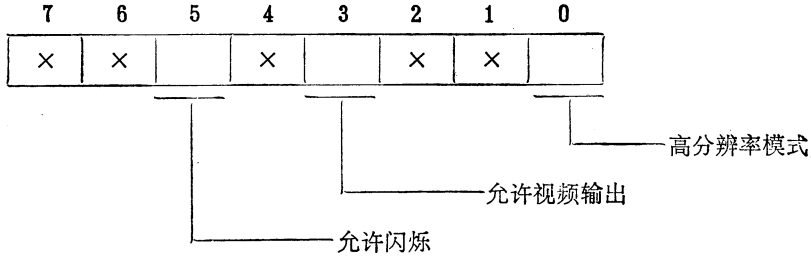
①识别若干控制字符，如回车、换行、退格、响铃等。

②当显示输出到行末时，应自动换行，特别是在输出到屏幕的最后一行最末一个字符时，还应自动实现滚行等处理。

显示控制器的操作模式以及工作参数均是由 CPU 通过输出指令来控制和设定的。系统分配给单色显示控制器的输入输出端口地址有四个，它们所对应的寄存器及其作用简单介绍如下：

（1）CRT 控制寄存器（I/O 地址为“3B8”）

这是一个用于控制显示器工作方式的寄存器，CPU 向输出端口“3B8”送出的控制字节有如下含义：

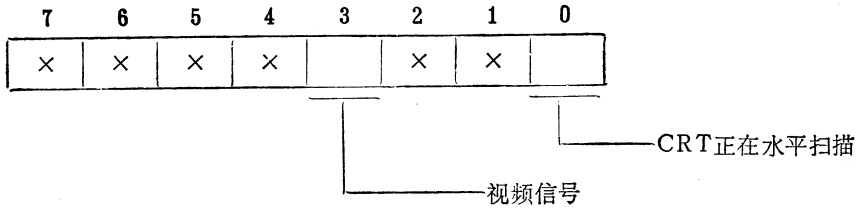


（“×”表示不使用的位）

系统加电后向单色显示控制器发出的第 1 条命令必须是置高分辨率模式为“1”，否则，CPU 将无休止地处于等待状态。

（2）CRT 状态寄存器（I/O 地址为“3BA”）

通过输入指令，CPU 可以从输入端口“3BA”处取得一个显示控制器的状态字节，其含义为：



这条输入指令极为有用。每当 CPU 需要输出显示一个新的字符时（即往字符缓冲存贮器中现行光标位置上写入一个字符代码时），总必须先用输入指令取回一个状态字节，判断 CRT 是否处于水平或垂直回扫状态。若是（此时状态字节最低位为“0”），则向字符缓冲/属性存贮器中写入，否则等待。下面是显示输出一个字符的流程图（图 1—18）。

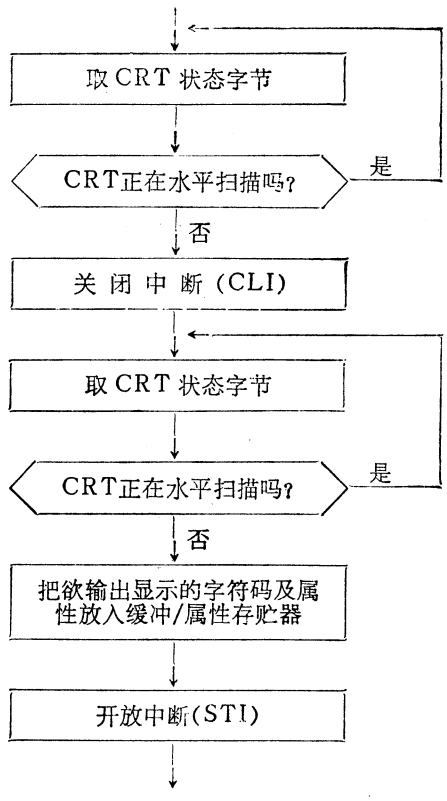
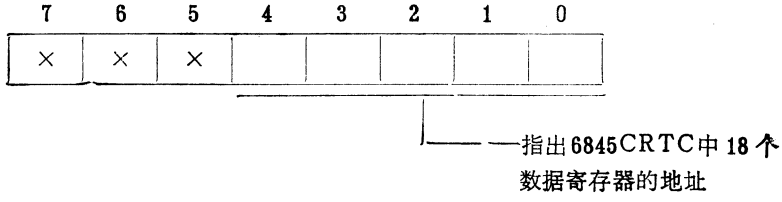


图 1—13 输出显示字符的流程

（3） 6845 CRTC 的索引寄存器（I/O 地址为“3B4”）

因为 6845 CRTC 芯片内部有 19 个寄存器，为了少占用 CPU 的 I/O 端口地址，因此其中有一个索引寄存器（也叫地址寄存器）用于指出其他 18 个数据寄存器中的某一个。

索引寄存器作为 CPU 的一个输出端口，其端口地址为“3B4”，CPU 在此端口所输出的一个字节其含义为：



（4） 6845 CRTC 的数据寄存器（I/O 地址为“3B5”）

6845 CRTC 有 18 个内部数据寄存器，它们的值均可由软件来设定，以便在不同操作模

式时产生出不同要求的输出控制信号、地址信号和同步信号。

CPU 在输出端口“3B5”输出的数据字节，将按照索引寄存器中的指针值送到6845 CRTC 的 R_0—R_{17} 之中的某一个。通常，这是在显示器初始化的时候完成的。下表给出单色显示控制器在初始化时有关寄存器应该设置的参数，表中参数的值均为十六进制。现对有

表 1-8　单色显示器 6845 初始化参数

编　号	寄存器名称	参　数	单　位
R_0	水平扫描总时间	61	字　符
R_1	每行字符数	50	字　符
R_2	水平同步位置	52	字　符
R_3	水平同步宽度	F	字　符
R_4	垂直扫描总时间	19	字符行
R_5	垂直总调节	06	扫描线
R_6	每帧显示行数	19	字符行
R_7	垂直同步位置	19	字符行
R_8	隔行扫描模式	02	——
R_9	最大扫描线地址	D	扫描线
R_{10}	光标开始	B	扫描线
R_{11}	光标结束	C	扫描线
R_{12}	起始地址(高位)	00	——
R_{13}	起始地址(低位)	00	——
R_{14}	光标(高位)	00	——
R_{15}	光标(低位)	00	——
R_{16}	不使用	—	——
R_{17}	不使用	—	——

关寄存器作简要介绍。

R_0、R_1、R_2 和 R_3 四个寄存器用来控制水平同步。R_0 为一次水平扫描的总时间（包括逆程在内），R_1 为每行的字符数目，R_2 用来决定水平同步信号在水平扫描线上的位置，以调节显示画面在屏幕上的左右位置，R_3 用作水平扫描宽度的补偿。

R_4、R_5、R_6、R_7、R_8 和 R_9 用作垂直同步的控制。R_4 和 R_5 决定了一次垂直扫描的时间，其中 R_5 起调节作用，使垂直扫描时间与额定的帧频相适应；R_6 是每帧显示的字符行行数；R_7 控制垂直同步信号在垂直扫描线上的位置；R_8 的作用是选择隔行扫描，当（R_8）=02 时，表示非隔行扫描；R_9 是每个字符行中扫描线的数目减 1。

图 1—19 是上述一些参数的物理意义的示意图。

光标控制寄存器 R_{10} 和 R_{11} 用来控制：要不要显示光标；光标是否闪烁；闪烁的周期；光标的形状大小等等。

R_{14} 和 R_{15} 是两个只读的寄存器，它们共 14 位，用来存贮光标的现行位置。CPU 读取这两个寄存器的内容，就可以了解到光标在屏幕上的位置。

R_{12} 和 R_{13} 是只写寄存器，它用来决定每次垂直扫描开始时，刷新存贮器的首地址。

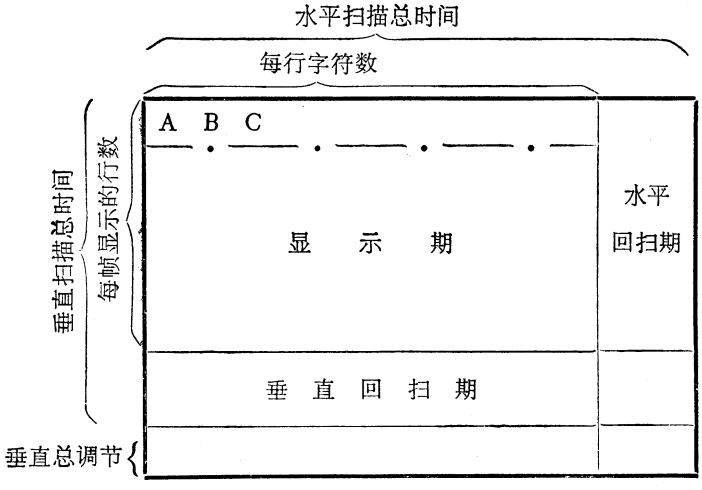


图 1—19 6845 扫描参数的物理意义

第五节 彩色图形显示控制器

IBM PC 的彩色图形显示器在系统中有两种使用方式，它既可以代替单色显示器作为控制台显示器使用，也可以与单色显示器同时使用，这时它主要用作系统的图形显示器，以满足事务管理、简单 CAD（计算机辅助设计）等应用领域对图形显示的要求。使用不同的显示器，用户应把系统底板上的微型组合开关 1（SWITCH1）中的第 5 和第 6 个开关设定在不同位置，具体规定见下图：

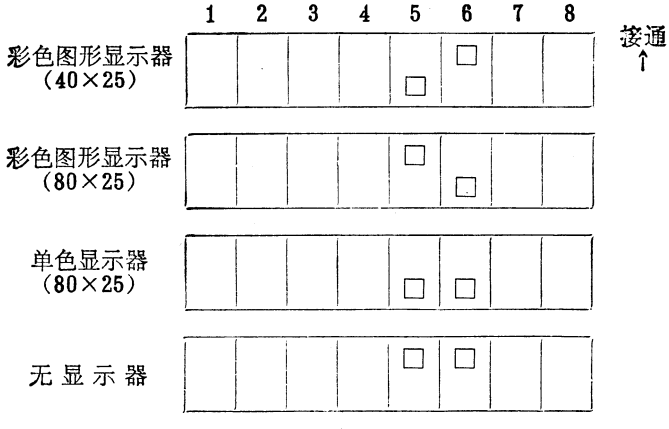


图 1—20 显示器类型的选择

当系统中同时使用两种显示器时，加电自举后，系统将根据开关的设定位置选择一个显示器作为系统的控制台显示器使用，然后可根据实际应用的需要，用软件在两种显示器之间进行切换，或者同时使用两种显示器进行工作。

1．工作模式

彩色图形显示器有两种基本工作模式：

（1）字母数字模式（A/N 模式）

在这种模式下，显示器可以在屏幕上显示 40×25 个字符（当连接低分辨率监视器或电视机时）或 80×25 个字符（连接较高分辨率监视器时）。每个字符块的大小为 8×8 点，其中字符由 5×7 点或 7×7 点的点阵构成。与单色显示器一样，彩色显示器控制器可以显示和处理所有 256 种组合的 8 位字符代码，每个字符还带有一个属性字符，当选择黑/白显示模式时，字符的属性有加亮、闪烁和反视频显示；而在彩色显示模式下，每个字符均有 16 种底色和 16 种显示色可以选用，而且仍可使用闪烁属性。

（2）图形显示模式（APA 模式）

在这种工作模式下，显示器屏幕上的每个点（称为“象素”或“象元”）均可由程序控制其亮度或颜色，因而能显示出质量较好的图形。APA 模式又可分成三种不同的分辨率：

* 高分辨率模式：每帧 200 线，每线 640 点，每点只能取黑白两种颜色。
* 中分辨率模式：每帧 200 线，每线 320 点，每点可以有 4 种不同的颜色。
* 低分辨率模式：每帧 100 线，每线 160 点，而每点则有 16 种不同的颜色。

目前，BIOS 中的驱动程序不能支持低分辨率显示模式，需要时应由应用软件人员自行开发。

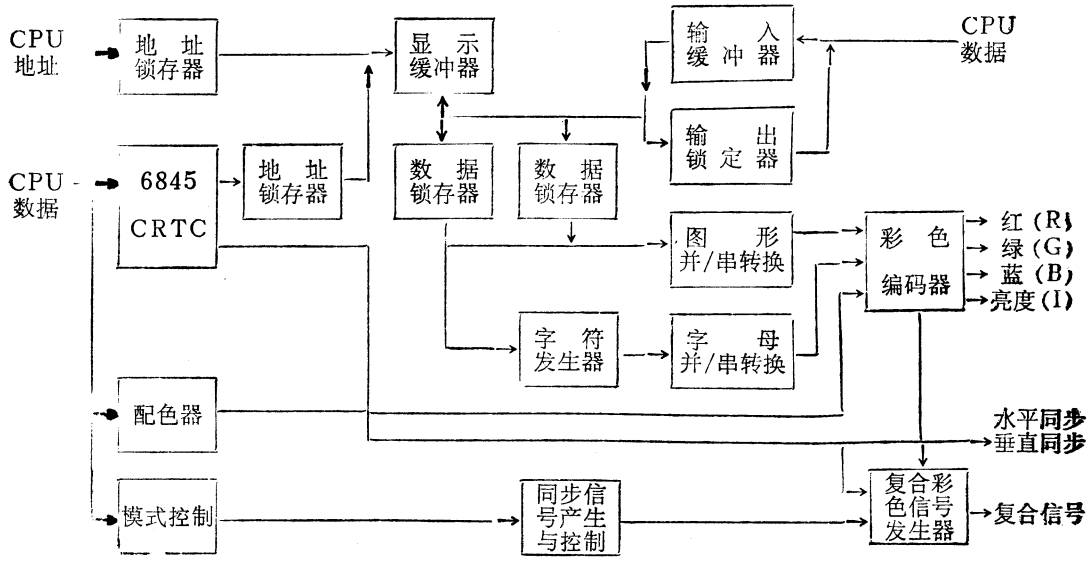


图 1—21 彩色图形显示控制器逻辑框图

2. 结构与工作原理

彩色图形显示控制器的逻辑结构与单色显示控制器类似（图1—21），它也采用6845 CRTC芯片作为控制器的核心，一方面产生显示缓冲器的地址码以读出其中的显示信息，另一方面同步地给出水平和垂直扫描信号去控制CRT显示器。由于6845芯片具有灵活的可编程序能力，因此，显示控制器具有多种操作模式。

显示缓冲器也叫图形刷新存贮器，它是一个容量为16KB的随机读写存贮器。在内存地址空间中，它占据的位置是B8000H—BBFFFH。在图形显示模式时，显示缓冲器中的每1个、2个或4个二进位，分别与高分辨率、中分辨率和低分辨率时的每一个象元相对应。图1—22是屏幕上的象元位置与显示缓冲器中信息的对应关系（低分辨率时仅使用缓冲器的前一半）。图1—23是高分辨率模式和中分辨率模式时，缓冲器中的每个字节的信息内容与屏幕上象元位置的对应关系。

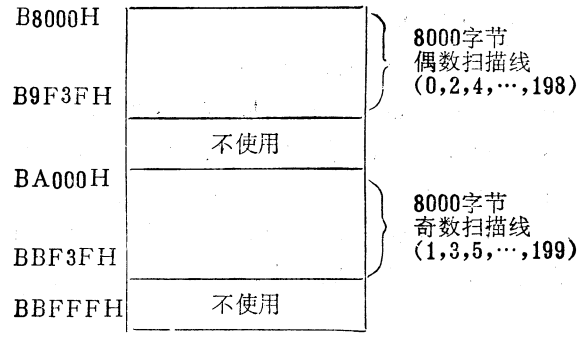


图1—22 显示缓冲器与屏幕扫描线的对应关系

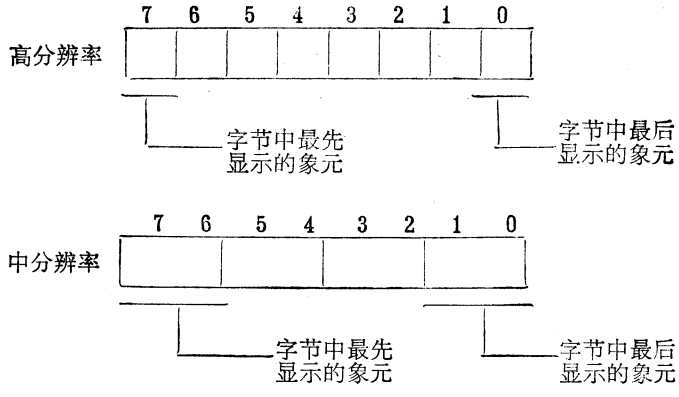


图1—23 缓冲器字节内容与象元位置的对应关系

在字符显示模式时，由于每个显示字符均附带有一个属性字符（与单色显示器一样，前者放入偶数地址单元，后者放入奇数地址单元），因此，当每屏显示40×25个字符时，

16KB 容量的显示缓冲器一共可以容纳八屏 显示信息（通 常称为八 个显示 “页” ），页号为 0—7； 当每屏显示 80×25 个字符时， 缓冲器中一共 可以容纳四页信息，编号为 0—3。

与单 色显示控制器一样， 彩色显示 控制器中 也有一 个标 准的 8K 字节的 字符 发生 器（MK 36000 ROM 存贮器）， MK36000 中有三套不同字体的字符模式， 即 7×9、7×7 和 5×7 。对于彩色图形显示器来说， 由于分辨率的原因， 只能选择后两种字 体。 系 统 一 般使用的是 7×7 点阵的字体，但通过插件板上的一个跨接插头可以选用 5×7 点阵的字体。

彩色图形显示控制器的一个重要组成部分是彩色编码器，它的功能是按照字符属性或象元的颜色以及软件所选择的某种配色器来产生所需要的颜色，并编码成相应的三元色（红、绿和蓝色）输出送到彩色监视器去。

彩色图形显示控制器有三种不同的接口方式与不同的显示器相连接。当使用分辨率较好的彩色（或单色）监视器时， 可以通过机箱背面的 D 形插头座与监视器互连。 这时， 控制器输出红、绿、蓝、亮度、水平和垂直同步信号（图 1—24 ），这种连接方式叫做“直接驱动”方式。 当使用低分辨率的监视器时， 可以通过机箱背面的话筒插口与监视器连 接。 这时，控制器已把颜色、 亮度及同步信号全部复合在一起成为一个信号。 这种连 接方式 叫做“复合信号输出”方式（图 1—25 ）。

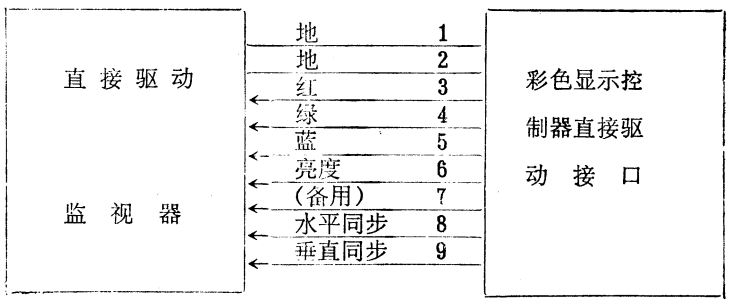


图 1—24 监视器的直接驱动方式

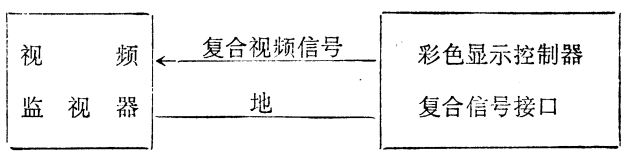


图 1—25 监视器的复合信号驱动方式

如果需要采用家用的彩色电视机（或黑白电视机）作为显示器的话，则在复合输出信号之后 ，还要使用一个射频信号调制器，然后再把调制后的输出信号连接到电视机的天线端口去（图 1—26 ）。

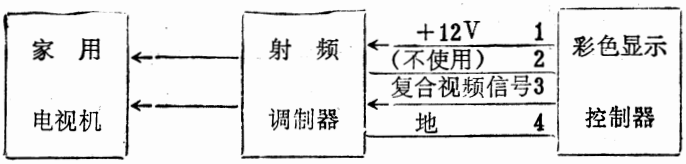


图 1—26 家用电视机作为显示器

3. 程序设计

CPU 对彩色图形显示器的控制是通过输入输出指令来进行的。彩色显示控制器占用 16 个输入输出端口地址（3D0—3DF），其中七个已有具体定义（表 1-9），

表 1-9 彩色显示控制器输入输出地址

输入输出地址	寄 存 器 名 称
3D0	6845CRTC 索引寄存器
3D1	6845CRTC 数据寄存器
3D8	操作模式控制寄存器
3D9	彩色选择寄存器
3DA	状态寄存器
3DB	清除光笔锁存器
3DC	预置光笔锁存器

对彩色图形显示器进行程序设计的一般步骤如下：

① 选定操作的模式；

② 把模式控制寄存器（3D8）中的第 3 位置“0”，即暂不允许视频信号输出；

③ 把所选模式对应的一组参数置入 6845CRTC 的寄存器中去；

④ 按选定的操作模式设置模式控制寄存器；

⑤ 按需要设置彩色选择寄存器；

⑥ 向显示缓冲器中写入要显示的信息（字符代码或图形信息）。

下面对有关问题作扼要的说明。

模式选择寄存器

模式选择寄存器用来决定彩色图形控制器的操作模式。它是由 CPU 通过输出指令向端口 3D8 输出 1 个字节来设置的，该寄存器的各位定义如下：

* Bit 0 “1”表示选择 80×25 字符模式，
  “0”表示选择 40×25 字符模式。
* Bit 1 “1”表示选择 320×200 图形模式，
  “0”表示选择字符显示模式。
* Bit 2 “1”表示选择黑白显示方式，

"0" 表示选择彩色显示方式。

* Bit 3　"1" 表示允许视频信号输出，
"0" 表示不许视频信号输出。

* Bit 4　"1" 表示 640×200 图形显示方式，
"0" 表示非 640×200 图形显示方式。

* Bit 5　"1" 表示允许字符模式下使用闪烁属性，"0" 则不许使用。

* Bit 6 和 7　不使用。

其中，各位的组合并不都是任意的，常用的操作模式总结如表 1-10 所示。

表 1-10　彩色显示器常用操作模式

模式选择寄存器 (7 6 5 4 3 2 1 0)	操作模式
1 0 1 0 1 1 0 0	40×25黑白字符模式
× × 1 0 1 0 0 0	40×25彩色字符模式
× × 1 0 1 1 0 1	80×25黑白字符模式
× × 1 0 1 0 0 1	80×25彩色字符模式
× × × 0 1 1 1 0	320×200黑白图形模式
× × × 0 1 0 1 0	320×200彩色图形模式
× × × 1 1 1 1 0	640×200黑白图形模式

表 1-11　6845 寄存器参数表

寄存器编号	寄存器名称	寄存器类型	参数单位	40×25字符模式	80×25字符模式	图形模式
R_0	水平扫描总时间	只写	字　符	38	71	38
R_1	每行字符数	只写	字　符	28	50	28
R_2	水平同步位置	只写	字　符	2D	5A	2D
R_3	水平同步宽度	只写	字　符	0A	0A	0A
R_4	垂直扫描总时间	只写	字符行	1F	1F	7F
R_5	垂直总调节	只写	扫描线	06	06	06
R_6	每帧显示行数	只写	字符行	19	19	64
R_7	垂直同步位置	只写	字符行	1C	1C	70
R_8	隔行扫描模式	只写	—	02	02	02
R_9	最大扫描线地址	只写	扫描线	07	07	01
R_{10}	光标开始	只写	扫描线	06	06	06
R_{11}	光标结束	只写	扫描线	07	07	07
R_{12}	起始地址(高位)	只写	—	00	00	00
R_{13}	起始地址(低位)	只写	—	00	00	00
R_{14}	光标(高位)	读写	—	××	××	××
R_{15}	光标(低位)	读写	—	××	××	××
R_{16}	光笔(高位)	只读	—	××	××	××
R_{17}	光笔(低位)	只读	—	××	××	××

6845 CRTC 寄存器

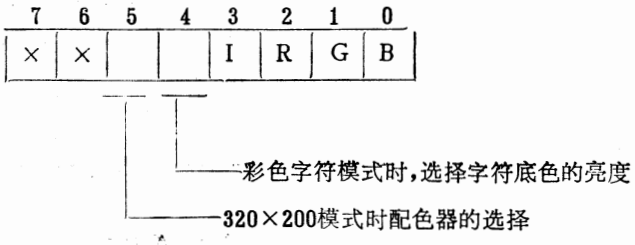
不同的操作模式，要求6845 CRTC产生不同的地址输出、水平扫描、垂直扫描等信号，因此就要向6845 CRTC芯片送入不同的参数。表1-11列出了6845 CRTC在三种常用情况下各有关寄存器应赋予的控制参数。

6845 CRTC中的每一个参数的设置都需要使用两条OUT指令。首先是向3D0（或3D4）端口送出CRTC寄存器的索引值（即序号），然后再向3D1（或3D5）端口送出相应的参数值。

在黑白字符显示模式时，无论是40×25的低分辨率方式还是80×25的高分辨率方式，除了16KB的显示缓冲器中可以分别容纳八页或四页以外，其它的程序设计都和上节所述的单色显示控制器一样，它们是相互兼容的，这里不再赘述。

彩色选择寄存器

彩色字符显示模式和彩色图形显示模式的程序设计都涉及到彩色的选择问题。彩色的选择用彩色选择寄存器来控制，这个寄存器的端口地址为“3D9”，其格式如下：



下面结合不同的操作模式给出其各位的解释。

第0，1，2，3四位分别代表蓝色、绿色、红色和加亮，它们在40×25的彩色字符模式下，代表屏幕上矩形显示区外部边界的颜色；而在320×200彩色图形显示模式下，则用来控制屏幕的底色。I、R、G、B可以组合成表1-12所示的16种不同的彩色。

注意，彩色字符显示模式下，每个字符本身的显示颜色以及它所在屏幕位置上（8×8点的方块）的底色并不是由彩色选择器中的I、R、G、B所决定的，而是由每个字符的属性字符的内容所决定。字符属性的定义如下：

其中，第3—0位用来选择16种彩色之一作为显示字符的颜色；第6—4位与彩色选择器中的第4位组合起来，用来决定每个字符的底色，属性字符中的最高位仍用来控制字符的闪烁。

彩色选择器中的第5位只在320×200的彩色图形显示模式时有用，它用于指出象元的

表 1-12　彩色编码表

I	R	G	B	彩　色
0	0	0	0	黑
0	0	0	1	蓝
0	0	1	0	绿
0	0	1	1	青
0	1	0	0	红
0	1	0	1	洋 红
0	1	1	0	棕
0	1	1	1	淡 灰
1	0	0	0	深 灰
1	0	0	1	浅 蓝
1	0	1	0	浅 绿
1	0	1	1	浅 青
1	1	0	0	淡 红
1	1	0	1	浅洋红
1	1	1	0	黄
1	1	1	1	白

（注：某些彩色监视器不能识别亮度信号，因此只有八种颜色。）

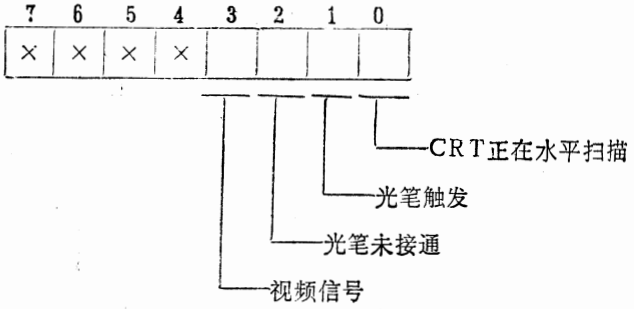
颜色是使用 0 号配色器还是 1 号配色器。前面已经讲过，320×200的彩色图形显示模式，每个象元对应着显示缓冲器中的两位（C_1 和 C_0），因此每个象元可以有四种取值，亦即代表着四种不同的颜色。但到底是什么颜色，需要由彩色选择器中的第 5 位来决定。表 1-13 是象元颜色的定义。

表 1-13　象元颜色定义表

彩色选择器第 5 位	象元的值 C_1	象元的值 C_0	象元的颜色
0	0	0	与底色相同
0	0	1	绿
0	1	0	红
0	1	1	黄
1	0	0	与底色相同
1	0	1	青
1	1	0	洋 红
1	1	1	白

状态寄存器

最后再介绍一下彩色显示控制器的状态寄存器，CPU 可以通过输入指令从 3DA 端口输入一个状态字节，其格式如下：



其中第 0 位和第 3 位的含义与单色字符显示器一样，第 0 位用来指出 CRT 显示器是否处在水平扫描或垂直扫描的回扫期，若是，则 CPU 可以向显示缓冲器中送入新的显示信息而不影响屏幕画面。第 3 位是字符显示时视频输出信号的瞬间状态，目的是检查一下是否有视频信号输出，以利于系统的故障诊断与分析。第 2 位和第 3 位都与光笔的操作有关，我们将在第八章中再进行讨论。

第六节　打印控制器与打印机

1．打印控制器结构与工作原理

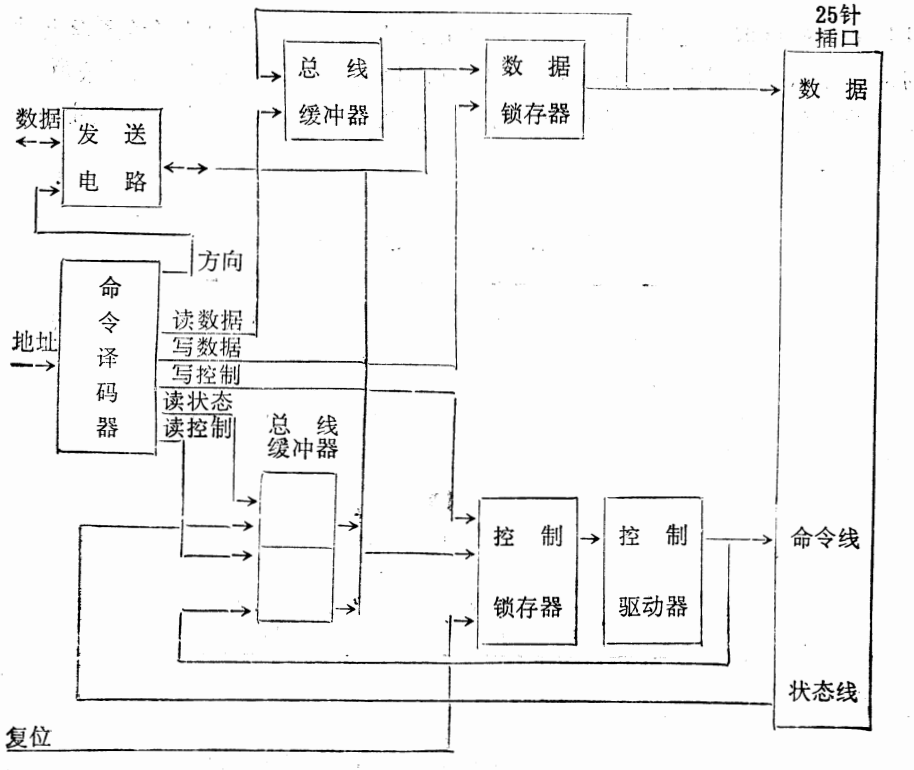
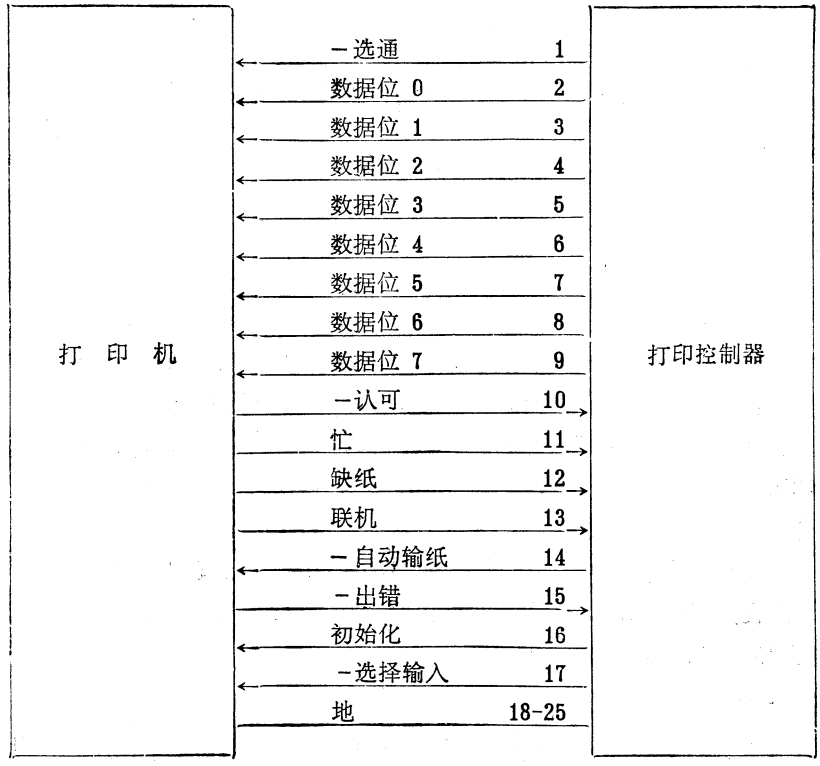


图 1—27　打印控制器的逻辑框图

打印机控制器是专门为连接一台具有并行接口的打印机而设计的。但也可以作为一个有一定通用性的并行接口来使用。图 1—27 是打印控制器的逻辑框图。

打印控制器的逻辑结构比较简单。它由命令译码器、总线缓冲器、数据锁存器、控制锁存器、控制驱动器以及接收发送器等组成。命令译码器负责识别从 CPU 送来的具有系统分配给打印控制器所规定的端口地址的输入输出指令，并译出“数据传送方向”、“读数据”、“写数据”、“写控制”、“读状态”、“读控制信号”等六个命令。数据锁存器用来暂存 CPU 送出的打印数据字节，然后一方面通过 D 形插头座送到打印机去打印，另一方面又回送到总线缓冲器，以便 CPU 必要时可以读入比较、进行故障测试与诊断。控制信号锁存器及开路集电极驱动器用来暂存并驱动四个控制信号——初始化、选通、自动输纸以及选择输入信号。这些控制信号在输出的同时也回送到另一个总线缓冲器，同样也是为了诊断时用。打印机的五个状态信息（联机，认可，忙，缺纸，出错）均为稳态电平信号，它们经由总线缓冲器被“读状态”命令选通后，即送往数据总线，由 CPU 的输入指令取回送入某寄存器中，供软件作分析打印机状态时使用。

打印控制器与打印机的接口是一个 25 芯的插头座，上面所介绍的一些信号名称及其对应脚码见图 1—28。这些信号都是标准的 TTL 电平。



（注：有“－”号表示负电平起作用）

图 1—28 打印控制器与打印机的接口

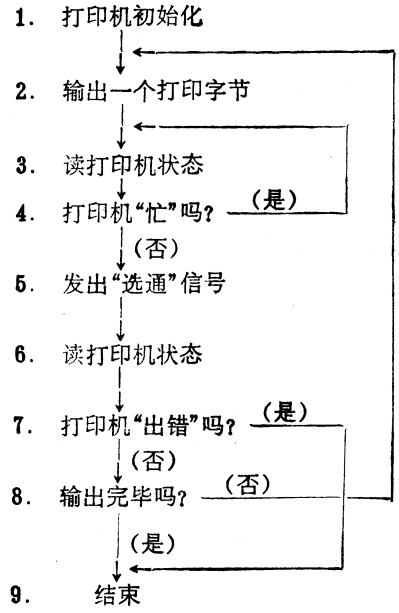
打印机控制器可以是一块独立的选件板，也可以是“单色显示控制器/打印控制器”选

件板的一个组成部分。但它们的逻辑结构、工作原理以及程序设计方法都一样，只是分配的输入输出端口地址有所不同。

2．程序设计

打印机的程序设计比较简单，通常有两种方式可以选用，一种是查询方式，另一种是中断方式。

用查询方式进行打印输出的程序设计时，过程大致为：



用中断方式进行打印输出时，系统的效率比较高。上面图中每输出一个打印数据字节并发出选通信号之后，CPU 即可进行其它任务的处理。在执行其它任务的过程中，若打印机发出“认可”信号（表示 CPU 发来的数据字节已被接收，可以再次发送数据），则打印控制器就会向 CPU 发出中断信号 INT 7。于是，CPU 再继续送出下一个打印数据字节及选通信号，不断重复上述过程直至结束。

IBM PC 的 ROM 常驻 BIOS 中，打印输出控制采用的是查询方式。

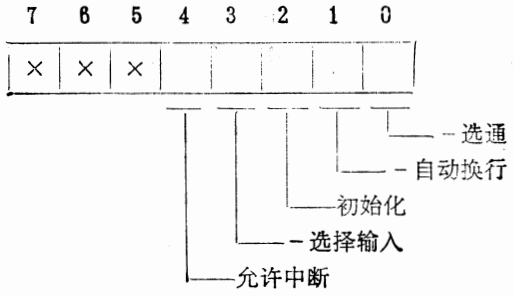
用于打印机程序设计的输入输出指令共五条，其中两条用于输出，三条用于输入，它们的使用方法及相应端口地址简述如下。

打印数据字节的输出

输出端口地址为“3BC”（若使用单独的一块打印控制板选件，则端口地址为“378”，以下仿此，不再另行说明），CPU 通过使用 OUT 指令在此端口输出的就是一个 8 位的打印数据字节。

打印控制字节的输出

输出端口地址为 3BE（或“37A”），CPU 输出的是一个控制字节，其格式及含义如下：



数据字节的输入

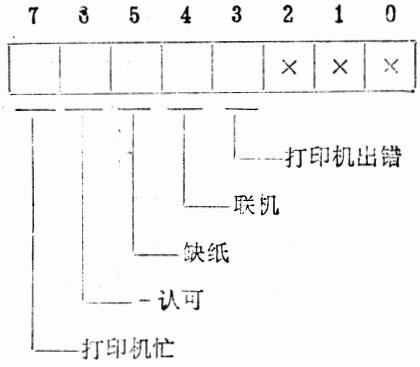
输入端口地址为“3BC”（或“378”），CPU使用输入指令所取得的是控制器与打印机接口上数据总线的状态，一般说来，它正是上一次刚送出的打印字节。本指令用于打印控制器的诊断。

控制字节的输入

输入端口地址为“3BE”（或“37A”），通过这个端口所取得的输入字节通常就是前次所输出的控制字节。本指令也在故障诊断时使用。

状态字节的输入

CPU从端口“3BD”（或“379”）可以取得打印机的状态信息字节，其格式和含义如下：



其中，“打印机忙”表示打印机正在接收数据、或正在打印、或处于脱机状态、或者是打印机出错。

3. IBM 80 CPS 打印机

IBM 80 CPS 打印机是一种经济实惠的点阵式打印机，以每秒可打印80个字符而得名。每个字符由9×9的点阵组成。以标准的字符尺寸输出时，每行可打印80个字符；若选择紧凑的字符尺寸输出，则每行可打印132个字符。两种情况下又都可以用“宽体”输出，这时每行分别可以打印40个或者66个字符。具有图形打印功能的打印机，通过软件送出一

定的控制字符，还可以实现图形和汉字的打印。

不同的打印机有不同的输出字符集。IBM 80 CPS 在打印机上可以打印出来的字符有 96 个标准的 ASCII 字符和 64 个图形字符，另外它还可以处理和执行若干控制字符。表 1-14 是所有有效字符的编码表。

表 1-14　打印机字符编码表

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	NUL		SP	0	@	P	`	p	NUL							
1	0001		DC1	!	1	A	Q	a	q		DC1						
2	0010		DC2	"	2	B	R	b	r		DC2						
3	0011		DC3	#	3	C	S	c	s		DC3						
4	0100		DC4	$	4	D	T	d	t		DC4						
5	0101			%	5	E	U	e	u								
6	0110			&	6	F	V	f	v								
7	0111	BEL			7	G	W	g	w	BEL							
8	1000		CAN	(	8	H	X	h	x		CAN						
9	1001	HT		)	9	I	Y	i	y	HT							
A	1010	LF		*	:	J	Z	j	z	LF							
B	1011	VT	ESC	+	;	K	[	k	{	VT	ESC						
C	1100	FF		'	<	L	\	l	:	FF							
D	1101	CR		-	=	M	]	m	}	CR							
E	1110	SO		.	>	N	^	n	~	SO							
F	1111	SI		/	?	O	_	o	DEL	SI							

所谓“控制字符”，是指那些能被打印机识别并使打印机执行某种特定功能的字符，一般它并不被打印出来，所以也可叫做“不可打印字符”。控制字符共 15 个，从编码表中可以看出，它们的编码与最高位无关，且后 7 位的值均小于 32。下面对控制字符作简单的介绍，因为这在应用软件开发时很有用处。

（1）CR（回车）使打印机开始输出打印缓冲器中已有的内容，打印后自动回车，是否换行需由打印控制字节中的第 1 位来决定。

（2）LF（换行）使打印机开始输出打印缓冲器中已有的内容，打印后自动换行。

（3）VT（纵向列表）缓冲器字符打印完毕后，纸前进到由“ESC B （见下文）指定的位置。

（4）FF（输纸）缓冲器内容打印完毕后，纸前进到下一页的第1行。

（5）HT（横向列表）使打印头横向移到由“ESC Dn”（见下文）所指定的位置上去。若未预先指定横向列表位置，则此码无效。

（6）CAN（取消）和DEL（删除）这两个控制码的作用是一样的，它们使打印缓冲器内的所有数据被清除不再打印。软件到底使用这两个控制码中的哪一个，应根据打印机内电路板上的开关位置而定。

（7）SO（宽体开始）和DC4（宽体结束）在这两个控制字符之间的所有可打印字符，均以双倍宽度的形式打印出来。

（8）SI（紧缩开始）和DC2（紧缩结束）在这两个控制字符之间的所有可打印字符，均按132列/行的紧缩形式输出，若同时又有宽体输出的要求，则每行打印66列。

（9）DC1（选择）和DC3（脱离）DC1控制码使打印机处于联机状态，此后送来的数据有效；DC3使打印机处于脱机状态，此后送来的数据忽略不计。下面的表1-15是打印机上的联机开关，打印控制字节中的第3位（-选择输入），与控制字符DC1和DC3组合使用时的各种情况，供程序设计时参考（“×”表示任意）。

表1-15　打印机联机/脱机的控制

联机开关	-选择输入	DC_1/DC_3	状态字节				可否向打印机送入数据
			-出错	忙	-认可	联机	
脱机	×	×	低	高	不产生	低	不可能
联机	低	×	高	高/低	可产生	高	可正常送入
	高	DC_1	高	同上	同上	同上	同上
	高	DC_3	高	同上	同上	低	可送数据但不进缓冲

（10）NUL（空）用作下面所述的纵向或横向制表设定序列的结束符。

（11）BEL（响铃）引起响铃3秒钟。

（12）ESC（换码）该控制码一定要跟一个或几个其它ASCII字符才会起到所规定的控制作用，所以往往也称为“换码”序列。常用的有如下几种：

* ESC D或ESC 1或ESC 2　使打印时行间距分别设置为1/8吋或7/72吋或1/6吋。ESC 2还用来执行由ESC An所设置的行间距。
* ESC 8　使打印机缺纸时也不停止打印操作。
* ESC 9　取消ESC 8的功能。
* ESC An　设置行间距为n×1/72吋，其中n满足1≤n≤85。此后当打印机收到代码ESC 2时，即按此行间距换行（1/72吋是打印头上面相邻两针之间的距离）。
* ESC B n_1 n_2 … n_R NUL　用来设定纵向列表的停止位置，n表示第n行。
* ESC Cn　指出纸的长度（n表示总的行数）。

* ESC D n_1 n_2 … n_R NUL　用来设定横向列表的位置，n 表示第 n 列。
* ESC E 和 ESC F　指出需要加重打印的字符的开始和结束。
* ESC G 和 ESC H　指出要打印的粗体字符的开始和结束（打印粗体字符是用打印一次后纸前进 1/216 吋，然后再重复打印一次来实现的）。
* ESC K n_1 n_2〈映象信息〉　打印 480×8 点映象的图形，其中 n_1 和 n_2 用来指出映象信息的字节个数，〈映象信息〉是一连串的字节。
* ESC L n_1 n_2〈映象信息〉　同上，但打印的是 960×8 点的映象。

第七节　异步通讯控制器

异步通讯控制器选件板为系统提供一个标准的 EIA RS—232—C 串行接口，它既可用于近程或远程的数据通讯，又可以用来连接附加的一些外部设备。如具有串行接口的绘图仪、数字化桌、汉字或西文终端、各式打印机等。它的主要性能和特点如下：

* 提供一个 RS—232—C 接口，需要时可选用电流环方式操作。
* 数据传送速率可在 50 波特— 9600 波特之间选择。
* 可传送 5 位、6 位、7 位或 8 位的字符。
* 可选用 1 个、1 个半或 2 个停止位。

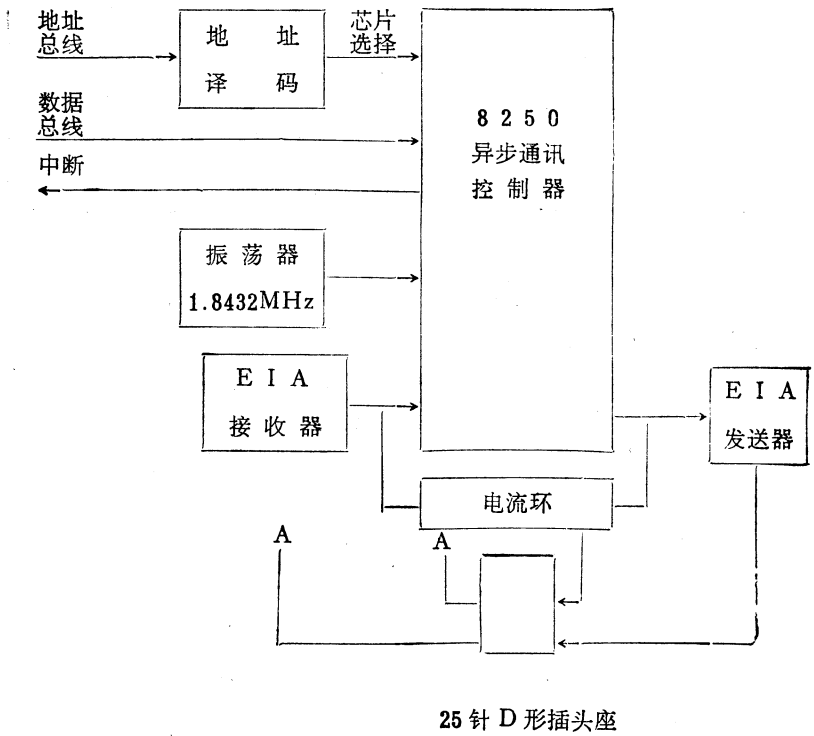


图 1—29　异步通讯控制器逻辑框图

* 可直接连接调制解调器（MODEM），实现远程通讯。
* 能产生或检测“线路暂停”信号。
* 有自行排优的中断控制。
* 具有一定的诊断能力。

1. 逻辑结构与工作原理

图 1—29 是异步通讯控制器的逻辑框图，从图中可以看出，它由振荡器、地址译码器、INS 8250 异步通讯控制器芯片及驱动器和接收器等电路组成。

地址译码器用来识别地址总线上的端口地址是否为3F8——3FE，以便启动（选通）8250中的有关寄存器进行工作。关于各个端口地址及其对应的输入输出字节，在程序设计一节中再作介绍。

INS 8250 芯片是一个大规模集成电路，它是整个控制器的核心。该芯片内部的寄存器结构及其数据通路可参见图 1—30，其中各个寄存器的功能与使用，也将在程序设计一节中介绍。

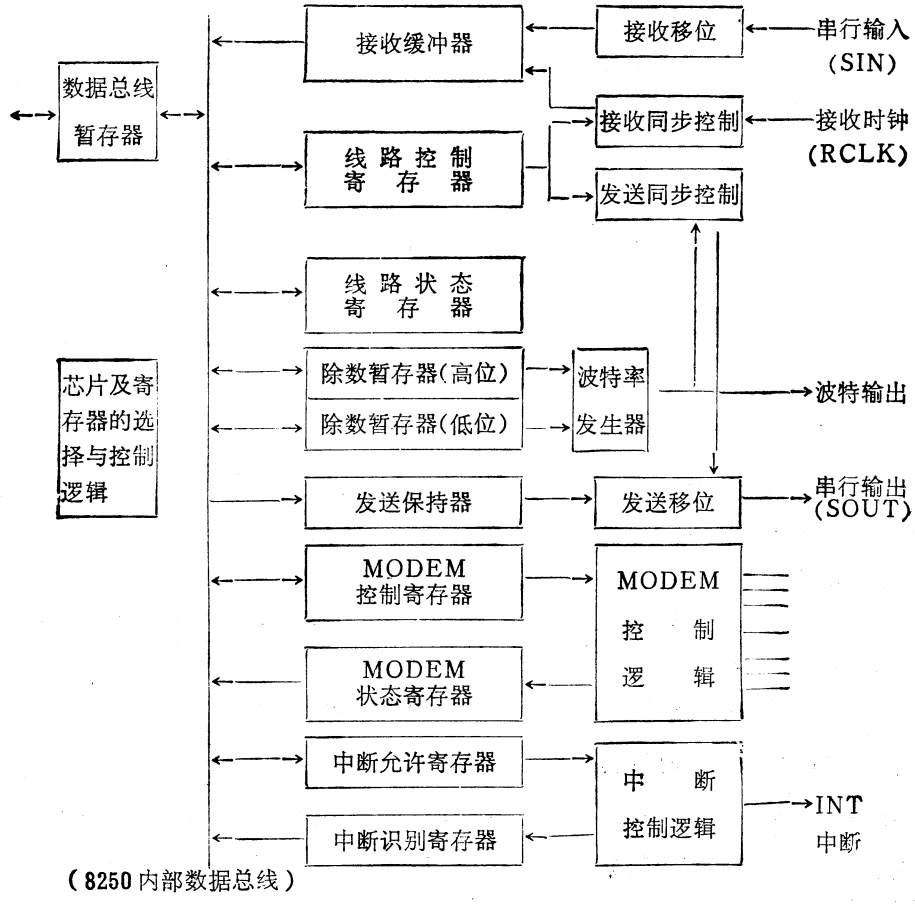


图 1—30　8250芯片的寄存器结构与数据通路

异步通讯控制器与通讯线路或外部设备的接口，是通过一个 EIA RS—232—C 标准的 25 芯 D 形插头座实现的。用户可以在电路板上改变“跨接块”的位置而选择接口的工作方式：电压接口方式或电流环接口方式。

两种接口方式所使用的信号如图 1—31 所示。

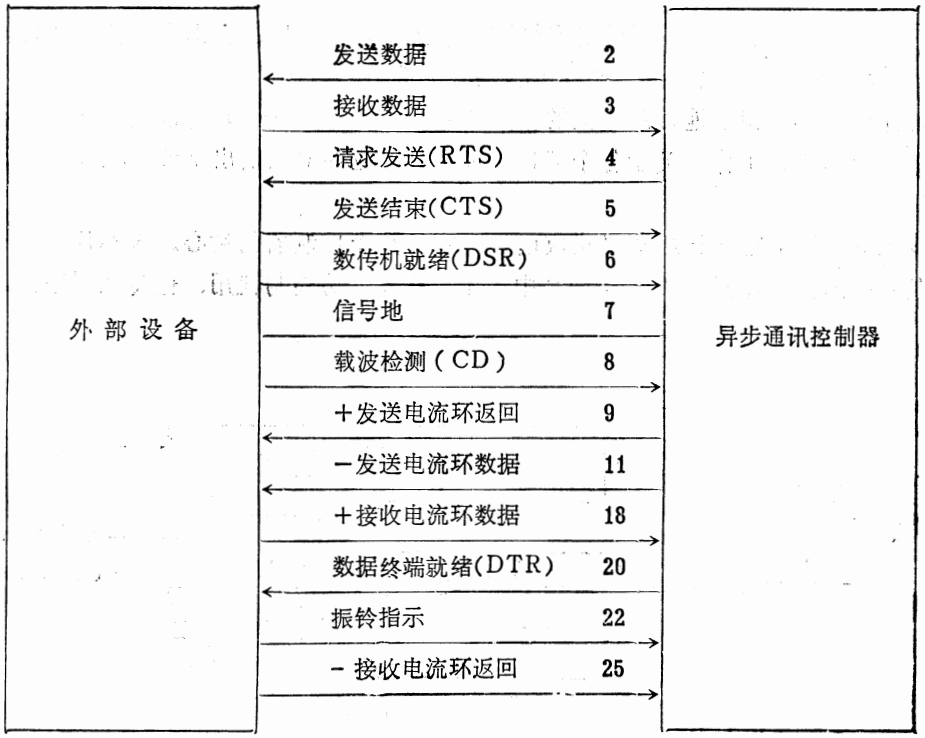


图 1—31 RS—232—C 接口信号

其中 2、3、7 为最基本的三根线（数据输出线、数据输入线和地线），其余信号线通常在应用 MODEM 或通讯控制器时才使用。关于它们的用法，在后面再作解释。

所谓电流环接口方式，即控制器提供一个 20mA 的稳流装置，从而允许与通讯控制器连接的外设可以远离主机而工作（约 1 公里左右，视线路的质量而定）。此时，发送数据由第 9 和第 11 脚送出，到达外部设备后再转换为电压信号；接收数据由第 18 和第 25 脚送入，用光隔离器转换成电压信号。图 1—32 是电流环接口的原理图。

2．异步通讯控制器的程序设计

异步通讯控制器有很灵活的程序设计能力来适应各种不同的应用场合。CPU 可以通过输入输出指令送出控制字节和数据字节，或者取得控制器的状态字节和收到数据字节。

异步通讯控制器程序设计的一般步骤如下：

（1）设定通讯的规程，如波特速率、奇偶校验方式、停止位的数目、数据字节长度等；

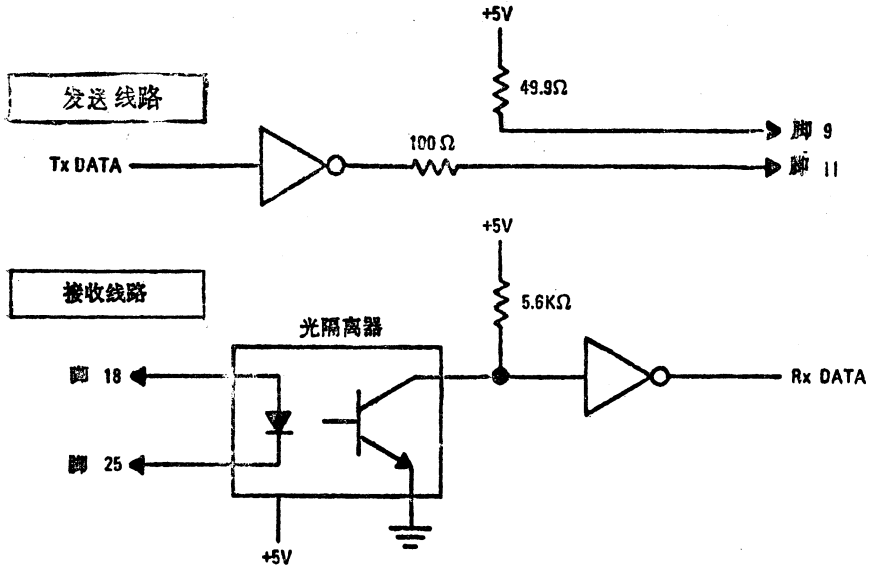


图 1—32　电流环接口的原理

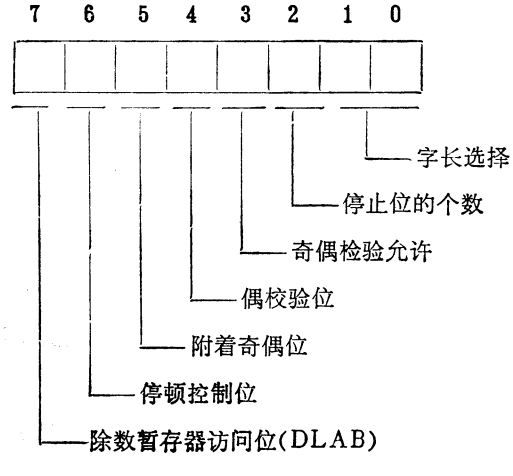
（2）读取通讯线路（或 MODEM）的状态，判断是否已可进行通讯；

（3）送出（或接收）一个数据字节；

（4）重复上述（2）和（3）直到通讯完毕。

当允许中断时，CPU 送出（或收到）一个字节之后，并不需要不断查询控制器的状态，而可转向执行其它任务。当有中断信号 INT4 发生并响应后，再按上述（2）、（3）处理即可。

系统分配给异步通讯控制器的输入输出端口地址为 3F8—3FE，它们都与 8250 芯片中的寄存器相对应。下面从程序设计的角度进行介绍。



数据发送保持寄存器和数据接收缓冲器

这两个寄存器分别用来保存（然后串行移位后发出）和接收正在串行接口上通讯的数据字节。它们的端口地址都是3F8，CPU分别使用输入和输出指令来进行操作。

通讯线控制寄存器和通讯线状态寄存器

CPU通过3FB端口送到通信线控制寄存器的字节，是用来控制通讯线路数据通讯格式的，例如字长、停止位个数、奇偶校验方式等。其内容也可以由CPU用输入指令来读取（端口地址相同），从而方便了程序设计。通讯线控制寄存器的格式如上页所示，其中，字长选择位具体规定为：

Bit 1	Bit 0	字长
0	0	五单位
0	1	六单位
1	0	七单位
1	1	八单位

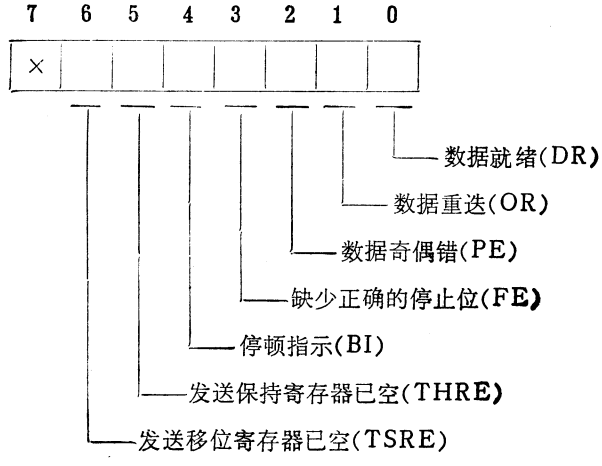
停止位的个数规定为：

Bit 2	Bit 1	Bit 0	停止位个数
0	×	×	1个
1	0	0	1个半
1	（其	它）	2个

停顿控制位用来使串行输出线强置为“空”（即逻辑“0”）状态并保持到这一位复位为止；除数暂存器访问位用来控制CPU对除数暂存器的读写，只有这一位是“1”时OUT指令和IN指令才能访问除数暂存器，以便改变数据传输的波特速率。

通讯线状态寄存器的端口地址为“3FD”，CPU可以用输入指令取得该寄存器的内容，然后通过分析，就可以了解异步通讯控制器的工作状况。

通讯线状态寄存器的格式为：



其中，第0—4位反映了数据接收的情况，包括出错产生的原因；第5和第6位是数据发送的情况，它们在程序中是非常有用的。

除数锁存器

由于8250芯片使用了1.8432MHz的时钟输入，所以需要采用分频的方法来产生所要求的波特速率。分频所用到的除数由CPU分两次分别送入除数锁存器的高位和低位部分，除数可以从下面的公式推算而得：

除数 = 1843200 ÷ （波特速率 × 16）

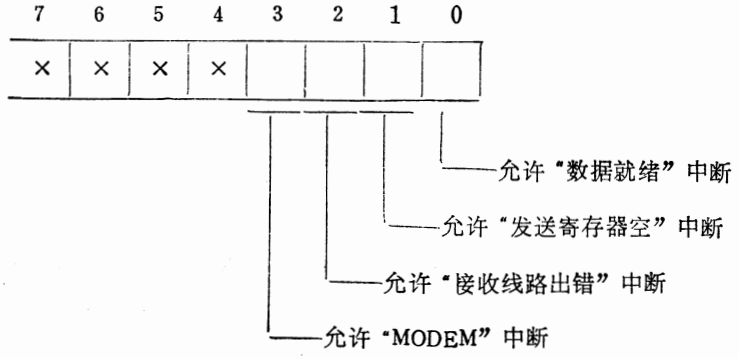
除数寄存器的端口地址是3F9和3F8，但程序必须预先把线路控制寄存器中的最高位（除数暂存器访问位）置“1”，然后才能对它们写入或读出，操作完毕后再把除数暂存器访问位恢复为“0”。

中断允许寄存器

8250芯片本身可以处理四种类型的中断，它们按优先次序排列为：

* 接收线路出错
* 接收数据就绪
* 发送保持寄存器已空
* MODEM中断

系统软件可以按照具体情况，选择开放一部分或全部类型的中断信号，也可以都不允许产生，这是通过向端口地址为3F9（注意，这时控制寄存器中最高位应为“0”）的中断允许寄存器送出一个字节来实现的，该字节的格式为：



中断识别寄存器

当系统允许8250的一种或几种中断信号发出时，由于8250芯片仅能向外输出一个总的中断请求信号，因此异步通讯控制器也只能向CPU发出一个中断信号IREQ4。程序为了能具体识别究竟是由哪一种原因引起的中断，以便分别处置，因此必须通过端口3FA从中断识别寄存器读入一个中断识别字节。中断识别字节的第3—7位恒为“0”（不使用），第0—2位的解释如表1—16所示。

MODEM控制寄存器与MODEM状态寄存器

异步通讯控制器可以通过连接一台调制解调器（MODEM）或多路通讯控制器而实现远程通讯。图1—33是使用MODEM实现数据通讯的示意图。

表 1-16　中断识别字节

中断识别寄存器（2 1 0）	中断类型与原因	中断源复位的控制
0 0 1	无中断	—
1 1 0	接收线路出错(数据重迭,奇偶错,缺停止位,停顿)	读通讯线状态寄存器后即可复位
1 0 0	接收数据齐备	读接收缓冲器后复位
0 1 0	发送保持寄存器空	向保持寄存器写入数据后即复位
0 0 0	MODEM中断(发送结束，数传机就绪,振铃指示,接收线信号检测)	读MODEM 状态寄存器后即可复位

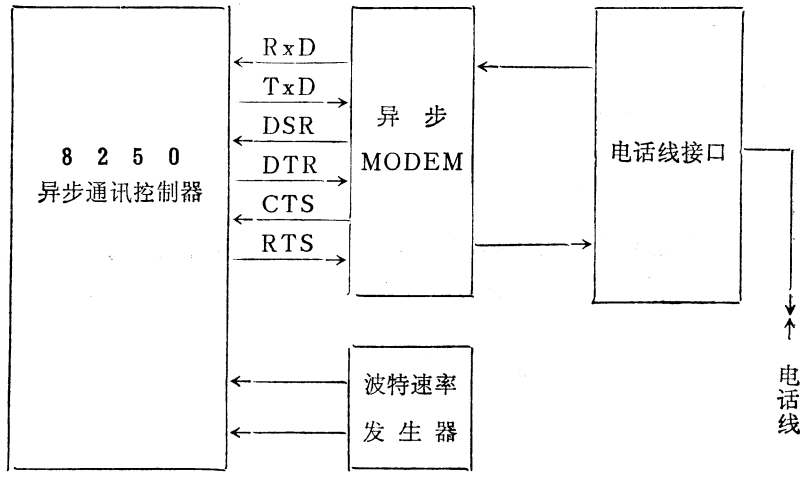


图 1—33　利用MODEM进行数据通讯

利用 MODEM 进行数据通讯，CPU 必须实现对 MODEM 的控制及读取它的状态，这是分别通过 MODEM 控制寄存器（MCR）和 MODEM 状态寄存器（MSR）来进行的。

MCR 的端口地址为 3FC，CPU 输出给它的控制字节中低 5 位分别起如下的控制作用（高 3 位不用）：

* 第 0 位　向 MODEM 指出异步通讯控制器已处于就绪状态（DTR）。
* 第 1 位　请求向 MODEM 发送数据（RTS）。
* 第 2 位和第 3 位　用来产生两个通用的输出控制信号（OUT 1 和 OUT 2），视不同需要而使用。
* 第 4 位　在诊断测试 8250 芯片时，用于使 MCR 和 MSR 不通过 MODEM 而自行构成一个闭合回路。

MSR 的端口地址为 3FE，CPU 可以从 MSR 中取得一个字节的有关 MODEM 的现行状态信息，其各位的含义如下：

* 第 7 位　　查出有接收线信号
* 第 6 位　　查出有振铃信号
* 第 5 位　　MODEM 处于就绪状态（DSR）
* 第 4 位　　MODEM 数据发送结束（CTS）
* 第 3—0 位　　分别用来指出第 7—4 位所对应的 MODEM 的四个输入信号线从 CPU 上次读过了 MSR 内容之后，状态又有了改变。

下图是 IBM PC 的 BIOS 中通过异步通讯控制器向外发送一个数据字节的大致处理过程（图 1—34）。

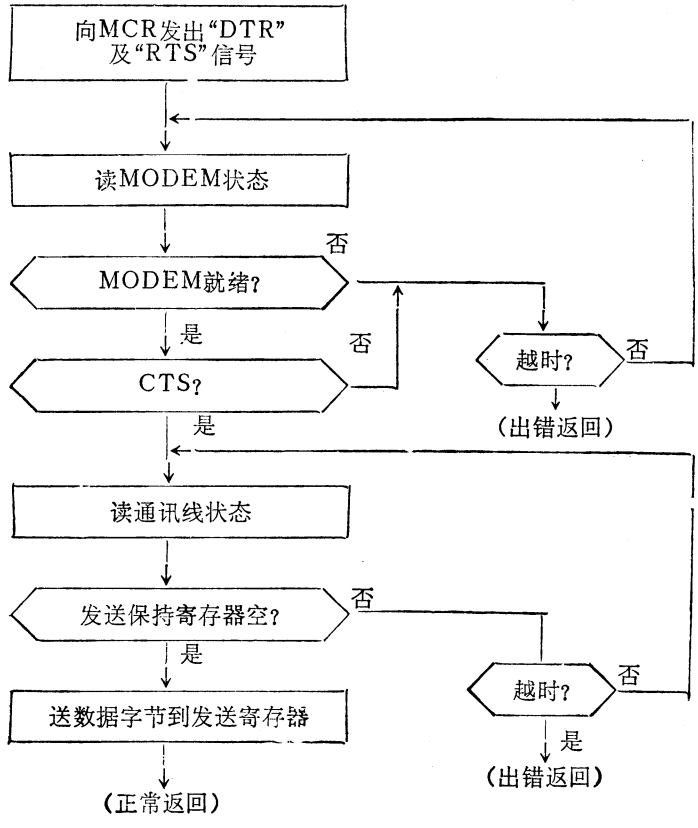


图 1—34　异步通讯控制器发送数据的流程

第八节　磁盘驱动器及其控制器

1．5 1/4 吋软盘驱动器

IBM PC 使用的软盘驱动器是单面双密度或双面双密度的、使用 5 1/4 吋软磁盘的一种外存贮器，每一面有 40 个磁道（也叫"磁轨"），按每扇区 512 个字节格式化之后，单面盘容量达 160 KB 或 180 KB，双面盘容量可达 320 KB 或 360 KB，它使用改进的频率调制

法（MFM）进行数据的读写。

盘驱动器由主轴驱动系统、磁头定位系统以及读／写／抹除系统组成。主轴由伺服机构控制的直流马达以每分钟 300 转的速度驱动。磁头的定位由四相步进马达及有关电路来进行，马达转动一步就带动磁头移过一条磁道。数据恢复电路中包括低电平读放大器、差分电路、零点检测器及数字化电路等，而所有的数据译码操作都在控制器插件板上通过数据分离电路完成。

盘驱动器与盘控制器的接口比较简单（图 1—35），其中驱动器提供给控制器的信号有：读出数据信号，写保护信号，索引信号（表示盘片旋转到某起始位置，每转一周，发出一次），0 号轨信号（表示磁头正停在 0 号磁道）。控制器发给盘驱动器的信号有：驱动器选择信号（使驱动器与控制器逻辑上接通），马达允许信号（控制驱动器的主轴马达旋转或停止），走步信号（使所选驱动器的读／写磁头按指定方向移动，一次一轨），方向信号（指出走步方向），写数据及写允许信号（允许把数据写入磁盘），磁头选择信号（选择两个磁头之一）等。

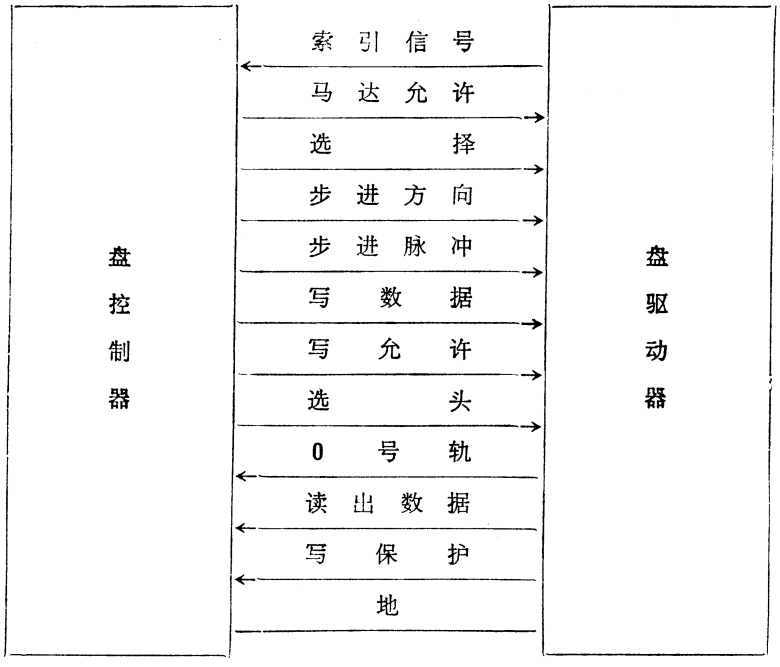


图 1—35 盘控制器与驱动器的接口

2．软盘驱动器的控制器

IBM PC 机箱中可以装入一台或两台软盘驱动器，而软盘驱动器的控制器选件板却可以最多连接四台驱动器——两台在主机箱内，另两台需要使用扩充机箱。

盘控制器的逻辑框图见图 1—36，它的核心部分是 NEC μPD 765 软盘控制器芯片（以下简称“FDC”），也可以使用其它与之兼容的软盘控制器芯片如 Intel 8272 等。

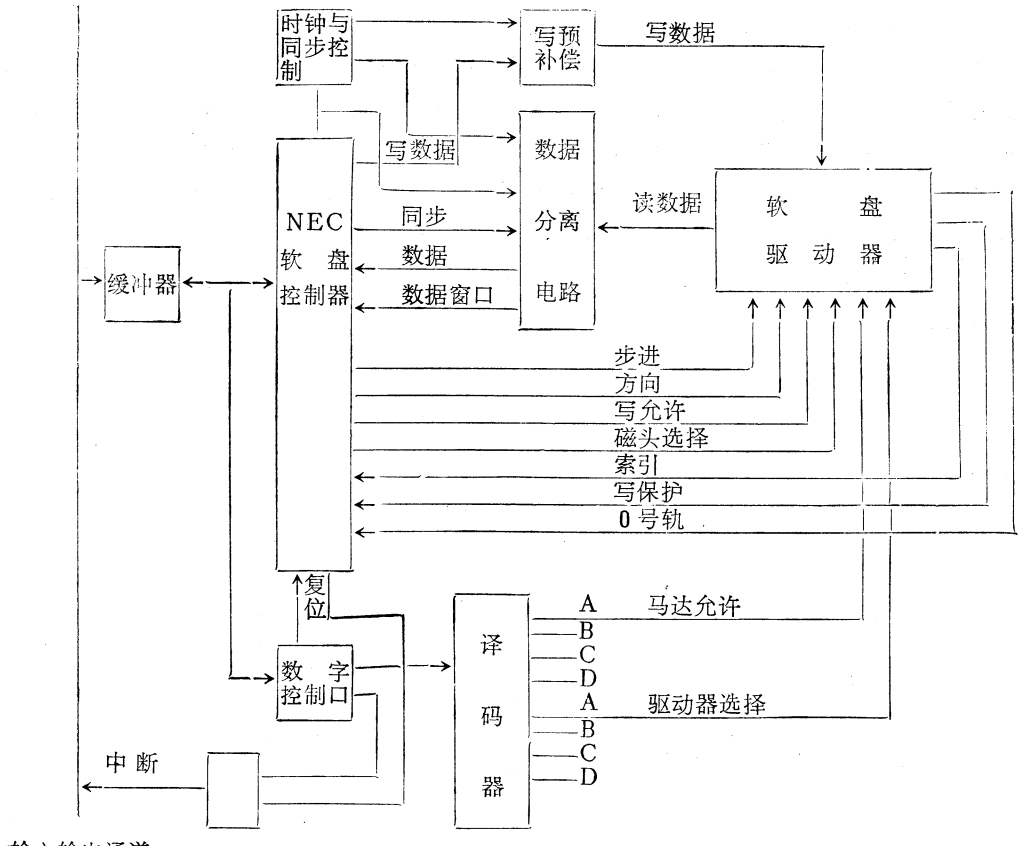


图 1—36　磁盘控制器逻辑框图

盘控制器与 CPU 的接口是输入输出通道。除了八根数据线和十根地址线（A_9—A_0）之外，还需要使用的控制信号有：

* DRQ2　　直接内存访问请求信号
* AEN　　直接内存访问标志
* DACK2　　直接内存访问认可信号
* T/C　　DMA 方式下计数已达到 0
* IOR 和 IOW　　CPU 执行输入输出指令
* IRQ 6　　中断请求信号
* RESET　　复位信号

磁盘控制器逻辑框图中的缓冲器用来暂存 CPU 要向软盘写入（或已从软盘读出）的数据。盘控与内存之间的数据传送是用 DMA 方式进行的，数据传送操作或其它操作的结束则通过发出中断信号 IRQ 6 使 CPU 进行相应的处理。

译码器的作用是选择驱动器和接通马达，这是根据 CPU 执行 OUT 指令送到端口“3F2”来的数据字节而决定的。

写预补偿电路用来实现双倍密度的 MFM 记录技术。数据分离电路则使用一个模拟的锁相回路把数据信号和时钟信号分离出来。它们都涉及到较复杂的电路技术，这里不再细述。

软盘控制器芯片（FDC）是一个大规模集成电路，它既可支持IBM 3740标准的单密度（调频制FM），也可支持IBM系统34采用的双倍密度格式（改进调频制MFM）。能用DMA方式与内存贮器传送数据，也能采用通常的非DMA方式，此时，每传送一据字节，芯片就会产生一次中断信号。FDC的其它特征还有：每个扇区的记录长度可程序进行设定为128、256、512或者1024字节；具有一次读写多个扇区的能力；具有扫描的能力——即存贮器中的数据与软盘上的数据逐个字节地读出并比较；几个驱动器进行查找操作等等。图1—37是μPD 765的方框图，其中着重列出了有关的接口信号名，它们的极性（正或负）并未强调指出。

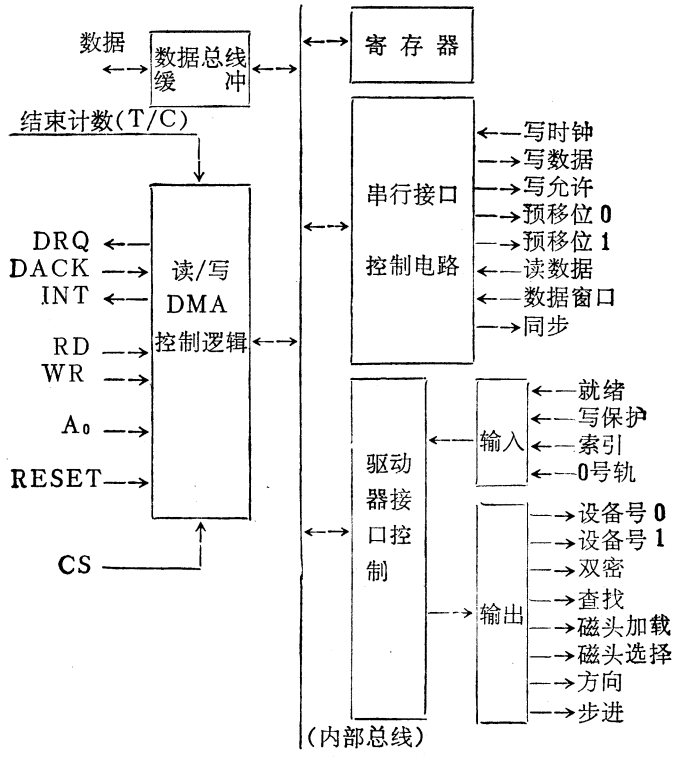


图1—37 μPD 765逻辑框图

3．软盘的程序设计

IBM PC软盘控制器的程序设计要比其它外围设备的编程复杂一些，一般按如下五个步骤来进行：

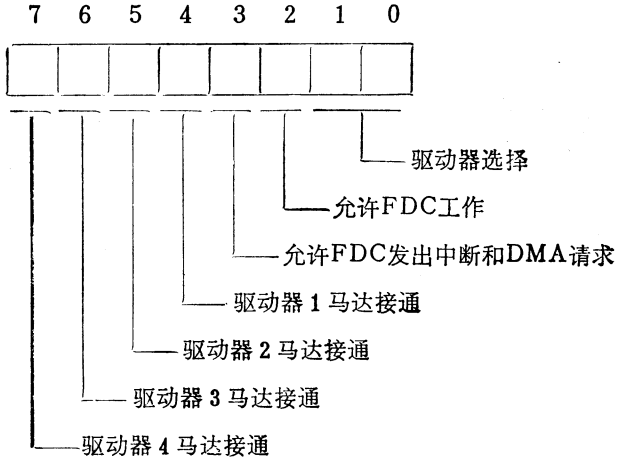
* 选择驱动器，设定控制器的操作模式。
* 读取FDC的主状态寄存器，以了解控制器和驱动器的当前状态。
* 向FDC发出需要执行的命令序列。
* 等待命令执行完毕（CPU在此期间可转向执行其它任务）。

* 取得结果状态，分析命令序列执行正确与否，再作适当处理。

下面依次介绍有关的内容。

（1）驱动器选择和操作模式的设定

这是通过 CPU 向盘控制器中的一个“数字输出寄存器（DOR）” 输出一个控制字节来实现的，输出端口地址规定为 3F2，DOR 的格式如下：



其中，第 0 位和第 1 位用来选择四个驱动器之一：

Bit 1	Bit 0	驱动器号
0	0	1
0	1	2
1	0	3
1	1	4

必须注意，只有已经接通了马达的驱动器才能被选择。

（2）主状态寄存器（MSR）

主状态寄存器在 FDC 芯片中，它用来反映 FDC 的当前状态，CPU 可以通过输入指令从端口地址 3F4 取得这个状态字节，该字节各位的含义是：

* 第 0 位　1 号驱动器正在查找。
* 第 1 位　2 号驱动器正在查找。
* 第 2 位　3 号驱动器正在查找。
* 第 3 位　4 号驱动器正在查找。
* 第 4 位　FDC 正在进行读写操作。
* 第 5位　FDC 采用非 DMA 方式传送数据。
* 第 6 位　表示 FDC 与 CPU 之间的数据传送方向，“1” 表示从 FDC 送到CPU；“0” 表示从 CPU 送到 FDC。
* 第 7 位　表示 FDC 的数据寄存器已可从 CPU 接收（或向 CPU 送出）数据。

（3）命令序列的发送

μPD765 有一组数据寄存器用来接收 CPU 送来的操作命令以及有关参数（叫做“命令序列”），这一组数据寄存器构成了一个“堆栈”，每次只有栈顶与数据总线接通，所以只需要使用一个输入输出端口地址“3F5”。

FDC 能够执行 15 种不同的命令序列。由于涉及的参数不同，故不同命令序列的长度并不一样。例如，向软盘写入数据的命令序列由 9 个字节组成。因此，为了起动盘控制器执行数据写入的操作，CPU 需要向“3F5”端口连续九次输出相应的命令代码及有关参数，然后FDC才会开始执行磁盘的写入操作。

下面对 15 种命令序列分类作概括的介绍。

① 读数据/读删除标记数据/写数据/写删除标记数据/读一条磁道/扫描相等/扫描小于等于/扫描大于等于

这八个命令序列都需要 9 个字节组成，除命令码字节（第一字节）各不相同外，其它 8 个字节的格式、顺序及意义都一样。下表是它们的命令码字节（表 1-17）。

表 1-17　读／写命令码

命令码字节								命　令
D7	D6	D5	D4	D3	D2	D1	D0	
MT	MF	SK	0	0	1	1	0	读数据
MT	MF	SK	0	1	1	0	0	读删除数据
MT	MF	0	0	0	1	0	1	写数据
MT	MF	0	0	1	0	0	1	写删除数据
0	MF	SK	0	0	0	1	0	读磁道
MT	MF	SK	1	0	0	0	1	扫描相等
MT	MF	SK	1	1	0	0	1	扫描小于等于
MT	MF	SK	1	1	1	0	1	扫描大于等于

写删除标记数据与写数据两条命令的功能都是把 CPU 送来的数据写入到磁盘上去，但前者是在扇区中数据场内放入删除数据寻找标记，而后者却是正常的数据寻找标记。

读删除标记数据与普通的读数据命令（当 SK＝0 时）的差别在于：前者可以读出具有删除数据寻找标记的数据场中的数据，而后者却不行，而且读操作不能继续下去。

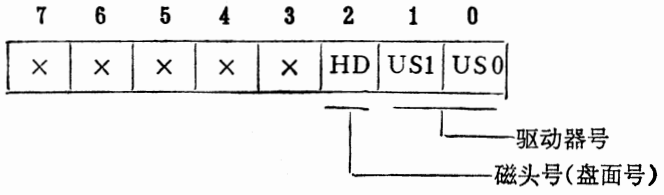
读磁道命令能把整个一条磁道上所有扇区中的数据全部读出。

三条扫描命令用来对磁盘上已有的数据与内存贮器中的数据逐个字节地作比较，看是否符合相等、小于等于或大于等于的条件。

命令码字节中的 MT＝0 表示读（或写）操作只在磁盘某一指定面的某条磁道上进行；而当 MT＝1 时，表示可在同一柱面上的两条磁道（分别位于“0”面和“1”面）上连续进行（先 0 面后 1 面）；MF＝0 表示单密度记录方式，MF＝1 表示双密度记录方式；SK＝1 表示读出时应跳过具有删除数据寻找标记的那些扇区。

这八条命令序列的第 2—9 个字节的格式和意义都是一样的，它们是：

* 第 2 字节　　其格式为：



* 第 3 字节（C）　柱面号（0—39）。
* 第 4 字节（H）　磁头号，它与 HD 相同。
* 第 5 字节（R）　读（写）的起始扇区号。
* 第 6 字节（N）　扇区中数据字节的个数，其表示方法为：

0——128 字节
1——256 字节
2——512 字节
3——1024 字节

* 第 7 字节（EOT）　柱面上最后一个扇区的号码。
* 第 8 字节（GPL）　扇区之间的间隔（间隔 3）的长度。
* 第 9 字节（DTL）　当 N＝0 时，它用来指出在一个扇区内应该读（或写）的字节数目；若 N≠0，则 DTL 无意义，并表示为 FF。

② 磁道格式化

本命令序列用来完成整个一条磁道的格式化，它一共有六个字节组成：

* 第 1 字节（命令码）

7	6	5	4	3	2	1	0
0	MF	0	0	1	1	0	0

（MF 指出单密度还是双密度）

* 第 2 字节　同第①组命令。
* 第 3 字节（N）　扇区中数据字节的个数。
* 第 4 字节（SC）　每轨的扇区数。
* 第 5 字节（GPL）　扇区之间的间隔 3 的长度。
* 第 6 字节（D）　在扇区的数据场中应填入的字节。

格式化命令执行过程中，CPU 应向 FDC 提供每个扇区的柱面号（C）、磁头号（H）、扇区号(R)和扇区长度（N）等信息，直到一条磁道的格式化全部做完为止。

③ 查找

查找命令序列使磁头定位到指定的磁道上，它由三个字节所组成：

* 第 1 字节（命令码）

7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1

* 第 2 字节（驱动器号）　同第①组命令。

* 第 3 字节（NCN）　需要磁头定位到达的道号。

注意，查找命令序列只有发送命令阶段和执行阶段，并无结束阶段，CPU 需通过“取中断状态”命令序列才能了解到查找命令执行的结果。

④ 指定参数

本命令序列用来规定 FDC 内部的三个定时器的初始值，这三个定时器分别用于控制磁头去载时间（HUT）、磁头步进速率时间（SRT）和磁头加载时间（HLT），此外它还用来选择 FDC 是否采用 DMA 方式传送数据。命令序列由以下三个字节组成：

* 第 1 字节（命令码）

7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	1

* 第 2 字节（参数）

7	6	5	4	3	2	1	0
SRT				HUT			

（其中参数 SRT 的单位 2ms，HUT 的单位 32ms）

* 第 3 字节（参数）

7	6	5	4	3	2	1	0
HLT							ND

（HLT 的单位 4ms，ND=1 表示选择非 DMA 方式）

⑤ 读标识场

本命令序列用来了解记录磁头的现行位置。其方法是 FDC 收到并识别了本命令序列之后，立即把读出的第一个标识场（ID场）中的标识信息（柱面号、磁头号、扇区号、扇区长度）分别放入结果状态寄存器之中，然后 CPU 通过一组输入指令来读取之。

本命令序列一共有两个字节：

* 第 1 字节（命令码）

7	6	5	4	3	2	1	0
0	MF	0	0	1	0	1	0

* 第 2 字节（设备号） 同第①组命令。

⑥ 磁头回零道

本命令序列用来使磁头退回到第 0 道，这是一个经常使用的命令，特别是在软盘操作出错时可以起到重新校正磁头定位的作用。它有两个字节：

* 第 1 字节（命令码）

7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1

* 第 2 字节（设备号）

7	6	5	4	3	2	1	0
×	×	×	×	×	0	US_1	US_0

本命令序列同查找命令一样，也无结束阶段，CPU 需通过“取中断状态”命令序列才能了解命令执行的结果。

⑦ 读驱动器状态

CPU 任何时候都可以使用此命令序列来了解软盘驱动器的状态，它有两个字节：

* 第 1 字节（命令码）

7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0

* 第 2 字节（设备号） 同第①命令序列。

⑧ 读中断状态

由于下列原因 FDC 会产生中断信号：

* 读数据/读删除标志数据/读一条磁道/读标识场/写数据/写删除标志数据/格式化/扫描比较等命令序列进入结束阶段时会产生中断信号。
* 盘驱动器的就绪信号（Ready）有变化时。
* 查找操作和回零道操作结束时。
* 非 DMA 方式下，命令序列的执行阶段也会产生中断（每当 FDC 需要与内存传送数据时）。

其中第 1 和第 4 种原因引起的中断 CPU 很易辨认（根据主状态寄存器的第 5 位），而第 2 和第 3 两类原因引起的中断就需要使用本命令序列来进行识别了。由于查找和回零道两个命令序列都没有结束阶段，因此当它们执行完毕发出中断信号之后，CPU 不能使用通常的输入指令来了解操作的结果，而应该使用本命令序列来读取两个字节的状态信息，以判别中断原因和现行磁头的位置。

如果没有中断信号待决时发出本命令序列，则 FDC 把它作为无效命令序列处理。

本命令序列只有一个字节组成：

* 第 1 字节（命令码）

7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0

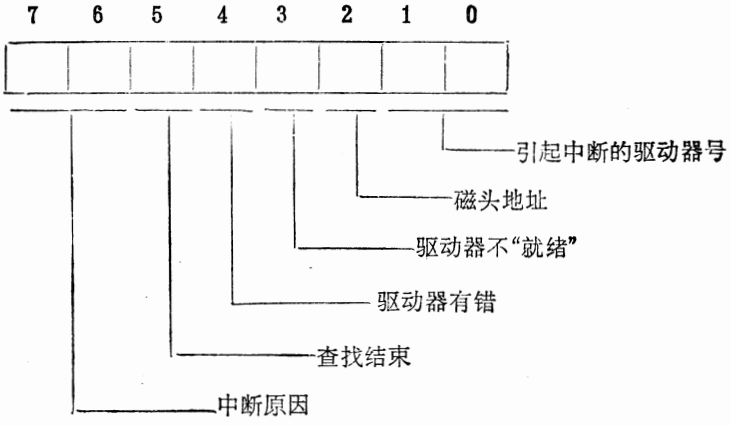
（4）结果状态的读取

上面介绍的 15 种命令序列中，除了查找、回零道、及指定参数三种命令序列之外，当软盘控制器 FDC 把命令序列执行完毕后，无论是正常情况结束还是出错而引起结束，都会在 FDC 的数据寄存器堆栈中形成若干按顺序存放的、有一定格式的状态字节，然后，CPU 可以用输入指令从端口地址“3F5”取得这些状态字节，有几个状态字节就需要使用几次输入指令，直到堆栈中取空为止。

命令序列执行完毕后在堆栈中形成的状态字节的个数、顺序和格式大都是一样的，只有“取驱动器状态”和“取中断状态”是例外。

这些状态字节共计有七个，下面按序作扼要的介绍。

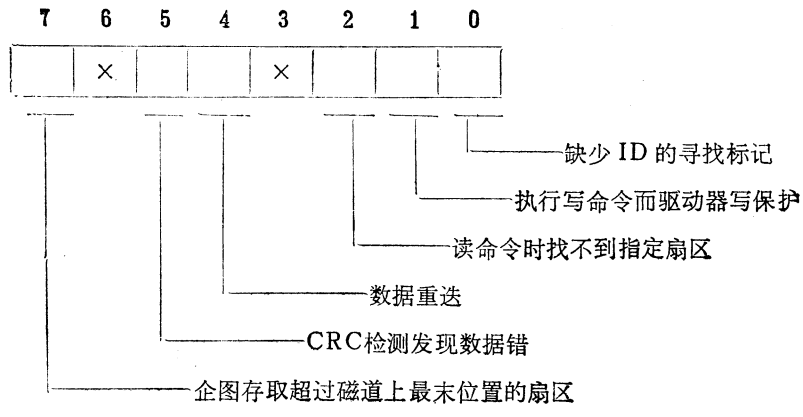
* 第 1 字节（ST0）　中断原因



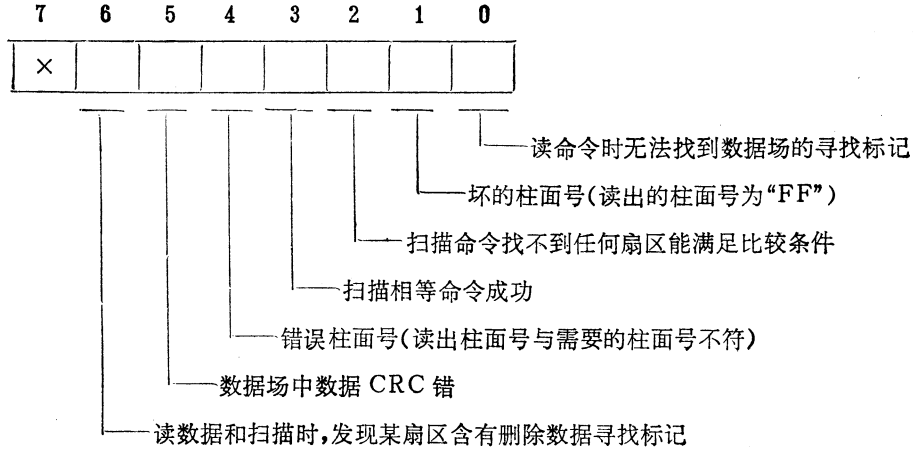
其中，第 6 和第 7 两位指出四种中断原因之一，具体规定为：

Bit7	Bit6	中 断 原 因
0	0	命令正常结束
0	1	命令异常结束
1	0	无效命令序列
1	1	由于驱动器原因命令异常结束

* 第 2 字节（ST1）　出错原因 1



* 第 3 字节（ST2）　出错原因 2



* 第 4 字节（C）　命令执行后扇区标识场中的柱面号。
* 第 5 字节（H）　命令执行后扇区标识场中的磁头号。
* 第 6 字节（R）　命令执行后扇区标识场中的扇区号。
* 第 7 字节（N）　命令执行后扇区标识场中的扇区长度。

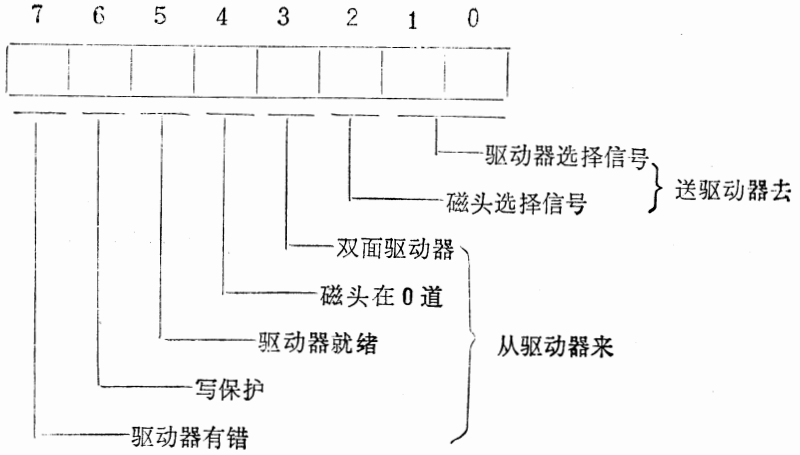
"取驱动器状态"命令序列执行完毕后，栈中只形成一个状态字节，它的格式如下页所示。

"取中断状态"命令序列执行后形成的两个状态字节是：

* 第 1 字节（ST0）　同前。
* 第 2 字节（PCN）　指出磁头所在的现行柱面号。

下面让我们给出 BIOS 中软盘驱动程序的"读操作"子程序的流程图。该子程序的功能是从软盘上的指定位置处读出一个或多个扇区（不能超过八个或九个）中的数据送入内存中的缓冲区。调用该子程序之前，必须在有关寄存器中放入下列参数：

（DL）＝驱动器号　　（DH）＝磁头号



（CL）＝扇区号　　（CH）＝磁道号

（AL）＝扇区数目

（ES:BX）＝内存中缓冲区地址

（AH）＝2　（表示进行磁盘的读操作）

读操作的流程图如下（图 1—38）：

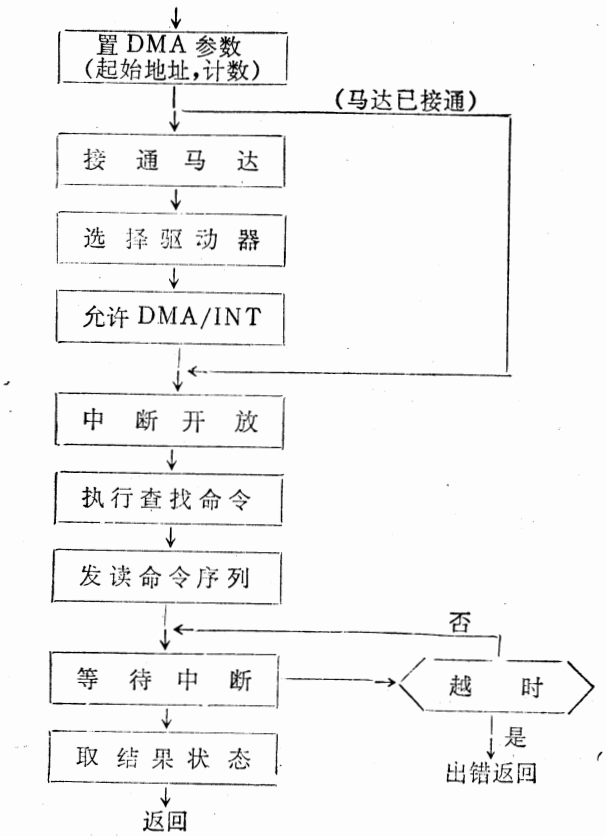


图 1—38　读子程序的流程

“读”子程序执行后返回时，AH中存放操作的状态，AL中是实际读出的扇区数目，标志位CY=0表示读出成功，CY=1表示读操作有错（AH中是错误原因）。通常，若读出有错，则可以反复再读几次，这叫做使用软件进行“重执”。

4. 硬盘及其控制器

IBM PC/XT(扩充型)与PC基本型的主要差别是增加了一台10MB的温彻斯特硬磁盘。温彻斯特磁盘是一种盘片不可更换的固定盘，它是微型机常用的一种高速大容量外存贮器。

（1）IBM 10MB硬盘驱动器及其控制器

PC/XT使用的5¼吋硬盘主要性能指标如下：

* 记录面总数　　4
* 磁道总数　　4×306=1224
* 每道扇区数　　17
* 扇区字节数　　512
* 总存贮容量　　10.4MB
* 数据传输速率　　5.0Mbit/秒

硬盘控制器选件板可以连接一台或两台硬盘。除了数据线和地址线之外，它与CPU的接口信号主要有直接访内请求信号（DRQ3和－DACK3）和中断请求信号（IRQ5）等，前者用于控制硬盘与内存之间的数据传送，后者在每次硬盘操作完毕后发出，表示指定的硬盘操作已经完成。硬盘控制器的逻辑结构与软盘相仿，这里不再赘述。

（2）硬盘的程序设计

系统要求硬盘完成读、写或定位等操作时，必须通过I/O指令进行。程序设计的一般步骤简要介绍如下。

① 选择硬盘控制器及设置其工作模式　　例如，是否采用DMA数据传送方式，是否允许发出中断请求信号等，这是通过OUT指令向端口323发出一个屏蔽字节进行设置的。在此之前，必须预先使用OUT指令选择该硬盘控制器（端口号322）。

② 向盘控发出操作命令　　与软盘类似，硬盘的操作命令由六个字节组成（叫做设备控制块DCB），DCB的格式大致如下：

	7 6 5 4 3 2 1 0
字节0	操作类型 | 操作码
字节1	0 0 d | 磁头号
字节2	柱面号(高) | 扇区号
字节3	柱面号（低）
字节4	读/写块数或交错因子
字节5	控制字段

其中d表示驱动器号。不同的操作命令其操作类型和操作码各不相同，主要的操作命令有：

* 读，长读（读出数据中还附有纠错码）。
* 写，长写（写入数据中还附有纠错码）。
* 定位。
* 连续格式化，单轨格式化。
* 取断定状态。

DCB 的六个字节是连续使用 OUT 指令通过端口 320 送给控制器去执行的。

③ 等待操作结束　CPU 发出操作命令后，硬盘控制器即可独立完成指定的操作。操作结束后，若处于中断允许方式，则立即向 CPU 发中断请求，否则，会在硬件状态寄存器的第 0 位置“1”，表示要求向 CPU 送出操作结束状态字节。

④ 取结束状态字节　硬盘每次操作完毕后，都要形成一个结束状态字节，用于指出操作是否正常结束。结束状态字节中第 1 位（最右面是第 0 位）若为“1”表示有错，第 5 位是驱动器号，其它各位均不使用。CPU 可以通过输入指令从端口 320 处取得这个字节。

⑤ 若操作有错可发出“取断定状态”命令了解出错的具体情况。断定状态由四个字节组成，字节 0 指出错误类型及错误码，若为全 0 则表示没有出错。它们的格式如下：

	7	6	5	4	3	2	1	0
字节 0	地址有效	0	错误类型		错	误	码	
字节 1	0	0	d	磁	头	号		
字节 2	柱面号(高)		扇	区	号			
字节 3	柱	面	号	（低）				

断定状态字节是在 CPU 发出命令后，通过四次使用输入指令从端口 320 读入的。此时，盘控制器应处于屏蔽 DMA 及中断的模式。

另外，硬盘控制器还提供一个硬件状态字节，它的格式为：

7	6	5	4	3	2	1	0
×	×		×				
		中断		控制器忙	总线忙	控制模式	请求传送

CPU 随时可以通过输入指令从端口 321 读入这个字节，以便了解盘控制器的状态。

（3）PC/XT 的扩充 BIOS 对硬盘的控制

PC/XT 硬盘控制器选件板上的只读存贮器中，存放着 BIOS 的扩充程序，主要包括：

* 硬盘的加电诊断程序
* PC/XT 的自举程序
* 硬盘的驱动子程序

当系统加电或总清时，扩充 BIOS 程序除了在地址 0034H—0037H 中填入硬盘中断服

务程序的起始地址之外，还将修改原基本系统的中断向量表。例如，使仅用于处理软盘调用的 INT13 改成 INT40，而把 INT13 作为软盘和硬盘统一处理的软中断入口；把 INT19 修改成转向扩充 BIOS 中的新的自举程序等等。

最后给出 PC/XT 的扩充 BIOS 中硬盘读出子程序的操作流程图（图 1—39），从中可以看出硬盘程序设计的若干主要环节。

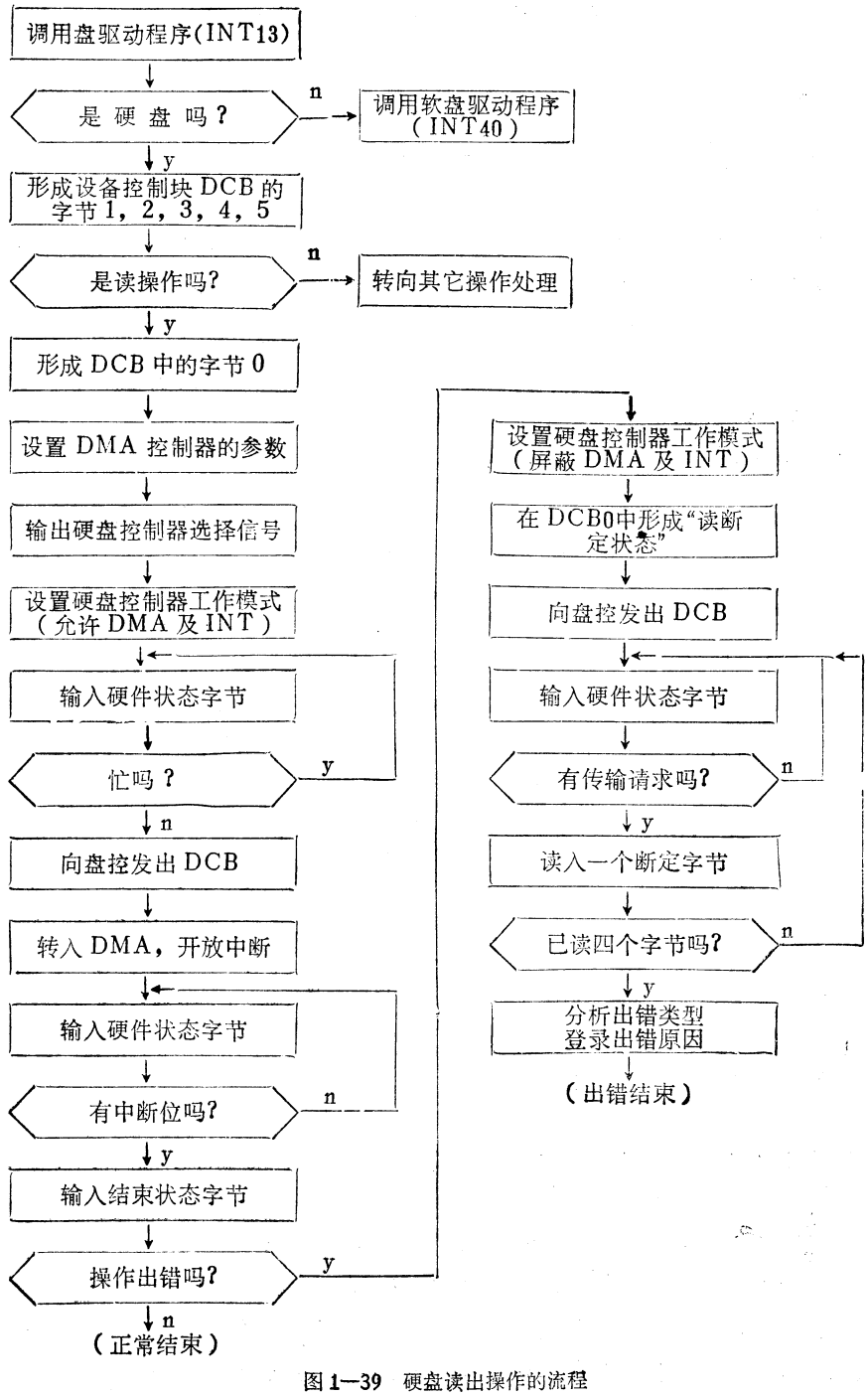


图 1—39　硬盘读出操作的流程

第二章　MS-DOS操作系统

第一节　概　述

1. 引　言

MS-DOS是美国MICROSOFT公司为IBM PC微机开发的磁盘操作系统，也称为IBM-DOS或PC-DOS。和其它单用户、单作业的微机操作系统一样，MS-DOS的功能主要是进行文件管理和设备管理。其中文件系统负责建立、删除、读写和检索各类文件，而I/O系统则负责驱动外围设备，例如显示器、键盘、磁盘、打印机以及异步通讯器等。

MS-DOS采用层次模块结构，它由三个层次模块和一个引导程序组成。这三个模块是：输入输出系统、文件系统（IBMDOS·COM）和命令处理程序（COMMAND·COM）。其中输入输出系统又由驻在ROM中的基本输入输出系统BIOS和系统盘上的BIOS接口模块IBMBIO·COM两部分组成。三个模块之间的层次关系如图2—1所示。

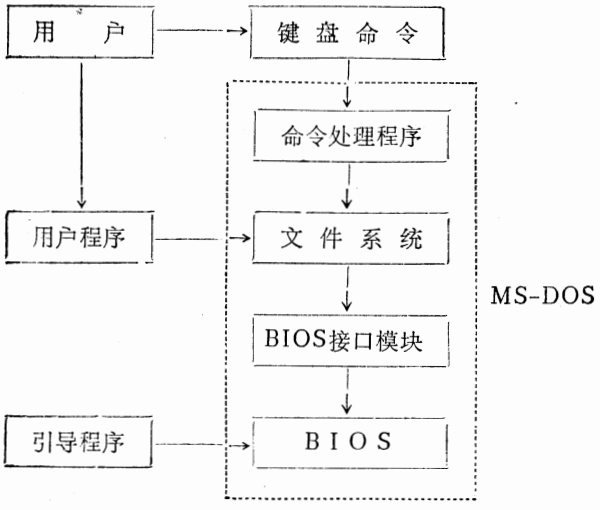


图2—1　MS-DOS的层次结构

MS-DOS是用户与物理机器的接口，用户通过使用键盘命令或用户程序（如汇编语言程序或高级语言程序）来使用MS-DOS。

MICROSOFT公司至今已开发了两个版本的MS-DOS。第2版比第1版在功能上有较大的扩充，即：增加了支持硬盘的功能；把软盘的容量从320K扩充到360K；文件系统增加了子目录操作、I/O重定向以及管道操作；内部命令和外部命令各增加了十几条；功能调用增加了30余条。本章，我们将主要以MS-DOS 2.00版为例介绍MS-DOS操作系统。

2. 系统启动

（1）初始启动

首先，把系统盘插入A驱动器，关上小门。随后，如果机器尚未加电，则打开机器的电源，否则就按住〈Ctrl〉键和〈Alt〉键同时按下〈Del〉键（即系统总清）。稍等片刻后，屏幕上显示出下列信息：

```
Current date is Tue 1—01—1980
Enter new date:
```

打入当天的日期（用减号“-”或斜杠“/”分隔月、日和年）。接着，系统又提示出：

```
Current time is 0:01:26.94
Enter new time:
```

输入时间后，MS-DOS显示出下列签到信息：

```
The IBM Personal Computer DOS
Version 2.00 (C)Copyright IBM Corp 1981, 1982, 1983
A>
```

其中A>是MS-DOS的提示符，表明系统已齐备，随时可以接收输入的键盘命令。

上文提示符中的A为当前盘盘符，它可用选盘命令来改变。例如下列命令：

```
A>B:〈CR〉
B>
```

表示把当前盘改为B盘，其中下横线标出部分表示用户输入的信息。

用户可用改换提示符命令PROMPT任意更改系统提示符。例如：

```
A>PROMPT #〈CR〉
#
```

将提示符改成“#”，而下例：

```
#PROMPT $n$g 〈CR〉
A>
```

又将提示符改成初始值。命令中的$n表示当前盘盘符；$g表示大于号“>”。

（2）自动启动

自动启动是从系统启动开始，引导MS-DOS并且自动地运行特定的批命令文件的一个过程。它常用于自动启动用户程序。

为了实现自动启动，用户必须事先使用编辑程序EDLIN或复制命令COPY建立一个名为AUTOEXEC·BAT的批命令文件，把需要自动启动的程序写在这个文件中。注意，AUTOEXEC·BAT必须建立在根目录中。

例如，如果希望系统初启后，立即执行 BASIC 程序 SAMPLES·BAS，则可用 COPY 命令建立下列文件：

```
A>COPY CON AUTOEXEC·BAT
DATE
TIME
BASIC SAMPLES
^Z
        1 File(s) copied
A>
```

其中 COPY 命令用于把键盘输入的数据送到文件 AUTOEXEC·BAT 中（详见第二节）。文件中的第 1 行和第 2 行是两条内部命令，分别用来提问日期和时间。因为在自动启动方式下，系统不再提问日期和时间，故在建立自动启动批命令文件时，最好一开始就使用 DATE 和 TIME 命令，使 MS-DOS 能够在更改文件时，写上新的日期和时间。这样，在下次启动后，系统将提问日期和时间，随后自动运行程序 SAMPLES·BAS。

（3）使用硬盘启动 MS-DOS

如果机器配有硬盘并且希望从硬盘上启动 MS-DOS，那么在进行系统启动时不要在A驱动器中插入盘片，机器将会从硬盘上引导MS-DOS（当然，硬盘必须事先初始化）。从硬盘上启动系统比从软盘上启动系统速度快，而且也较方便。如果你的机器配有硬盘，建议你使用硬盘启动。

概括地说，MS-DOS 的启动过程如图 2—2 所示。

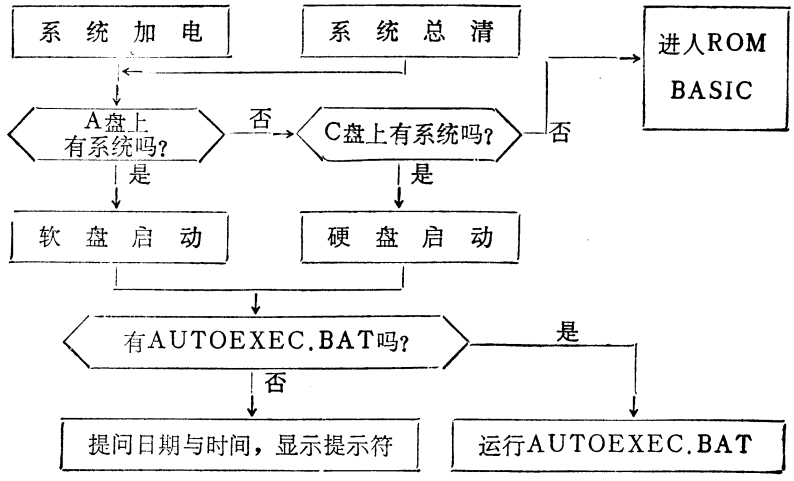


图 2—2 MS-DOS 的启动过程

3．系统盘复制与硬盘的使用

（1）系统盘的复制

在初次使用MS-DOS时，首先应复制一份备份盘，以防止原盘被破坏。用户可以使用全盘复制命令DISKCOPY复制一张系统盘，下面是操作过程：

```
A>DISKCOPY A: B: <CR>
Insert source diskette in drive A:
Insert target diskette in drive B:
Strike any key when ready <CR>
Copying 9 sectors per track, 2 side(s)
Formatting while copying
Copy complete
Copy another (Y/N)? N
```

注意，应将源盘插入A驱动器，而B驱动器则插入一张空盘（可以是未初始化的新盘）。对于复制好的盘，还可以使用全盘比较命令DISKCOMP与源盘进行比较，检查一下复制得是否正确，其过程如下：

```
A>DISKCOMP A: B: <CR>
Insert first diskette in drive A:
Insert second diskette in drive B:
Strike any key when ready <CR>
Comparing 9 sectors per track, 2 side(s)
Diskettes compare ok
Compare more diskettes (Y/N)? N
```

比较无误后，系统盘的复制工作就完成了。这时，将原始盘保存起来，平时就使用复制的盘工作。

这种方法也可用来复制用户的数据盘。但应注意DISKCOPY命令仅用于复制完全相同的盘片，对于个别文件的复制需要用COPY命令进行（参见第二节）。

（2）盘片初始化

MS-DOS一般都使用$5^1/_4$时软盘记录数据，新盘在使用前必须初始化，下面是初始化盘片的一个例子：

```
A>FORMAT B:/S/V <CR>
Insert new diskette for drive B:
and strike any key when ready <CR>
Formatting···Format complete
System transferred
Volume label (11 characters, ENTER for none)? MyDataDisk <CR>
    362496 bytes total disk space
     40960 bytes used by system
    321536 bytes available on disk
```

```
Format another (Y/N)? N
```

注意 新盘必须插在B驱动器中。命令中的/S开关表示需要在初始化盘片后，写上操作系统，如果初始化的盘为数据盘，则可将/S开关省去。/V开关表示要写“卷名”。

随后，用CHKDSK命令检查磁盘的状态以及内存的状态：

```
A>CHKDSK B: <CR>
Volume MYDATADISK created May 22, 1984 1:05p
   362496 bytes total disk space
    22528 bytes in 3 hidden files
    18432 bytes in 1 user files
   321536 bytes available on disk
   131072 bytes total memory
   106496 bytes free
```

在显示出的信息中，第2行表示盘空间总共360K（362496 bytes）；第3和第4行表示盘上已存在的三个隐含文件和一个普通文件的大小；第5行表示盘上剩余空间为320K；最后两行给出内存总空间和可用空间的大小。

（3）硬盘的使用

如果系统配有硬盘，而且准备在MS-DOS下使用硬盘，则必须按以下步骤进行。

首先，使用实用程序FDISK在硬盘上为DOS分配一个区域，打入：

```
A>FDISK <CR>
```

屏幕上即出现信息：

```
IBM Personal Computer
Fixed Disk Setup Program Version 1.00
(C) Copyright IBM Corp. 1983
FDISK Options
Current Fixed Disk Drive: 1
Choose one of the following:
      1. Create DOS Partition
      2. Change Active Partition
      3. Delete DOS Partition
      4. Display Partition Data
      5. Select Next Fixed Disk Drive
Enter choice: [1]
```

打入一个<CR>键，表示在硬盘上建立DOS区（IBM PC允许几个操作系统分区使用硬盘）。接着屏幕又出现信息：

```
IBM Personal Computer
Fixed Disk Setup Program Version 1.00
(C) Copyright IBM Corp. 1983
Create DOS Partition
Current Fixed Disk Drive: 1
Do you wish to use the entire fixed disk
for DOS (Y/N)…? (Y)
```

打入<CR>键，表示整个硬盘全部分配给 DOS 使用，然后显示出：

```
Insert DOS diskette in drive A:
Press any key when ready…
```

打入任意键后，硬盘就分配好了。

然后，使用 FORMAT 命令把 DOS 复制到硬盘，操作过程如下：

```
A>FORMAT C: /S/V <CR>
Press any key to begin formatting drive C: <CR>
Formatting…
Format complete
System transferred
Volume label (11 characters, ENTER for none)?
```

输入一个不长于 11 个字符的卷名，例如 MyFixedDisk。这样，硬盘就可以使用了。

4. 常用的控制键

MS-DOS 为用户提供了一组控制键，使用它们可以对系统的运行进行一定程度的干预。

控制键多数由几个键同时动作组合而成。例如，Ctrl-C 表示按住 <Ctrl> 键的同时按下 C 键。<Ctrl> 键一般简记为 ^，例如：Ctrl-C 简记为 ^C，Ctrl-Z 简记为 ^Z 等。

在 MS-DOS 下，控制键的含义如表 2-1 所示。

表 2-1　MS-DOS 的控制键

控　制　键	功　　能
Ctrl-Alt-Del	系统复位
^C或Ctrl-Break	终止当前操作
^P或Ctrl-PrtSc	把标准输出同时送到打印机和屏幕
^S或Ctrl-NumLock	暂停标准输出设备的输出
Shift-PrtSc	在打印机上产生屏幕的硬拷贝
↵或<Return>	结束命令行或结束逻辑行
^H或<BS>	退格并删除一个字符
^J或Ctrl-CR	结束物理行但不结束逻辑行

表中的控制键多数都列了两个键，在MS-DOS的命令行中一般说来是等效的，用户可任选一个使用。为简明起见，以后将主要使用较短的形式，如 ^C、^P、^S和‹BS›。另外，回车键‹Return›将记为‹CR›，并在不混淆的情况将它省去。

5．键盘命令格式及命令行编辑

（1）键盘命令格式

键盘命令是操作员从键盘上输入的、要求MS-DOS完成一定功能的会话命令。由于每条命令都是以‹CR›结尾的逻辑行，故又称为命令行。命令行的格式是：

〔‹盘符›〕‹命令›{ /‹开关› }{ ‹文件名› }‹CR›

其中盘符为A：、B：、C：或D：，当使用内部命令或使用的命令在当前盘上时，可以省去。开关和文件名可有可无，也可为多个，视具体命令而定。命令行格式中方括号内是任选项，花括号内也是任选项，但可以不止一项。下列都是命令行：

A>B: CHKDSK
A>COPY CON: AUTOEXEC.BAT
A>FORMAT B:/S/V

在MS-DOS中，命令分成内部命令、外部命令和批命令三类。内部命令常驻内存，使用时不占用分配给用户的内存空间；外部命令驻留盘区，使用时才调入内存，用完后退出并归还内存，MS-DOS把具有类型名.COM和.EXE的文件都视为外部命令；批命令则是由一组内部命令或外部命令组成的复合命令，批命令文件的类型为.BAT。

下文中如不特别说明，则所说的命令就是指外部命令。

（2）命令行编辑

MS-DOS具有较强的命令行编辑功能，允许操作员修改或重复上一命令行。键盘命令的输入过程是：首先操作员把一行命令从键盘上输入，打入‹CR›键后，MS-DOS把命令行同时送入命令处理程序和命令暂存器（template）；下次如果要执行类似的命令或修改错误的命令，则可使用编辑功能键来修改，以节省输入的时间。输入过程如图2—3所示。

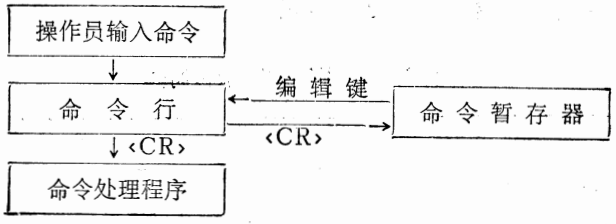


图2—3 键盘命令的输入过程

命令行的编辑主要是通过键盘左侧的一组功能键‹F1›—‹F10›来进行的，这些功能键在运行不同的软件时有不同的定义，运行MS-DOS时，它们的作用就是进行命令行编辑，所以也称为“编辑键”。常用的编辑键有下列几类：

① 复制类：复制字符键 <F1>；复制到指定字符键 <F2>；复制到行末键 <F3>。

② 插入类：插入若干字符键 <Ins>。

③ 删除类：删除字符键 <Del>；删除到指定字符键 <F4>。

例如，假如在建立自动启动批命令文件时，命令打错成：

```
A>COPYT AUTOEXEC.BAT
```

为了改正这个命令行，可以打入：

```
A><F2>T<Del><Ins> CON<F3>
```

其中，<F2> T 表示复制出“COPY”，<Del> 表示删去“T”，<Ins> CON 表示插入四个字符“ CON”，<F3> 表示复制余下的部分。改正后的命令行为：

```
A> COPY CON AUTOEXEC.BAT
```

批命令文件输入完毕后，如希望显示一下这个文件，则可以打入：

```
A>TY <F1> E <F1> <F4> A <F3>
```

其中 <F1> 表示复制一个字符，<F4> A 表示删除到“A”。结果如下：

```
A> TYPE AUTOEXEC.BAT
```

如要再执行一遍这条命令，仅打入 <F3>和<CR>即可：

```
A> TYPE AUTOEXEC.BAT
```

上述编辑键除用于编辑命令行外，还可用于行编辑程序 EDLIN。关于EDLIN 的使用，在第四节中再作介绍。

MS-DOS 操作系统为用户提供了几种不同类型的操作命令：选盘命令、控制键、内部命令、外部命令以及批处理命令。现把本节中介绍的命令归纳如下：

（1）选盘命令

<盘符> <CR>　　选择当前盘

其中盘符为 A：、B：、C：或 D：，分别表示 A 盘、B 盘、C 盘和 D 盘。

（2）控制键

① Ctrl-Alt-Del　　系统复位

② ^C　　结束当前操作

③ ^P　　联机打印

④ ^S　　暂停输出

⑤ Shift-PrtSc　　硬拷贝屏幕

⑥ <CR>　　回车换行

（3）内部命令

① COPY CON 〈文件名〉　建立一个ASCII码文件
② PROMPT〔〈字符串〉〕　修改系统提示符
③ DATE　置日期
④ TIME　置时间

（4）外部命令

① DISKCOPY 〔〈盘符〉〕〔〈盘符〉〕　全盘复制
② DISKCOMP 〔〈盘符〉〕〔〈盘符〉〕　全盘比较
③ FORMAT 〔〈盘符〉〕〔/V〕〔/S〕　初始化磁盘
④ CHKDSK 〔〈盘符〉〕　检查磁盘状态
⑤ FDISK　硬盘分区

（5）批处理

AUTOEXEC.BAT　自动启动批处理文件

第二节　文件及其操作命令

1. 文件与文件名

文件是具有名字的一组相关信息的集合。MS-DOS下的所有程序和数据都是以文件的形式存贮在磁盘上的。

为了区别不同的文件，以便文件的执行、修改和检索，文件必须有一个标记，我们把这个标记称为文件引用名，简称引用名。引用名由盘符、文件名和类型名三部分组成，其中盘符和文件类型名可缺省，盘符缺省时表示文件所在的盘为当前盘。引用名的格式如下：

〔〈盘符〉〕〈文件名〉〔.〈类型名〉〕

其中盘符为A：、B：、C：或D：，表示文件所在的盘分别为A盘、B盘、C盘或D盘。文件名和类型名分别由不多于8个和3个ASCII字符组成。这些字符可以是下列字符：

A··z, 0··9, $, #, &, @, !, %, (,), —, {, }, ”等。

用户命名文件时，最好选用与文件内容或性质相关的文件名。例如，用ADDRLIST.BAS表示一个打印通讯录地址一览表的BASIC程序的文件名就较好。

文件引用名又分成单义引用名和多义引用名两类。单义引用名仅和一个文件对应，而多义引用名则通过使用替代符模糊地对应着多个文件。

MS-DOS中使用的替代符有两个：“?”代替所在处的任一字符；“*”则代替从所在位置到下一间隔符（·或空格）之间的一串字符。例如，ADDRLST?·DAT表示文件ADDRLST1.DAT，ADDRLST2.DAT，…。*·BAS则表示所有类型名为·BAS的文件。

对于类型名，MS-DOS有一定的约定。下面是若干常用的文件类型名及其含义：

.COM　可执行的二进制代码文件
.EXE　可执行的浮动代码文件
.BAT　可执行的批处理文件
.SYS　系统文件

.BAK　编辑程序的后备文件
.OBJ　汇编语言或高级语言的目标码文件
.LIB　库文件
.MAP　目标程序模块全局量列表文件
.ASC　ASCII 码文件
.LST　源程序列表文件
.PRG　dBASE Ⅱ程序文件

MS-DOS 除磁盘文件外，还把一些常用的标准外部设备也看作文件（称为“设备文件”），以便于和磁盘文件统一进行操作和处理。这些设备文件可使用在数据传输类的键盘命令中。设备文件的引用名为：

〈设备名〉〔〈序号〉〕〔:〕

其中冒号“:”可有可无。常用的设备名及其含义如表 2-2 所示。

表 2-2　常用的设备文件

设备文件名	可进行的操作	对应的物理设备
CON:	输入输出	控制台键盘/CRT显示器
PRN: 或LPT1:	输出	并行打印机
AUX: 或COM1:	输入输出	串行输入输出设备(RS232C)
NUL:	输入输出	虚设备(不产生输入输出)

设备文件使用起来十分方便，例如第二节的建立自动启动批文件的例子中，打入

A>COPY CON: AUTOEXEC.BAT

这条命令后，操作员紧接着在键盘上打入的内容就会全部从控制台“拷贝”到磁盘文件 AUTOEXEC.BAT 中去，这比使用行编辑程序 EDLIN 来建立磁盘文件简便得多。

2．目录和路径名

（1）目录结构

MS-DOS 的文件系统采用树型目录结构，树中的每个结点都有一个名字以供访问。树的结点分为三类：根结点表示根目录；树枝结点表示子目录；而树叶则表示普通文件，如图 2—4 所示。

根目录又称为系统目录，每张盘上只有一个根目录，它是在盘片初始化时自动建立的。单面盘的根目录最多允许 64 个文件或子目录，而双面盘则允许多达112个。

子目录是包含在根目录或其它子目录中的目录。子目录是操作员使用“建立子目录命令”建立的，MS-DOS允许在同一目录中建立多个不同名的子目录。由于MS-DOS是把子目录作为文件（即目录文件）来处理的，故子目录中文件的数目仅受盘空间的限制。如果使用硬盘则可以允许使用数千个文件。此外，如果甲目录包含了乙目录，我们就说：甲目录是乙目录的上级目录；而乙目录则是甲目录的下级目录。

为了标识子目录，子目录必须具有目录名。目录名的格式与文件名相类似，也是由八个

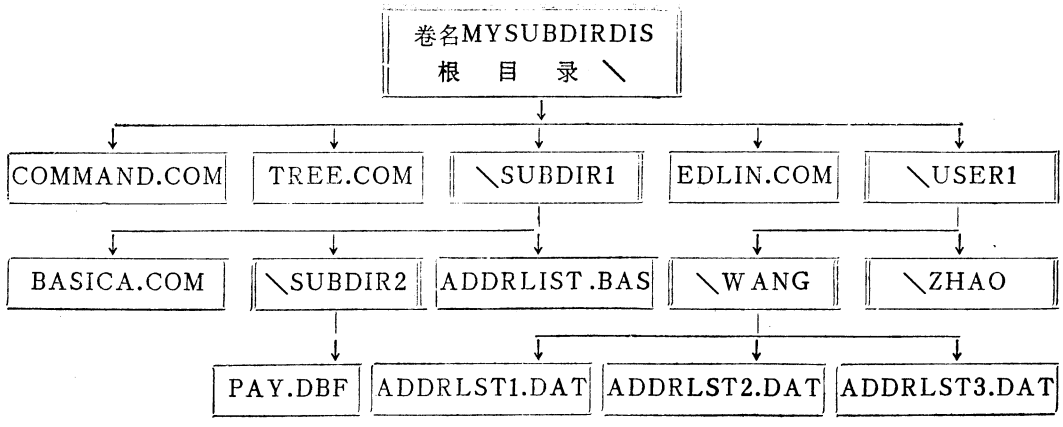


图2—4 MS-DOS的目录结构

ASCII字符组成，例如：

SUBDIR1
SUBDIR2

都是合法的目录名。

每个子目录中必定有两个特殊的目录项，可以看成已存在的两个特殊文件。第1个的文件名为“·”，表示自己是一个子目录；第2个文件名为“··”，表示自己归属于那一个上级目录。这两个文件在子目录建立时自动形成，可以用列目录命令DIR把它们列出来：

```
A>DIR \SUBDIR1\SUBDIR2
  Volume in drive A is MYSUBDIRDIS
  Directory of A:\SUBDIR1\SUBDIR2
.            <DIR>      5—28—84   10:46a
..           <DIR>      5—28—84   10:46a
             2 File(s)     284672 bytes free
```

（2） 路径和路径名

对于树状结构的文件系统，为了对文件进行操作，例如建立或寻找一个文件，DOS必须知道该文件所在的盘符、文件名以及包含该文件的目录名。如果该文件就在当前目录中，则仅指出文件名即可，DOS将自动地在当前目录中寻找该文件。如果文件不在当前目录中，则还必须指出从当前目录(或根目录)到文件所在目录的路径(分别叫做相对路径和绝对路径)。

路径用反斜杠相互隔开的一组目录名来表示。路径若以“\”开始，则表示绝对路径，否则就表示相对路径。例如：

\SUBDIR1
\SUBDIR1\SUBDIR2
\SUBDIR1\SUBDIR2\PAY.DBF

都是绝对路径，而

PAY.DBF

..\SUBDIR2\PAY.DBF

却是相对路径。在系统初始启动之后，当前目录即为根目录，此后，通过使用“改变当前目录命令（CD）”可以对它进行修改。上面最后一例中的“..”，表示当前目录的上级目录。

为了在键盘命令中指出要求DOS进行处理的某个文件或目录，输入命令行中通常应该打入：

〔<盘符>〕〔<路径>〕<文件名>　　{指定某个文件}

〔<盘符>〕<路径>　　{指定某个目录}

为了表达方便起见，下文将把它们分别称为“文件路径名”（或简称“路径名”）和“目录路径名”。

3．目录操作命令

（1） 显示目录命令DIR

这条内部命令用来列出指定盘、指定目录或指定文件的目录。它的命令格式有以下三种：

DIR 〔<盘符>〕〔/P〕〔/W〕

DIR 〔<目录路径名>〕〔/P〕〔/W〕

DIR 〔<文件路径名>〕〔/P〕〔/W〕

其中 /P 表示逐屏显示；/W 开关表示多列显示文件名。例如：

```
A>DIR
 Volume in drive A is MYSUBDIRDIS
 Directory of A:\
COMMAND  COM     17664    3—08—83  12:00p
TREE     COM      1513    3—08—83  12:00p
EDLIN    COM      4608    3—08—83  12:00p
SUBDIR1          〈DIR〉    5—28—84  10:46a
USER1            〈DIR〉    5—28—84  10:47a
        5 File(s)      311296 bytes free
```

如不选 /W 开关，DIR 命令将依次列出文件名、类型名、以字节为单位的文件长度以及文件最后修改的日期和时间。目录中带有〈DIR〉标记的文件为目录文件，DIR命令不给出它的长度。DIR命令列目录时，在第 1 行列出所列盘的卷名；在第 2 行列出所列目录的路径名；在最末一行给出所列目录的文件数及盘上剩余空间的字节数。

应当注意：在MS-DOS的DIR命令中，对于多义文件的处理有些特殊的规定，例如：

DIR或DIR *　　等效于　DIR　*.*

DIR FILENAME　等效于　DIR　FILENAME.*

DIR　.EXT　　等效于　DIR　*.EXT

DIR　.　　等效于　DIR　*.

如果在使用DIR命令时选用 /W 开关，则每行列出五个文件，不过仅列出文件名和类型名。例如：

```
A>DIR B:*.COM/W
 Volume in drive B has no label
 Directory of  B:\
COMMAND  COM  FORMAT   COM  CHKDSK  COM  …  DISKCOPY  COM
DISKCOMP COM  COMP     COM  EDLIN   COM  …  FDISK    COM
BACKUP   COM  RESTORE  COM  PRINT   COM  …  ASSIGN   COM
TREE     COM  GRAPHICS COM  MORE    COM  …  BASICA   COM
KEYBUK   COM  KEYBFR   COM  KEYBSP  COM  …  KEYBGR   COM
WTDATIM  COM  GRAFTABL COM  DEBUG   COM
          29 File(s)        40960 bytes free
```

（2） 文件换名命令 REN

该命令用来更改文件名，也是一条内部命令，格式为：

REN <旧文件路径名> <新文件名>

或 RENAME <旧文件路径名> <新文件名>

其中新文件名由文件名加类型名组成。文件名可为多义文件名。例如：

```
A>REN *.LST *.PRN
A>REN \USER1\ADCBE ?B?D?
```

其中第 1 行命令把当前目录中所有类型名为.LST的文件换名为类型名是 .PRN的文件；第 2 行则把子目录\USER1中的ADCBE文件换名成ABCDE。

（3） 列卷名命令VOL

这条内部命令用以显示出指定盘的卷名。卷名是用户使用FORMAT命令初始化盘片时，选用/V开关写入的盘标识信息，在显示目录或显示目录结构时显示出来。命令的格式为：

VOL 〔<盘符>〕

（4） 建立子目录命令MD

要建立图 2—4 所示的树形文件系统，必然涉及到建立目录文件（即子目录）。建立目录文件使用下列内部命令来完成：

MD <目录路径名>

或 MKDIR <目录路径名>

例如：

```
A>MD SUBDIR1
A>MD \SUBDIR1\SUBDIR2
A>MD \USER1
```

其中第 1 行命令在当前目录中建一个名为SUBDIR1的子目录；第 2 行在子目录 SUBDIR1 中

建立子目录SUBDIR2；第3行则表示在根目录中建立子目录USER1。

（5） 显示或改变当前目录命令CD

这条内部命令用以显示或改变当前目录。命令格式为：

CD〔<目录路径名>〕

或 CHDIR〔<目录路径名>〕

其中目录路径名缺省时，表示显示当前目录的路径名。例如：

```
A> CD USER1
A> CD ..\SUBDIR1\SUBDIR2
A> CD ..
A> CD
A：\SUBDIR1
```

其中第1行表示把当前目录改为USER1；第2行表示把当前目录改为SUBDIR2；第3行表示把当前目录改成其上级目录；第4行显示当前目录。

（6） 删除子目录命令RD

这条内部命令仅用于删除目录文件，不能删除普通文件。RD命令一次可删除一个空目录（即只含有特殊文件“.”和“..”的目录），但不允许删除根目录和当前目录。命令格式为：

RD〈目录路径名〉

或 RMDIR〈目录路径名〉

例如：

```
A> RD \USER1\LI
```

表示删除子目录USER1下的子目录LI。

（7） 显示目录结构命令TREE

TREE是一条外部命令，用来显示指定盘的所有路径及其层次关系。命令格式如下：

TREE〔<盘符>〕〔/F〕

其中/F开关表示在每个子目录下列出下属的所有文件的文件名，如不选/F开关，则仅列出所属的子目录名。如果某目录下没有目录或文件，则显示出“None”。下面给出与图2—4对应的所有路径：

```
A> TREE
DIRECTORY PATH LISTING FOR VOLUME MYSUBDIRDIS
Path: \SUBDIR1
Sub-directories: SUBDIR2
Path: \SUBDIR1\SUBDIR2
Sub-directories: None
Path: \USER1
Sub-directories: WANG
                 ZHAO
```

Path: \USER1\WANG

Sub—directories: None

Path: \USER1\ZHAO

Sub—directories: None

（8） 设定外部命令的搜索路径 PATH

用户在自己的目录中工作时，时常会使用别的目录中的外部命令，MS-DOS 提供的内部命令PATH则可满足这一要求。PATH 用来指出，假如在当前目录中找不到某一外部命令文件时应进一步去查找的目录。命令格式是：

PATH 〈目录路径名〉〔;〈目录路径名〉…〕

如果命令不带参数，将显示现行的命令路径；若仅有“;”参数，则表示把命令路径置为空（即仅在当前目录中寻找外部命令文件），如果参数中有多个路径名，MS-DOS 的命令处理程序将逐个查找它们，直至所输入的外部命令找到为止。

例如，可把所有的命令文件全部放在子目录\BIN中，在自动启动批文件中插入命令：

PATH \BIN

则在重新启动后，在任何目录下，都可直接使用\BIN目录中的外部命令。

4. 文件操作命令

文件操作是指对文件本身的操作，即建立、复制、删除和输出一个文件。而前面介绍的目录操作命令，仅能处理文件的目录，不改变文件的内容。

除了文件操作命令能够加工处理文件外，MS-DOS 还允许使用实用程序（如编辑程序）来处理文件。关于实用程序将在第四节中介绍。下面介绍一组文件操作命令。

（1） 文件复制命令COPY

这条内部命令处理文件与文件、文件与设备和设备与设备之间的信息交换，也可以把几个文件联结成一个文件。其格式有两种：

COPY 〈路径名〉〔〈路径名〉〕〔/A〕〔/B〕〔/V〕

COPY 〈路径名〉〔+〈路径名〉…〕 〔〈路径名〉〕〔/A〕〔/B〕〔/V〕

第 1 格式用来复制文件。如果其中第 2 参数缺省，则表示以原名记在当前盘上，但这时第 1 参数不能为当前盘的文件。

第 2 格式用来联结文件。如第 2 个参数缺省，表示联结后的文件记在第 1 参数中的第 1 个文件上。

参数/A表示ASCII文件；/B表示二进制文件；/V表示复制过程中要进行校验。

例如：

```
A> COPY B:EDLIN.COM  A:
        1 File(s) copied
A> COPY B:BASIC.COM  B:SUBDIR1
        1 File(s) copied
```

其中第 1 条命令表示把 B 盘上的 EDLIN·COM 复制到 A 盘上去。第 2 条命令表示将 B 盘

当前目录中的BASIC·COM复制到B盘的子目录SUBDIR1中。

COPY命令还可以用来联结几个文件，例如：

A>COPY \USER1\ALL·PRN+ * ·PRN

表示把子目录\USER1下的所有·PRN文件联结成ALL·PRN（ALL·PRN原先必须存在）。

COPY还可以用来完成设备间的数据交换。例如：

A> COPY CON MYFILE

A> COPY MYFILE PRN

A> COPY CON PRN

第1行表示把键盘输入的数据送到文件MYFILE中去；第2行表示把文件MYFILE从打印机输出；第3行表示把键盘上输入的数据直接拷贝到打印机上去打印。

（2） 系统复制命令SYS

MS-DOS的两个隐含文件可在盘片初始化时，由选择/S开关来复制。如当时未加/S开关，事后也可用这个外部命令来复制，但这条命令不复制COMMAND·COM这个文件，需要时要用COPY命令复制。SYS命令的格式为：

SYS 〈盘符〉

盘符缺省表示当前盘。例如：

A> SYS B:

就表示把系统复制到B盘。

（3）文件比较命令 COMP

这条命令用于文件间的比较，可用来检查文件复制的正确性。命令格式如下：

COMP 〈路径名〉 〈路径名〉

（4）文件删除命令 DEL

这条内部命令用于删除一个或一组文件，但它不能用于删除子目录。格式是：

DEL 〈路径名〉

或 ERASE 〈路径名〉

例如：

A> DEL * ·BAK

A> DEL \SUBDIR1\SUBDIR2\LIST1 .DAT

其中第1行表示将当前目录中所有的 .BAK 文件删除；第2行则表示将子目录\SUBDIR2下的LIST1 .DAT删去。

（5）文件打印命令 TYPE

这是一条内部命令，用于把ASCII文件按原来的格式输出到屏幕上或打印机上。要输出到打印机上须事先按一个 Ctrl-P 键把打印机接通。这条命令的格式是：

TYPE 〈路径名〉

例如：

```
A> TYPE MYFILE.ASC
```

表示把 MYFILE.ASC 显示出来。

（6）假脱机输出命令 PRINT

这条命令用来把文件送出排队打印，即所谓假脱机打印，从而实现打印输出与 CPU 任务的重叠。该命令还允许用来删除打印队列中已有的文件和使新文件加入到队列中去。命令格式如下：

PRINT 〔< 路径名 > 〔/T〕〔/C〕〔/P〕…〕

其中 /T 表示清除打印队列中的所有文件；/C 表示进入删除方式，删除打印队列中指定的文件；/P 表示打印方式，其后送入的文件将被打印出来。如不指定任何开关，则认为是打印方式。在 PRINT 工作期间，其它打印命令（如 Ctrl-P ）不允许执行。

假脱机缓冲区至多只允许十个文件。如队列中已有十个文件，还要再加入文件的话，屏幕会出现出错信息："PRINT queue is full"（打印队列满）。例如：

```
A> PRINT/T
PRINT queue is empty          {打印队列空}
A>PRINT AUTOEXEC.BAT *.ASM
A>PRINT TEMP1.TST/C TEMP2.TST/P TEMP3.TST
```

其中第 1 条命令将队列全部清除；第 2 条命令把一个 .BAT 文件和所有 .ASM文件加到队列中去；第 3 条命令将队列中的 TEMP1.TST 删除，加入 TEMP2.TST 和 TEMP3.TST 两个文件到队列中去。

（7）写校验命令 VERIFY

这条内部命令类似 COPY 命令的 /V 开关，当置成校验方式后，此后 MS-DOS 对盘进行写操作时均加以正确性检查。但校验方式比非校验方式要花费更多的时间，故一般仅在较重要的数据写盘时才用这种方式。命令格式如下：

VERIFY 〔ON | OFF〕

如参数缺省则表示显示当前校验方式。

（8）硬软盘间的复制命令

除了 COPY 命令可以用于硬软盘间交换数据外，还可以使用 BACKUP 命令把硬盘上的文件复制到软盘上，也可以用 RESTORE 命令把软盘上的文件复制到硬盘上去。这两条命令的格式为：

```
BACKUP  <路径名> <盘符> 〔/S〕〔/M〕〔/A〕
RESTORE  <盘符> <路径名> 〔/S〕〔/P〕
```

其中/S 开关表示复制包括子目录在内的所有文件；/M 开关表示仅复制修改过了的文件；/A 开关表示把文件添加到备份盘中去；/P表示复制时给出提示。例如：

```
A>BACKUP C:*.COM B:
A>RESTORE B: C:\BIN
```

第三节　批处理与输入输出操作

1. 批处理

（1）批处理的基本概念

在使用键盘命令过程中，有时候需要连续使用几条命令，有时候则要多次地重复使用若干条命令，还有的时候是要有选择地使用不同的命令。为了满足这些要求，MS-DOS 提供了一种特殊的文件——.BAT 文件。该文件允许用户组织键盘命令语言程序，一次建立，多次执行。这个 .BAT 文件可用 EDLIN 建立或用 COPY 命令建立。例如，要建立一个盘片初始化的批处理文件，可这样进行：

```
A>COPY CON NEWDISK.BAT
REM This is a batch file to init a new disk
REM It is named NEWDISK.BAT
PAUSE Insert new disk in drive B:
FORMAT B: /S/V
CHKDSK B:
^Z
        1 File(s) copied
```

上例中的 REM 子命令是注释命令，用来提供说明信息；PAUSE子命令是暂停命令，暂停操作并给出提示信息：

```
Insert new disk in drive B:
Strike a key when ready……
```

用户可以在 B 驱动器中插入一张新盘，并任意敲一个键。随后批处理就会继续进行下去，执行“FORMAT”和“CHKDSK”命令。

当需要执行这个批命令时，打入

```
A>NEWDISK
```

即可。因为 MS-DOS 是把批命令文件 .BAT 作为普通的命令一样处理的。

MS-DOS 的批文件中允许带形式参数，它们是 %0—%9。其中 %0 是用来指出批文件自己。如果十个参数不够，则可用子命令 SHIFT 来扩充，SHIFT 是通过改变形参与实参对应关系来实现参数扩充的。请参见下例：

```
A>COPY CON SHIFTSTR.BAT
CLS
REM This is a batch file to shift the strings
ECHO OFF
ECHO
ECHO %0 %1 %2
```

```
SHIFT
ECHO %0 %1 %2
SHIFT
ECHO ON
ECHO %0 %1 %2
^Z
        1 File(s) copied
```

批文件中的第1句CLS是清除屏幕命令。随后的ECHO子命令的格式如下：

ECHO 〔ON | OFF | 〈信息〉〕

它有三种方式，本例中第5行用来显示一些信息；第2行的ECHO OFF用来阻止命令回送信息在屏幕上显示。本例的执行结果为：

```
A>SHIFTSTR STRING11 STRING22 STRING33 STRING44
A>CLS
A>REM This is a batch file to shift the strings
A>ECHO OFF
ECHO is off
SHIFTSTR STRING11 STRING22
STRING11 STRING22 STRING33
A>ECHO STRING22 STRING33 STRING44
STRING22 STRING33 STRING44
```

批文件的最后一行可以是一个批命令，也可以是自己的名字，这样就可以使批命令连续不断地执行下去。

例如：

```
A> COPY CON LOOPVER.BAT
CLS
REM ----- Loop Batch -----
VER
%0
^Z
        1 File(s) copied
```

这条批命令使用后，会连续不断地重复下去，若想终止它的执行，输入Ctrl-C。在当前一条命令执行完毕以后，显示屏幕上会出现下列信息：

```
Abort batch job ( Y/N ) ?
```

若输入“Y”则批命令终止。该例中的VER子命令表示显示操作系统版本号。

MS-DOS对批命令文件AUTOEXEC.BAT是特殊处理的，即系统在初始启动之后会

自动执行该文件中的命令组，详见第一节。

（2）批处理的控制子命令

批处理文件中允许出现所有的内部命令和外部命令，以及前面所述的子命令REM, PAUSE, ECHO, SHIFT。除此之外，MS-DOS还允许在批处理文件中使用一组批处理顺序控制子命令。下面逐一介绍这组子命令。

① FOR子命令

FOR子命令用于实现某条命令的循环执行，它的格式如下：

FOR %%〈字符〉 IN 〈循环表〉 DO 〈命令〉

这里的FOR, IN, DO一定要大写。格式中的%%〈字符〉为循环变量，它的取值为循环表中给出的每一个文件名；循环表可以是一系列单义文件名，也可以是一个多义文件名。FOR功能是对循环表中每一个文件执行DO后面的命令。

例如：

```
A> COPY CON FORCBAS.BAT
FOR %%A IN ( BASCOM.COM LINK.EXE ) DO ECHO %%A OK!
REM
FOR %%F IN ( *.ASC ) DO BASCOM %%F
FOR %%F IN ( *.ASC ) DO LINK %%F
^Z
        1 File(s) copied
```

上例表示：在当前盘上查找，当发现BASCOM.COM和LINK.EXE时，则显示出其文件名和“OK!”，并把所有.ASC文件编译连接成.EXE文件。

② GOTO子命令

这条子命令用于改变批命令中子命令的执行顺序，格式为：

GOTO 〈标号名〉

标号由一个“:”号跟随一个标号名组成。例如：

:LOOP1, :Wang

都是标号。使用GOTO子命令的例子如下：

```
A> COPY CON LOOPVER.BAT
:LABEL
CLS
REM ---- Loop Batch
GOTO LABEL
^Z
        1 File(s) copied
```

本例产生一个死循环，直至按下Ctrl-C为止。

③ IF子命令

条件子命令用来选择执行某条命令。其格式为：

IF 〔NOT〕〈条件〉〈命令〉

其中〈条件〉有三种形式：

〈字符串 1〉==〈字符串 2〉	左右字符串相同
EXIST 〈文件名〉	指定的文件存在
ERRORLEVEL 〈出错号〉	发生指定出错号的错误

如果选择 NOT，则条件为假时执行随后的语句，否则为真才执行。

例如：

```
IF NOT EXIST %1 ECHO %1 not found!
IF %1==MYPROG.ASC TYPE %1
```

本例首先检查盘上是否有名为第 1实参的文件，若无打出文件名和“not found! ”，再检查批命令的第 1 个实参是不是 MYPROG.ASC，若是则显示出 MYPROG.ASC。

（3）几个批处理的例子

① 复制若干个 BASIC 盘

```
:LOOP
REM This batch file will copy BASIC disks
FORMAT B: /S/V
ECHO OFF
COPY A:BASICA.COM B:
COPY A:BASCOM?.* B:
COPY A:BASRUN?.* B:
COPY A:LINK.EXE B:
PAUSE Copy another?
GOTO LOOP
```

一张盘复制完后要求输入一个键。若输入 Ctrl-C 将终止复制操作，否则再复制一张盘。

② 使用编译 BASIC 的操作过程

```
:LOOP
ECHO OFF
CLS
REM This batch file will debug BASIC programs
IF EXIST %1.ASC BASICA %1.ASC
IF NOT EXIST %1.ASC BASICA
IF NOT EXIST %1.ASC GOTO BASSTOP
BASCOM %1.ASC;
LINK %1;
%1
ECHO ON
```

```
PAUSE Debug it again!
GOTO LOOP
:BASSTOP
ECHO BASIC Debug Stop!
```

该例首先调入高级 BASIC（如果要调试的程序已存在，则同时调入），用户就可以编制和调试程序。调试完毕记盘后，系统调用编译 BASIC 对它进行编译，生成目标程序。若发生错误则停止。

③ 数据库程序文件的运行

首先检查 DBASE·COM 是否在当前盘上，不在则显示出

```
=====Not found the DBASE·COM=====
```

并结束操作。否则，依次执行用户的 .PRG 程序。执行结束后，显示

```
# # # # END # # # #
```

并停止执行。这个批文件的内容如下：

```
ECHO OFF
CLS
REM This batch file will do all of
REM the dBASE II .PRG files
IF NOT EXIST DBASE.COM GOTO STOP
FOR %%F IN (*.PRG) DO DBASE %%F
ECHO ON
ECHO # # # # END # # # #
GOTO DBASEEND
:STOP
ECHO ON
ECHO ==== Not found the DBASE.COM ====
:DBASEEND
```

2. 输入输出重定向

MS-DOS 的输入输出，通常是在标准设备（键盘和显示器）上进行的，但也可以定义为对文件进行。该文件可以是磁盘文件或是设备文件，这就是所谓的输入输出重定向。

（1）输出改向

输出改向可以在命令行中打入一个大于号“>”加上输出文件引用名或设备名即可。如果输出文件已存在，该命令将重写这个文件。

使用连续的两个大于号“>>”，可用来添加一个文件。如文件已存在，则输出将置于原有文件之后，而不会破坏原文件。如文件不存在，这条命令类似于上条命令。

例如：

```
A> DIR *.COM>MYDIRS.DIR
A> DIR *.EXE>>MYDIRS.DIR
```

这两条重定向的列目录命令，可产生一个 MYDIRS·DIR 文件，它将包含第 1 条命令输出的四个 ·COM 文件目录及第 2 条命令加入的两个 ·EXE 文件目录。其内容为：

```
A> TYPE MYDIRS.DIR
   Volume in drive A is MYDATADISK
   Directory of A: \
COMMAND     COM     17664    3—08—83    12 : 00P
BASCOM      COM     41472    7—06—83     1 : 42P
EDLIN       COM      4608    3—08—83    12 : 00P
BASICA      COM     25984    3—08—83    12 : 00P
          4 File(s)     81920 bytes free
   Volume in drive A is MYDATADISK
   Directory of A:\
LINK        EXE     39936    3—08—83    12 : 00P
BASRUNG     EXE     22272    5—18—83
          2 File(s)     81920 bytes free
```

（2）输入改向

类似输出改向，在命令行中使用小于号“<”及输入文件名，可进行输入改向。例如有两个文件 MAX.BAS 和 INFILE，其内容分别为：

```
A> TYPE MAX.BAS
10 REM Print the max. number of input
20 MAX=0
30 INPUT NUM
40 IF NUM=0 THEN 80
60 IF NUM>MAX THEN MAX=NUM
70 GOTO 30
80 PRINT "The max. number is" ; MAX
90 END
A> TYPE INFILE
19
25
67
8
23
0
```

使用下列命令：

```
A> BASICA MAX < INFILE > OUTFILE
```

在运行BASIC 程序 MAX 时，则从 INFILE 文件中输入六个数据，并将输出结果送到文件 OUTFILE 中。OUTFILE 的内容如下：

```
A> TYPE OUTFILE
? 19
? 25
? 67
? 8
? 23
? 0
The max. number is 67
```

（3）重定向到设备

重定向为使用系统中的其它设备提供了一种方便的途径，在调用程序执行的命令行中，只需给出重定向命令，就可以把程序的输入输出转换到其它的设备上去。

例如：

```
A > DIR > PRN:
A > PROGRAM < INFILE > PRN:
```

都将命令或程序的结果送到打印机去。

3. 管道操作与过滤处理

（1）管道操作

使用管道操作可以把一个命令或程序的输出送到另一命令或程序的输入中去，并依次类推形成一个连续的处理过程。命令或程序一个接着一个，象自来水管道一样，而输入和输出的数据就好象管道中的水流，故称为管道操作。MS-DOS用垂直线“|”作为管道符号，表示左面程序的输出送到右面程序的输入中去。

例如：

```
A> PROGRAM1 | PROGRAM2
```

表示程序 PROGRAM1 的输出作为程序 PROGRAM2 的输入，它等效于下列三条命令：

```
A> PROGRAM1 > TEMP
A> PROGRAM2 < TEMP
A> DEL TEMP
```

应当注意，MS-DOS 管道操作仅当左边的程序向控制台输出，而右边的程序由控制台输入时才有效。例如，下述命令是不允许的：

```
A>DIR | PRINT
```

因为后一条命令不要求控制台输入，但过滤器命令除外。

管道技术还可用来简化某些实用程序的操作过程。首先将操作命令写成一个文本文件，随后可多次使用来操作某一特定的实用程序。例如：

首先，用COPY命令建立一个操作命令文件如下：

```
A> COPY CON DEBUGOP
D0L100
U100L100
R
D200L100
Q
^Z
```

再打入带管道操作的命令行：

```
A> TYPE DEBUGOP | DEBUG DISKCOPY A : B :
```

则 MS-DOS 调入 DEBUG 并依次执行 DEBUGOP 文件中规定的操作。（关于DEBUG的使用，请见第四节）。本例中的操作依次为显示 00—FF 的内存单元内容，反汇编出 100 到 1FF 的程序，显示各寄存器的内容，显示 200 到 2FF 的内存内容。

（2）过滤处理

MS-DOS 还提供一种叫过滤器的输入输出工具，它们对输出的表格等数据进行过滤处理。过滤器是一组命令，可以从控制台或从用户文件中取得数据，经过一定的变换和加工后，再输出到控制台或文件中。用户的输出数据仿佛是经过“过滤器”筛过一遍一样，如图 2—5 所示。

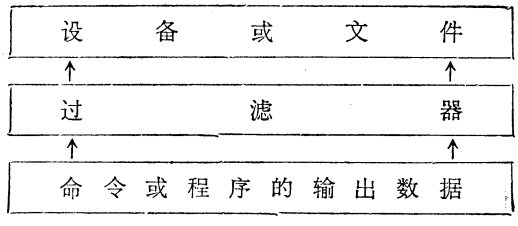


图 2—5 MS-DOS 的过滤处理

过滤处理一般都与 I/O 重定向操作或管道操作结合进行，下面分别介绍 MS-DOS 的三条过滤处理命令。

① FIND 命令

这条外部命令对输入文件逐行检索，选择满足条件的行输出。它的格式为：

FIND〔/V〕〔/C〕〔/N〕“字符串”〔<路径名>〕

其中/V 表示输出不含有“字符串”的行；/C 开关表示仅仅输出包含所给定“字符串”的行的数目；/N 表示输出含有“字符串”的行，并在行首加上这些行在原文件中的相对序号。

如果<路径名>省略，FIND 将从控制台或管道的上一级的输出中取得它的输入数据。

例如，对文件 CURSOR·HLP：

```
ESC[#;#H        Cursor Position
ESC[#A          Cursor Up
ESC[#B          Cursor Down
ESC[#C          Cursor Forward
ESC[#D          Cursor Backward
ESC[#;#1        Horizontal & Vertical Position
ESC[6n          Device Status Report
ESC[#;#R        Cursor Position Report
ESC[s           Save Cursor Position
ESC[u           Restore Cursor Position
ESC[2J          Erase in Display
ESC[k           Erase in Line
```

使用下列 FIND 命令，其命令格式和输出结果为：

```
A>TYPE CURSOR.HLP | FIND/N "Erase"
〔11〕ESC[2J                Erase in Display
〔12〕ESC[k                 Erase in Line
A>FIND/C "Erase" CURSOR.HLP
----------CURSOR.HLP:2
```

其中第 1 条命令选用管道操作方式并选择开关/N；第 2 条命令则用直接方式，并选择/C 开关，仅输出含有“Erase”的行的数目。

② MORE 命令

MORE 命令的使用范围较小，一般仅用于当屏幕输出长于一屏时的暂停处理。它从标准输入设备或管道中读出数据，一次送一屏到屏幕上去，并输出提示信息：

```
--MORE--
```

当输入<CR>键后，就显示出下一屏。

例如：

```
A>TYPE CURSOR·HLP | MORE
A>DIR | MORE
```

都是使用 MORE 的例子，其中第 2 行类似于 DIR/P 命令。

③ SORT 命令

这条命令对输入数据按字典顺序排序后输出，也可用于对表格的某一列进行排序。格式为：

SORT〔/R〕〔/+n〕

开关/R 表示反向排序，即按字典顺序的相反方向排序；/+n 开关表示以输入数据的第 n 列进行排序，用于对表格的某一栏排序。

使用直接方式，SORT 命令可将控制台输入的数据排序。

例如：

```
A>SORT/R
MORE
FILE
SORT
^Z
```

它把控制台输入的三个单词按逆字典顺序排列输出，输出的结果如下：

```
SORT
MORE
FILE
```

过滤处理常常在管道操作方式或重定向方式时使用。例如，下例就是这类方式对目录表中文件长度进行排序的结果（第 14 列是目录表中文件长度栏所在的位置）：

```
A>DIR *.COM | SORT/+14
EDLIN       COM      4608     3—08—83     12:00p
COMMAND     COM     17664     3—08—83     12:00p
BASICA      COM     25984     3—08—83     12:00p
BASCOM      COM     41472     7—06—83      1:42p
  Directory Of  A:\
        4 File(s)     62464 bytes free
  Volume in drive A is MYDATADISK
```

4．其它输入输出命令

MS-DOS 还提供了一些特殊的 I/O 命令，可以用来实现控制台转让、驱动器转让以及设备参数设置。这些命令主要有以下几种：

（1）控制台转让命令 CTTY

MS-DOS 允许用户把系统控制台（通常是键盘和单色显示器）定义成其它设备，以便在其它设备上对系统进行操作。例如，电传打字机、汉字显示终端、图形显示终端等均可作为控制台，假设这些设备连接在 1 号异步通讯接口上，一般过程是：先用 MODE 命令设置好串行口的波特率等参数，再使用 CTTY 命令替换。例如：

```
A> MODE COM1 : 9600, N, 8
A> CTTY COM1
```

（2）驱动器指派命令 ASSIGN

ASSIGN 命令用以把一个驱动器设置成另一个盘符。例如 A 盘上有下列目录：

```
A> DIR A:.DOC
  Volume in drive A is MYDATADISK
  Directory Of A:\
TIMER DOC 245 12—15—83 2:02p
    1 File(s) 61440 bytes free
```

使用驱动器指派命令ASSIGN B=A 后，再列出 B 盘的目录如下：

```
A> DIR B:.DOC
  Volume in drive B is MYDATADISK
  Directory of B:\
TIMER DOC 245 12—15—83 2:02p
    1 File(s) 61440 bytes free
```

可见把 B 盘指向 A 驱动器之后，一切关于 B 盘的操作也都在 A 驱动器上完成。显然，驱动器指派命令给操作带来很大的灵活性。

（3）BREAK 命令

在 MS-DOS操作系统下，输入 Ctrl-C（或 Ctrl-Break）表示终止系统的现行操作，返回 MS-DOS。BREAK 命令用以设置 Ctrl-C（或 Ctrl-Break）检查方式。BREAK OFF 表示仅在控制台、打印机或串行口输入输出操作时，才检查并处理 Ctrl-C。而 BREAK ON 则表示 MS-DOS在执行任何功能时都检查和处理 Ctrl-C 的输入。如果仅输入 BREAK，表示显示当前 Ctrl-C 检查状态。

例如：

```
A> BREAK
BREAK is off
A> BREAK ON
```

其中第 1 条命令表示显示当前 Ctrl-C 检查状态；第 2 条命令打开 Ctrl-C 检查状态。

（4）MODE 命令

这条命令主要用于设置异步通讯器的通讯规程，格式如下：

MODE COMn：〈波特率〉[，〈校验方式〉][，〈数据位个数〉][，〈停止位个数〉]

其中波特率可在 75 至 9600 波特之间选择。

例如：

```
A> MODE COM1: 9600, N, 8, 1
```

（5）GRAPHICS 命令

如果你的系统配有图形显示器和图形打印机，这条命令可用来把图形显示器的内容硬拷贝输出。这条命令的格式为：

GRAPHICS

为了自动地执行一组键盘命令以及提高使用I/O的灵活性，MS-DOS 提供了较强的批处理能力和 I/O 处理功能。现把 MS-DOS 的批处理、重定向操作、管道操作和过滤处理小结如下。

常用的批处理子命令有：

* FOR　　循环子命令
* IF　　条件子命令
* GOTO　　转移子命令
* SHIFT　　改变形参与实参的对应关系
* PAUSE　　暂停批处理

重定向操作和管道操作的调用符为：

* >　　输出重定向
* >>　　添加方式输出重定向
* <　　输入重定向
* |　　管道操作

过滤处理有三条命令：

* FIND　　检索
* SORT　　排序
* MORE　　分页显示

第四节　几个常用的实用程序

所有操作系统都向用户提供一组经常需要用到的、功能相当完善的程序，通常叫做“实用程序”。例如：行编辑程序 EDLIN，动态调试程序 DEBUG，目标模块连接程序 LINK，库管理程序 LIB 和快速比较程序 FC 等。下面对使用频率较高的几个实用程序予以扼要的介绍。

1. 行编辑程序 EDLIN

（1）引　言

在使用高级语言或汇编语言编写程序时，首先必须采用编辑程序来建立和修改源程序文件。所以编辑程序是一种十分重要的实用程序。

EDLIN 是一种行编辑程序，也就是说，编辑以行为单位进行。这里所说的行是指以<CR>结尾的逻辑行，它的最大长度可达 253 个字符。

调用 EDLIN 的格式是：

EDLIN <路径名>

对于新文件，调入 EDLIN 以后，屏幕上显示出：

```
New file
*
```

而对于老文件，EDLIN 把文件装入内存并显示：

```
End of input file
*
```

或者，当内存编辑缓冲区装不下老文件时，仅装满内存的 3/4 空间，然后，显示出一个星号“*”。

这里的星号“*”是EDLIN的提示符。出现“*”以后，表示EDLIN可以接收编辑命令。

（2）基本编辑命令

在介绍编辑命令以前，先说明几个有用的概念。在 EDLIN 中使用的行号分为绝对行号和相对行号两种。绝对行号是行在正文中的顺序号，而相对行号则是指与当前行的相对位置，用加上正负号的数字来表示。这里所说的当前行，是行指针所指向的行（屏幕上行号后面标有“*”号）。并常作为命令执行的缺省行。在编辑命令中，一般以“·”表示当前行，用“#”表示最末行。

例如：

8　　表示正文第 8 行

·　　表示当前行

+ 3　表示当前行后面的第 3 行

在编辑正文时，还常要用到编辑功能键，它们的定义与命令行编辑功能键一样，见表 2-3。

表 2-3　编辑功能键

编辑键	功能
‹Del›	删除一个字符
‹Esc›	删除当前行
‹Ins›	插入若干字符
‹F1›或→	复制一个字符
‹F2›	复制到指定的字符
‹F3›	复制到行末
‹F4›	删除到指定的字符为止
‹F5›	终止当前行并存入暂存器
‹F6›	^Z

下面着重介绍五条最基本的编辑命令，这五条命令掌握后，就能完成一般的编辑工作。

① 插入命令 I

I 命令使 EDLIN 进入插入工作方式，而使用控制字符 ^C 可退出 I 命令。I 命令的格

式为：

〔n〕I

其中n表示要插入处的行号，缺省值为当前行。如n大于文件的行数或为“*”，表示在文件末插入。

例如：

```
A>EDLIN EDLIN.HLP
New file
*I
   1:  〔n〕                    Edits line
   2:  〔n〕A                   Appends lines
   3:  〔n〕,〔n〕,n,〔c〕C       Copies lines
   4:  〔n〕〔,n〕D              Deletes lines
   5:  E                       Ends editing
   6:  〔n〕I                   Inserts text
   7:  〔n〕〔,n〕L              Lists text
   8:  〔n〕,〔n〕,nM            Moves text
   9:  〔n〕〔,n〕〔?〕R<str><^Z><str>
  10:                          Replaces lines
  11:  〔n〕〔,n〕〔?〕S<str><CR>
  12:                          Searches text
  13:  〔n〕T<file>             Transfers text
  14:  〔n〕W                   Writes lines
  15: *
*
```

在插入过程中，若下一行与上一行相似，可使用编辑功能键减少输入的键数。例如，第2行仅输入下列键：

<F1><F1><F1>A<F2>EAppe<Ins>nd<F3>s

即可。

有时要在正文中输入控制符，EDLIN采用前导符 ^V 加上一个大写字母表示控制符，例如：

^VC　表示 ^C

^VV　表示 ^V

② 退出命令E

在编辑完毕后，使用E命令退出编辑并将编辑后的正文记盘。盘上若存在老文件，则老文件被改名为 .BAK 文件作为后备，原来的 .BAK 文件被删去。

③ 显示命令L

L 命令用来显示不多于23行的一段正文。命令格式为：

〔n_1〕，〔n_2〕L

其中 n_1 的缺省值为当前行的前 11 行；n_2 的缺省值为 n_1 行的后 22 行。如果 n_2 与 n_1 的差大于 22，则 n_2 取其缺省值。例如：

－4，＋6L	列出当前行的前 4 行至其后 6 行
4L	列出 4—26 行
L	列出－11至＋11 行

④ 删除命令 D

D 命令用以删除一行或数行正文，格式为：

〔n_1〕，〔n_2〕D

其中 n_1 的缺省值为“·”（当前行）；n_2 的缺省值为 n_1。例如：

,8D	表示从当前行删至第 8 行
8D	表示删除第 8 行
D	表示删除当前行

⑤ 修改命令

命令的格式为：

〔n〕

其中 n 的缺省值为＋1。输入该命令后，即可对指定的行使用编辑键进行修改，修改完成后，输入一个 〈CR〉。

例如：

```
* 9
  9: *〔n〕〔,n〕〔?〕R〈str〉〈^Z〉〈str〉
```

为了把第 9 行中的两个“str”改成“STR”打入下列键：

〈F2〉sSTR〈F2〉sSTR〈F1〉

即可得到下面的结果：

```
9: *〔n〕〔,n〕〔?〕R〈STR〉〈^Z〉〈STR〉
```

注意，修改命令与其它命令联用时，应当用分号“；”分隔两条命令。而其它命令的联用，则可不用“；”分隔。

（3）其它命令

① 查找、替换字符串命令 S，R

S 是查找命令，用来在指定的正文段中查找一个字符串；而 R 是替换命令，用以把一段正文的旧字符串替换成新字符串。两条命令的格式分别为：

〔n_1〕〔,n_2〕〔?〕S〈字符串〉

〔n_1〕〔,n_2〕〔?〕R〈老字符串〉^Z〈新字符串〉

其中 n_1 的缺省值均为＋1（即当前行的下一行），n_2 的缺省值是最后一行。问号“？”表示提问方式。

例如，设当前行为第9行，对下文

```
*1,#L
   1:  〔n〕              Edits line
   2:  〔n〕A             Appends lines
   3:  〔n〕,〔n〕,n,〔c〕C Copies lines
   4:  〔n〕〔,n〕D        Deletes lines
   5:  E                 Ends editing
   6:  〔n〕I             Inserts text
   7:  〔n〕〔,n〕L        Lists text
   8:  〔n〕,〔n〕,nM      Moves text
   9: *〔n〕〔,n〕〔？〕R‹str› ‹^Z› ‹str›
  10:                    Replaces lines
  11:  〔n〕〔,n〕〔？〕S‹str› ‹CR›
  12:                    Searches text
  13:  〔n〕T‹file›       Transfers text
  14:  〔n〕W             Writes lines
  15:
```

分别使用命令

Rstr^ZSTR 和 1R str^ZSTR

后得到的结果分别为：

```
*Rstr^ZSTR
   11: 〔n〕〔,n〕〔？〕S‹STR›‹CR›
*1Rstr^ZSTR
    9: 〔n〕〔,n〕〔？〕R‹STR› ‹^Z› ‹str›
    9: 〔n〕〔,n〕〔？〕R‹STR› ‹^Z› ‹STR›
   11: 〔n〕〔,n〕〔？〕S‹STR› ‹CR›
*
```

第1例仅替换了一处，而第2例则全部替换了。

如果需要有挑选地替代一组字符串，则选用“？”。下例是将第6行和第8行中的 text 改为 lines 的操作过程：

```
*6,8？ Rtext^Zlines
        6:  〔n〕I          Inserts lines
O. K. ？ Y
        7:  〔n〕〔,n〕L     Lists lines
```

```
O. K. ? N
        8:  〔n〕,〔n〕,nM  Moves lines
O. K. ? Y
*6, 9L
        6:  〔n〕I         Inserts lines
        7:  〔n〕〔,n〕L    Lists text
        8:  〔n〕,〔n〕,nM  Moves lines
        9:  〔n〕〔,n〕〔? 〕R‹STR› ‹^Z› ‹STR›
*
```

② 复制与移动命令 C，T，M

EDLIN 允许把一段正文或另一个文件复制到指定的地方。C 命令用来复制一段正文，T 命令用来复制一个文件到当前文件中。这两条命令格式分别为：

〔‹始行›〕，〔‹末行›〕，‹目的行›，〔‹复制次数›〕C

〔‹目的行›〕T ‹文件名›

这里可缺省的行号其值都是当前行。

另外，EDLIN 还提供一条移动命令 M，它与 C 命令的区别是：传送之后把原来位置上的正文删掉。M 命令的格式为：

〔‹始行›〕，〔‹末行›〕，‹目的行› M

例如：

```
*1, 5, 16C      { 把第 1—5 行复制到第 16 行处 }
*1, 5, 16M      { 把第 1—5 行移动到第 16 行处 }
*16T ED.HLP     { 把文件 ED.HLP 复制到第 16 行处 }
```

③ 输入输出命令 A，W

前面提到，在输入文件大于编辑缓冲区的情况下，调入 EDLIN 仅装满 3/4 缓冲区。如需要编辑不在缓冲区内的正文，就须用 A 命令将其装入。如缓冲区已装满，须用 W 命令写一部分正文到盘上去。这两条命令的格式为：

〔n〕A

〔n〕W

其中 n 表示添加到缓冲区或写入到磁盘的行数，若省略，则表示装满 3/4 缓冲区或空出 1/4缓冲区。

2. 文字处理程序 WORD STAR 简介

（1）引言

WORD STAR 是一种全屏幕通用文字编辑程序，它使用“菜单”方式操作，有联机求助能力，使用十分方便。WORD STAR 不仅可以用来编辑源程序文件，而且也可以用来编辑西文的书信和文章。下面仅介绍使用 WORD STAR 编辑源程序文件的方法。如果读者要详细了解 WORD STAR 的使用方法，请参阅 WORD STAR 的使用手册。

（2）编辑文件的过程

欲使用 WORD STAR，当前盘上必须至少具备以下三个文件：

WSMSGS.OVR，WS.COM，WSOVLY1.OVR

其中，若 WSMSGS.OVR 不存在，WORD STAR 就不会提供提示信息以及命令选择清单。要启动 WORD STAR仅需打入：

A> WS

随后在屏幕的上部显示出主命令清单，同时在屏幕的下部显示出当前盘如下所示的目录。关于这些命令的含义，请参见后文中的表 2-4。

```
                editing no file
                  < < <  NO-FILE MENU  > > >
  —Preliminary Commands—  |  —File Commands—  |  -System Commands-
L  Change logged disk drive                    R  Run a program
F  File directory    off  (ON)    P  Print a file    X  EXIT to system
H  Set help level
  —Commands to open a file—  |  E  RENAME a file  |  -WordStar Options-
   D  Open a document file   |  O  COPY    a file |   M  Run MailMerge
   N  Open a non-document file  |  Y  DELETE a file  |  S  Run SpellStar
DIRECTORY of disk A:
     WS.COM        WSOVLY1.OVR  WSMSGS.OVR
```

接着输入命令 N，表示需要编辑的文件是源程序文件。屏幕上显示出：

```
N        editing no file
  Use this command to create and alter program source files
    and other non-documents. Word wrap defaults off;
    tabbing defaults to fixed ( TAB chars in file; 8-col stops );
    page breaks not shown; hi bit flags not used in file.
  For normal word processing uses, use the "D" command instead.
    A file name is 1-8 letters/digits, a period,
    and an optional 0-3 character type.
    File name may be preceded by disk drive letter A-D
    and colon, otherwise current logged disk is used.
    NAME OF FILE TO EDIT? PYRAMID.BAS
DIRECTORY of disk A:
      WS.COM        WSOVLY1.OVR  WSMSGS.OVR
```

这段文字的大意是：这条命令用来建立和修改源程序文件，而不是进行文章编辑，并请输入需要编辑的文件的文件名。这时，应该输入被编辑的文件名 PYRAMID.BAS，随后对新文件WORD STAR显示出“New file”，接着进入编辑状态，显示出：

```
A: PYRAMID.BAS FC=1 FL=1 COL 01        INSERT ON
              < < <   M A I N   M E N U   > > >
  --Cursor Movement--   -Delet-   -Miscellaneous-   -Other Menus-
^S char left  ^D char right  ^G char   ^I Tab   ^B Reform   (from Main only)
^A word left  ^F word right  DEL chr lf   ^V INSERT ON/OFF   ^J Help  ^K Block
^E line up    ^X line down   ^T word rt   ^L Find/Replce again   ^Q Quick  ^P Print
      --Scrolling--          ^Y line   RETURN End paragraph   ^O Onscreen
^Z line up    ^W line down              ^N Insert a RETURN
^C screen up  ^R screen down            ^U Stop a command
```

这张清单称为编辑命令清单，它给出各种编辑命令键的含义。各键的含义请见后文中的表2-5。在此状态下便可输入源程序。输入完毕后，按下^K退出编辑状态，进入块处理状态。下例是输入一段BASIC源程序后，进入块处理状态时显示出的结果：

```
^K          A: PYRAMID.BAS FC=195 FL=11 COL 01        INSERT ON
                    < < <   B L O C K   M E N U   > > >
  -Saving Files-   -Block Operations-   -File Operations-   -Other Menus-
S Save & resume   B Begin K End   R Read  P Print   (from Main only)
D Save--done      H Hide/Display  O Copy E Rename   ^J Help   ^K Block
X Save & exit     C Copy Y Delete J Delete          ^Q Quick  ^P Print
Q Abandon file    V Move W Write    -Disk Operations-   ^O Onscreen
  -Place Markers-   N Column on (OFF)   L Change logged disk   Space Bar returns
0-9 set/hide 0-9                      F Directory on (OFF)   you to Main Menu.
```

```
10 'Draw PYRAMID                                          <
20 CLS : WIDTH 40                                         <
30 C1=20                                                  <
40 FOR COUNT=1 TO 21 STEP 2                               <
50    PRINT TAB( C1 );                                    <
60    FOR STARS=1 TO COUNT                                <
70      PRINT "^" ;                                       <
80    NEXT                                                <
90    PRINT                                               <
120 END                                                   <
```

最后，再打入X命令，便结束编辑并将源程序记盘。显示下列信息后，返回MS-DOS：

```
^KX          WAIT
     SAVING FILE A: PYRAMID.BAS
A>
```

(3) WORD STAR的常用命令

① 主命令

主命令用以处理文件级的操作，例如：打印、换名、复制及删除一个文件。使用这些命令时，仅需键入一个字母，随后 WORD STAR 会作出简明的提示，帮助完成所需要的操作。这些命令大多数类似于 MS-DOS 的键盘命令，这里不作详细介绍。下面仅举改变当前盘命令“L”一例来说明主命令的使用方法。

为了将当前盘从 A 改到 B，打入 L 命令后屏幕上显示出：

```
L          editing no file
  The LOGGED DISK ( or Current Disk or Default Disk ) is the
  disk drive used for files except those files for which
  you enter a disk drive name as part of the file name.
  WordStar displays the File Directory of the Logged Disk.
  THE LOGGED DISK DRIVE IS NOW A :
  NEW LOGGED DISK DRIVE ( letter, colon, RETURN ) ?
```

回答“B:”之后，当前盘就改为 B。

下面列出主命令及其功能的对照表，以供读者查找。

表 2-4 WORD STAR 的主命令

命 令	功 能
D	编辑文章
E	文件改名
F	改变目录显示状态
H	设置联机求助(help)详略程度
L	改变当前盘
N	编辑源程序文件
O	复制一个文件
P	打印一个文件
R	执行一条 MS-DOS 命令
X	退出 WORD STAR
Y	删除一个文件

② 编辑命令

编辑命令在编辑状态下使用，对源程序进行编辑。这些命令完成光标移动、删除和插入等功能。常用的编辑命令键如表 2-5 所示。

此外，可用 ^J 进入求助状态，这对于在操作中忘记某条命令含义的用户来说，是十分有用的。

③ 双字符编辑命令

双字符编辑命令以 ^Q 打头，后接一个字母，这些命令可用来加快编辑过程的实现。

较常用的双字符编辑键如表 2-6 所示。

表 2-5　常用的编辑键

编辑键	功能
← 或 ^S	光标左移一个字符
→ 或 ^D	光标右移一个字符
↑ 或 ^E	光标上移一行
↓ 或 ^X	光标下移一行
‹Del›或 ^G	删除光标处的一个字符
^Y	删除一行
‹Tab›或 ^I	制表符
‹Ins› 或 ^V	进入或退出插入工作方式

表 2-6　双字符编辑键

双字符键	功能
‹Home› 或 ^QE	光标移到屏首
‹End› 或 ^QX	光标移到屏末
‹F9› 或 ^QR	光标移到文件开头
^QC	光标移到文件末尾
^QB	光标移到块首
‹F10› 或 ^QK	光标移到块末
^QA	字符串替换
^QF	查找字符串

在这些命令中，字符串替换符较为重要。下面举例说明 ^QA 的使用方法。
如果想要在已存在的程序中将变量 C1 改成 Column，则打入 ^QA，屏幕上显示出：

```
^QA     A: PYRAMID.BAS FC=1 FL=1 COL 01                INSERT ON
  ^S=delete character    ^Y=delete entry    ^F=File directory
  ^D=restore character   ^R=Restore entry   ^U=cancel command
   FIND? C1  REPLACE WITH? COLUMN  OPTIONS?  (? FOR INFO) G
   10 ' Draw PYRAMID
   20 CLS : WIDTH 40
   30 C1=20
   40 FOR COUNT=1 TO 21 STEP 2
   50    PRINT TAB( C1 );
   60    FOR STARS=1 TO COUNT
```

```
 70            PRINT "^";
 80          NEXT
 90          PRINT
120     END
```

对屏幕上提出的问题作出上述回答后，WORD STAR从文件头开始查找“C1”，查到文件末尾。（因G选择开关表示在全文中查找）。若找到时，则找到的“C1”将不断闪烁，直至按下“Y”（替换）或“N”（不替换）为止。因程序中C1出现两次，故上述替换过程将重复两次。

^QA的开关主要有：B从后向前查；G在全文中查；N不询问；<n>则表示查找n次。

④ 块处理双字符键

块处理双字符键由^K加一个字母组成，用以处理一个程序块或者一个文件。在“编辑文件的过程”那一段中介绍的退出命令^KX，就是这组键的一例。

这里所说的程序块是指用<F7>指定的块首与<F8>指定的块尾之间的程序段。程序块可以被移动、复制、删除或写入到另一个文件中去。表2-7列出了常用的^K双键命令及其功能。

表2-7　块处理及文件处理双字符命令

命　令	功　能
^KS	保存文件并重新打开文件
^KD	保存文件
^KX	保存文件并返回MS-DOS
^KQ	撤消文件
<F7>或^KB	置块首
<F8>或^KK	置块末
^KC	复制块到光标处
^KY	删除块
^KV	移动块至光标处
^KW	将块写入文件

例如，要把下列程序的80到90句复制到120句之前，则首先把光标移至80行首，打入^KB或<F7>，然后把光标移至90行末，打入^KK或<F8>，这样程序块就建好了。再移动光标到120行首，用^KC将这两行复制一下。最后，再用光标移动命令将光标移上两行，逐次把这两句的行号改成100和110，结果如下：

```
10 'Draw PYRAMID
20 CLS : WIDTH 40
30 COLUMN=20
```

```
 40 FOR COUNT=1 TO 21 STEP 2
 50   PRINT TAB ( COLUMN ) ;
 60   FOR STARS=1 TO COUNT
 70     PRINT "^" ;
 80   NEXT
 90   PRINT
100 NEXT
110 PRINT
120 END
```

随后，再打入 ^KX 退出编辑。这样，该程序就全部编辑好了。

3．连接程序 LINK

（1）引言

LINK 程序用以连接宏汇编程序或高级语言编译程序生成的若干目标模块，并与指定的库文件连接，产生一个可执行文件。

在 LINK 程序运行时，主要涉及四种类型的文件，即：目标文件 ·OBJ、库文件 ·LIB、可执行文件 ·EXE 和公共变量列表文件 ·MAP。它们之间的关系如图 2-6 所示。

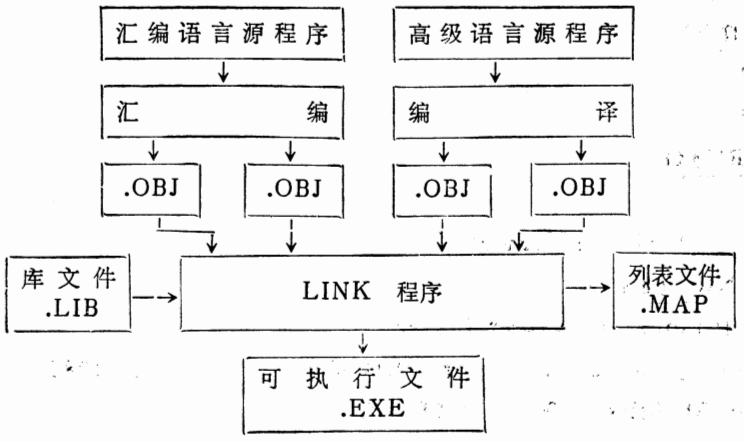


图 2—6　LINK 运行时诸文件间的关系

（2）命令格式

LINK 程序的调用有以下三种形式：

LINK

LINK〈目标文件〉，〈可执行文件〉，〈列表文件〉，〈库文件〉〔/开关…〕

LINK @〈路径名〉

第 1 种形式是提问方式，LINK 将逐次提问目标、可执行、列表和库文件的名字；第 2 种形

式要求在命令行中给出所有参数；第3种形式中@后面指出的文件是用户响应文件，响应文件中包含着提问方式下的回答信息，它由用户事先建立。

这三种方式效果一样，例如下列三例效果完全相同。

① 提问方式

```
A>LINK
IBM Personal Computer Linker
Version 2.00 (C) Copyright IBM Corp 1981, 1982, 1983
Object Modules [.OBJ]: MYPROG
Run File [MYPROG.EXE]: MYP
List File [NUL.MAP]: MYP
Libraries [.LIB]: BASRUNG
```

② 命令行方式

```
A>LINK MYPROG, MYP, MYP, BASRUNG
IBM Personal Computer Linker
Version 2.00 (C) Copyright IBM Corp 1981, 1982, 1983
```

③ 响应文件方式

```
A>COPY CON: RESP
MYPROG
MYP
MYP
BASRUNG
^Z
        1 File(s) copied
A>LINK @RESP
IBM Personal Computer Linker
Version 2.00 (C) Copyright IBM Corp 1981, 1982, 1983
Object Modules [.OBJ]: MYPROG
Run File [MYPROG.EXE]: MYP
List File [NUL.MAP]: MYP
Libraries [.LIB]: BASRUNG
```

如果要连接多个目标模块，可将模块分别用加号“+”连起来做为目标文件参数输入。如在参数中出现一个分号“；”，则表示以后的参数全部选用缺省值。例如：

```
A>LINK MYPROG1+MYPROG2, , PROGLIST, BASRUNS
A>LINK MYPROG;
```

其中第1行表示把两个目标模块（假设是编译BASIC生成的）连接起来，运行程序命名

为 MYPROG1.EXE,第 2 行则表示源文件为 MYPROG.OBJ,运行文件名为 MYPROG.EXE,不产生公共变量列表文件，使用的是缺省的 BASIC 运行库（BASRUNG.LIB）。

（3）连接开关

LINK 有七个开关，用以指出对连接操作的附加要求。开关用斜杠“/”隔开，置于文件名参数之后。这些开关的含义如表 2-8 所示。

表 2-8　LINK 的开关

开　关	功　　能
/D	把数据装到数据区 DS 的上部
/H	把运行程序装入到内存的高区
/L	在列表文件中加入行号和地址
/M	以字典顺序在列表文件中列出全部公共变量
/P	暂停连接
/S:n	把可执行文件的栈的大小置为 n
/N	禁止连接缺省库

4. 动态调试程序 DEBUG

（1）DEBUG 的启动

汇编语言程序产生的目标代码，调试时比较困难。动态调试程序 DEBUG 就是调试汇编语言程序的一个有力工具。

DEBUG 的启动格式如下：

DEBUG〔<路径名>〔<参数>〕〕

其中路径名指的是被调试的程序，参数就是该程序所涉及的参数。下面是几个启动 DEBUG 的例子：

```
A>DEBUG B:DISKCOPY.COM A: B:
A>DEBUG MYPROG.EXE
A>DEBUG
```

DEBUG 装入内存后，接着就把被调试的程序从盘上找到并将它装入内存（对于 .COM 文件，DEBUG 将它装入到最低可用的区段中，并从 100H 开始装入），然后显示出提示符“-”；而第 3 行所示的启动方式则直接显示“-”，等待用户进一步输入命令。

在 DEBUG 命令中使用的地址其格式约定如下：

〔<段地址>:〕<位移量>

其中<段地址>可以是段寄存器名，也可以是十六进制的值，也可以缺省。例如：

```
CS:100
4BA:100
```

而地址范围的格式为：

〈段地址〉：〈始位移量〉〈末位移量〉

或　〈段地址〉：〈始位移量〉L〈长度〉

例如：

```
CS：100 110
4BA：100 L10
```

（2）汇编与反汇编命令

DEBUG 提供一条汇编命令 A 和一条反汇编命令 U。

① 汇编命令 A

A 是一条逐行汇编命令，主要用于小段程序的汇编以及修改目标程序。使用逐行汇编命令汇编程序时，一般不允许使用标号和伪指令。但在 MS-DEBUG 中允许使用 DB 和 DW 这两条伪指令。汇编命令的格式如下：

A〔〈地址〉〕

A 命令用 ^C 退出。汇编中发现错误时，显示出一个“?”并要求重新输入。

使用 A 命令来汇编小段程序往往比使用汇编、连接程序方便。汇编好的程序也可用第（6）小段讲的写盘命令保存到盘上去。

例如：

```
A> DEBUG
-A
08F1：0100 MOV AL,40
08F1：0102 MOV CH,24
08F1：0104 JMP 110
08F1：0106 DB 'DATA' 00
08F1：010C DW 1234 5678
08F1：0110 MOV AH, 01
08F1：0112 INT 20
08F1：0114 ^C
```

注意，A 命令后面不指出地址时，表示程序的起始地址就是 DEBUG 指针所指向的当前地址。

② 反汇编命令 U

U 命令可以对二进制代码程序作反汇编，常用于分析和调试目标程序。它的格式如下：

U〔〈地址范围〉〕

例如，上例中的程序可用 U 命令反汇编出：

```
-U 100 104
08F1：0100 B040          MOV     AL, 40
08F1：0102 B524          MOV     CH, 24
```

```
08F1 : 0104 EB0A          JMP     0110
-U110 112
08F1 : 0110 B401          MOV     AH, 01
08F1 : 0112 CD20          INT     20
```

（3） 显示和修改内存与寄存器命令

这一组命令常用于检查或设置寄存器以及数据区。

① 显示内存命令 D

这条命令按照十六进制的形式以及对应的 ASCII字符形式显示内存内容。

例如：

```
-DCS : 106 10F
08F1 : 0106  44 41—54 41 20 00 34 12 78 56    DATA·4·xV
```

或

```
—D8F1 : 106 10F
08F1 : 0106  44 41—54 41 20 00 34 12 78 56    DATA·4·xV
```

② 修改内存命令 E

E 命令用来修改内存，它的格式为：

E 〈地址〉〔〈字节串〉〕

其中字节串为一系列用空格隔开的十六进制字节，或者是用单引号括起来的字符串。

例如：

先用 E 命令修改内存：

```
-E112  00 'END OF PROGRAM'
```

再用 D 命令检查：

```
-D112 120
08F1 : 0112 00 45 4E 44 20 4F-46 20 50 52 4F 47 52 41  .END OF PROGRA
08F1 : 0120 4D                                          M
```

③ 显示和修改寄存器命令 R

这条命令的格式为：

R〔〈寄存器〉〕

当 R 命令后面不带参数时，显示各寄存器的内容，否则修改指定寄存器的内容。

在显示寄存器内容时，首先显示 13 个 16 位寄存器的内容，随后是标志寄存器的内容，最后一行是下一条要执行的指令地址及指令内容。

例如：

```
—R
AX=0140  BX=0000  CX=2400  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS= 8F1  ES=08F1  SS=08F1  CS=08F1  IP=0104  NV  UP  DI  PL  NZ  NA  PO  NC
08F1 : 0104 EB0A            JMP     0110
```

要修改某寄存器，就在R命令后打入寄存器名，DEBUG显示出这个寄存器原来的值，然后打入新值即可。

例如：

```
—RAX
AX 0140
:1234
```

（4）运行和跟踪命令

① 运行命令G

G命令用来启动运行一个程序或程序的一段。它的格式如下：

G〔=<始址>〕〔<断点>…〕

其中断点最多允许设置十个。如果G命令不带参数，则从头运行装入的程序，运行后仍返回DEBUG。如G命令后有断点参数，则程序执行到断点地址时暂停并显示出各寄存器状态。

例如：

```
—G=100 110
AX=1240  BX=0000  CX=2400  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS= 8F1  ES=08F1  SS=08F1  CS=08F1  IP=0110  NV  UP  DI  PL  NZ  NA  PO  NC
08F1: 0110 B401             MOV     AH, 01
```

② 跟踪命令T

T命令用来逐条跟踪程序，它的格式是：

T〔=<地址>〕〔<跟踪条数>〕

每条指令执行后都将显示各寄存器内容。

例如：

```
—T=100 2
AX=0140  BX=0000  CX=2400  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=08F1  ES=08F1  SS=08F1  CS=08F1  IP=0102  NV  UP  DI  PL  NZ  NA  PO  NC
08F1 : 0102 B524            MOV     CH, 24
AX=0140  BX=0000  CX=2400  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=08F1  ES=08F1  SS=08F1  CS=08F1  IP=0104  NV  UP  DI  PL  NZ  NA  PO  NC
08F1 : 0104 EB0A            JMP     0110
```

（5）查找、比较、填充和移动内存命令

这组命令对内存进行比较、移动、填充固定的值以及在内存中查找一个特定的字节。这

些命令的格式分别是：

① 移动内存命令 M

M 〈源地址范围〉〈目的地址〉

② 填充内存命令 F

F 〈地址范围〉〈要填入的字节或字节串〉

③ 比较命令 C

比较命令用以比较两块内存区域的内容。若不相同，则显示出两者的地址和内容。命令格式是：

C 〈源地址范围〉〈目标地址〉

④ 查找命令 S

查找命令在指定的内存范围内查找一个字节串。若找到，则显示出它们的地址。其格式为：

S 〈地址范围〉〈要查找的字节或字节串〉

（6） 磁盘文件操作命令

① 指定文件名命令 N

这条命令用来设置文件名以供读写。格式如下：

N 〈文件名〉〔〈文件名〉…〕

② 读盘命令 L

命令格式为：

L〔〈地址〉〔〈盘号〉〈相对扇区号〉〈扇区数〉〕〕

其中有关磁盘的参数只有在按物理地址读盘（不是读文件）时才使用，盘号 0 表示 A 盘，1 表示 B盘，扇区数为读入的扇区总数。

在读文件时，L 命令常与 N 命令结合使用，〈地址〉缺省时表示 CS:100。

例如，先用 N 和 L 命令读入文件：

```
-NMYPROG .COM
-L
```

再用 U 命令检查一下：

```
-U100 104
08F1:0100 B040          MOV     AL,40
08F1:0102 B524          MOV     CH,24
08F1:0104 EB0A          JMP     0110
```

③ 写盘命令 W

写盘命令的格式为：

W〔〈地址〉〔〈盘号〉〈相对扇区号〉〈扇区数〉〕〕

参数的含义同L命令。

使用 W 写一个文件时，要先使用 N 命令给予文件名，再用 R 命令把文件的长度送到寄存器 BX 和 CX 中。

例如：

```
-R BX
BX 0100
: 0000
-R CX
CX 0000
: 2400
-NMYPROG.COM
-W
Writing 2400 bytes
```

（7）其它命令

① 十六进制运算命令H

H命令用来计算两个十六进制数的和与差，例如：

```
-H 12 34
0046 FFDE                    {0046为12与34之和，FFDE为差}
```

② 输入命令I

I命令用来显示从某端口取得的输入数据字节，命令格式为：

I〈端口号〉

③ 输出命令O

O命令把指定字节送到指定的端口去。命令格式为：

O〈端口号〉〈字节〉

例如：

```
-O 3BC 7                     {把07送到端口3BC去}
```

第五节 MS-DOS的内部结构及与用户程序的接口

从使用的角度来看，MS-DOS有两个面向用户的接口：一个是面向操作员的键盘命令，已在前面几节中介绍过；另一个是面向程序员的系统功能调用（以下简称为“功能调用”），将在这一节中说明。在讨论MS-DOS与用户程序的接口之前，先把MS-DOS的内部结构作些介绍。

1. MS-DOS的内部结构

（1）MS-DOS的组成

从第一节中得知，MS-DOS主要由四个部分组成，即：引导程序（BCOT）、输入输出系统（BIOS和IBMBIO.COM）、文件系统（IBMDOS.COM）以及键盘命令处理程序（COMMAND.COM）。

引导程序是在磁盘初始化时，由FORMAT命令写在软盘或硬盘的0柱0面1扇区上

的。它在系统启动时用来查找和装入 MS-DOS（IBMBIO·COM 和 IBMDOS·COM），如未找到则给出出错信息。

IBMBIO·COM 是 ROM 中 BIOS 的低级接口模块。在系统启动过程中，IBMBIO.COM 负责初始化设备状态、填写中断向量表以及装入 COMMAND·COM。它和 IBMDOS·COM 都是隐含文件，一般不能用列目录命令列出。

IBMDOS·COM 是操作系统的核心部分，主要由文件管理、磁盘读写和其它外设管理三方面的功能子程序组成。COMMAND·COM 和用户程序通过使用它所提供的 75 条系统功能调用来使用 DOS。系统功能调用是 DOS 提供给用户程序的高级接口。

COMMAND·COM 是键盘命令处理程序，它是操作员与系统的接口，用来接收并解释执行键盘命令。COMMAND·COM 处理的命令有四种：选盘命令，内部命令，外部命令（.COM 和 .EXE 文件）和批处理命令（.BAT 文件）。

（2）盘区分配

MS-DOS 2.00 版支持单面和双面$5^1/4$时软盘以及硬盘，不同类型盘的盘空间分配不尽相同。下面仅以双面 $5^1/4$ 时软盘为例，介绍 MS-DOS 的盘空间分配。

双面的 $5^1/4$ 时软盘，每面各有 40 道，每道又分成九个扇区（第 1 版 MS-DOS 的盘仅有八个扇区），每个扇区 512 字节。盘上除文件区外，还记录着引导程序、两份文件定位表（FAT1 和 FAT2）和根目录，如表 2-9 所示。

表 2-9　MS-DOS 的盘区分配

起始位置	长度	内容
0 柱 0 面 1 区	1	引导程序 BOOT
0 柱 0 面 2 区	2	第一文件定位表 FAT1
0 柱 0 面 4 区	2	第二文件定位表 FAT2
0 柱 0 面 6 区	7	根目录区 ROOTDIR
0 柱 1 面 4 区 39 柱 1 面 9 区	708	（MS-DOS 系统文件区） 用户文件区

下面介绍文件定位表以及根目录的结构。

① 文件定位表 FAT

MS-DOS 的文件结构是链表结构，链表的指针保存在文件定位表中。在 MS-DOS 中盘空间分配单位是“簇”，簇的大小由盘的类型决定。对于双面软盘，簇的大小是 1KB。每簇在 FAT 中占一个 12 位的项。

FAT 的前两项用来标记盘的类型，双面软盘为 FDFFFF。从第 3 项开始表示盘簇的分配状态，它们的含义为：

000　未用

FF8-FFF　文件的最末簇

×××　文件下一簇的簇号

文件的第 1 簇的簇号记录在该文件的文件目录中。

采用链表结构有两个特点：其一是文件目录仅记录文件的首簇号，即使很大的文件也仅需一个目录项；其二是文件长度仅受盘空间的限制。这种文件结构如图 2-7 所示。

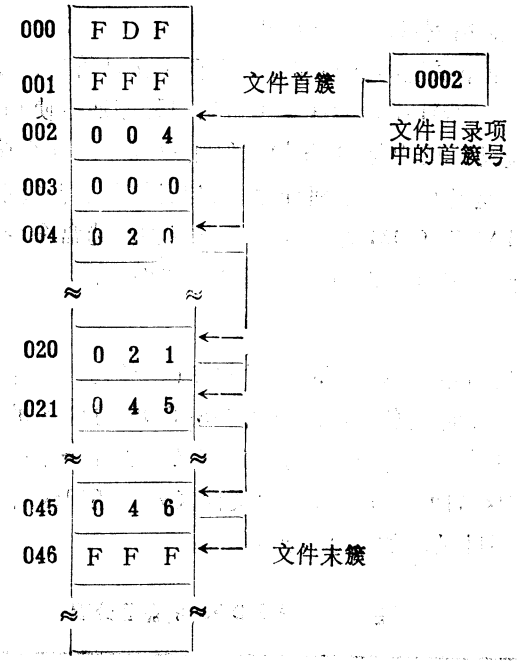


图 2—7　文件定位表 FAT

② 根目录

根目录是盘片在初始化时，由 FORMAT 程序建立的，它有固定的长度。对于双面 5¼ 吋软盘，长度是 3.5K，可存放 112 个目录项。在 MS-DOS 中，一个文件仅占用一个目录项。

应当注意，子目录是作为文件处理的，所以它的目录项的个数仅受盘空间的限制，可以大于 112 个。

每个目录项占 32 个字节，它们的内容及其含义如下：

0			7	8		A	B	C		F
文	件	名		类型名			属性	未	用	
未	用		时	间	日	期	簇	号	文件长度	
10		15	16	17	18	19	1A	1B	1C	1F

图 2—8　文件目录项

* 文件名：其中第 1 个字节表示目录项的状态：

00H　从未用过的目录项

E5H　已删除文件的目录项

2EH　子目录标记项（如果第2字节也为2E，表示上级目录标记项）

其它　文件名的首字符

* 属性：共六种文件属性，其中有些属性可组合使用，含义如下：

b7	b6	b5	b4	b3	b2	b1	b0
×	×	更改位	子目录	卷标位	系　统	隐　含	只　读

其中隐含、系统和子目录文件不能用一般的目录操作检索；只读文件只能读不能写；卷标位为1表示该目录项记录着这张盘的卷名；更改位在文件修改后置1。

* 时间和日期：记录文件最后一次更改的时间和日期
* 簇号：文件的第1簇的簇号
* 文件长度：以字节为单位的文件长度

（3）内存分配

除命令处理程序的可覆盖部分（内部命令的解释程序）驻于高区外，MS-DOS的主要部分常驻于内存的低地址部分。MS-DOS的内存映象如图2—9所示。

0000：0000 0040：0000 0050：0000 0060：0000	中断向量表 ROM 通讯区 DOS 通讯区 IBMBIO.COM IBMDOS.COM DOS 工作区 命令处理程序(常驻部分)
	用户区
	命令处理程序(可覆盖部分)
F000：0000	ROM BIOS

图2—9　MS-DOS的内存映象

下面介绍程序前缀区以及文件控制块FCB。

① 程序前缀区

当执行一条外部命令或执行用户程序时，命令处理程序要在可用内存的最低区处的00—FFH单元建立一张程序前缀表，然后把用户程序装入100H开始的内存中。前缀表的内容如图2—10所示。

其中INT 20是结束用户程序软件中断，类似于将要介绍的功能调用00H和4CH。用

户程序的最后一条指令可以是“JMP 0000”。

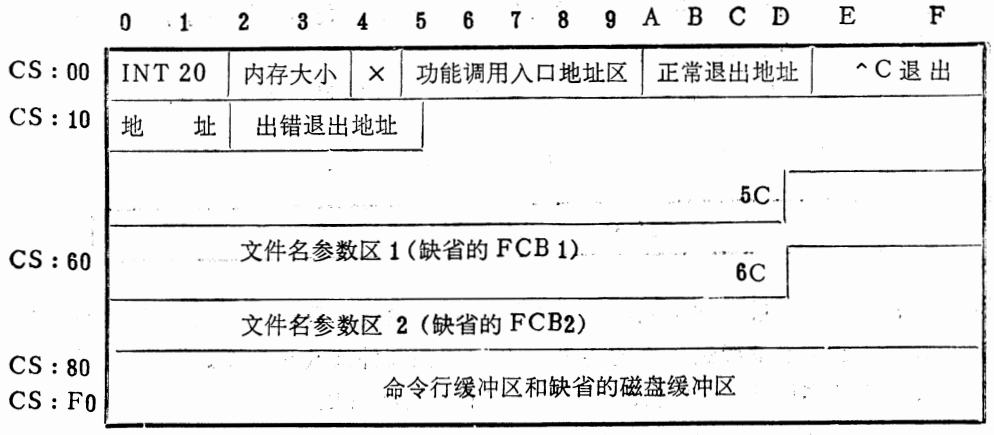


图 2—10 程序前缀区

在处理命令行时，命令处理程序把命令行的头两个文件参数送入 FCB1 和 FCB2，其余参数放在从 81H 开始的命令行缓冲区，80H 单元指出命令行的长度。

② 文件控制块 FCB

文件控制块 FCB 是查找文件的依据，用户在使用功能调用访问文件时，通常都要建立 FCB。FCB 的内容及其含义如下：

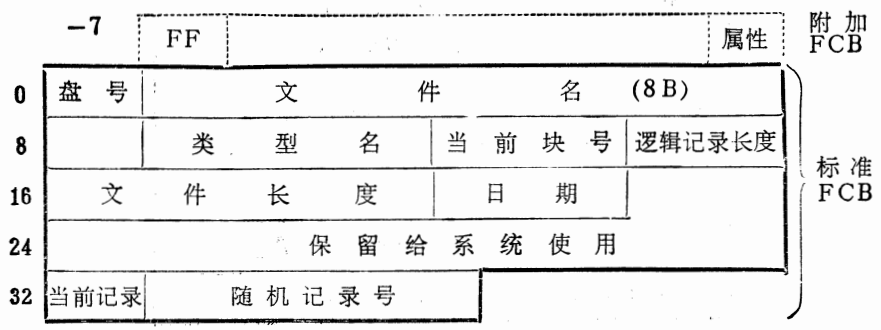


图 2—11 文件控制块 FCB

* 盘号：0 表示当前盘（文件打开后改成对应的盘号）；1—4 表示 A—D 盘
* 当前块号：当前相对块号，每块由 128 个记录组成（用于顺序文件读写）
* 逻辑记录长度：以字节为单位的逻辑记录长度（使用功能调用前置成 80H）
* 当前记录：当前块内的当前记录的相对序号（用于顺序文件读写）
* 随机记录号：当前随机记录号（用于随机文件读写）
* 属性：含义同目录项的属性，只有当 FCB—7 为 FF 时才有效

2. MS-DOS 的软件中断

软件中断是程序自行安排的中断，它常在 CPU 执行到程序中的“INT n”指令时产生，CPU 根据中断类别的编号 n 转去运行不同的中断服务程序。

MS-DOS 使用编号从 20H 到 3FH（中断向量表存放在内存的 80—FFH 区域中）的软件中断作为操作系统调用，为其它程序提供不同类型的服务。下面讨论常用的几条软件中断。

（1）系统功能调用

MS-DOS 的 21H 号软件中断用于为用户提供系统功能调用。使用的方法是先在 AH 中放入功能号，再按照该功能调用的要求设置好其它有关寄存器，然后执行 INT 21。关于具体的功能调用的使用方法，将在本节第 3 段作介绍。

例如：

```
A> DEBUG <CR>
-A
907:100 MOV AH, 2        ;  功能号2→AH
907:102 MOV DL, 41       ;  “A”→DL
907:104 INT 21           ;  系统功能调用
907:106 INT 20           ;  程序正常结束
```

表示要在控制台上输出一个“A”，然后返回操作系统（INT 20）。

（2）磁盘读写

INT 25 和 INT 26 用来读写盘上的若干个扇区，这是两条 BIOS 调用。在使用这两条命令前，必须按下列要求设置参数：

（AL）　驱动器号（0=A，1=B……等）
（CX）　读写的扇区数
（DX）　开始扇区的逻辑号（0 号表示 0 柱 0 面 1 区）
（DS:BX）　内存起始地址

例如，从双面盘上读出目录的程序如下：

```
MOV AL, 0          ;  把盘号置为 A
MOV CX, 7          ;  双面盘根目录长为 7个扇区
MOV DX, 5          ;  目录区从 0 柱 0 面 6 区开始
MOV BX, 1000       ;  传送到 1000H 起的内存中去
INT 25             ;  读盘
JMP 0              ;  返回操作系统
```

（3）程序的退出

退出软中断有两条，正常退出 INT 20 和驻留退出 INT 27，现分别介绍如下。

① 正常退出 INT 20

用户程序执行完毕后，可使用这条软中断结束用户程序，退回操作系统。

② 驻留退出 INT 27

用于用户自己写的中断处理程序的初始化。用这种方法退出后，留下的程序被MS-DOS视为自身的一部分，不会被其它程序覆盖。在其它用户程序中，可以通过使用软中断而调用。

（4）其它

这组有 Ctrl-C 处理 INT 23；结束退出处理 INT 22 以及出错处理 INT 24。这里就不再细述。

3. MS-DOS 的系统功能调用

系统功能调用是 MS-DOS 为用户程序提供的一组常用子程序，汇编程序员可通过相当简单的一些规则来调用这些子程序。

使用功能调用的一般过程为：把调用号放入寄存器 AH 中，设置入口参数，然后执行软件中断 INT 21，最后分析出口参数，如图 2—12 所示。

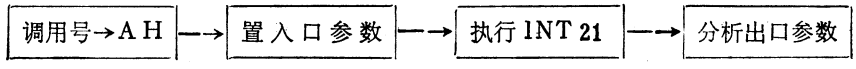


图 2—12 系统功能的调用过程

MS-DOS 的功能调用主要有设备管理、文件管理和目录管理三类，关于它们的使用方法分述如下。

（1）设备管理

① 字符 I/O 设备的功能调用

这一组包括键盘、显示器、异步通讯控制器以及打印机输入输出的 12 条系统功能调用，现分别介绍。

* 控制台字符 I/O：这组调用共有五个，其调用号分别为1，2，6，7，8。它们的入口参数 DL 用于放置输出字符，返回参数AL中保存输入字符（各调用的功能请见第 4 段）。

* 控制台字符串 I/O：这组调用中，09H 号用于把 DS:DX 指定的内存中以 “$” 结束的字符串在屏幕上显示出来；调用 0AH 则用于把键盘上输入的一行字符（以<CR>结尾）读入缓冲区中。调用 0AH 使用的缓冲区由用户设定，缓冲区始址放于 DS:DX，缓冲区长度填放在缓冲区第 1 个字节中。调用后，行的实际长度在第 2 个字节中，从第 3 个字节开始是输入的字符串。

* 其它设备字符 I/O：调用 05H 的功能为在打印机上输出 DL 中的字符。调用 03H 和 04H 的功能分别为异步通讯器的输入和输出，输入字符和输出字符分别在AL和 DL 中。通讯口的初始状态是 2400 波特、无校验、1 位停止位和 8 位数据位。

例如：

```
—U100 117
093A: 0100 B409                MOV     AH, 09
```

```
093A: 0102 BA0002          MOV     DX, 0200
093A: 0105 CD21            INT     21              ; 输出 "Do you want to go on? "
093A: 0107 B401            MOV     AH, 01
093A: 0109 CD21            INT     21              ; 输入一个字符
093A: 010B 3C4E            CMP     AL, 4E
093A: 010D 7408            JZ      0117
093A: 010F B409            MOV     AH, 09
093A: 0111 BA1802          MOV     DX, 0218
093A: 0114 CD21            INT     21              ; 输出 "##### GO ON #####"
093A: 0116 90              NOP
093A: 0117 CD20            INT     20              ; 返回操作系统
-D200L30
093A: 0200  0D 0A 44 6F 20 79 6F 75-20 77 61 6E 74 20 74 6F    ..Do you want to
093A: 0210  20 67 6F 20 6F 6E 3F 24-0D 0A 0D 0A 23 23 23 23    go on? $....####
093A: 0220  23 20 47 4F 20 4F 4E 20-23 23 23 23 23 24 00 00    # GO ON #####$..
```

这段程序用来询问是否要继续下去，若输入“N”则停止，否则显示出“# # # # # GO ON # # # # #”。

② 磁盘设备的功能调用

这组调用常用来设置磁盘参数，以供读写盘块的软中断使用。主要的几种有：

* 盘初始化（0DH）
* 选择当前盘（0EH）
* 取当前盘盘号（19H）
* 置盘缓冲区地址（1AH）
* 取盘缓冲区地址（2FH）

（2）文件管理

为了使程序员能方便地按名使用文件，MS-DOS提供了两组有关文件管理的功能调用，分别称为传统的文件操作和新增的文件操作。传统的文件操作类似于(CP/M①)，通过使用文件控制块FCB来访问文件；新增的文件操作则类似于高级语言，直接使用文件名或文件号来访问文件。后一组显然比前一组容易使用，但使用后一组调用写出的程序不能在MS-DOS 1.10下运行。

① 传统的文件操作

使用这组功能调用读写文件，首先必须在内存中建立一个FCB，并填入文件名、类型名和盘号，随后再使用各种文件操作的功能调用即可。FCB的构成如表2-10所示。

这组调用除建立FCB调用（29H）外，都使用寄存器DS:DX指向FCB，返回参数AL用来表示操作的成败，等于00时表示成功，否则表示失败。下面分别介绍。

* 建立FCB（29H）：当用户使用命令行中的文件名参数建立FCB时，使用这条调用.调用前，首先把DS:SI指向输入的命令行,再把ES:DI指向FCB并将AL置为0F（表

① CP/M是Digital Research公司开发的8位机上最流行的操作系统

示略去文件名前的间隔符）。调用成功后，则可使用填好的 FCB 进行其它文件操作。

表 2-10　FCB 的构成

名　称	位　置	说　明
FCBAT	−1	文件属性(仅当 FCB−7=FF 时有效)
FCBDK	0	盘号(0 表示当前盘)
FCBFN	1- 8	文件名
FCBFT	9-11	类型名
FCBCB	12-13	当前块号(每块 128 个记录)
FCBRZ	14-15	逻辑记录长度
FCBFS	16-19	文件长度(低字在前高字在后)
FCBDT	20-21	文件建立的日期
FCBCR	32	当前记录号
FCBRR	33-36	随机记录号(低字在前高字在后)

* 建立、打开、关闭和删除文件：要在磁盘上记录新文件，首先必须使用调用 16H 建立文件，调用前必须填好 FCB。如果使用已存在的文件，则必须使用 0FH 号调用首先打开文件，调用前也需填好 FCB。文件被修改后必须使用调用 10H 关闭，以便把文件的目录写到盘上。文件不再使用时，可用 13H 号调用删除。

* 顺序读写文件（14H 和 15H）：这两条调用可用来以顺序的方式读写 FCBCB 和 FCBCR 指定的一个记录。读写完毕后 FCBCR 自动加 1，若 FCBCR 已为 127，则 FCBCB 加 1 而 FCBCR 恢复为 0。（14H）用来读，（15H）用来写。其中 14H 号调用的返回参数为 01 时，表示读到文件结束符且最后读入的记录已不含有效数据，若为 03 则表示还有不满一个记录的数据在缓冲区中。

* 随机读写文件（21H 和 22H）：如希望以随机方式读写文件，可使用这组调用。（21H）为读一记录，（22H）为写一记录。在调用这两个功能以前，必须在 FCBRR 中填入要读写的记录号。FCBRR 由双字组成，低字在前，高字在后。读写完毕后，FCBCB 和 FCBCR置成与FCBRR 相对应的值。

② 新增的文件操作

这组文件操作与上组操作功能基本相同，但使用的方式不同，且使用时比上组简便而灵它具有以下特点：

——在建立、打开或删除文件时，使用一个字符串表示文件，这个字符串包括盘符和文件路径名。

——文件打开成功后，返回一个文件号，下次对这个文件读写时，仅使用文件号。

——允许使用路径名。

——文件读写以字节为单位，一次可以读写直到 64KB 数据。

——读写数据的方式为随机方式，读写的起始位置由读写指针决定，这个指针可以由用户任意设置。

——通过标准设备文件号可以直接使用设备。标准设备文件号的定义如表 2-11 所示。

表 2-11　标准设备文件号

文件号	设备名
0 0 0 0	标准输入设备
0 0 0 1	标准输出设备
0 0 0 2	出错信息输出设备
0 0 0 3	异步通讯控制器
0 0 0 4	打印机

这组调用多数都使用 DS:DX 表示路径名首址，BX 表示文件号，都使用进位位 CY 表示操作的成败。如果失败则 CY 置 1，AX 中为错误类型（见表 2-12），否则 CY 为0。

表 2-12　常见出错表

AX	错误类型
1	非法的调用号
2	文件未找到
3	路径未找到
4	同时打开的文件太多
5	访问方式错
6	非法文件号
11	非法格式
12	非法的访问码
13	非法数据
15	非法驱动器
16	企图删除当前目录

* 文件的建立、打开、关闭与删除：这组调用的功能与传统文件操作一样。（3CH）为建立文件的调用，使用 DS:DX 指出文件路径名，CX 表示要打开文件的属性，这些属性常用的含义有：

01　只读文件
02　隐含文件
04　系统文件

执行这条调用后，MS-DOS 建立并按读写方式打开文件，并回送分配的一个文件号给调用程序（在 AX 中），供以后的读写操作使用。打开文件的调用（3DH）使用 DS:DX 指出文件路径名，AL 指出访问方式（0 为只读，1 为只写，2 为读写）。返回的文件号也放于 AX 中。关闭文件的调用（3EH）仅需在 BX 中给出欲关闭文件的文件号。调用 41H 用来删除文件，DS:DX 指出欲删除的文件名。

* 文件读写：文件建立或打开后，可以使用读文件调用(3FH)和写文件调用（40H）来访问。这两条调用的入口参数为：

BX：文件号

CX：希望读写的字节数

DS:DX：缓冲区首址

读写完毕后，AX 中给出实际读写的字节数。这两个调用都从读写指针处开始读写，读写指针的初值为文件首字节。读写指针可通过调用 42H 来修改，修改的位移量放在 CX:DX中，修改方式由 AL 指定，共有三种方式：

AL=0　从文件头向下移动

AL=1　从当前指针向下移动

AL=2　从文件尾向上移动

修改之后，就可以使用于读写命令中。

下面是把文件 COPYEX1.COM 复制到文件 COPYEX2.COM中去的一个例子。

```
0905：0100  JMP  120
0905：0102  DB   'COPYEX1.COM' 00
0905：010E  DB   'COPYEX2.COM' 00  ；文件名字符串
0905：011A  DW   0 0 0
0905：0120  MOV  AX,  3D00
0905：0123  MOV  DX,  0102
0905：0126  INT  21                ；打开 COPYEX1.COM
0905：0128  MOV  BX,  AX
0905：012A  MOV  AH,  3F
0905：012C  MOV  CX,  8000
0905：012F  MOV  DX,  0200
0905：0132  INT  21                ；读入 COPYEX1.COM
0905：0134  MOV  BX,  AX           ；（必须小于 32K）
0905：0136  MOV  AH,  3C
0905：0138  MOV  DX,  010E
0905：013B  MOV  CX,  0000
0905：013E  INT  21                ；建立 COPYEX2.COM
0905：0140  MOV  CX,  BX
0905：0142  MOV  BX,  AX
0905：0144  MOV  AH,  40
0905：0146  MOV  DX,  0200
0905：0149  INT  21                ；写入 COPYEX2.COM
0905：014B  MOV  AH,  3E
0905：014D  INT  21                ；关闭 COPYEX2.COM
0905：014F  INT  20                ；返回操作系统
```

（3）目录管理

MS-DOS 提供一组目录操作供用户查找文件、更改目录以及建立、删除子目录。这组

调用的出错信息与新增的文件操作的出错返参类似，必要时请查阅表 2-12。

① 查找文件

调用 4EH 的功能为查找与 DS:DX 指定文件相匹配的第 1 个文件，这里的文件名允许是多义文件名。调用前还应将使用的属性放入 CX。调用后，文件参数在盘缓冲区中放置情况如下：

DTA 0—15	DOS内部使用
DTA 16—17	文件属性
DTA 18—19	文件建立的时间
DTA 1A—1B	文件建立的日期
DTA 1C—1F	文件长度（高位字节在后）
DTA 20—2C	文件名与类型名
DTA 2D	00

对于多义文件，还可以使用 4FH 号调用查找下一匹配的文件，如果找到下一个匹配的文件，则重填 DTA，如果未找到则返回出错号 18。

② 更改目录

调用 56H 可用来更改文件名，使用前把 DS:DX 指向旧路径名，ES:DI 指向新路径名。新旧路径名可以有不同的路径，这时旧文件被更名并移到新路径名所在的目录中。但新旧路径名前不允许有不同的盘符。

调用 43H 用来显示或修改文件的属性。如果把入口参数 AL 置成 00，则表示把指定文件的属性读入到 CX 中；如果 AL 置成 01，则把 CX 中的属性字写入到 DS:DX 指定的文件上去。

例如，程序：

```
0907：0100  JMP   110
0907：0102  DB    'MYFILE.DAT', 0, 0, 0, 0
0907：0110  MOV   AX, 4301
0907：0113  MOV   DX, 102              ；置字符串地址
0907：0116  MOV   CX, 02               ；隐含属性
0907：0119  INT   21
0907：011B  INT   20                   ；返回操作系统
```

用来把文件 MYFILE·DAT 变成隐含文件，以便保密。

③ 子目录操作

这组操作用来建立、删除子目录以及设置或询问当前目录。入口参数 DS:DX 用来指出目录路径名（对 39H、3AH 和 3BH）而言。

39H 号调用用来建立一个子目录。如果路径名的路径是存在的，则在路径的末端建立一个空目录。

调用 3AH 用来删除一个空目录，如果目录不空，则不进行任何操作，返回出错号 5。

调用 3BH 用以把当前目录改为 DS:DX 指出的目录路径名。

47H 号调用可以把当前目录的绝对路径名取到一块64个字节大小的内存中。调用前必须把DS:SI指向这块内存的首址。

4．软件中断和功能调用一览表

MS-DOS 中常用的软件中断有八条，现列出如下：

中　断	功　能	入口参数	出口参数
INT 20	程序正常退出		
INT 21	系统功能调用	AH=调用号 功能调用入口参数	功能调用出口参数
INT 22	结束退出		
INT 23	Ctrl-Break 退出		
INT 24	出错退出		
INT 25	读盘	AL=盘号 CX=读入扇区数 DX=起始逻辑扇区号 DS:BX=缓冲区地址	CY=1 出错 (CY是进位标志位)
INT 26	写盘	AL=盘号 CX=写盘扇区数 DX=起始逻辑扇区号 DS:BX=缓冲区地址	CY=1　出错
INT 27	驻留退出		

使用 MS-DOS 功能调用的过程为：把调用号放入 AH 中，设置入口参数，再执行软件中断 INT 21，最后分析返回参数。对同一功能有两个调用时，通常使用后一个，因为它往往比前者更为简便和灵活。

MS-DOS 2.00 版共有 75 条系统功能调用，其中 00H—2EH 是与 MS-DOS 1.10 版兼容的，而其中 00H到 24H 又是与 CP/M 操作系统类似的。

下面按照功能分成四组，把所有 MS-DOS 2.00 版的系统功能调用列出。

一、　关于设备 I/O 的功能调用

编号	功　能	入口参数	出口参数
01H	键盘输入字符		AL=输入字符
02H	显示器输出字符	DL=输出字符	
06H	直接控制台 I/O	DL=FF（输入） DL=字符（输出）	AL=输入字符

（续表一）

编号	功　　能	入口参数	出口参数
07H	直接控制台输入(无回显)		AL=输入字符
08H	键盘输入字符(无回显)		AL=输入字符
09H	显示字符串	DS:DX=缓冲区首址	
0AH	输入字符串	DS:DX=缓冲区首址	
0BH	检查标准输入状态		AL=00 无键入 AL=FF 有键入
0CH	清输入缓冲区并执行指定的标准输入功能	AL=功能号(01,06,07,08 或 0A)	
03H	串行设备输入字符		AL=输入字符
04H	串行设备输出字符	DL=输出字符	
05H	打印机输出字符	DL=输出字符	
0DH	初始化盘状态		
0EH	选择当前盘	DL=盘号	AL=系统中盘的数目
19H	取当前盘盘号		AL=盘号
1AH	置磁盘缓冲区	DS:DX=缓冲区首址	
1BH	取文件定位表(FAT)(当前盘)		DS:BX=盘类型字节地址 DX=FAT 表项数 AL=每簇扇区数 CX=每扇区字节数
1CH	取指定盘的文件定位表(FAT)	DL=盘号	DS:BX=盘类型字节地址 DX=FAT 表项数 AL=每簇扇区数 CX=每扇区字节数
2EH	置写校验状态	DL=0，AL=状态	
54H	取写校验状态		AL=状态
36H	取盘剩余空间数	DL=盘号	BX=可用簇数 DX=总簇数 CX=每扇区字节数 AX=每簇扇区数
2FH	取磁盘缓冲区首址		ES:BX=缓冲区首址

二. 有关文件操作的功能调用

编 号	功 能	入 口 参 数	出 口 参 数
29H	建立 FCB	DS:SI=字符串地址 ES:DI=FCB 首址 AL=0E 非法字符检查	ES:DI=FCB首址 AL=00 标准文件 AL=01 多义文件 AL=FF 非法盘符
16H	建立文件	DS:DX=FCB 首址	AL=00 成功 AL=FF 目录区满
0FH	打开文件	DS:DX=FCB首址	AL=00 成功 AL=FF 未找到
10H	关闭文件	DS:DX=FCB 首址	AL=00 成功 AL=FF 已换盘
13H	删除文件	DS:DX=FCB 首址	AL=00 成功 AL=FF 未找到
14H	顺序读一个记录	DS:DX=FCB 首址	AL=00 成功 AL=01 文件结束 AL=03 缓冲不满
15H	顺序写一个记录	DS:DX=FCB 首址	AL=00 成功 AL=FF 盘满
21H	随机读一个记录	DS:DX=FCB 首址	AL=00 成功 AL=01 文件结束 AL=03 缓冲不满
22H	随机写一个记录	DS:DX=FCB 首址	AL=00 成功 AL=FF 盘满
27H	随机读若干记录	DS:DX=FCB 首址 CX=记录数	AL=00 成功 AL=01 文件结束 AL=03 缓冲不满
28H	随机写若干记录	DS:DX=FCB 首址 CX=记录数	AL=00 成功 AL=FF 盘满
24H	置随机记录号	DS:DX=FCB 首址	
3CH	建立文件	DS:DX=字符串地址 CX=文件属性字	AX=文件号
3DH	打开文件	DS:DX=字符串地址 AL=0 读 AL=1 写 AL=2 读/写	AX=文件号
3EH	关闭文件	BX=文件号	

（续表二）

编　号	功　　能	入　口　参　数	出　口　参　数
41H	删除文件	DS:DX＝字符串地址	
3FH	读文件	BX＝文件号 CX＝读入字节数 DS:DX＝缓冲区首址	AX＝实际读出的字节数
40H	写文件	BX＝文件号 CX＝写盘字节数 DS:DX＝缓冲区首址	AX＝实际写入的字节数
42H	改变文件读写指针	BX＝文件号 CX:DX＝位移量 AL＝0　绝对移动 AL＝1　相对移动 AL＝2　绝对倒移	DX:AX＝新的指针位置
45H	复制文件号	BX＝文件号 1	AX＝文件号2
46H	强制复制文件号	BX＝文件号 1 CX＝文件号 2	CX＝文件号 1
4BH	装入一个程序	DS:DX＝字符串地址 ES:BX＝参数区首址 AL＝0 装入执行 AL＝3 装入不执行	
44H	设备文件 I/O 控制	BX＝文件号 AL＝0 取状态 AL＝1 置状态 DX AL＝2 读数据 AL＝3 写数据 AL＝6 取输入状态 AL＝7 取输出状态	DX＝状态

三、　关于目录操作的功能调用

编　号	功　　能	入　口　参　数	出　口　参　数
11H	查找第 1 个目录项	DS:DX＝FCB 首址	AL＝00 成功 AL＝FF 未找到
12H	查找下一个目录项	DS:DX＝FCB 首址	AL＝00 成功 AL＝FF 未找到
23H	取文件长度(结果在 FCBRR 中)	DS:DX＝FCB 首址	AL＝00 成功 AL＝FF 未找到

（续表三）

编　号	功　　能	入　口　参　数	出　口　参　数
17H	文件更名	DS:DX=FCB 首址 (DS:DX+17)=新名	
4EH	查找第1个文件	DS:DX=字符串地址 CX=属性	DTA
4FH	查找下一文件	DTA	DTA
43H	置/取文件属性	DS:DX=字符串地址 AL=0 取文件属性 AL=1 置CX=属性	CX=文件属性
57H	置/取日期和时间	BX=文件号 AL=0 读 AL=1 写 DX:CX	DX:CX=日期和时间
56H	文件更名	DS:DX=字符串地址 ES:DI=新名地址	
39H	建立一个子目录	DS:DX=字符串地址	CY=00 成功
3AH	删除一个子目录	DS:DX=字符串地址	CY=00 成功
3BH	改变当前目录	DS:DX=字符串地址	CY=00 成功
47H	取当前目录路径名	DL=盘号 DS:SI=字符串地址	DS:SI=字符串地址

四、　其它功能调用

编　号	功　　能	入　口　参　数	出　口　参　数
00H	退出用户程序并返回操作系统		
31H	终止用户程序并驻留在内存	AL=退出码 DX=程序长度	
4CH	终止当前程序并返回调用程序	AL=退出码	
4DH	取退出码		AX=退出码
33H	置/取Ctrl-Break检查状态	AL=00 取状态 AL=01 置状态 DL	DL=状态
25H	置中断向量	AL=中断类型号 DS:DX=入口地址	
35H	取中断向量	AL=中断类型号	ES:BX=入口地址
26H	建立一个程序段	DX=段号	

（续表四）

编　号	功　　能	入　口　参　数	出　口　参　数
48H	分配内存空间	BX＝申请内存数量	AX：0 分配内存首址 BX＝最大可用内存空间 (失败时)
49H	释放内存空间	ES＝内存始址	
4AH	修改已分配的内存空间	ES＝原内存始址 BX＝再申请的数量	BX＝最大可用空间（失败时）
2AH	取日期		CX：DX＝日期
2BH	置日期	CX：DX＝日期	AL＝00 成功 AL＝FF 失败
2CH	取时间		CX：DX＝时间
2DH	置时间	CX：DX＝时间	AL＝00 成功 AL＝FF 失败
30H	取 DOS 版本号		AL＝版号，AH＝发行号
38H	取国别信息	DS：DX＝信息区首址 AL＝0	

第三章　IBM PC的BASIC语言

第一节　概　述

BASIC 语言是一种通用的交互式计算机语言，它主要用于数值计算、数据处理以及教学与游戏。BASIC 语言简单而实用，深受广大计算机使用者的欢迎。

微型计算机上配置的 BASIC 语言，一般都具有以下几个特点：

* 语言成分简单明了，易于初学者学习和掌握。
* 人机通讯采用会话方式，易于编制和调试程序，使用者甚至可以直接在机器上编写程序，边写边调试直至满意为止。
* 有较强的数据处理能力，用于小型事务管理，既方便又灵活。
* 有丰富的作图语句和音响语句，可用于辅助教学和娱乐游戏。

鉴于 BASIC 语言的上述特点，它已得到了广泛的应用，并将经久不衰。目前，许多微型机已把 BASIC 语言固化在只读存贮器中，所以也有人把它叫做微型计算机的缺省语言。

IBM PC 微机已由美国著名的软件公司 MICROSOFT 公司配置了四个向上兼容的 BASIC语言，它们依次为：磁带 BASIC、磁盘 BASIC、高级 BASIC 和编译 BASIC。这四个版本的 BASIC 中，除磁带 BASIC 常驻于 32K ROM 中并可独立运行外，其余三个 BASIC 都存放在磁盘上且都必须在 MS-DOS 操作系统下运行。表 3-1 列出了这四个版本 BASIC 的功能及其差别。

表 3-1　四个向上兼容的 BASIC 语言

版　本	主　要　功　能
磁带BASIC	* 提供基本BASIC的所有功能 * 可显示 256 种不同的字符，除普通的 ASCII 字符外，还可显示图形符号、希腊字母、数学符号和其它特殊符号 * 支持图形显示器，可以直接使用特定的语句作图 * 支持扬声器、光笔和游戏操纵杆等设备，交互能力较强
磁盘BASIC (BASIC.COM)	* 提供磁带 BASIC 的所有功能 * 支持磁盘文件系统，可使用磁盘记录用户的程序和数据 * 设置了一个内部时钟，可供读写日期和时间 * 支持异步通讯器和两个附加的打印机
高级BASIC (BASICA.COM)	* 提供磁盘 BASIC 的所有功能 * 提供了事件陷井的设置功能，允许用户捕俘键盘、异步通讯器等设备发出的中断 * 增加了几条作图语句和音乐语句

（续 表 3-1）

版　本	主　要　功　能
编译BASIC (BASCOM.COM、BASRUNG.EXE等)	* 具有高级 BASIC 的大部分功能 * 使用的方式为编译方式 * 执行速度比解释 BASIC 快三至十倍 * 比解释 BASIC 节省内存空间 * 为用户程序提供了保密措施

在这一章中，主要介绍 PC BASIC 对基本 BASIC 的扩充部分，如果读者不熟悉基本 BASIC 的内容，请先阅读有关的书籍。

大家知道，学习程序设计语言的最好方法是使用程序设计语言。如果条件允许，在阅读这一章的同时，最好能直接在机器上运行书中的例子（一般要在 BASICA 2.00下运行），或运行自己编制的程序。这样，对掌握这门语言是有帮助的。

1. BASIC 语言的基础

在介绍 BASIC 语言的基本成分之前，首先看一个用 PC BASIC 写的一个打印质数的例子。通过阅读这个简单的例子，读者可以对 IBM PC 微型机上的 BASIC 程序的结构有个直观的了解。

例3—1　打印前 29 个质数

```
100  ' EX3-1 Prime Numbers Program
110  N=29
120  PRINT : PRINT "The firs " ; N; "prime numbers are : "
130  PRINT : PRINT " " , 2, 3,
140  I=3
150  FOR J=1 TO N-2
160  I=I+2
170  K=3
180  Q=I\K  ' Integer Division
190  R=I-Q*K
200  IF R=0 THEN 160
210  IF K>=Q THEN 240
220  K=K+2
230  GOTO 180
240  PRINT I,
250  NEXT
260  END
```

程序运行的结果如下：

The first 29 prime numbers are:

	2	3	5	7
11	13	17	19	23
29	31	37	41	43
47	53	59	61	67
71	73	79	83	89
97	101	103	107	109

从上例可以看出：BASIC 程序一般由若干行语句组成，而每条语句又由行号、语句关键字和操作对象等部分组成。下面扼要地介绍 BASIC 语言的数据类型和语句类型。

（1）数据类型及其表示

BASIC 语言使用的数据类型主要有整型、实型、长实型、字符串以及数组，前四种类型属于简单类型，数组类型为复合类型。

① 简单类型

PC BASIC 一般使用特殊的变量尾标来区分变量的类型，若不加尾标，BASIC 就当作实型变量处理。几种简单变量的表示范围和变量尾标分述如下：

* 整型变量的表示范围为 −32768 到 +32767，在内存中占两个字节，变量尾标为“%”。整型常数又分为十进制数、十六进制数和八进制数三种，其中十六进制数和八进制数前分别冠以&H和&O（或&），例如：

  &HFF，&77，98

  分别表示十六进制数 FF、八进制数 77 和十进制数 98。
* 实型变量可表示具有 7 位有效数字的实数，在内存中占 4 个字节，变量尾标为“!”。采用科学记数法表示的实型常数用字母 E 表示乘方底数。例如：

  46.8，−1.09E−06，22.5!

  都表示实型常数。
* 长实型变量则可表示具有 16 位有效数字的实数，内存中占 8 个字节，变量尾标为“#”。为了与实型数相区别，乘方底数表示为 D。例如：

  34567890，　−1.09D−06，　3.0#

  都是长实型数。
* 字符串变量用来表示一串字符，它的尾标为“$”，它的长度为 0 到 255 个字符。字符串常量为用双引号括起来的一串字符。

除了使用变量尾标可以区分变量类型之外，还可使用定义类型语句来限定变量的类型，语句的格式如下：

〈行号〉　DEF〈类型〉　〈字母〉[−〈字母〉]{，〈字母〉[−〈字母〉]}

其中类型可以是 INT、SNG、DBL 或 STR，分别表示以指定字母打头的变量为整型、实型、长实型或字符串型。下文的例 3—2 就是使用这条语句定义变量类型的。

② 数组类型

数组由一组同类型的数或字符串组成，可使用下标变量访问。数组在使用前一般都要用

DIM 语句说明，DIM 语句的格式为：

DIM 〈变量〉(〈下标表〉){，〈变量〉(〈下标表〉)}

下标表的维数最多可达 255，每维元素则最多允许 32767 个。

如果不经 DIM 语句定义而直接使用数组，则 BASIC 认为它的维数最多为 4，而下标最多只能到10。如使用维数大于 4 或下标值大于 10 的下标变量，BASIC将给出“下标溢出”出错信息。

下标的起始值一般为 0，但可使用语句

OPTION BASE 1

把下标起始值改为 1。

（2）数据的运算与处理

对于每种数据类型，BASIC 都提供了一组运算符和一组标准函数。关于字符串的处理在第二节再作专门讨论。而 PC BASIC不允许直接对数组进行运算，所以这里只介绍数值变量的运算与处理。

对数值变量（整型、实型和长实型）一般可施行以下五类操作：

* 算术运算：+，−，*，/，\（整除），^（乘方）和MOD（取模）。
* 关系运算：=，<>，>，<=和>=。
* 逻辑运算：NOT，AND，OR，XOR，EQV（恒等）和IMP（蕴涵）。
* 内部函数：见表 3-2。

表 3-2　PC BASIC 的内部函数

函　数	功　能	举　例
CDBL(X)	转换成长实型数	CDBL(454.67)=454.6699829101563
CSNG(X)	转换成实型数	CSNG(454.66998)=454.6700
CINT(X)	舍入成整型数	CINT(45.67)=46,CINT(−2.89)=−3
FIX(X)	截断成整型数	FIX(45.67)=45, FIX(−2.89)=−2
INT(X)	取<=X的最大整数	INT(45.6)=45, INT(−2.89)=−3
ABS(X)	取X的绝对值	ABS(−35)=35
SQR(X)	取X的平方根	SQR(10)=3.162278
EXP(X)	指数函数	
LOG(X)	对数函数	
SIN(X)	正弦函数	
COS(X)	余弦函数	
TAN(X)	正切函数	
ATN(X)	反正切函数	
SGN(X)	符号函数	SGN(0)=0, SGN(−20)=−1
RND(X)	0—1间的随机数	

* 自定义函数：使用语句

〈行号〉 DEF FN〈函数名〉[〈参数表〉]=〈表达式〉

定义的函数。例如：

```
10 DEF FNARCSIN(X)=ATN(X/SQR(1-X*X))
```

定义了反正弦函数arcsin(x)。

(3)语句及其分类

语句是BASIC程序的组成和执行单位。BASIC程序在执行时，通常依语句的行号从小到大逐条执行，直至END语句为止。PC BASIC中提供了80多种不同的语句，它们按功能可分为赋值、控制、I/O、说明和专用语句等五类。

① 赋值语句

赋值语句可用来给变量直接提供数据，它的格式是：

〔LET〕<变量>=<表达式>

其含义为：把右部表达式的值赋予左部的变量。

此外，PC BASIC还提供了一条交换语句，可用来交换两个变量的值，例如：

```
10 SWAP A, B
```

等效于：

```
10 TEMP=B
20 B=A
30 A=TEMP
```

② 控制语句

控制语句用以改变程序的正常执行顺序。常见的有无条件转移语句、条件语句、循环语句和转子语句。

PC BASIC有以下几条控制语句：

* IF…THEN…ELSE	条件语句
* GOTO	无条件转移语句
* GOSUB…RETURN	转子与返回语句
* FOR…TO…STEP	步长循环语句
* WHILE…WEND	当循环语句

此外，PC BASIC还提供了程序链接语句、情况语句和中断陷井语句来控制程序的执行顺序。这几条语句将在第四节中介绍。

③ 输入输出语句

在BASIC程序的执行过程中，有时需要用户提供一些数据，有时又要把程序执行的结果输出出来，这些功能都是通过输入输出语句实现的。PC BASIC可以支持多种外设，故其输入输出语句也特别丰富。

PC BASIC中最常见的输入输出语句有以下几条：

* READ…DATA	读数与置数语句
* INPUT	键盘输入语句

* PRINT　　显示器输出语句
* LPRINT　　打印机输出语句
* LINE INPUT　　键盘行输入语句

除此以外，在第二节中将要介绍的格式输出语句，第三节中的文件I/O语句以及第五节中的扬声器和异步通讯器I/O语句，也都是输入输出语句。

④ 说明语句

PC BASIC的说明语句主要有以下几种：

* REM 或 ’　　注释语句
* DIM　　数组说明语句
* DEF〈类型〉〈字母范围〉　　变量类型限制语句
* DEF FN〈函数名〉=〈表达式〉　　自定义函数语句
* DEF SEG=〈地址〉　　定义代码段地址语句
* COMMON　　公共变量定义语句
* KEY n, x$　　功能键定义语句

⑤ 专用语句

这组语句可用来完成一些特殊的初始化工作。较常用的有以下四个：

* RESTORE n　　初始化DATA数据区
* RANDOMIZE　　初始化随机数序列
* DATE$=x$　　置日期
* TIME$=x$　　置时间

（4）几个例子

下面再举几个例子，通过这些例子可以大概地看出PC BASIC的特点。

例3—2　计算三个数的平均值

```
10 ' EX3-2 Print the average of 3 numbers
20 DEFINT I-K, N
30 DEFSTR S
40 DEFDBL A-D
50 READ D1, D2, D3
60 AVERAGE= ( D1+D2+D3 ) /3
70 STRING= " numbers. Average is"
80 NUM=10/3
90 PRINT NUM ; STRING ; AVERAGE
100 DATA 1, 0.5, 0.5
110 END
```

程序运行后的结果为：

```
 3 numbers. Average is .6666666666666667
```

说明：20行限定以字母I--K和N打头的变量为整型变量。30行限定以S打头的变量为字符串变量。40行则表示以A—D开头的变量为长实型变量。50行READ语句把100行DATA语句中的数据分别读到D1、D2和D3中。60行计算平均值。90行输出计算结果。

例3—3 打印电话号码表

```
100 ' EX3-3 Telephone Directory Program
110 PRINT TAB(6); "Telephone Directory"
120 PRINT : PRINT "Unit", "Room", "Extension"
130 PRINT : NUM=0
140 READ N$, R$, T
150 WHILE N$<>"END"
160 NUM=NUM+1
170 PRINT N$, R$, T
180 READ N$, R$, T
190 WEND
200 PRINT : PRINT "Number of Entries"; NUM
210 DATA The Office, 2-09, 2345
220 DATA Centre Lab, 1-02, 3232
230 DATA MCS Lab, 2-08, 4546
240 DATA END, ,
250 END
```

程序运行的结果为：

```
    Telephone Directory
Unit          Room        Extension

The Office    2-09        2345
Centre Lab    1-02        3232
MCS Lab       2-08        4546

Number of Entries 3
```

说明：110—120行打印表头。140行读入第1行数据。150—190行生成和打印表格主体，使用当循环语句实现。200行则打印项数。210—240行为数据。

例3—4 猜数游戏

```
10 ' EX3-4 Guess my number -- A computer game
20 RANDOMIZE TIMER
30 GOSUB 120
40 PRINT "I'm thinking of a number from 1 to 100."
```

```
 50 PRINT "Guess my number! "
 60 PRINT : INPUT "Your guess" ; G
 70 IF G<X THEN PRINT "Try a bigger number." : GOTO 60
 80 IF G>X THEN PRINT "Try a smaller number." : GOTO 60
 90 IF G=X THEN PRINT "That's it! You've guessed my number."
100 END
110 ' Generate a random number
120 X=INT(100 * RND(1))+1
130 RETURN
```

说明：20行初始化随机数序列。30行调用120—130行随机数生成子程序。40—60行给出提示信息并要求猜测计算机“想”的一个1到100间的数。70—90行分析所输入的数，若正确则显示出：“That's it! You've guessed my number.”（对！你猜中我“想”的数了）。否则，需要用一个更大的或更小的数试验。程序执行的结果如下：

```
I'm thinking of a number from 1 to 100.
Guess my number!
Your guess? 50
Try a smaller number.
Your guess? 30
Try a smaller number.
Your guess? 20
Try a bigger number.
Your guess? 25
Try a smaller number.
Your guess? 23
That's it!  You've guessed my number.
```

2. 源程序的准备与程序文件的管理

在准备源程序以前，先要调入BASIC的解释程序（假设使用高级BASIC 2.00）。调入的方法是在引入MS-DOS操作系统后，打入：

```
A>BASICA<CR>
```

BASIC的解释程序就调入了内存并显示出：

```
The IBM Personal Computer Basic
Version A2.00 Copyright IBM Corp. 1981, 1982, 1983
60865 Bytes free
OK
```

其中，第2行中的Version A2.00表示使用的BASIC是高级BASIC 2.00版。最后出现

的“OK”是BASIC的提示符，表示可以开始与计算机对话了。

（1）全屏幕行编辑

在运行一个程序以前，首先要用编辑程序建立和修改源程序。PC BASIC提供了一种功能较强的编辑程序，它允许用户使用编辑键编辑和修改屏幕上的任意行。编辑以行为单位，只有在输入〈CR〉键后，编辑工作才有效。由于编辑以行为单位，且允许在全屏幕范围内进行，故称为全屏幕行编辑。表3-3列出了PC BASIC的常用编辑键。

表3-3　BASIC的编辑键

编辑键	功　　能
↑	光标上移一行
↓	光标下移一行
←	光标左移一符
→	光标右移一符
Ins	插入若干字符
Del	删除光标处的一个字符
BS	反向删除一符
Esc	删除一行
Home	光标回左上角
Ctrl-Home	光标回左上角并清除屏幕
End	光标移到行末
Ctrl-End	从光标处删到行末
Ctrl-→	光标右移一个单词
Ctrl-←	光标左移一个单词
Ctrl-Break	终止编辑，不作任何修改
→| (Tab)	光标移到下一个制表位置

（2）编辑命令

为了配合编辑键对源程序编辑，PC BASIC提供以下几条编辑命令：

* LIST 〔〈行号〉—〈行号〉〕〔，“文件名”〕　列出程序清单到指定的文件或显示器上。
* LLIST 〔〈行号〉—〈行号〉〕　从打印机上列出程序清单。
* AUTO 〔〈行号〉，〈增量〉〕　自动编行号。
* RENUM 〔〈新行〉,〈旧行〉,〈增量〉〕　重新编行号。
* EDIT 〈行号〉　编辑指定的行。
* DELETE 〔〈行号〉—〈行号〉〕　删除若干行。
* NEW　清除内存中的程序。

（3）源程序文件的管理

PC BASIC还提供一组命令，用以管理源程序文件。它们分别为：

* FILES“路径名”　显示文件目录。

* LOAD “文件路径名” 装入一个源程序文件。
* SAVE “文件路径名”〔,A〕〔,P〕 把内存中的程序保存到盘上。其中“,A”表示以 ASCII 码形式保存；而 “,P” 则表示以密码形式存盘。
* NAME “老文件名” AS “新文件名” 更换一个文件名。
* KILL “文件路径名” 删除文件。
* MKDIR “目录路径名” 建立一个子目录。
* CHDIR “目录路径名” 改变当前目录。
* RMDIR “目录路径名” 删除一个空目录。
* MERGE “文件路径名” 装入一个 ASCII 码形式的源程序文件，并与内存中原有程序合并。

3. BASIC 程序的运行、控制与调试

（1）启动运行方式

BASIC 程序共有三种启动运行方式。下面分别叙述。

其一是使用 RUN 命令运行。RUN 命令的使用格式如下：

RUN〔<行号>〕

RUN “文件名”〔, R〕

第 1 式当程序已在内存时使用；第 2 式则当程序在盘上时使用，若选 “, R” 开关，表示运行前不关闭已打开的文件。

其二是使用 LOAD 命令。可以先用 LOAD 命令装入程序，再用 RUN 命令运行，也可直接使用命令

LOAD “文件名”, R

运行程序。这样的方式类似于 RUN “文件名” 命令。

运行程序的最后一种方式是使用 BASIC 命令行。在 MS-DOS 下，BASIC 解释程序是作为命令来执行的，它的格式为：

BASICA〔<路径名>〕〔<I/O 重定向>〕〔/F:n〕〔/S:n〕〔/C:n〕〔/M:〔n〕〔, n〕〕〔/D〕

其中，路径名为进入 BASIC 状态后立即装入执行的程序文件名。重定向 I/O 可用数据文件代替键盘和显示器作为 I/O 设备。例如：

A>BASICA MYPROG<MYIN.DAT>MYOUT.DAT

表示装入 BASIC 后，立即执行程序 MYPROG.BAS，并从文件 MYIN.DAT 中取出数据作为键盘输入，显示器的输出则送入文件 MYOUT.DAT。

BASIC 命令行的开关可用来改变 BASIC 的运行环境。其中 /F：n 表示可同时打开的数据文件的个数为 n，n 的最大值为 15，如果这个开关缺省表示同时打开的文件个数不超过 3个。/S：n可用以设置文件缓冲区长度。/C：n则用以设置异步通讯器的缓冲区长度。开关 /M 可用以设置内存。开关 /D 表示函数 SIN，COS，TAN，ATN，EXP，LOG 和 SQR 使用长实型量进行计算。

退出 BASIC 使用 SYSTEM 命令来完成。

（2）运行控制

在程序的运行过程中，可以使用控制键进行人工干预。常用的控制键有以下几个：

* Ctrl-Break　　终止程序的执行或退出 AUTO 状态。
* Ctrl-NumLock　　显示器暂停输出数据。
* Ctrl-PrtSc　　接通/断开打印机。
* Alt-Ctrl-Del　　系统总清。

（3）调试

调试手段对于有效地定位和排除程序中的故障具有十分重要的意义。对于一些不易由静态观察出来的错误，一般可由动态调试查出。

PC BASIC 设有逐行跟踪和设置断点等调试设施，使用起来颇为方便。有关命令如下：

* TRON　　打开逐行跟踪。
* TROFF　　取消逐行跟踪。
* STOP　　暂停语句，可用于设置断点。
* CONT　　从断点继续执行下去。

例如，使用命令 TRON 跟踪例 3—3 产生的结果如下：

```
〔100〕〔110〕
     Telephone Directory
〔120〕
Unit          Room          Extension
〔130〕
〔140〕〔150〕〔160〕〔170〕 The Office        2—09        2345
〔180〕〔190〕〔160〕〔170〕 Centre Lab        1—02        3232
〔180〕〔190〕〔160〕〔170〕 MCS Lab           2—08        4546
〔180〕〔190〕〔200〕
 Number of Entries 3
〔210〕〔220〕〔230〕〔240〕〔250〕
```

4. BASIC的功能键

为方便用户，PC BASIC 提供两组功能键用来表示常用的一些语句名和命令。使用这两组键，可加快程序的编制和调试。

（1）语句功能键

语句功能键由〈Alt〉键加上语句的第 1 个字母组成，它们的含义如表 3-4 所示。

（2）命令功能键

命令功能键为〈F1〉到〈F10〉的十个功能键，通常它们的含义如表 3-5所示。

表 3-4　语句功能键

功能键	含　义	功能键	含　义
Alt-A	AUTO	Alt-M	MOTOR
Alt-B	BSAVE	Alt-N	NEXT
Alt-C	COLOR	Alt-O	OPEN
Alt-D	DELETE	Alt-P	PRINT
Alt-E	ELSE	Alt-R	RUN
Alt-F	FOR	Alt-S	SCREEN
Alt-G	GOTO	Alt-T	THEN
Alt-H	HEX$	Alt-U	USING
Alt-I	INPUT	Alt-V	VAL
Alt-K	KEY	Alt-W	WIDTH
Alt-L	LOCATE	Alt-X	XOR

表 3-5　命令功能键

功能键	含　义	功能键	含　义
F1	LIST	F6	,"LPT1:" ‹CR›
F2	RUN‹CR›	F7	TRON‹CR›
F3	LOAD"	F8	TROFF‹CR›
F4	SAVE"	F9	KEY
F5	CONT‹CR›	F10	SCREEN 0,0,0‹CR›

这十个功能键可以用 KEY 语句重新定义，KEY 语句的格式为：

KEY　n, x$

其中 n 为 1—10，x$ 为欲定义的字符串。

第二节　字符串处理与报表生成

随着计算机性能的不断提高，计算机处理的对象已从单一的数值数据拓宽为数据、文字、图形、声音乃至于实时物理量。在微型计算机的主要应用领域之一——事务管理中，除了处理大量的数值数据外，还要加工大量的文字信息，处理后的结果又常要以报表的形式输出。这一节主要介绍字符串处理技术及其在报表生成中的应用。

1．字符串与字符串变量

（1）字符串

通常的文字在计算机中一般使用字符串来表示。所谓字符串是指用双引号括起来的一组有序的字符。例如：

```
"HELLO"
"$25,000.00"
"Number of employees"
""
```

都是正确的字符串，其中“”是一个特殊的字符串——空字符串。

字符串中的每个字符在内存中均占一个字节，而字符串的长度为0到255，所以一个字符串视其长度占0—255个字节的内存。PC BASIC的每个字符都以其对应的ASCII码的形式存放，共允许使用256种字符。

（2）字符串变量

在第一节中曾指出，字符串变量有两种定义形式，其一是采用变量尾标（$），其二是使用DEFSTR语句。下文中主要采用第1种形式。

PC BASIC的字符串的长度限定在255以内，它的实际长度可用函数LEN$求出。例如：

```
OK
X$= "Number of Employees"
OK
PRINT LEN(X$)
19
OK
```

注意，字符串中的空格也算一个字符。

PC BASIC允许使用字符串数组，这对于处理字符串型表格特别有益。一般说来，使用一个两维字符串数组就可以表示出一张表格。用这种方法表示的表格易访问、易加工。如何使用字符串数组表示一张通讯录，可用例3—5来说明。

例3—5 输入一份通讯录。通讯录由200行组成，每行又分为年龄、地址和电话三项。

```
10 'EX3-5 Input the Address Book
20 DIM ADDRBK$(199, 2)
30 PRINT " ==== Input the Address Book ===="
40 PRINT " Name, Address, Phone-number<CR>"
50 FOR I=0 TO 199
60 INPUT ":", ADDRBK$(I,0), ADDRBK$(I,1), ADDRBK$(I,2)
70 NEXT
80 END
```

程序运行的部分结果如下：

```
==== Input the Address Book ====
Name, Address, Phone-number<CR>
```

: WANG XING, "29, SHANGHAI Road, NANJING", 43259
: ZHAO MING, "2, RENMING Road, NANJING",45245

说明：20行定义一个两维字符串数组，用以存放通讯录。30—40行输出提示信息。50—70行输入通讯录数据。

（3）字符串的输入

字符串在赋值以前为空。字符串取得新值一般有三种方法：第1种是使用赋值语句，第2种是使用READ和DATA语句，第3种则是使用键盘输入语句。下面分别介绍。

对于程序运行前已确定其值的字符串变量，数量较少时直接用赋值语句赋值，较多时则采用READ和DATA语句取值。

对于程序运行前还不能确定的字符串变量，通常采用键盘输入方式赋值，例3—5就是使用INPUT语句输入字符串的。

除了INPUT语句外，还可用LINE INPUT语句输入字符串，该语句的格式为：

LINE INPUT〔;〕〔"提示" ;〕<串变量>

这条语句用来输入以<CR>结尾的一行字符，不计逗号等间隔符。

如果仅需输入一个字符且不回显，则可使用语句

D$=INKEY$

来完成。如要输入n个字符而不回显，则用语句

D$=INPUT$(n)

输入。

2．字符串的处理

和数值变量一样，串变量也可以进行比较、运算和转换。关于字符串的几种操作和处理介绍如下。

（1）字符串的比较

字符串的比较操作是根据其对应的ASCII码的大小进行的，例如：

"ABCD" < "ABCE"
"FILENAME" = "FILENAME"
"SMYTHE" > "SMYTH"

除了<、=和>外，还可用<>、<=和>=进行比较。

字符串的比较操作常用于从一组字符串中检索出所需的字符串。下面的例子说明如何使用字符串比较操作检索特定的字符串。

例3—6 在例3—5建立的通讯录中查出某人的地址

```
100 'EX3-6 Search a line from the Address Book
110 INPUT "Name" ; ANAME$
```

```
120 FOR I=0 TO 199
130 IF ANAME$=ADDRBK$(I,0) THEN 170
140 NEXT
150 PRINT "Not found! "
160 GOTO 190
170 PRINT : PRINT "---- Address Book ----"
180 PRINT ANAME$, ADDRBK$(I,1), ADDRBK$(I,2)
190 END
```

程序的运行结果如下：

```
Name? WANG XING

---- Address Book ----
WANG XING     29, SHANGHAI Road, NANJING     43256
```

说明：110行请求输入需查找的人的姓名。120—140行在通讯录中查找，若未找到则打印出："Not found! "（未找到），否则打印出被查人的地址和电话号码。

（2）字符串的运算

对于字符串，BASIC仅提供了一个运算，即并置（+）运算。这个运算用于把两个字符串合并，并形成一个新的字符串。例如，B盘上名为FILENAME$的BASIC程序文件可以表示为：

```
F$= "B:"+ FILENAME$ +"·BAS"
```

应当注意，PC BASIC允许对字符串进行一种类似于算术累加的运算——"累置"。例如，执行程序

```
10 TYPE$= "IBM"
20 TYPE$=TYPE$+"Personal"
30 TYPE$=TYPE$+"Computer"
40 PRINT TYPE$
```

的结果为：

```
IBM Personal Computer
```

（3）字符串函数

一般来说，BASIC都提供一组字符串加工函数，它们可以用来取子字符串或者生成一个特殊的字符串。PC BASIC提供的字符串加工函数主要有以下几个：

* LEFT$(x$, n)　取x$中最左的n个字符。
* RIGHT$(x$, n)　取x$中最右的n个字符。

* MID$(x$, n, m)　取x$中从n开始的m个字符。
* SPACE$(n)　产生一个由n个空格组成的字符串。
* STRING$(n, m)　产生一个由n个ASCII码为m的字符组成的字符串。
* STRING$(n, x$)　产生一个由n个x$中第1个字符组成的字符串。
* INSTR(n, x$, y$)　从第n个字符开始对x$进行检查，若有y$子串存在则返回其位置，否则返回0。n缺省时表示0。

其中最后一个函数INSTR，有时可用于从一个字符串中检索出若干个字符。例3—7是利用INSTR函数来检索汉字编码的。

例3—7　汉字编码检索程序

① 功能：假设127个汉字的国标码，以每码两符的形式存放在字符串CODETAB$中，输入的汉字已转化成国标码放于CODE$中。程序的功能是在CODETAB$中查找，看有无与CODE$中代码一致的代码，若有在变量NUM中给出序号，否则NUM为0。

② 程序：（以子程序的形式给出）

```
10 ' EX3-7 Search for a Chinese Character
50 NUM=INSTR(CODETAB$, CODE$)
60 IF (NUM MOD 2)=1 THEN 100
70 IF NUM=0 THEN PRINT "Not found! " :GOTO 110
80 NUM=INSTR(NUM+1, CODETAB$, CODE$)
90 GOTO 60
100 NUM=(NUM+1)/2
110 RETURN
```

③说明：50行在CODETAB$中查找CODE$。60行判断NUM是否为奇数，若是，则转100行计算序号，否则还要继续查找（因为国标码以两个字节为一个单位，NUM偶数时为误码，须重查）。70行表示未查到时打印出："Not found! "（未找到）并返回主程序。80行为误码的重新检索。100行为计算查到的汉字的序号。

（4）字符串与数值量之间的转换

字符串与数值量可以通过两对函数进行相互转换。第1对为字符与其对应的ASCII码的转换，另一对则为字符串与其对应的数值数据之间的转换。

* ASC(x$)　求出x$中第1个字符所对应的ASCII码。例如：

```
PRINT ASC( "ABC" )
PRINT ASC( "TEST" )
```

的结果分别是65和84。

* CHR$(n)　求出ASCII码为n的字符，n的取值范围为0—255。这个函数可用来产生一些特殊的字符，例如：

CHR$(65)　表示字符A
CHR$(13)　表示回车

CHR$(7)　　表示使扬声器发出鸣声的字符

CHR$(224)　　表示希腊字母α

* VAL(x$)，把数字形式的字符串转换成数值量，如x$中的第1个字符不为数字或加减号，则函数VAL(x$)的值为0。例如：

```
VAL( "-3" )
```

的值为-3。

* STR$(V)　把数值量转换成对应形式的字符串，也可用来测试数字的位数。例如：

```
200  INPUT "TYPE A NUMBER"; N
210  PRINT "The digits of";N; "is" ;LEN(STR$(N))-1
```

运行的结果为：

```
TYPE A NUMBER? 1234<CR>
The digits of 1243 is 4
```

下面是利用上述函数编制的一个程序。

例3—8　把阿拉伯数转换为罗马数

① 功能：把1—1999之间的任意一个阿拉伯数转换为对应的罗马数。

② 程序：

```
10  ' EX3-8 Convert a decimal number to a Roman numeral
20  INPUT "Number, not over 1999" ;A:X1%=0
30  IF A>1999 OR A<1 THEN 10
40  A$=STR$(A):B$=" ":A$=RIGHT$(A$, LEN(A$)-1)
50  FOR X%=LEN(A$) TO 1 STEP -1
55  X1%=X1%+1
60  GOSUB 100
70  B$=B$+C$
80  NEXT
90  PRINT "The answer is" ; B$
95  END
100 D$=MID$(A$, X1%, 1)
110 ON X% GOSUB 300, 310, 320, 330
120 ON VAL(D$)+1 GOTO 130, 140, 150, 160, 170, 180, 190, 200,
    210, 220
130 C$="":RETURN
140 C$=S$:RETURN
150 C$=S$+S$:RETURN
```

```
160 C$=S$+S$+S$ : RETURN
170 C$=S$+L$ : RETURN
180 C$=L$ : RETURN
190 C$=L$+S$ : RETURN
200 C$=L$+S$+S$ : RETURN
210 C$=L$+S$+S$+S$ : RETURN
220 C$=S$+V$ : RETURN
300 S$= "I" :L$= "V" : V$= "X" : RETURN
310 S$= "X" : L$= "L" : V$= "C" : RETURN
320 S$= "C" : L$= "D" : V$= "M" : RETURN
330 S$= "M" : RETURN
```

③ 说明：20—30行输入阿拉伯数，若超出处理范围则要求重输。40行中的第1个语句把输入的数转换成对应的字符，第3句用来除去A$中开头的空格。50—80行用来由高位到低位逐位把阿拉伯数字转换成罗马数字，其中60行调用的子程序为100—330行的1位数字的转换程序。90行输出转换的结果。从100行开始为子程序，100行从A$取出1位数字，110行根据该位的权，分别调用300到330行的初始化程序，120行根据该数位为0—9之一分别处理。程序的运行结果如下：

```
Number, not over 1999? 29<CR>
The answer is XXIX
```

3．报表生成

在数据处理类的应用中，经常需要把处理的结果以报表的形式输出。下面分别讨论如何利用BASIC的输出语句打印报表以及报表的生成过程。

（1）输出语句

最常用的输出语句是显示器输出语句PRINT，这条语句的格式为：

```
PRINT〔<表达式表>〕
或    ?〔<表达式表>〕
```

其中，表达式表是一系列由分隔符隔开的数值或字符串表达式，如表达式表省略则仅输出一个<CR>。

表达式表中使用的分隔符共有三个：逗号（，）、分号（；）和空格。其中 逗号（，）表示下面要打印的内容从下一打印区段开始（PC BASIC中规定每14列为一个打印区段），分号或空格均表示后面的输出紧接着前面的输出内容，且不留间隔。若PRINT语句的最末字符为间隔符，则下一个输出语句仍将在这一行中输出。

如果输出的是算术表达式，则输出的数据之前留符号位，其后还要留一个空格。

例3—9 设有五道选择题，每题可选择1到4的四种答案，统计出每人答对的题数。

其程序和运行结果为：

```
10 ' EX3-9 Compute the number of right answers
20 DIM KY(4), ANS(4)
30 PRINT "Question #",
40 FOR I=1 TO 5 : PRINT I; : NEXT : PRINT
50 PRINT "KEY",
60 FOR I=0 TO 4 : READ KY(I) : PRINT KY(I); : NEXT : PRINT
70 READ NAME1$ : IF NAME1$= "END" THEN END
80 PRINT NAME1$,
90 FOR I=0 TO 4 : READ ANS(I) : PRINT ANS(I); : NEXT
100 PRINT
110 NUM=0
120 FOR I=0 TO 4
130 IF KY(I)=ANS(I) THEN NUM=NUM+1
140 NEXT
150 PRINT NUM; "answers are correct! "
160 GOTO 70
200 DATA 3, 3, 4, 2, 1
210 DATA CHANG, 1, 4, 2, 2, 1
220 DATA LI, 2, 3, 3, 1, 2
230 DATA WANG, 3, 1, 4, 2, 1
240 DATA END
RUN
Question # 1 2 3 4 5
KEY       3 3 4 2 1
CHANG     1 4 2 2 1
2 answers are correct!
LI        2 3 3 1 2
1 answers are correct!
WANG      3 1 4 2 1
4 answers are correct!
OK
```

输出语句除PRINT外，还有WRITE和LPRINT语句。LPRINT语句的格式与PRINT语句一样，只是输出到打印机。WRITE语句类似PRINT语句，也输出到显示器，但要输出表达式表中的间隔符，字符串的双引号也要与它们的值一起输出。例如：

```
10 A=80 : B=90 : C$= "That's all."
20 WRITE A, B, C$
```

的输出结果为：

```
80, 90, "That's all."
```

（2）自选格式输出语句

PC BASIC具有几种格式输出语句，使输出格式灵活多样、规范整齐。常用的格式输出语句有三种：

① 空格语句PRINT SPC（n）

这条语句用来在显示器上显示出n个空格，n的取值范围为0至255，如果n超过实际屏幕的宽度，则输出（n MOD 屏宽）个空格。

SPC（n）函数只能控制两个输出内容之间的相对位置，要决定它们之间的绝对位置，则要使用TAB（n）函数。

② 制表语句PRINT TAB（n）

该语句把打印指针移到绝对位置的n列，如当前打印位置已大于n，则打印指针移到下一行的这个位置上。

TAB（n）后常跟随一个分号再加一个字符，用于输出一个由字符组成的曲线或图案。下例就是使用TAB（n）函数输出的一幅图案。

例3—10　输出一幅金字塔图案

该例的程序如下：

```
10 ' EX3-10 Draw Pyramid
20 CLS
30 COLUMN=20
40 FOR COUNT=1 TO 21 STEP 2
50 PRINT TAB(COLUMN);
60 FOR STARS=1 TO COUNT
70 PRINT "^";
80 NEXT
90 PRINT
100 COLUMN=COLUMN-1
110 NEXT
120 END
```

程序运行的结果见下页。

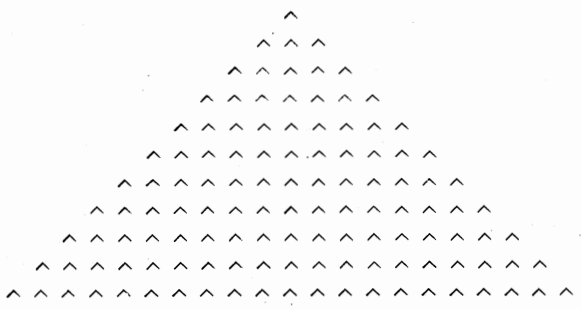
③ 格式输出语句PRINT USING

BASIC还允许使用自选格式输出，自选格式输出语句的格式为：

PRINT USING 〈格式串〉;〈表达式表〉

格式串为一个字符串变量（或字符串常量），其中包含各种格式符号以及需直接输出的字符串。

格式输出语句在报表输出中十分有用，它可以用来决定数值数据的输出位数以及小数点的位置，也可以用来决定字符串数据的输出的字符个数。



格式输出语句中使用的格式字符主要有以下几种：

* （ # ）　用来决定打印数字的宽度，如果输出的数据短于设定的长度，则左边填入空格；如果长于设定的长度，则输出一个百分号并输出原数。
* （ ** ）　出现在（#）前，表示输出数据的位数不足时，左边填入星号。
* （ $$ ）　表示在数据前加上一个美元符。
* （ , ）　表示输出数据每三位加一个逗号。
* （ . ）　表示数中小数点所在的位置。
* （+）或（−）　表示在数前加上符号位。
* （^ ^ ^ ^）　出现在格式场后部，表示输出数据以指数方式输出。

例如：

格　式	数　据	结　果
###	12	12
#####,.##	1200	1,200.00
**###	12	***12
$$##.##	12	$12.00
#.##	−12	%−12.00
+###	12	+12
+.##^^^^	123	+.12E+03

* （!）　用来输出字符串的第 1 个字符。
* （\ \）当中间的空格为 n 时，输出字符串的前 n+2 个字符。
* （ &）　输出整个字符串。

例如：

格　式	数　据	结　果
\ \	"ABCD"	AB
!	"ABCD"	A
&	"ABCD"	ABCD

（3） 有关打印机的语句

PC BASIC 主要有三个关于打印机的语句，它们分别是LPRINT、WIDTH "LPT1："和V=LPOS(0)。其中语句

WIDTH "LPT1："，　<宽度>

用来设定打印机的输出宽度。当输出的一行数据超过设定的宽度时，自动插入一个<CR>符。V=LPOS(0)语句可用来确定打印头的当前位置。例如，程序

```
10  WIDTH "LPT1: ", 20
20  FOR I=0 TO 99 :LPRINT "-" ; : NEXT
```

与程序

```
10  FOR I=0 TO 99 : LPRINT "-" ;
20  IF LPOS(0)=20 THEN LPRINT
30  NEXT
```

是等效的，都表示每输出20个减号后插入一个<CR>符。

LPRINT语句除了可以输出数值和字符串数据外，还可用来输出打印机控制信号，这可通过语句

LPRINT CHR$(<控制码>)

实现。如果使用的是IBM 80 CPS点阵打印机，则其主要控制字符如表3-6所示。

表3-6　IBM 80 CPS打印机的控制符

ASCII码	功　能
7	BEL(响铃)
10	LF(换行)
13	CR(回车)
12	FF(换页)
14	设置打印机为宽体方式，每行40符
20	取消宽体方式
15	设置打印机为窄体方式，每行132符
18	取消窄体方式
ESC-E	设置打印机为加重方式
ESC-F	取消加重方式
ESC-0	设行间隔为 1/8"
ESC-1	设行间隔为 7/72"
ESC-2	设行间隔为 1/6"(正常间隔)
ESC-C	设置页长(缺省值为66)

例3—11　打印节日明信片

① 功能：按通讯录中的地址打印若干份节日明信片。

② 程序：

```
10  ' EX 3-11 Print some postcards
20  WIDTH "LPT1:", 80
30  LPRINT CHR$(27)+"1"
40  READ NAME1$, ADDR$: IF NAME1$="END" THEN END
50  LPRINT  LPRINT: LPRINT: LPRINT
60  LPRINT SPC(25); CHR$(14); "POSTCARD"
70  LPRINT: LPRINT: LPRINT: LPRINT TAB(35); "|"
80  S1$=" With Best Wishes": S2$=" To:"+NAME1$: GOSUB 220
90  GOSUB 200
100 S1$="    for": S2$="      "+ADDR$: GOSUB 220
110 GOSUB 200
120 S1$="A Merry Christmas": S2$=" ": GOSUB 220
130 LPRINT TAB(35); "|"
140 S1$="       and": S2$="From: Computer Dept.,": GOSUB 220
150 GOSUB 200
160 S1$="A Happy New Year": S2$=" NANJING UNIV., NANJING"
170 GOSUB 220: GOSUB 200
180 FOR I=0 TO 10: LPRINT TAB(35); "|": NEXT
190 GOTO 40
200 LPRINT TAB(35); "|  ----------------------"
210 RETURN
220 LPRINT CHR$(14); S1$; TAB(18); CHR$(20); "|";
230 LPRINT CHR$(15); S2$; CHR$(18)
240 RETURN
300 DATA WANG WU, "19, SHANGHAI Road, NANJING"
310 DATA CHAO LIN, "34, BEIJING Road, NANJING"
390 DATA END, END
```

③ 说明：20—30 行把打印机置为 80 符，行间隔为 $^{7}/_{72}$″。40 行读入一个人的地址并判断是否已结束，若结束，则停止运行。50—60 行以大号字打印标题。70—180行输出明信片的内容。200 -210 行的子程序输出一行下划线。220—240 行的子程序先以大号字输出S1$，再以小号字输出S2$。300—390 行为通讯录数据。

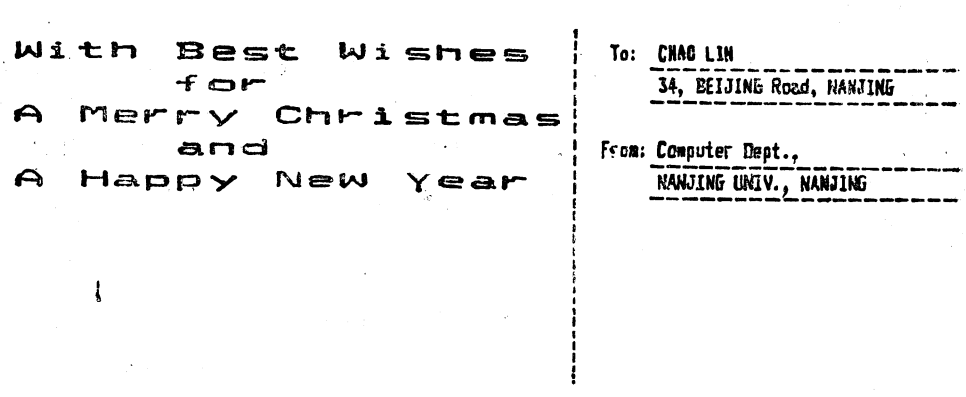
④ 结果：每张卡片输出的格式如下页所示。

（4）报表生成

利用上述输出语句就可以生成并打印报表。下面通过实例说明报表生成程序的设计方法。

例 3—12　打印某校教员统计报表

① 功能：假设统计结果（百分比）已放在D(n, m)中，其中n为报表的行数，m为报表的列数。D(n, m)中的数据格式为：D(0, m)为行总计，D(n, 0)为列总计．其



它各项为数据。标题："REPORT——TEACHER"放在TITLE$中；列标题：Total（总计）、Prof.（教授）、As Prof.（副教授）、Lecturer（讲师）、Assist.（助教）和Other（其它）放在C$（n）中；行标题：Total、Dept. 1—4（系1到系4）放在L$（m）中。程序以小号字的形式打印出报表（报表的列数不允许超过12列）。

② 程序：

```
100 ' EX3-12 Print a report
110 ' D(N, M)--Data, N--Line#, M--Col#
120 ' TITLE$--Title, L$--Linename, C$--Colname
130 WIDTH "LPT1:", 132
140 IF M>12 THEN PRINT "Too many col! " :END
150 LPRINT:LPRINT:LPRINT CHR$(27)+"0" ' 1/8" LF
160 ' Print the title
170 LPRINT TAB(5);CHR$(14);
180 LPRINT USING "\                    \";TITLE$
190 LPRINT CHR$(20):LPRINT:LPRINT:LPRINT CHR$(15)
200 ' Print the report
210 GOSUB 300:LPRINT "|Items|";
220 FOR J=0 TO M:LPRINT USING "\     \|";C$(J);:NEXT
230 LPRINT:GOSUB 300
240 FOR I=0 TO N
250 LPRINT USING "|\     \|";L$(I);
260 FOR J=0 TO M:LPRINT USING "###.##%|";D(I, J);:NEXT
270 LPRINT:GOSUB 300
280 NEXT
290 LPRINT CHR$(18):RETURN
300 LPRINT "+";
310 FOR J=0 TO M+1:LPRINT "---------+";:NEXT
```

```
320 LPRINT
330 RETURN
```

③ 说明：130行把打印机的宽度置为132符。140行检查要输出的列数是否超过12列，若超过则打印："Too many col! "（列数太多了！）并停止程序的运行。150行置行间隔为1/8"。160—180行打印报表标题。210—230行打印列名。240—280行打印报表内容和行名，其中数据的格式为："# # #, # # %"。

④ 结果：打印出的报表如下：

REPORT--TEACHER

Items	Total	Prof.	As Prof.	Lecturer	Assist.	Other
Total	100.00%	3.75%	10.82%	65.08%	17.32%	3.03%
Dept. 1	100.00%	3.62%	9.95%	67.87%	15.84%	2.71%
Dept. 2	100.00%	2.82%	11.97%	62.68%	19.72%	2.82%
Dept. 3	100.00%	5.02%	11.42%	61.64%	18.72%	3.20%
Dept. 4	100.00%	2.70%	9.91%	69.37%	14.41%	3.60%

附：报表生成程序：

```
10 ' Gendata
20 N=4 : M=5 : TITLE$= "REPORT--TEACHER"
30 DIM D(N, M), C$(M), L$(N)
40 FOR I=0 TO N : FOR J=0 TO M : READ D(I, J) : NEXT J, I
45 FOR I=0 TO N : FOR J=M TO 0 STEP -1 :
   D(I,J)=D(I, J)/D(I, 0)*100 : NEXT J, I
50 FOR J=0 TO M : READ C$(J) : NEXT
60 FOR I=0 TO N : READ L$(I) : NEXT : GOSUB 100 : END
68 DATA 693, 26, 75, 451, 120, 21
70 DATA 221, 8, 22, 150, 35, 6
72 DATA 142, 4, 17, 89, 28, 4
74 DATA 219, 11, 25, 135, 41, 7
76 DATA 111, 3, 11, 77, 16, 4
80 DATA Total, Prof., As Prof., Lecturer, Assist., Other
90 DATA Total, Dept. 1, Dept. 2, Dept. 3, Dept. 4
```

第三节　数据文件

前面已经介绍过，BASIC 语言不仅可以用于科学工程计算，还可以用于非数值计算的应用领域。无论是数值计算还是非数值处理，常常需要处理输入的大量数据，并把产生的大量数据输出，这些数据通常以数据文件的形式组织起来（DATA 语句一般仅适合于存贮少量的数据）。下面介绍数据文件的使用方法。

1．数据文件的概念

（1）数据文件

数据文件是一批存放在计算机外存贮器中的数据。每个数据文件都有一个供用户访问的名字，叫做文件名。一个数据文件通常由若干记录组成。记录是数据文件的存取单位，它通常又由若干字段组成。字段则是数据的最小组成单位，用来记录一项数据。例如，如果用一个文件表示某系的人事档案，则每人的档案就可以占一个记录，而其中姓名、性别、年龄和职称等项目就各占一个字段。

PC BASIC数据文件的存贮介质为磁盘或磁带（下文均以磁盘文件为例说明文件的使用方法）。数据可以用 ASCII 码或用 BASIC 内码存贮。文件的存取方式允许为顺序和随机两种。

数据文件除了可以是磁盘文件外，还可以是下列设备文件：

* KYBD：　　键盘
* SCRN：　　显示器
* LPT1：—LPT3：　　打印机
* COM1：—COM2：　　异步通讯器

（2）顺序文件

顺序文件中的每个记录以其输入的先后次序顺序存放，它建立起来较容易，但访问时间长且不灵活。读写顺序文件必须从头至尾一个记录接一个记录地读写，十分不方便。顺序文件允许存放数值和字符串数据，文件长度仅受盘空间限制。

顺序文件建立和使用的步骤如图 3—1 所示。

（3）随机文件

随机文件的存取是随机的，要访问文件中的任意记录仅需给出记录号即可，而不象顺序文件那样要从头找起，故其访问时间快得多。此外，随机文件一般比顺序文件节省盘空间，这是因为随机文件可采用 BASIC 内码存放数据，而顺序文件则必须用 ASCII 码存放数据。

随机文件只允许直接存贮字符串数据。数值数据在存入数据文件之前，必须转换成字符串．而从数据文件读回时又要转换回来。

随机文件的记录长度可以是 1 到 32767 个字节。每个文件最多允许 16，777，215 个记录（对于 BASICA 2.00 版而言）。

随机文件的使用比顺序文件显得麻烦，它的建立和访问的过程如图 3—2 所示。

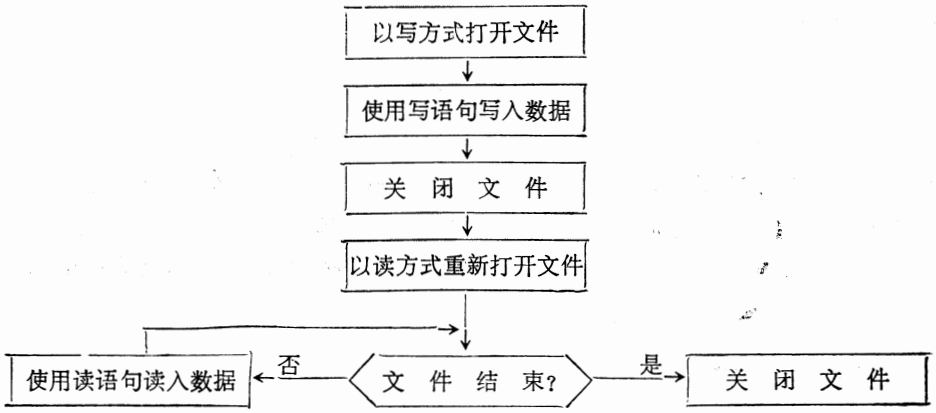


图 3—1　顺序文件的使用过程

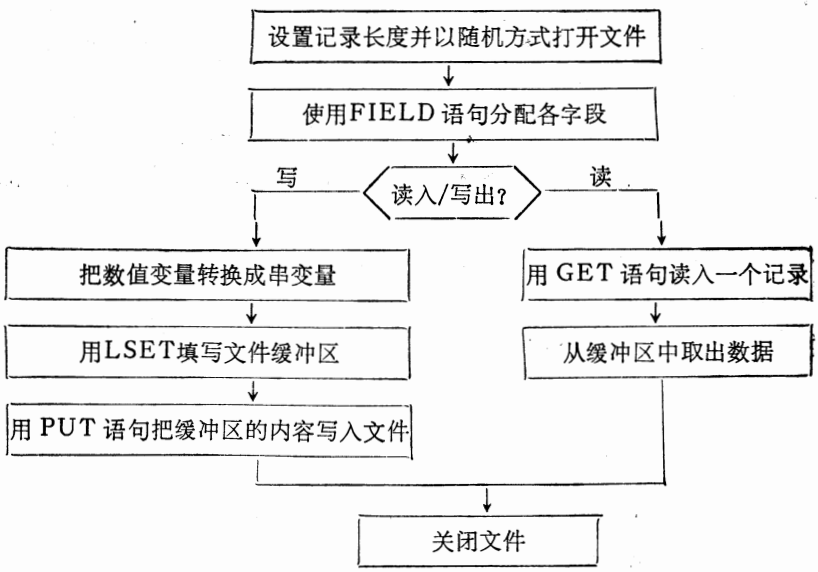


图 3—2　随机文件的使用过程

2．顺序文件的处理

（1）文件的建立与打开

建立和打开文件都使用 OPEN 语句，OPEN 语句的格式为：

OPEN〈路径名〉〔FOR 〈方式〉〕AS〔#〕〈文件号〉〔LEN=〈记录长度〉〕

或　　OPEN 〈方式〉,〔#〕〈文件号〉,〈路径名〉〔,〈记录长度〉〕

第 1 式的方式有四种：OUTPUT（顺序写）、INPUT（顺序读）、APPEND（顺序添加）和缺省（随机）。第 2 式的方式有 O（顺序写）、I（顺序读）和 R（随机）三种。两式的文件号均可为 1—15，记录长度的缺省值为 128 字节。

使用OPEN语句后，BASIC在盘上查找指定的文件。若找到则打开（若用OUTPUT方式打开则清除原文件），否则就建立一个新文件。随后，BASIC为打开的文件分配一个缓冲区，并把文件号与文件名联系起来，在以后的读写等操作中，使用文件号代替文件名。

顺序文件的打开是写互斥的，即已用OUTPUT或APPEND方式打开的文件在未关闭之前不允许再次打开。

（2）文件的关闭

文件在使用完毕后，一般都需要关闭，以释放所占的文件号并防止文件的意外损坏。文件关闭语句的格式为：

CLOSE〔<文件号>{，<文件号>}〕

如果文件号缺省，则关闭所有打开的文件。

执行END，NEW，RESET，SYSTEM或不带“，R”开关的RUN命令，将自动关闭所有打开的文件。

（3）文件的读写

顺序文件在打开之后、关闭之前，可以依照文件的打开方式进行读或者写。

写文件通常使用语句

WRITE #<文件号>，<表达式表>

来完成，有时也用PRINT # 语句来写。建立文件的方法用下例说明。

例3—13 建立一个通讯录文件

```
10 ' EX3-13 Create a Address Book File
20 OPEN "ADDRBK.DAT" FOR OUTPUT AS #1
30 READ N$, A$, P$
40 IF N$= "END" THEN 70
50 WRITE #1, N$, A$, P$
60 GOTO 30
70 ' Close the file
80 CLOSE #1
90 END
100 DATA WANG XING, "29, SHANGHAI Road, NANJING", 43259
110 DATA ZHAO MING, "2, RENMING Road, NANJING", 45245
120 DATA CHEN TING, "214, HUAIHAI Road, NANJING", 13578
130 DATA ZHOU HONG, "371, NANGHAI Road, NANJING", 98765
190 DATA END, ,
```

说明：20行把文件ADDRBK.DAT打开成OUTPUT方式，若该文件已存在则删去原来的文件并重新建立一个新文件，否则建立一个新文件。30—40行从100行开始的DATA语句中读入一行通讯录，并判断是否已结束，若结束转70行做结束处理，否则继续下去。50行写一行通讯录到文件中去。70—90行关闭文件并结束程序的执行。注意，在写语句中用的变量只允许使用一个字母，如例中的N$等。

文件建立好了之后，就可使用读语句读出文件中的数据，常用的读语句有三个：

INPUT #〈文件号〉，〈变量〉{，〈变量〉}

LINE INPUT #〈文件号〉，〈串变量〉

V$=INPUT(n, #〈文件号〉)

后两式只用于读入字符串数据，其中LINE INPUT# 用于读入一行以〈CR〉结尾的字符串，而INPUT$ 则读入文件中的n个字符。

例3—14 读出一个通讯录

```
10 ' EX3-14 Access the Address Book File
20 OPEN "ADDRBK.DAT" FOR INPUT AS #1
30 IF EOF(1) THEN 70
40 INPUT #1, N$, A$, P$
50 PRINT N$, A$, P$
60 GOTO 30
70 ' Close the file
80 CLOSE #1
90 END
```

说明：20行以读方式打开文件。30行检查文件是否结束，若是，则关闭文件，其中EOF为标志文件结束的一个逻辑函数。有关文件的函数还有两个：

LOF(〈文件号〉)　　文件长度

LOC(〈文件号〉)　　文件当前记录号

40行为文件读入语句。该程序的运行结果如下：

```
WANG    XING    29,   SHANGHAI  Road,   NANJING  43259
ZHAO    MING     2,   RENMING   Road,   NANJING  45245
CHEN    TING   214,   HUAIHAI   Road,   NANJING  13578
ZHOU    HONG   371,   NANGHAI   Road,   NANJING  98765
```

对于已建立的文件，允许在文件的末尾添加若干个记录，这是通过在打开文件时使用添加方式而实现的。下面就是顺序文件添加的一个例子。

例3—15 在通讯录文件中添加若干个记录

```
10 ' EX3-15 Append the Address Book File
20 OPEN "ADDRBK.DAT" FOR APPEND AS #1
60 INPUT N$, A$, P$
70 IF N$= "END" THEN CLOSE: END
80 WRITE #1, N$, A$, P$
90 GOTO 60
OK
```

```
RUN
? SUN YAN, "12, XINMING Road, BEIJING", 234567
? DING DING, "34, FUJIAN Road, BEIJING", 765941
? END, ,
```

添加两行数据后，再运行例 3—14 的结果如下：

```
RUN "EX3—14"
WANG    XING    29,   SHANGHAI   Road,   NANJING   43259
ZHAO    MING     2,   RENMING    Road,   NANJING   45245
CHEN    TING   214,   HUAIHAI    Road,   NANJING   13578
ZHOU    HONG   371,   NANGHAI    Road,   NANJING   98765
SUN     YAN     12,   XINMING    Road,   BEIJING  234567
DING    DING    34,   FUJIAN     Road,   BEIJING  765941
```

顺序文件的修改和插入是比较困难的，通常都必须建立一个中间文件，所以修改和插入的速度较慢。下面给出一个顺序文件修改记录的示范例子。修改指定记录、指定项以及插入若干记录的程序都可仿此编制。

例 3—16　修改通讯录文件

程序及其运行结果如下：

```
10 ' EX3—16 Change the Address Book File
20 OPEN "ADDRBK.DAT" FOR INPUT AS #1
30 OPEN "TEMP" FOR OUTPUT AS #2
40 IF EOF(1) THEN 120
50 INPUT #1, N$, A$, P$
60 PRINT N$, A$, P$
70 INPUT "Change(y/n)", R$
80 IF R$<>"y" THEN 100
90 INPUT N$, A$, P$
100 WRITE #2, N$, A$, P$
110 GOTO 40
120 CLOSE
130 KILL "ADDRBK.DAT"
140 NAME "TEMP" AS "ADDRBK.DAT"
150 END
OK
RUN
WANG XING  29, SHANGHAI Road, NANJING 43259
Change(y/n) y
? WANG XING, "27, SHANGHAI Road, NANJNG", 43459
```

```
ZHAO MING  2, RENMING Road, NANJING 45245
Change ( y/n ) n
CHEN TING 214, HUAIHAI Road, NANJING 13578
Change ( y/n )
    ⋮
```

说明：30行建立一个中间文件TEMP。50—90行输出一行数据并询问是否需要修改，若要修改则要求输入新数据。100行把修改好的数据送到中间文件中去。130—140行删除旧文件并把中间文件改名成ADDRBK·DAT。注意，这个程序不适用于处理有大量数据的数据文件。

3．随机文件的处理

（1）文件的打开与关闭

随机文件在使用前必须打开，随机文件的打开语句为：

OPEN 〈文件名〉 AS〔#〕〈文件号〉〔LEN=〈记录长〉〕

或 OPEN "R"，〔#〕〈文件号〉，〈文件名〉〔，〈记录长〉〕

文件打开成功后，BASIC分配一个缓冲区，其长度为一个记录。接着用FIELD语句把这缓冲区分配给各个字段。若字段为字符串型，则其长度为字符串的长度；若为数值型，则其长度为数值的内码长度（整型为2，实型为4，长实型为8）。FIELD语句的格式为：

FIELD〔#〕〈文件号〉{，〈长度〉 AS 〈串变量〉}

FIELD语句可看成是记录结构的定义语句。例如，要打开一个文件，这个文件的每个记录由五个字段组成，它们的项名和长度依次为：姓名（N$：16B）、性别（S$：1B）、年龄（A$：2B）、工资（P$：4B）和地址（D$：24B）。其程序为：

```
10  OPEN "ADDRLIST.DAT" AS #1 LEN=47
20  FIELD #1, 16 AS N$, 1 AS S$, 2 AS A$, 4 AS P$, 24 AS D$
```

随机文件的关闭语句与顺序文件一样，这里就不重复了。

（2）文件的读写

随机文件的读写语句有两条：

PUT〔#〕〈文件号〉〔，〈记录号〉〕

GET〔#〕〈文件号〉〔，〈记录号〉〕

其中记录号的缺省值为当前记录的下一记录。

在这两条语句中，PUT语句用来把缓冲区的内容写到文件的指定记录中去，而GET语句则是把文件的指定记录读到缓冲区中。

在使用PUT语句进行写之前，必须用LSET或RSET语句填缓冲区。这两条语句的格式为：

LSET 〈字段变量〉=〈串变量〉

RSET 〈字段变量〉=〈串变量〉

其中 LSET 表示向左对齐，RSET 表示向右对齐。例如：

```
30 INPUT "Name, Sex, Address" ; NAME$, SEX$, ADDR$
40 LSET N$=NAME$ : LSET S$=SEX$ : LSET D$=ADDR$
```

表示输入一个姓名、性别和地址并向左对齐放入缓冲区。

数值数据在放入缓冲之前必须转换成串变量，而在取出缓冲区之后，则又要将其转换成数值数据。上节中介绍的 STR$ 和 VAL 函数，可以完成这对转换工作，这时数值变量将以 ASCII 码的形式存盘。为了节省空间，通常使用另一组转换函数来完成这对转换，这组函数有以下三对：

* MKI$(I%)　把整型量转换成两个字节长的内码串，逆函数为 CVI(A$)。
* MKS$(S!)　把实型量转换成四个字节长的内码串，逆函数为 CVS(A$)。
* MKD$(D#)　把长实型量转换成八个字节长的内码串，逆函数为 CVD(A$)。

例如，程序：

```
50 INPUT "Age, Salary" ; Age%, Salary!
60 LSET A$=MKI$(Age%) : LSET P$=MKS$(Salary!)
70 PUT #1, 1
```

可以把年龄和工资写入到 1 号记录中，而下列程序则用于读出写入的数据：

```
80 GET #1, 1
90 ? USING "###" ; CVI(A$)
100 ? USING "###.##" ; CVS(P$)
```

在随机文件的使用过程中，不能使用函数 EOF( ) 来判断文件是否结束，而必须使用函数 LOF( ) 来推算，例如：

```
200 IF NUM%>LOF(#1)/RECLEN% THEN 290
210 GET #1, NUM%
    ⋮
290 CLOSE : END
```

就是这样来判断文件读出是否结束的。

(3) 一个随机文件的实例

下面介绍随机文件应用的一个完整实例。通过这个例子可以进一步地了解随机文件的操作方法。

例 3—17　教师档案管理程序

程序由总控(MENU)、输入(INPUT)、显示(DISPLAY)、修改(CHANGE)、删除(DELETE)和统计(COUNT)六个模块组成，下面分别进行介绍：

(I) 总控模块(MENU)

① 功能：询问需要处理的文件名，并打开这个文件，随后显示各功能的清单，再根据用户选择的功能号调用各功能块。若选择退出，则进行文件的关闭处理。

程序中使用的数据文件具有以下结构：

区段号	名称	长度	含　　义
1	M$	1B	标志，为255时表示已删记录
2	N$	16B	教师姓名
3	S$	1B	姓别，F表示女性；M表示男性
4	A$	2B	年龄，整型数
5	T$	16B	职称
6	P$	4B	工资，实型数
7	D$	24B	地址

② 程序：

```
10 'EX3-17 Member data file management program
20 CLS : PRINT : PRINT "File Management Program"
30 RECLEN%=64 : PRINT : PRINT "Which file do you want to
   manage? "
40 INPUT "Filename " ; FILENAME$ : FILENAME$=FILENAME$+
   ".DAT"
50 OPEN FILENAME$ AS #1 LEN=RECLEN%
60 FIELD #1, 1 AS M$, 16 AS N$,1 AS S$,2 AS A$, 16 AS T$,
      4 AS P$, 24 AS D$
70 FILELEN%=LOF(1)/RECLEN%
80 CLS : PRINT : PRINT "Options : " : PRINT
90 PRINT 1,  "Input records"
100 PRINT 2,  "Display records"
110 PRINT 3,  "Change a record"
120 PRINT 4,  "Delete a record"
130 PRINT 5,  "Count"
140 PRINT 6,  "Exit"
150 PRINT : PRINT : INPUT "Choice" ; CHOICE%
160 IF CHOICE%<1 OR CHOICE%>6 THEN 70
170 ON CHOICE% GOSUB 200, 400, 600, 800, 900, 190
180 CLOSE : GOTO 50
190 CLOSE : END
```

③ 说明：40行输入一个文件名并加上".DAT"。50—60行打开文件并安排各字段的

位置。70行计算文件长度。80—140行显示功能清单。150—170行根据用户输入的功能号，分别调用各功能模块。190行为退出处理。

（Ⅱ）输入模块（INPUT）

① 功能：输入或在文件末尾添加若干个记录。

② 程序：

```
200 ' Input records
210 CLS : PRINT : PRINT
220 PRINT "Input the data, when end enter END, , , , , "
230 PRINT : PRINT
240 PRINT "Name, Sex ( M/F ), Age ( 1—100 ), Title, Salary ( xxx.xx ),
        Address"
250  INPUT " : " , NAME1$, SEX$, AGE%, TITLE$, SALARY! ,
         ADDR$
260 IF NAME1$= "end" OR NAME1$= "END" THEN 350
270 LSET M$=CHR$ ( 0 ) : LSET N$=NAME1$ : LSET S$=SEX$
280 LSET A$=MKI$ ( AGE% ) : LSET T$=TITLE$
290 LSET P$=MKS$ ( SALARY! ) : LSET D$=ADDR$
300 FILELEN%=FILELEN%+1
310 PUT #1, FILELEN%
320 GOTO 250
350 ' END
390 RETURN
```

③ 说明：210—240行输出提示信息。250—260行输入一个人的档案，每项以逗号分隔，空项仅输入一个逗号，结束时输入："END, , , , , "。270—310行把输入的记录存盘。

（Ⅲ）显示模块（DISPLAY）

① 功能：显示若干行数据。

② 程序：

```
400 ' Display records
410 CLS : PRINT : PRINT "File Length is " ; FILELEN%
420 INPUT "Start rec #, end rec #" ; R1%, R2% : PRINT
430 IF R1%<1 OR R2%>FILELEN% THEN 410
440 PRINT "Name Sex Age Title Salary Address"
450 PRINT "----  --- --- ----- ------ -------"
460 IF R1%=R2% THEN I%=R1% : GOSUB 510 : GOTO 490
470 FOR I%=R1% TO R2%
```

```
480 GOSUB 510 : NEXT
490 IF INKEY$= "" THEN 490
500 RETURN
510 GET #1, I%
520 IF M$=CHR$(255) THEN 590
530 PRINT USING "\          \" ; N$;
540 PRINT S$; : PRINT USING " # # # " ; CVI(A$);
550 PRINT USING "\          \" ; T$;
560 PRINT USING " # # # . # # " ; CVS(P$);
570 PRINT D$
590 RETURN
```

③ 说明：410行显示出文件的长度。420—430行要求输入欲显示的数据范围，如超出文件的范围则要求重新输入。440—480行输出要显示的数据。490—500行等待输入任意键返回主控程序。510—590行为读入并显示一行数据的子程序，其中520行表示不显示已删除的数据。

（Ⅳ）修改模块（CHANGE）

① 功能：修改指定记录中的指定项。

② 程序：

```
600 ' Change a record
610 CLS : PRINT : PRINT
620 INPUT "Which record ===>" , R1%
630 PRINT : PRINT "1 : Name  2 : Sex  3 : Age  4 : Title  5 : Salary
    6 : Address"
640 I%=R1% : GOSUB 510
660 PRINT : INPUT "Which item ===>" , N%
670 ON N% GOTO 680, 690, 700, 710, 720, 730
680 INPUT "Name=" , NAME1$ : LSET N$=NAME1$ : GOTO 750
690 INPUT "Sex=" , SEX$ : LSET S$=SEX$ : GOTO 750
700 INPUT "Age=" , AGE% : LSET A$=MKI$(AGE%) : GOTO 750
710 INPUT "Title=" , TITLE$ : LSET T$=TITLE$ : GOTO 750
720 INPUT "Salary=" , SALARY! : LSET P$=MKS$(SALARY!)
    : GOTO 750
730 INPUT "Address=" , ADDR$ : LSET D$=ADDR$ : GOTO 750
750 PUT #1, R1%
790 RETURN
```

③ 说明：620行请求输入要修改的记录号。640行调用510行开始的子程序读入和显示指定的记录。660—670行输入欲修改的字段号，并分别转不同的处理程序。680—730行为

各项的修改程序。750 行写入修改的记录。

（5）删除模块（DELETE）

```
800 ' Delete a record
810 CLS : PRINT
820 INPUT "Which record===>" , R1%
830 GET #1, R1% : LSET M$=CHR$(255)
840 PUT #1, R1%
890 RETURN
```

说明：删除的方法是在该记录的标志字段中写入一个 CHR$（255）。记录删除后，不能显示和修改，也不参加统计工作，但仍占一个记录的位置。这时，文件长度与实际有效的记录数并不一致。

（6）统计模块（COUNT）

```
900 ' Count
910 AGE=0 : PAY=0 : COUNT%=0
920 FOR I%=1 TO FILELEN% : GET #1, I% : IF M$=CHR$(255)
    GOTO 940
930 COUNT%=COUNT%+1 : AGE=AGE+CVI(A$) : PAY=PAY+
    CVS(P$)
940 NEXT : CLS
950 PRINT : PRINT "Total of record=" ; COUNT%
960 PRINT : PRINT USING "Average of age= # #.#" ; AGE/COUNT%
970 PRINT : PRINT USING "Average of pay= # #.# #" ;PAY/COUNT%
980 IF INKEY$= "" THEN 980
990 RETURN
```

说明：本模块只进行几项最简单的统计：计算总数（Total of record）、平均年龄（Average of age）和平均工资（Average of pay）。

第四节　BASIC 程序的编程技术

随着微机软件技术的不断发展，特别是商品软件的出现，软件质量的衡量标准也发生了较大的变化。目前一般认为，一个好的软件不仅要速度快和省内存，更重要的是还要具有模块化、坚固性、通用性和可用性。下面介绍的是在 BASIC 语言中实现这些要求的编程技术。

1．“菜单”技术

当一个程序具有若干项供用户选择的功能时，一般都使用交互技术进行分枝处理。实现

的过程是程序首先显示出所能完成的功能名称，用户根据需要指出希望完成的功能号，程序分析用户的输入并调用不同的功能块进行处理。因为这种方法酷似餐馆的点菜方式，故称为“菜单”技术。

在介绍“菜单”技术之前，首先说明与之密切相关的显示器控制语句。

（1）显示器控制语句

PC BASIC 可以支持两种显示器——字符显示器和图形显示器。本节只介绍字符显示器的控制语句，关于图形显示器的控制语句将在第八章中介绍。

字符显示器每屏由 25 行组成（第 25 行为监视行），每行可显示 80 个字符。屏幕的左上角为光标的初始位置，其行列坐标为：（1，1）。

下面是几条常用的显示器控制语句：

* CLS　　清除屏幕且光标返回初始位置。
* V=SCREEN（〈行〉，〈列〉）　取屏幕上指定位置处的字符的 ASCII 码。
* V=CSRLIN　　取光标的行坐标。
* V=POS（0）　　取光标的列坐标。
* KEY OFF　　终止显示第 25 行的功能键提示信息。
* COLOR n，m　　修改字符属性（下文中有详细的介绍）。
* LOCATE〔〈行〉，〈列〉〕〔，〈方式〉，〈光标始线〉，〈光标末线〉〕　把光标移到指定的位置并改变光标的属性。若方式项为 0，则光标不可见。光标的始末线决定光标的大小，它们的取值范围为 0—13。当始线大于末线时，为分裂光标。例如：

```
10  LOCATE 1, 1 { 光标移回初始位置 }
20  LOCATE, , 1, 0, 13 { 把光标改成一个字符大小的闪烁白方块 }
30  KEY OFF : LOCATE 25, 1 : PRINT "OK! " {在第25行上显示出：
"OK! " }
```

（2）“菜单”技术

在“菜单”技术中，显示程序功能清单的方式一般有两种，第 1 种是在屏幕的首行或监视行显示一行功能名，这种方式不影响屏幕上的其它操作，但提供的信息较少，例如功能键〈F1〉—〈F10〉的提示行就是采用的这种方法。第 2 种方式是屏幕上给出若干行提示信息，每行包括功能名（或功能号）及其含义，例 3—17 就是采用的这种方式。

用户选择的方式也有两种，第 1 种是输入被选功能的功能名或功能号（例 3—17使用的是功能号），如输入功能名，通常仅需输入一个字符即可。第 2 种方式则是通过定位输入设备（如键盘、光笔、定位器和数字化桌等）把光标移到所选功能所在的行上，按下一个键（或一个开关）来选择的。这两种方法各有特色，但后一种使用起来更方便。

程序在分析用户输入的键之后，一般都要做分枝处理，这种情况下的分枝处理，最好使用情况语句。情况语句的格式为：

```
ON 〈条件值〉 GOTO 〈行号〉 {，〈行号〉}
ON 〈条件值〉 GOSUB 〈行号〉 {，〈行号〉}
```

其中条件值为数值量，取值范围为 0—255。如使用字符串条件，则须先转换成数值量。例如：

```
10 FOR I=1 TO 10 : READ N(I) : NEXT
20 ? "INPUT"
30 ? "DISPLAY"
40 ? "CHANGE"
50 INPUT "Your choice" ; C$
60 N=ASC(MID$(C$, 1, 1)) - &H40
70 IF N<1 OR N>10 THEN 20
80 ON N(N) GOSUB 1000, 2000, 3000
90 GOTO 20
100 DATA 0, 0, 3, 2, 0, 0, 0, 0, 1, 0
```

就是使用了字符串条件。假如要选择 INPUT 功能，可以输入 INPUT，IN,I 甚至 IM，因为程序只识别第 1 个字母 "I"。该例的字母与条件值的转换是通过查对照表实现的，因此十分灵活和方便。

下面举例说明如何使用键盘设备移动光标来选择指定功能。

例 3—18　英制到公制的转换程序

① 功能：通过移动光标键↑和↓选择把英吋化为厘米、英呎化为米、英哩化为公里或者英磅化为公斤，按下〈CR〉键表示选择生效。若要退出这个程序，请选EXIT。

② 程序：

```
10 ' EX3—18 English standard measures to metric system
20 CLS : LOCATE 3, 17
30 PRINT CHR$(201)+STRING$(38, 205)+CHR$(187)
40 FOR COUNT=1 TO 6
50 PRINT TAB(17) ; CHR$(186) ; TAB(56) ; CHR$(186)
60 NEXT
70 PRINT TAB(17) ; CHR$(200)+STRING$(38, 205)+CHR$(188)
80 LOCATE 5, 24 : PRINT "Measurement Conversions"
90 LOCATE 7, 21 : PRINT "(C) Copyright 1984, NANJING UNIV."
100 LOCATE 12, 21 : PRINT "Select conversion from this list : "
110 PRINT TAB(21) ; " * Inches to Centimeters"
120 PRINT TAB(21) ; " * Feet to Meters"
130 PRINT TAB(21) ; " * Miles to Kilometers"
140 PRINT TAB(21) ; " * Pounds to Kilograms"
160 PRINT TAB(21) ; " * Exit"
170 PRINT : PRINT TAB(21); "Move the cursor up/down"
180 LOCATE 13, 21, 1, 0, 13
190 K$=INKEY$ : IF K$= "" THEN 190
200 L=CSRLIN
210 IF K$=CHR$(&HD) THEN 250
```

```
220 IF MID$(K$, 2, 1)=CHR$(72) THEN IF L>13 THEN PRINT
    CHR$(30);
230 IF MID$(K$, 2, 1)=CHR$(80) THEN IF L<17 THEN PRINT
    CHR$(31);
240 GOTO 190
250 LOCATE 21, 1, 1, 13, 13 : PRINT SPC(70) : PRINT
260 PRINT SPC(70) : LOCATE 21, 1
270 ON L-12 GOSUB 300, 340, 380, 420, 290
280 GOTO 100
290 END
300 '* * * INCHES TO CM
310 INPUT "How many inches" ; I
320 PRINT I; "inches=" ; I*2.54; " cm"
330 RETURN
340 '* * * FEET TO METERS
350 INPUT "How many feet" ; F
360 PRINT F; "feet=" ; F*.3048; " m"
370 RETURN
380 '* * * MILES TO KM
390 INPUT "How many miles" ; M
400 PRINT M; "miles=" ; M*1.609; " km"
410 RETURN
420 '* * * POUNDS TO KG
430 INPUT "How many pounds" ; P
440 PRINT P; "pounds=" ; P*.45; " kg"
450 RETURN
```

③说明：20—90行输出一个由特殊字符组成的方框，在方框中填入程序名。100—170行显示出功能清单。180行把光标改为闪烁的方块形式并移到"菜单"的第1个功能前的星号上。190行等待输入。200行取光标所在的行号。210行表示若键入<CR>键，则转250行处理。220—240行处理键入的光标移动键↑和↓，对用户输入的其它键都不予响应。250—260行清除屏幕上的21行和22行，为转换程序做准备。300—450行为英制到公制的转换程序。

2．链接技术

在设计一个较大的程序时，经常根据程序的功能把它分成若干功能块，这就是所谓的模块化技术。一个好的模块化程序，不仅易编易调，也易读和易维护。另外，还可利用覆盖技术在较小的内存中运行较大的程序。模块化程序的各个模块一般都使用链接技术进行链接，使其成为一个可运行的整体。下面就介绍PC解释BASIC中所使用的链接技术。

（1）无参模块的链接

要把几个相互完全独立的程序链接起来，方法十分简单（因为模块间无参数传递）。一般先设计一个主控程序，在程序中利用“菜单”技术分枝。对于每个分枝，可以直接使用RUN语句或CHAIN语句（链接语句）来调用所对应的程序模块，接着分别编制各功能模块，各个模块执行完毕后，再执行一条CHAIN“主控模块”语句返回主控模块。

IBM公司提供的MS-DOS磁盘上有一个名为SAMPLES·BAS的程序，它就是该盘上BASIC表演程序的主控程序。为了明确起见，可把这个程序简化为例3—19的形式。

例3—19 表演程序的主控程序

```
10 ' EX3—19 Samples
20 KEY OFF : CLS : LOCATE 5 , 10 : PRINT "SAMPLE PROGRAMS"
   : PRINT
30 PRINT TAB(10); "M—MUSIC"
40 PRINT TAB(10); "O—MORTGAGE"
50 PRINT TAB(10); "C—COMM"
60 PRINT : PRINT TAB(10); "ESC KEY—EXIT"
70 K$=INKEY$ : IF K$= "" THEN GOTO 70
80 IF MID$(K$, 1, 1)= "M" THEN CHAIN "MUSIC" , 1000
90 IF MID$(K$, 1, 1)= "O" THEN CHAIN "MORTGAGE" , 1000
100 IF MID$(K$, 1, 1)= "C" THEN CHAIN "COMM" , 10
110 IF MID$(K$, 1, 1)=CHR$(27) THEN GOTO 130
120 GOTO 20
130 END
```

说明：20—60行显示“菜单”，其中20行中的第1句表示终止显示第25行的功能键清单。70行等待输入。80—110行表示根据用户的不同选择链接不同的程序，例如80行中的CHAIN“MUSIC”，1000表示装入名为“MUSIC·BAS”的程序并转至该程序的1000行开始执行。在这段程序中，若链接成功主控程序就被覆盖，被链接的程序运行完毕后应再装入主控程序运行。

(2) 有参模块的链接

在多数情况下，模块之间都需要交换信息。模块之间的参数传递一般可用两种方法实现，一种是使用CHAIN语句中的ALL选项，另一种是使用COMMON语句说明。CHAIN语句的格式为：

CHAIN〔MERGE〕〈路径名〉[,[〈行号〉]][, [ALL][, DELETE 〈范围〉]]]

若选MERGE则表示部分覆盖（仅覆盖DELETE后指出的程序段）。ALL表示要传递所有变量，否则仅传递由COMMON语句指明的公共变量。COMMON语句的格式如下：

COMMON 〈变量〉{ , 〈变量〉}

其中变量为数组名时，后面要跟一对圆括号。例如：

```
10 CHAIN "PROG1" , ALL
```

表示把所有变量传送到程序PROG1中去，而

```
10  COMMON A，DE( )，C$
20  CHAIN "PROG2"
```

表示仅把变量A、数组DE和字符串C$传送到程序PROG2中去。

例3—20是用来说明部分覆盖的一个例子。

例3—20　　性别/年龄分析

① 功能：在人口普查等应用中，常需要分析各项调查数据的相关性。本例就是利用调查数据分析性别与年龄相关性的一个程序。设调查表的格式如下：

调　查　表	
性　别	年　龄　组
□ 1. 男性	□ 1. <20　□ 4. 41-50
□ 2. 女性	□ 2. 21-30　□ 5. 51-60
	□ 3. 31-40　□ 6. >60

② 程序：

* 主模块（统计程序）

```
10 ' EX3—20 Population sex/age analysis
20 N=2：M=6
30 DIM D(N，M)，L$(N)，C$(M)
40 FOR I=0 TO N：FOR J=0 TO M：D(I，J)=0：NEXT J，I
50 READ SEX，AGE：IF SEX=0 THEN 90
60 D(SEX，AGE)=D(SEX，AGE)+1：D(0, 0)=D(0, 0)+1
70 D(SEX，0)=D(SEX，0)+1：D(0, AGE)=D(0, AGE)+1
80 GOTO 50
90 FOR I=0 TO N：READ L$(I)：NEXT
100 FOR J=0 TO M：READ C$(J)：NEXT
110 TITLE$= "SEX/AGE ANALYSIS"
190 CHAIN MERGE "REPCRT" , 200, ALL, DELETE 200—990
200 DATA 2, 1, 1, 2, 2, 3, 1, 4, 2, 5, 2, 6, 2, 1, 1, 2, 2, 1, 1, 1
210 DATA 1, 2, 2, 2, 1, 3, 1, 3, 2, 3, 1, 3, 2, 3, 1, 4, 1, 5, 1, 3
970 DATA 0, 0
980 DATA Total, MALE，FEMALE
990 DATA Total, <20, 21—30, 31—40, 41—50, 51—60, >60
```

* 子模块（报表打印）

```
200 ' REPORT
220 LPRINT：LPRINT：LPRINT CHR$(27)+ "0"
230 LPRINT TAB(20)；CHR$(14)；
```

```
240  LPRINT USING "\                    \" : TITLE$
250  LPRINT : LPRINT : LPRINT : LPRINT CHR$ (20)
260  GOSUB 350 : LPRINT " | Items | " ;
270  FOR J=0 TO M : LPRINT USING "\     \| " ; C$(J) ; : NEXT
280  LPRINT : GOSUB 350
290  FOR I=0 TO N
300  LPRINT USING " | \     \| " ; L$ (I) ;
310  FOR J=0 TO M : LPRINT USING " # # # # | " ; D(I , J) ; : NEXT
320  LPRINT : GOSUB 350
330  NEXT
340  END
350  LPRINT "+" ;
360  FOR J=0 TO M+1 : LPRINT "----------+" ; : NEXT
370  LPRINT
380  RETURN
990  REM
```

③ 说明：

* 主模块：20—40 行初始化结果数组。50 行读入一对性别年龄数据，若性别为 0 则结束。60—70行进行统计。90—110 行填入标题。190 行装入模块 REPORT·BAS（必须是 ASCII 码文件）并覆盖 200—990 行的数据，然后转 REPORT·BAS 的 200 行开始执行。

* 子模块：这段程序类似于例 3—12，这里不再说明。

④ 结果：这个程序的执行结果如下：

SEX/AGE ANALYSIS

Items	Total	<20	21-30	31-40	41-50	51-60	>60
Total	20	4	4	7	2	2	1
MALE	11	1	3	4	2	1	0
FEMALE	9	3	1	3	0	1	1

3. 输入输出技术

编制程序的目的最终是交给用户使用，而用户与计算机交换信息一般是通过程序的输入输出来实现的，所以从“用户第一”的观点来看，一个程序的输入输出质量至关重要，有时会直接影响到整个程序的质量。那么，又如何来衡量一个程序的输入输出设计得好不好呢？一般认为可以从程序的可用性和坚固性两个方面来判断出一个程序在输入输出方面的优劣。

(1) 程序的可用性（Availability）

程序的可用性是指用户使用该程序的难易和方便程度。提高程序可用性的措施主要有以下几条：

* 要求用户输入时，应给出指示信息。
* 尽量为用户提供输入缺省值。
* 采用用户最方便、最自然的格式输入数据，要方便用户而不是方便程序。
* 用户输入出错时，应给出错误提示信息。
* 提供联机的求助功能(HELP)，使用户不查手册就能很好地使用这个程序。
* 输出信息的格式要清晰明了易懂。

实现上述功能一般都要用到一些特殊的显示器操作，下面首先介绍字符属性更改语句，这条语句的格式为：

COLOR〔<字符属性>，<背景属性>〕

对于字符显示器，常用的属性组合及其功能如表3-7所示。

表 3-7　字符显示器的属性码

字符属性	背景属性	功　能
7	0	正常(黑底白字)
1	0	下横线
15	0	加亮
23	0	闪烁
31	0	闪烁并加亮
0	7	反视频(白底黑字)
0	0	不显示

下例是使用COLOR语句修改字符属性而实现输入提示和出错提示的。通过这个例子能够清楚地看出可用性的实现方法。

例3—21　核对密码与登录日期

```
10  ' EX3—21 Check Password
20  CLS : PRINT : PRINT
30  COLOR 7, 0 : PRINT "Enter your" ;
40  COLOR 15, 0 : PRINT "PASSWORD" ; ' highlight
50  COLOR 7, 0 : PRINT "here : " ; ' default
60  COLOR 0, 0 : INPUT PASSWORD$ ' invisible
70  IF PASSWORD$ = "SeCret" THEN 110
80  ' Blink and hightlight error message
90  COLOR 31, 0 : PRINT "Wrong password"
100 GOTO 30
110 COLOR 0, 7 : PRINT "Program continues..." ' Reverse image
120 COLOR 7, 0
```

```
130 DATE$= "6/20/84"
140 INPUT "Enter new date〔6/20/84〕" ; D$
150 IF D$= "" THEN 170
160 DATE$=D$
170 PRINT : PRINT "* * * * CURRENT DATE * * * *"
180 PRINT "* * * * * *" ; DATE$ ; "* * * * * *"
190 END
```

说明：30—50行显示出输入提示信息："Enter your PASSWORD here:"（输入你的口令），其中PASSWORD为高亮度显示。60行输入口令，不回显输入的口令，使别人看不见你的口令。70行核对口令，若不对，则以高亮度闪烁形式给出出错信息，并请你重新输入口令，否则继续执行下去。110行以反视频方式打印出："Program Continues…"（程序继续下去…）。130行把日期置成缺省值。140—150行要求输入新日期，若仅输入一个〈CR〉键表示取缺省值。170—180行打印当前日期。

（2）程序的坚固性（Robustness）

程序的坚固性是指对于用户的任意输入，无论其对错都不影响程序的执行，并且都能产生有益的结果。当然，对于错误的输入，不会得到正确的结果，正所谓"输入是垃圾，输出也是垃圾。"

改善程序坚固性的手段主要有：

* 对用户的输入进行正确性检查，对非法输入给出提示信息，并要求重新输入。
* 使用容错技术，以防止因输入错误而造成程序的运行错误。
* 对于会破坏用户的大量数据或会破坏用户文件的操作应给出警告，防止因用户的一时失误而造成不必要的损失。
* 对一问题有多种类似的回答时，应允许用户输入任意一种回答。

4．陷井技术

PC BASIC允许用户设立中断陷井，以捕俘可能发生的某些随机事件。陷井使用的一般过程是首先编制好预想事件的续元处理程序，再在程序的开头放一条陷井设置语句指出续元程序的入口。当程序运行完每一条语句，BASIC解释程序都去查找一下期待的中断事件发生了没有，如发生了，则转该事件的续元处理程序，处理完毕后，再返回被打断的程序继续执行。

允许用户程序捕俘的事件主要有程序性出错、键盘上有功能键输入、设定的时间片已到以及异步通讯器中断等。下面仅介绍前三种事件的陷井技术。

（1）出错陷井

在正常情况下，如果程序发生错误，则自动停止程序的执行并打印出错误信息。但有时程序员可利用出错陷井技术捕俘可能会发生的错误或使用错误模拟语句产生的错误点。

下面是几条与出错陷井技术有关的语句：

* ON ERROR GOTO〈行号〉 出错陷井设置语句

* ERROR n　　错误模拟语句
* V=ERR　　取当前的错误号
* V=ERL　　取当前的出错行号
* RESUME〔<行>〕　恢复到指定的行执行，缺省行号表示恢复到出错的行。若行号为NEXT，则表示恢复到出错行的下一行。

根据上述几条语句，就可以设立出错陷井。利用出错陷井编制的程序可参见例3—22。

例3—22　检查DATA语句中的数据个数

```
10 ' EX3-22 Check the data number
20 ON ERROR GOTO 100
30 N=0
40 READ D
50 N=N+1
60 GOTO 40
70 DATA 2, 1, 2, 2, 2, 3, 1, 4
80 PRINT "Data number is" ; N
90 END
100 IF ERR<>4 THEN PRINT "Error :" ; ERR : END
110 PRINT "End of data"
120 RESUME 80
```

说明：20行为出错陷井设置语句。30—60行为数据个数的计数程序。80行为输出结果。100—120行为出错陷井的续元程序。

下面是PC BASIC的常见出错信息表：

错误号	含　义	错误号	含　义
1	NEXT without FOR (没有配对的FOR语句)	15	String too long (字符串太长)
2	Syntax error (语法错)	16	String formula too complex (字符串表达式太复杂)
3	RETURN without GOSUB (没有配对的GOSUB语句)	24	Device timeout (设备超时)
4	Out of data (数据读完)	26	FOR without NEXT (没有配对的NEXT语句)
5	Illegal function call (非法的功能)	50	FIELD overflow (字段溢出)
6	Overflow (数值上溢)	53	File not found (文件未找到)
7	Out of memory (内存溢出)	55	File already open (文件已经打开)

（续表）

错误号	含　义	错误号	含　义
8	Undefined line number (行号未定义)	58	File already exists (文件已存在)
9	Subscript out of rang (下标溢出)	61	Disk full (磁盘满)
10	Duplicate definition (多重定义)	64	Bad file name (非法的文件名)
11	Division by zero (除数是零)	67	Too many files (文件太多)
13	Type mismatch (类型不匹配)	70	Disk Write Protect (写保护盘)
14	Out of string space (字符串空间溢出)	71	Disk not ready (盘未准备好)

(2) 功能键陷井

与功能键陷井有关的语句主要有：

* ON KEY(n) GOSUB <行>　当指定的功能键输入后，转续元入口。
* KEY(n) ON　允许发出第 n 号功能键中断。
* KEY(n) OFF　禁止发出第 n 号功能键中断。
* KEY(n) STOP　停止发出第 n 号功能键中断，但要记录已发生的事件，以备使用。

上列语句中n的含义是：1—10 为功能键<F1>—<F10>，11—14 表示光标上下左右移动键，15—20 为自定义的功能键。

功能键陷井有时可用来处理特殊的功能键，比如例 3—23 的功能键陷井就是用来处理光标移动键的。

例 3—23　汉字造字程序

```
10  ' EX3—23 Get Image of Chinese Character
20  ON KEY(11) GOSUB 200 : ON KEY(14) GOSUB 220
30  ON KEY(12) GOSUB 240 : ON KEY(13) GOSUB 260
40  KEY(11) ON : KEY(12) ON : KEY(13) ON : KEY(14) ON
50  DIM D%(15, 15)
60  KEY OFF : CLS : LOCATE 2,30 : PRINT " == KEY =="
70  PRINT TAB(21); "<CR>——End; <SP>——Del; Other——( * )"
75  PRINT TAB(21); " A B C D E F G H I J K L M N O P"
80  PRINT TAB(21); CHR$(218)+STRING$(33,196)+CHR$(191)
90  FOR COUNT=1 TO 16 : PRINT TAB(19); USING " # #" ; COUNT;
100 PRINT TAB(21); CHR$(179); TAB(55); CHR$(179)
110 NEXT
```

```
120 PRINT TAB(21); CHR$(192)+STRING(33, 196)+CHR$(217)
130 LOCATE 6, 23, 1, 0, 13
140 K$=INKEY$ : IF K$= "" THEN 140
150 IF K$=CHR$(27) THEN END
160 IF K$= " " THEN PRINT " " ; : GOTO 190
170 IF K$=CHR$(13) THEN 300
180 PRINT "*" ;
190 LOCATE ,POS(0)-1 : GOTO 140
200 IF CSRLIN>6 THEN PRINT CHR$(30);
210 RETURN
220 IF CSRLIN<21 THEN PRINT CHR$(31);
230 RETURN
240 IF POS(0)>23 THEN PRINT CHR$(29); CHR$(29);
250 RETURN
260 IF POS(0)<53 THEN PRINT CHR$(28); CHR$(28);
270 RETURN
300 LOCATE 24, 1, 1, 13, 13
310 FOR I=0 TO 15 : FOR J=0 TO 15
320 D%(I, J)=1
330 IF SCREEN(I+6, J*2+23)=32 THEN D%(I, J)=0
340 NEXT J, I
350 END
```

说明：20—30行设立四个光标移动键的事件陷井。40行允许发出这四个键的事件中断。50行定义一个数组以存放一个汉字的点阵。60—70行显示出提示信息：〈CR〉键用来结束字型的生成，空格键可删去一个点，其它键可生产一个点。75—120行显示一个方框，给出造字的范围。130行把光标改为闪烁的方块并置于造字区的右上角。140—190行检查用户敲入的键，分别进行处理。200—270行为陷井的续元程序。300—340行把构造好的汉字点阵存入数组D%( )中。利用这个程序构造一个“国”字的结果如下页所示。

（3）时钟陷井

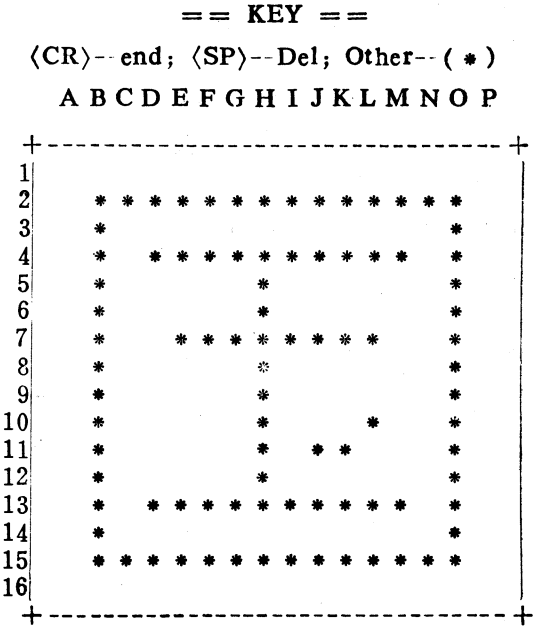
在处理与时间有关的问题时，可以使用时钟陷井。时钟陷井的设置语句为：

ON TIMER(n) GOSUB〈行号〉

其中，n的取值范围为1秒到86,400秒（24小时），表示产生时间片已到事件的时间值。

时钟陷井技术可用于定时数据采样、限时回答等环境中。它的设置过程是，编制续元程序，在ON TIMER(n) GOSUB语句中设置时间间隔，最后用TIMER ON语句启动时钟计时。当时间计到与设定的时间相同时，则发出时钟中断，由ON TIMER语句捕俘并转续元处理。这时，时钟归零重新开始计时，以发出第2次中断。这样不断往复下去，直到遇到TIMER OFF为止。

有些微机（如日本产的Ai-M16）使用第25行显示日期和时间，以方便用户。下面举一



个在 IBM PC 上实现这一功能的例子。

例 3—24　每隔一分钟在第 25 行上显示日期和时间

```
10 'EX3-24
20 ON TIMER(60) GOSUB 1000
30 KEY OFF : TIMER ON
 ⋮
1000 OLDROW=CSRLIN
1010 OLDCOL=POS(0)
1020 LOCATE 25, 40
1030 PRINT DATE$; "    " ; TIME$;
1040 LOCATE OLDROW, OLDCOL
1050 RETURN
```

说明：10 行设置时钟陷井，其时钟间隔为 60 秒（1分钟）。20 行停止显示屏幕上第 25 行的信息。1000—1010 行保存光标的当前位置。1020—1030 行在第 25 行的右部显示日期和时间。1040行把光标移回至中断前的位置上。

第五节　PC BASIC 的专用功能

1. 利用特殊字符画条形图

许多应用场合要求计算机能产生图形输出。的确，图形以其独具的直观性和形象化深受

人们的欢迎。例如，在管理信息系统中，人们喜爱采用折线图、扇形图和条形图表示统计数据，因为这比使用统计报表更为直观。

例3—25是一个利用字符设备实现条形图的程序。如果能使用图形设备（例如，图形显示器、图形打印机或绘图仪），则图的效果会更好。

例 3—25　年度销售额条形图

① 功能：在打印机或显示器上输出某公司的年度销售额条形图。程序首先提问“Print it on Printer(Y/N)？”（在打印机上输出吗？），如果回答是（Y），则从打印机上输出，否则从显示器输出。图的标题为“ANNUAL SALES TREND”（年度销售额趋势），纵坐标为1到12月的月名，横坐标为销售额（0至800）。条形图以两种不同的特殊字符交替打印或显示（注意，本例使用的打印机特殊字符是采用STAR gemini—10xf打印机的特殊字符，若使用其它类型的打印机，则需更换这几个特殊字符）。

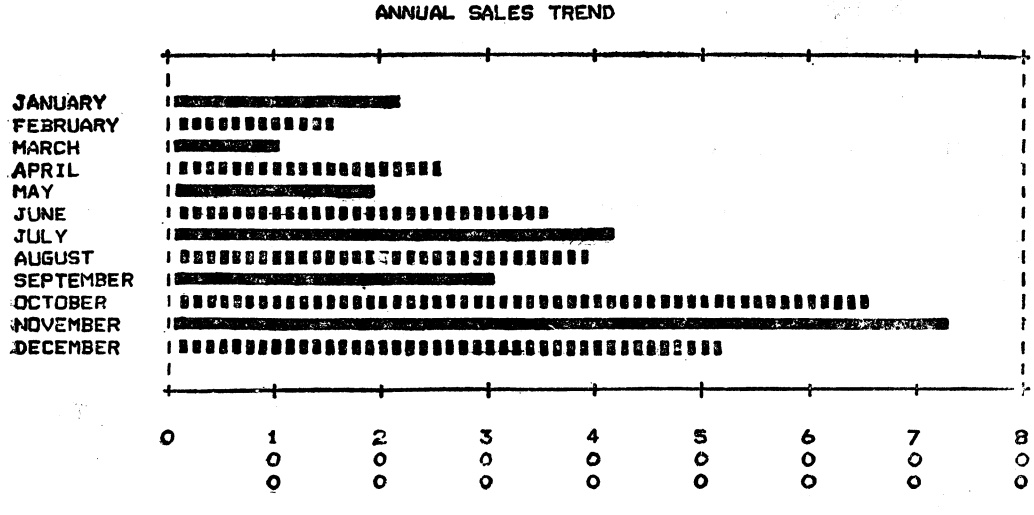
② 程序：

```
10 'EX3-25 Labeled Bar Graph Using Character Graphics
20 DV$="SCRN:" : A=197 : L=196 : C=179 : F=177 : S=219
30 PRINT : PRINT : INPUT"Print it on Printer ( Y/N ) ";M$
40 IF M$<>"Y" AND M$<>"y" THEN 60
50 DV$="LPT1:" : A=250 : L=241 : C=245 : F=234 : S=239
60 OPEN DV$ FOR OUTPUT AS #1
70 CLS : PRINT #1, : PRINT #1, TAB(28); "ANNUAL SALES TREND"
80 PRINT #1,: PRINT #1,TAB(12);
90 FOR REPEAT=1 TO 8
100     PRINT #1, CHR$(A);STRING$(7,L);
110 NEXT
120 PRINT #1, CHR$(A)
130 PRINT #1, TAB(12);CHR$(C);TAB(76);CHR$(C)
140 RANGERATIO=(76-12)/(800-0)
150 CODE=F
160 FOR MONTH=1 TO 12
170     READ MONTHNAME$, SALES
180     POSITION=INT((SALES-0)*RANGERATIO+12+.5)
190     PRINT #1, MONTHNAME$;TAB(12);CHR$(C);
200     IF CODE=F THEN CODE=S : GOTO 220
210     IF CODE=S THEN CODE=F
220     FOR COLUMN=13 TO POSITION
230         PRINT #1,CHR$(CODE);
240     NEXT
250     PRINT #1, TAB(76);CHR$(C)
260 NEXT
```

```
270 PRINT #1, TAB(12);CHR$(C);TAB(76);CHR$(C)
280 PRINT #1, TAB(12);
290 FOR REPEAT=1 TO 8
300     PRINT #1, CHR$(A);STRING$(7,L);
310 NEXT
320 PRINT #1, CHR$(A)
330 PRINT #1,
340 PRINT #1,TAB(11);0;:FOR REPEAT=1 TO 8: PRINT #1,
    STRING$(5, " " );REPEAT;:NEXT
350 PRINT #1,TAB(14);:FOR REPEAT=1 TO 8: PRINT #1,
    STRING$(5," ");0;:NEXT
360 PRINT #1,TAB(14);:FOR REPEAT=1 TO 8: PRINT #1,
    STRING$(5," ");0;: NEXT
365 PRINT #1,
370 DATA "JANUARY",210,"FEBRUARY",150,"MARCH",99
380 DATA "APRIL",250,"MAY",183,"JUNE",352
390 DATA "JULY",410,"AUGUST",390,"SEPTEMBER",300
400 DATA "OCTOBER",651,"NOVEMBER",724,"DECEMBER",516
410 IF INKEY$= "" THEN 410
420 END
```

③ 说明：20行为显示器输出条形图准备初值：设备名“SCRN：”，边框字符197、196和179，条形图字符177和219。30—40行询问是否从打印机输出，若是则把初值换成打印机的参数：设备名“LPT1：”，边框字符250、241和245，条形图字符234和239。60行打开指定的设备文件。70行输出标题。80—330行输出条形图。340—365行输出横坐标的标尺刻度。370—400行为数据，改变数据可以得到不同的条形图。

④ 结果：打印机上输出的结果如下：



2．音响与音乐

当程序需要提醒用户注意时（例如请求输入），可采用发出音响的方式提示。在游戏程序中常常加入一些音响效果，以提高游戏的趣味性。即使是一般的应用程序，也可以插入一段音乐，以解除等待中的烦恼。

PC BASIC为用户提供下列几条音响和音乐语句：

* BEEP 或 PRINT CHR$(7) 发出频率为 800Hz，音长为$^1/_4$秒的声音。
* SOUND〈频率〉,〈音长〉 发出指定频率、指定音长的声音，其中频率范围从37到32767Hz，音长范围从 0 到 65535 个时钟单位（$^1/_{18.2}$秒）。
* PLAY“字符串” 演奏由给定字符串表示的音乐。

用 PLAY 语言演奏的乐曲，必须用一种特定的音乐语言记谱。这个音乐语言由若干单字符命令组成，可表示音符的音高和音长以及演奏方式。

在这个音乐语言中，基本音符采用字母 A—G 表示，它们与简谱对应如下（C调）：

简　谱	1	2	3	4	5	6	7
音乐语言	C	D	E	F	G	A	B

升音和降音号 # 和 b，分别用“#”（或“+”）和“-”表示。音乐的八度用命令 On 表示，其中 n 的取值范围为 0 到 6。日常使用的中音用 O3 表示，提高一个八度用“>”表示，降低一个八度用“<”表示。

音乐中的音长用命令 Ln 表示，其中 n 的取值范围为 1—64，分别表示音长 1—$^1/_{64}$ 个全音。对于加点的延长音，仍用加点表示。休止符用 Pn 表示，n 为休止的长度，其含义同 Ln 命令中的 n。

此外，音乐语言中还有一些辅助命令，用来控制演奏方式。节奏用 Tn 表示，其中 n 为 32 到255（缺省值为 120）。演奏速度分正常（MN）、快速（MS）和慢速（ML）三种。演奏方式有前台（MF）和后台（MB）两种，其中后台方式可在不影响正常工作的情况下演奏音乐（不超过32个音符）。对于需要重复演奏的乐曲片段，可先把这段谱子放在一字符串变量中，再用X 命令反复引用。

如果希望在不影响正常工作的情况下演奏长于32个音符的音乐，就必须使用音乐陷井。其设置语句为：

ON PLAY(n) GOSUB〈行号〉

表示当后台音符缓冲区中音符少于 n 个时，发出中断并转续元处理。在续元程序中，依次填入每行谱子，从而实现了整首乐曲的后台演奏。

由例3—26可具体地说明后台演奏的方法。

例 3—26 故事片《城南旧事》的插曲——“送别”

① 功能：在显示随机字符图案的同时，反复演奏乐曲“送别”。程序采用音乐后台缓冲陷井技术实现连续后台演奏。

“送别”的原谱为：

送　别

1=C　4/4

5 $\overset{\frown}{35}$ i —	6 $\overset{\frown}{16}$ 5 —	5 $\overset{\frown}{12}$ 3 $\overset{\frown}{21}$	2 — — —
5 $\overset{\frown}{35}$ i · 7	6 i 5 —	5 $\overset{\frown}{23}$ 4 · 7	1 — — 0
6 i i —	7 $\overset{\frown}{67}$ i —	$\overset{\frown}{67}$ $\overset{\frown}{16}$ $\overset{\frown}{65}$ $\overset{\frown}{31}$	2 — — 0
5 $\overset{\frown}{35}$ i · 7	6 $\overset{\frown}{16}$ 5 —	5 $\overset{\frown}{23}$ 4 · 7	1 — — 0

② 程序：

```
10  ' EX3—26 Display random pattern and play music
20  PP=0
30  PLAY "MB P1P1"
40  ON PLAY(5) GOSUB 9000
50  PLAY ON
100 ' Display random pattern
110 KEY OFF
120 RANDOMIZE TIMER
150 PRINT CHR$(RND*220+32);
190 GOTO 150
9000 TEMPO=120
9010 REP$= "L4G L8E G"
9020 PLAY "MN MB T=TEMPO; O3"
9030 PP=PP+1
9040 IF PP>9 THEN PP=1
9050 ON PP GOTO 9060, 9070, 9080, 9090, 9100, 9110, 9120, 9130, 9140
9060 PLAY "XREP$;L2>C< L4AL8>C<AL2G" : RETURN
9070 PLAY "L4GL8CDL4EL8DC L1D" : RETURN
9080 PLAY "XREP$;L4>C.<L8B L4A>C<L2G" : RETURN
9090 PLAY "L4GL8DEL4F.L8<B> L2C.P4" : RETURN
9100 PLAY "L4A>CL2C< L4BL8ABL2>C<" : RETURN
9110 PLAY "L8AB>C<AAGEC L2D.P4" : RETURN
9120 PLAY "L4GL8EGL4>C.<L8B L4AL8>C<AL2G" : RETURN
9130 PLAY "L4GL8DEL4F.L8<B> L2C.P4" : RETURN
9140 PLAY "P1P1" : RETURN
```

③ 说明：20行初始化音乐演奏指针。30行休止两个全音。40—50行设立陷井并允许

捕俘音乐中断事件。100—190 行不停地显示由随机字符产生的图案。9000—9140 行为中断续元程序，其中 9000—9020 行设置初值，把演奏环境置成：中速、后台演奏形式和 120 的节奏。9030—9050 行根据当前演奏指针装入一段谱子。9060—9140 行的 PLAY 后的字符串为歌谱，其中用>和<括起来的部分需升高一个八度，而用<和>括起来的部分需降低一个八度。9060 行和 9080 行中的 XREP$ 表示把 REP$ 中的谱子插入到指定的位置演奏，这样就可以实现重复演奏指定的片段。

3．异步通讯

如果机器配有异步通讯接口板，可以使用串行接口(RS232C)与另一台IBM PC微机（或其它微机）进行通讯，也可以通过 RS232C 接口使用其它标准串行设备（例如串行打印机、汉字终端、绘图仪和数字化桌等）。

在 PC BASIC 中，异步通讯器是当作一个设备文件来使用的，它在使用前必须先用 OPEN“COM1：…”语句打开，使用后一般也要用 CLOSE 语句关闭。

OPEN“COM1：…”语句的格式是：

OPEN “COM1：〔波特率〕〔，校验〕〔，数据位〕〔，停位〕〔，RS〕〔，CS〔n〕〕〔，DS〔n〕〕〔，CD〔n〕〕〔，LF〕〔，PE〕” AS〈文件号〉〔LEN=〈记录长〉〕

其中波特率可以为 75、110、150、300、600、1200、1800、2400、4800 和 9600bps．缺省值为300bps。校验的方式有恒 0(S)、恒 1(M)、奇校验（O）、偶校验（E）和不校验(N)，缺省值为E。数据位从 4 到 8 可任选，缺省值为 7，而停止位为 2 或 1，110bps 以上的缺省值为 1。RS、CS、DS、CD、LF、PE 选项主要用于控制通讯状态和控制信号，它们的含义如下：

* RS　　禁止请求发送（RTS）。
* CS〔n〕　等待发送结束信号(CTS)n毫秒。
* DS〔n〕　等待数传机就绪信号(DSR)n毫秒。
* CD〔n〕　等待载波检测信号(CD)n毫秒。
* LF　　在回车符(0C)后自动加换行符(0D)。
* PE　　允许检测奇偶错。

例如，波特率为 9600 的设备可用语句

```
10  OPEN “COM1：9600，N，8，1，CS，DS，CD” AS #1
```

打开。

由于异步通讯器是作为文件使用的，故所有文件读写语句都可以进行通讯设备的输入和输出。但最常用的输入和输出语句是以下两条：

```
10  A$=INPUT$(LOC(1)，  #1)
20  PRINT  #1，B$
```

若希望在不停止当前程序运行的情况下接收对方发送来的信息，可采用下列陷井技术

```
10  ON COM(1)  GOSUB 500
```

```
20  COM(1) ON
 ⋮
500  REM Incoming characters
 ⋮
590  RETURN
```

来实现。关于通讯器陷井技术的详情，感兴趣的读者可参阅有关的使用说明书，这里不作详细介绍。

下面举例说明如何编制 IBM PC 的通讯程序。

例 3—27　异步通讯程序

① 功能：这个程序允许使用自行设置的波特率等参数与另一台 IBM PC 通讯，也允许用同样的方式与 PC 所配置的串行设备交换数据。程序的输出数据由用户从键盘输入，接收到的数据在屏幕上显示出来。

② 程序：

```
10 ' EX3—27 Communications
15 FOR I=1 TO 10 : KEY I, " " : NEXT
20 KEY OFF : CLS
30 FALSE=0 : TRUE=NOT FALSE : XOFF$=CHR$(19)
40 XON$=CHR$(17) : ON ERROR GOTO 500 : T=0 : ECH$= ""
50 WIDTH "com1 : " , 255
60 LOCATE 4 : PRINT TAB(23) "COMMUNICATIONS MENU"
70 LOCATE 8, 23 : PRINT "Choose one of the following : "
80 PRINT TAB(23) "1 IBM Personal Computer"
90 PRINT TAB(23) "2 Other service"
100 PRINT TAB(23) "3 End program"
110 LOCATE 16 : PRINT SPACE$(60) : LOCATE 16, 23, 1 : PRINT
    "choice" ;
120 A$=INKEY$ : IF A$= "" THEN 120
130 IF LEN(A$)=1 THEN LT=VAL(A$)
140 IF LT=3 THEN CLS : PRINT TAB(30) "— COMMUNICATION
    ENDED—" : END
150 IF LT=1 OR LT=2 THEN 180
160 PRINT : PRINT TAB(23) "Invalid choice, try again"
170 FOR I=1 TO 1000 : NEXT : LOCATE 17 : PRINT SPACE$(60) :
    GOTO 110
180 CLS : LOCATE 1, 23 : PRINT "USER DEFINED LINK"
190 IF LT=1 THEN LOCATE, 23: PRINT "TO ANOTHER IBM PERSONAL
    COMPUTER"
200 LOCATE 4, 23, 1 : INPUT "BAUD RATE" ; SPEED$
210 LOCATE 5, 23, 1 : INPUT "PARITY" ; PARITY$
220 LOCATE 6, 23, 1 : INPUT "NUMBER OF BITS PER CHAR" ; BITS$
```

```
230 LOCATE 7, 23, 1 : INPUT "NUMBER OF STOP BITS" ; STP$
240 LOCATE 8, 23, 1 : INPUT "CHARS ECHOED TO SCREEN(Y/N)" ;
    ECH$
250 LOCATE 10, 23, 1 : INPUT "DEVICE CORRECTLY(Y/N)" ;CR$
260 IF CR$= "N" OR CR$= "n" THEN 180 ELSE GOSUB 590
270 LOCATE 21, 3 : COMFIL$= "COM1 : " +SPEED$+ ", " +
    PARITY$+ ", " +BITS$+ ", " +STP$
280 COMFIL$=COMFIL$+ ", cs, ds, cd"
290 PRINT COMFIL$
300 OPEN COMFIL$ AS #1
310 OPEN "SCRN : " FOR OUTPUT AS #2
320 LOCATE , , 1
330 PAUSE=FALSE : ON ERROR GOTO 500
340 B$=INKEY$ : IF B$= " " THEN 380
350 IF MID$(B$, 2, 1)=CHR$(68) THEN 490
360 IF B$=CHR$(8) THEN LOCATE , POS(0)-1, 1 : PRINT " " ; :
    LOCATE, POS(0)-1, 1
370 PRINT #1, B$; : IF ECH$= "Y" OR ECH$= "y" THEN PRINT # 2,
    B$;
380 IF EOF(1) THEN 340
390 IF LOC(1)>128 THEN PAUSE=TRUE : PRINT #1, XOFF$;
400 A$=INPUT$(LOC(1), #1)
410 FOR I=1 TO LEN(A$)
420 IF (ASC(MID$(A$, I, 1))<31 AND MID$(A$, I,1)<>CHR$(13) )
    THEN 450
430 IF MID$(A$, I, 1)=CHR$(10) THEN MID$(A$, I, 1)= " "
440 PRINT MID$(A$, I, 1);
450 NEXT I
460 IF LOC(1)>0 THEN 340
470 IF PAUSE THEN PAUSE=FALSE : PRINT #1, XON$;
480 GOTO 340
490 CLOSE : ON ERROR GOTO 0 : GOTO 10
500 IF ERR<>24 THEN PRINT "ERR : " ; ERR : STOP
510 CLS : LOCATE 12, , 1 : PRINT "Device timeout error! "
530 GOSUB 630 : CLS : RESUME
590 CLS : LOCATE 1, 32 : PRINT : PRINT : PRINT
600 PRINT TAB(23) "Get your device or another PC ready."
610 PRINT TAB(23) "Then press ENTER to begin. " : PRINT : PRINT
620 PRINT TAB(23) "—PRESS F10 TO GO TO MENU" : PRINT
624 GOSUB 630
626 RETURN
630 A$=INKEY$ : IF A$= " " THEN 630
635 IF MID$(A$, 2, 1)=CHR$(68) THEN 490
640 RETURN
```

③ 说明：15行把<F1>—<F10>这十个功能键定义为空，以便以后使用。20—50 行 进行初始化。60—170 行显示功能清单并处理用户输入的键。180—260 行由用户设置波特率 等参数。

270—310行打开“COM1：”作为 #1文件，打开“SCRN：”作为 #2文件。320行表示光标消隐。330行设立出错陷井。340—390行把用户从键盘上送入的信息从串行口发送出去，如果用户输入〈F10〉，则停止通讯返回20行。400—450行接收对方发送的信息并从显示器上输出。490—530行为出错处理程序，对于设备超时错误打印出：“Device timeout error! ”（设备超时错！）。590—626行为提示子程序。630—640行为输入一个键的处理子程序。

④ 使用：程序运行后立即显示出下列的“菜单”：

```
COMMUNICATIONS MENU
Choose one of the following:
1 IBM Personal Computer
2 Other service
3 End program

choice
```

如果想要与另一台IBM PC微机通讯则选择1，如要使用串行设备则选择2，退出则选3。对于其它输入，屏幕上都显示出：

Invalid choice, try again {非法输入，请再输入一次}

并要求重新输入。

如果选择3，表示退出，屏幕上显示出

```
-- COMMUNICATION ENDED --

{—— 通讯结束 ——}
```

并结束这个程序。

如果选择1或2，程序都会提问波特率等参数。例如，要使用9600波特与另一台IBM PC通讯，其操作过程为：

```
USER DEFINED LINK
TO ANOTHER IBM PERSONAL COMPUTER
BAUD RATE? 9600
PARITY? n
NUMBER OF BITS PER CHAR? 8
NUMBER OF STOP BITS? 1
CHARS ECHOED TO SCREEN(Y/N)? y
DEVICE CORRECTLY(Y/N)? y
```

输入完毕后，屏幕上显示出：

```
Get your device or another PC ready.
Then press ENTER to begin.
-PRESS F10 TO GO TO MENU
```

要求准备好通讯线路，按下〈CR〉键开始进行通讯。并告诉如要返回“菜单”状态请按〈F10〉键。接着就可以进行通讯。这时，从键盘上输入的数据都送到了对方的机器，而对方发来的数据全部在显示器上显示出来。结束时，按下〈F10〉。

4．机器级语句

PC BASIC提供了一组与机器直接有关的语句，以便用来直接与硬件打交道。在介绍这组语句之前，首先说明 PC BASIC 变量的存贮方式以及 BASIC 数据文件的文件控制块的格式。

PC BASIC 存放变量的形式如下图所示。

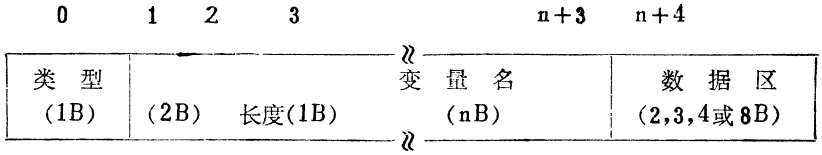


图 3—3　变量的存贮方式

图中的类型按照变量是整型、字符串型、实型和长实型，分别为 2、3、4 和 8（这个数为其数据区的长度）。因为 PC BASIC 的变量名长度允许多达 255 个字符，故变量名以不定长的方式存放。数据区用于存放数值变量的内码或存放字符串变量的长度和地址。

PC BASIC 数据文件的文件控制块与 MS-DOS 的文件控制块 FCB 不同，它的若干主要内容如表 3-8 所示。

表 3—8　数据文件的文件控制块

位置	长度	内　　容
0	1	文件的打开方式 （4＝随机文件）
1	38	MS-DOS的FCB
39	2	读写扇区号或下一记录号(随机文件)
41	1	每扇区字节数
42	1	输入缓冲中余下的数据数目
46	1	设备号(0,1—A:,B:盘；253—LPT1: 251—COM1:；　254—SCRN:; 255—KYBD:)
47	1	设备行宽
51	128	物理数据缓冲区
179	2	记录长度
181	2	当前物理记录号
183	2	当前逻辑记录号
188	n	逻辑数据缓冲区(FIELD区)

知道变量的存贮方式和文件控制块的结构后，就可以使用机器级语句直接访问这些变量或文件中的数据了。

现将几条常用的机器级语句列出：

* DEF SEG〔=<段地址>〕　　定义当前内存的段地址。
* V=PEEK(<地址>)　　读指定内存单元的内容送入V中。
* POKE <地址>, <数据>　　写数据到指定的内存中去。
* BLOAD <文件名>〔,<地址>〕　　把内存映象文件装入内存。
* BSAVE <文件名>,<地址>,<长度>　　以文件的形式保存内存中的数据。
* V=INP(<端口号>)　　读输入端口。
* OUT <端口号>, <数据>　　写输出端口。
* V=VARPTR(<变量>)　　取指定变量在内存中的地址（其格式如图3—3所示）。
* V=VARPTR(<文件号>)　　取指定文件的文件控制块始址（文件控制块的内容如表3-8所示）。

下面举例说明如何使用机器级语句编写程序。必须指出，使用这组语句是较危险的，搞不好会破坏系统区的内容，所以在使用时应当格外小心。

例3—28　保存显示缓冲区中的数据

① 功能：在显示器上显示一幅由星号组成的图案，再把这幅图案以名"PIC·DAT"记盘，随后再把这幅图案显示出来。

② 程序：

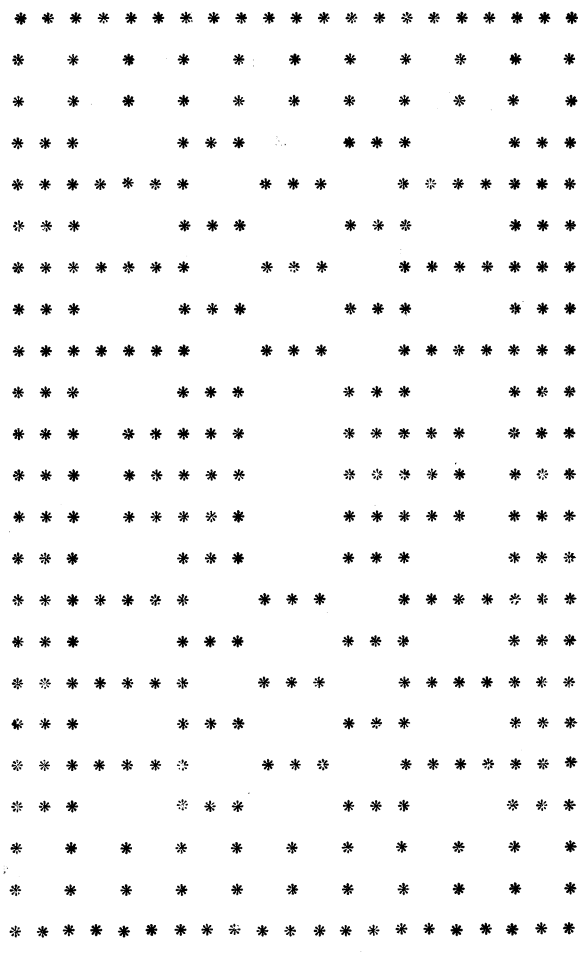
```
10  ' EX3-28 Save the pattern
20  READ X : IF X=-1 THEN 100
30  ON X GOTO 40, 50, 60, 70, 80
40  PRINT TAB(20) ; "*******  ***  *******" : GOTO 20
50  PRINT TAB(20) ; "***  *****  *****  ***" : GOTO 20
60  PRINT TAB(20) ; "**********************" : GOTO 20
70  PRINT TAB(20) ; "* * * * * * * * * * *" : GOTO 20
80  PRINT TAB(20) : "***  ***  ***  ***" : GOTO 20
90  DATA 3, 4, 4, 5, 1, 5, 1, 5, 1, 5, 2, 2, 2, 5, 1, 5, 1, 5, 1, 5,
    4, 4, 3, -1
100 DEF SEG=0
110 IF (PEEK(&H410) AND &H30)=&H30 THEN MONO=1 ELSE
    MONO=0
120 BUFSEG%=&HB000 : BUFLEN%=&H1000
130 IF MONO=0 THEN BUFSEG%=&HB800 : BUFLEN%=&H4000
140 DEF SEG=BUFSEG%
150 BSAVE "PIC.DAT", 0, BUFLEN%
```

```
160 CLS
170 BLOAD "PIC.DAT" , 0 : PRINT "Press any key" ;
180 IF INKEY$= " " THEN 180
190 END
```

③ 说明：20—90行输出一幅图案。100—110行检查显示器的类型，如果是字符显示器则把MONO置为1，否则置为0。120—130行根据显示器的类型设置显示缓冲区首址和缓冲区长度。140行把段地址置成显示缓冲区的首址。150行把显示缓冲的内容保存到盘上，命名为PIC.DAT。160行清除屏幕。170行再把文件PIC.DAT装入到显示缓冲区中，这时屏幕上又显示刚才保存的那幅图案，接着显示出："Press any key"，按下任意键就结束这个程序。

④ 结果：第2次显示出的结果为：

Press any key



5、调用机器语言子程序

有时为了更方便地使用外设或为了提高程序的执行速度，要求调用一段机器语言子程序。

下面介绍 PC BASIC 调用机器语言子程序的方法。

调用机器语言子程序的一般过程是首先编制汇编子程序并汇编成机器代码，再分配一块内存，接着把代码程序装入内存，最后由 BASIC 程序调用执行。

（1）分配内存

如果机器具有 96K 以上的内存，通常都把代码子程序按排在 BASIC 工作区之后，例如可用语句

```
10  DEF SEG= &H1700
```

把代码程序区定义到 17000H 开始的区域中。如果机器仅有 64K 内存，那么使用 BASIC 工作区的高区放置代码程序，这可以通过下列两条语句来实现：

```
10  CLEAR ,&HF000
20  DEF SEG= &HF00
```

其中 10行中的语句表示把 BASIC 的工作区放在前 60K 中，以便为代码程序空出 4K 空间。

（2）装入代码程序

把代码程序装入内存通常有两种方法，第 1 种是把汇编好的代码程序用 DATA 语句写在 BASIC 程序中，调用前用 POKE 语句装入内存。下例就是采用这种方法实现的。

例 3—29　调用汇编子程序做加法

```
10 ' EX3-29 C=A+B
20 DEFINT A-Z
30 DEF SEG= &H1700
40 FOR I=0 TO 21
50 READ J
60 POKE I, J
70 NEXT
80 SUBRT=0
90 A=2 : B=3 : C=0
100 CALL SUBRT(A, B, C)
110 PRINT A; "+" ; B ; "=" ; C
120 END
130 ' PUSH BP
135 ' MOV BP, SP
140 ' MOV SI, [BP+0A]
145 ' MOV AX, [SI]
150 ' MOV SI, [BP+08]
155 ' ADD AX, [SI]
160 ' MOV DI, [BP+06]
165 ' MOV [DI], AX
```

```
170 ' POP BP
175 ' RETF 6
200 DATA &H55, &H8B, &HEC, &H8B, &H76, &H0A
210 DATA &H8B, &H04, &H8B, &H76, &H08
220 DATA &H03, &H04, &H8B, &H7E, &H06
230 DATA &H89, &H05, &H5D, &HCA, &H06, &H00
```

说明：30行把段地址置为 &H1700。40—70行把代码程序装入内存。80行把程序的起始地址赋给 SUBRT。90行给变量 A，B 和 C 赋初值。100行调用代码子程序。110行打印计算结果。130—175是200—230行代码程序的汇编形式的注释。

把代码程序调入内存的第2种方法是使用 BLOAD 语句直接把二进制的内存映象文件装入内存。这种方法的操作过程为：

* 把汇编子程序汇编连接成 .EXE 文件。
* 用 DEBUG 装入 BASIC 的解释程序并用 R 命令记下寄存器 CS, IP, SS, SP, DS 和 ES的内容。
* 用 DEBUG 装入 .EXE 文件并记下 CS 和 IP 的内容。
* 用 R 命令把寄存器再改成装入 BASIC 时的值并用 G 命令执行 BASIC 的解释程序。
* 在 BASIC 状态下用 BSAVE 语句保存汇编子程序的映象文件。
* 装入 BASIC 程序，使用 BLOAD 语句装入内存映象文件。

(3) 调用代码子程序

代码子程序的调用也有两种方法，第1种是使用 USR(n) 函数调用，这种方法类似于其它 BASIC 代码子程序调用函数 USR，这里不再作介绍。另一种是使用 CALL 语句，该语句的格式为：

CALL 〈变量〉[(参数表)]

这里的变量代表数值变量，用来指出被调子程序的起始地址。参数表为传递给代码子程序的参数，在执行 CALL 语句时，分别压入堆栈。

代码子程序通常使用指向堆栈的寄存器 BP 来取得参数。如果有 n 个参数，则第 m 个参数的地址为 (BP)+2*(n−m)+6。

如何用 CALL 语句调用代码子程序，请看下例。

例 3—30 调用代码子程序在打印机上打印一个字符串

* BASIC 程序：

```
100 DEF SEG=&H1700
110 BLOAD "SUBRT.DAT", 0
120 SUBRT=0
130 A$= "HELLO! "
140 CALL SUBRT(A$)
150 END
```

* 汇编程序

```
1700 : 0000 PUSH BP                ; 保存BP
1700 : 0001 MOV BP, SP             ; 把栈指针赋给BP
1700 : 0003 MOV BX, [BP+06]        ; 取A$的数据区地址
1700 : 0006 MOV CL, [BX]           ; 取AS的长度
1700 : 0008 MOV BX, [BX+1]         ; 取A$内容的首址
1700 : 000B MOV AH, 5
1700 : 000D AND CL, CL
1700 : 000F JZ 1B
1700 : 0011 MOV DL, [BX]
1700 : 0013 INT 21                 ; 输出一个字符
1700 : 0015 INC BX
1700 : 0016 DEC CL
1700 : 0018 JMP 0D
1700 : 001A NOP
1700 : 001B MOV DL, 0A
1700 : 001D INT 21
1700 : 001F POP BP
1700 : 0020 RETF 2                 ; 返回BASIC
```

说明：100行把段地址置为 &H1700。110行把文件 SUBRT·DAT 文件装入内存.120行把调用地址置为 0。130—140 行调用汇编子程序输出："HELLO! "。

第六节　编译 BASIC 简介

1. 引　言

第一节中已经指出，编译 BASIC 具有两大优点——速度快和省内存。PC的编译BASIC又与解释 BASIC基本兼容，所以可利用编译 BASIC 来开发应用程序，以便提高程序的运行质量。

（1）编译 BASIC 的组成

编译BASIC 通常由以下五个文件组成：

BASCOM.COM	编译程序
LINK.EXE	连接程序
BASRUNG.EXE	运行包
BASRUNG.LIB	运行库（要带运行包）

BASCOMG.LIB　　运行库（不带运行包）

如果在连接的时候使用BASRUNG库，那末生成的目标必须在盘上具有BASRUNG.EXE的情况下才能运行，如果使用BASCOMG.LIB连接，即使不用运行包也可以运行。

（2）编译BASIC程序的开发过程

使用编译BASIC开发程序的过程如图3—4所示。首先用解释BASIC或行编辑程序EDLIN建立一个源程序文件，再用解释BASIC对程序进行调试，调试完毕后进行编译和连接，最后执行生成的.EXE文件。

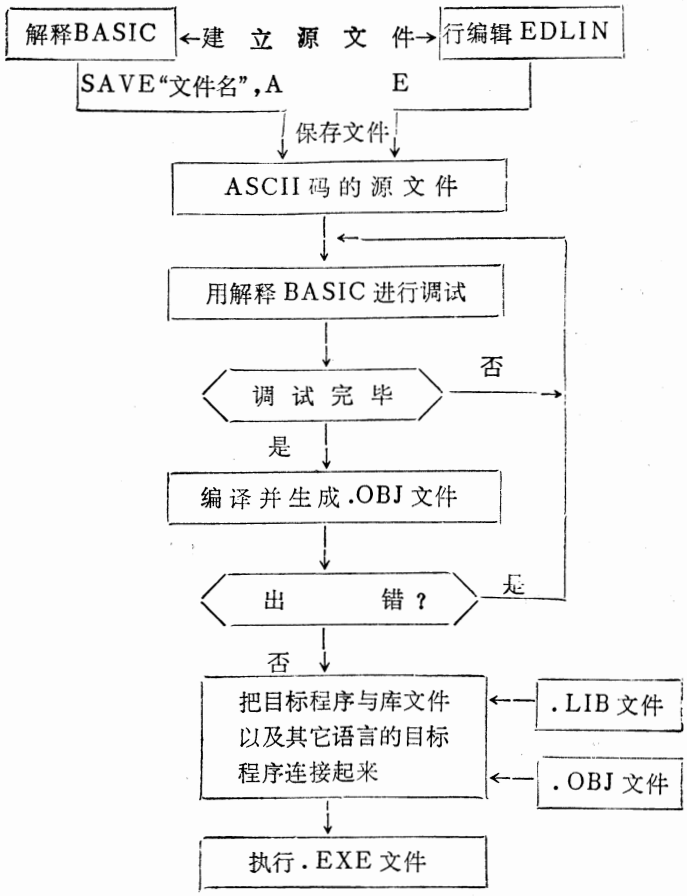


图3—4　使用编译BASIC开发程序的过程

例如，要开发一个名为PROG.BAS的程序，其操作过程为：

① 装入BASIC解释程序，编制和调试这个程序。

② 调试成功后，使用命令

SAVE "PROG", A

把程序以ASCII码的形式存盘，再用SYSTEM命令退出BASIC。

③ 在 MS-DOS 下打入命令

```
A>BASCOM PROG;
```

进行编译，产生目标文件 PROG·OBJ。

④ 用连接程序连接，打入

```
A>LINK PROG;
```

生成 PROG·EXE 文件。

⑤ 最后打入

```
A>PROG
```

执行这个程序。如果 A 盘上没有运行包 BASRUNG.EXE，则屏幕上显示出：

```
Cannot find  A : BASRUNG.EXE
Enter new driver letter :
```

这时只要输入 BASRUNG.EXE 所在盘的盘符即可。

2．编译 BASIC 与解释 BASIC 的主要区别

使用解释 BASIC 写的程序当利用编译 BASIC 编译时，应注意这两个 BASIC 的不同之处，否则有通不过的可能。下面介绍的是它们之间的几个主要区别，至于其它的不同点请参阅解释 BASIC 的使用手册。在手册中，编译 BASIC 版本栏下凡标有（＊＊）者，都表示两者有一定的区别。

（1）操作命令

下列文件操作、程序编辑和程序调试等命令，在编译 BASIC 中不允许使用。

- ＊ 程序文件操作命令：SAVE、LOAD、KILL 和MERGE
- ＊ 目录操作命令：MKDIR、CHDIR 和 REDIR
- ＊ 程序编辑命令：LIST、LLIST、AUTO、RENUM、NEW、DELETE 和 EDIT
- ＊ 程序调试命令：CONT

（2）说明语句

在编译 BASIC 中，说明语句 COMMON、DEF、DIM 和 REM 语句一般都放在程序的首部。对于几个程序模块共同使用的公共变量的说明部分，常常放在一个单独的包含文件中（见下），并在这几个程序模块前都用包含命令$INCLUDE 把包含文件嵌入。

除此以外，某些说明语句还有一些特殊的要求，比如：

- ＊ COMMON 语句必须出现在所有可执行的语句之前。
- ＊ DIM 语句中的下标范围只允许使用整型常数。
- ＊ REM语句后随 $〈命令名〉表示编译命令，例如上文中提到的包含命令可表示为：

```
10 REM $INCLUDE 'DEFINE.HD'
```

（3）程序的连接

在编译 BASIC 中，不仅可把几个 BASIC 程序或 BASIC 程序与汇编程序连接起来，还可以把 BASIC 与 PASCAL 或 FORTRAN 子程序连接起来。此外，编译 BASIC 的子程序调用语句和链接语句与解释 BASIC 也不完全相同。例如：

* CALL 语句可用来调用汇编子程序或调用 PASCAL 语言的过程。被调用程序不必在调用前装入内存，而在连接时自动装入。CALL 语句后的变量名需与汇编子程序 PUBLIC 名或 PASCAL 的过程名一致。
* CHAIN 语句不允许包含ALL，MERGE，DELETE 和〈行号〉等任选项，所有公共变量都必须用 COMMON 语句说明。

（4）其它

编译 BASIC 对解释 BASIC 还有以下一些扩充和限制：

* 不必每行都写行号。行号仅在被 GOTO 或 GOSUB 转去的行才是必须的，这时应选用编译开关/N。
* 字符串的长度允许多达 32767 个字符。
* 不允许使用 ON PLAY(n)和 ON TIMER 语句。

3．编译 BASIC 的编译开关和编译命令

编译 BASIC 还通过编译开关和编译命令提供一些特殊功能，下面分别介绍它们的使用方法。

（1）编译开关

编译程序的命令格式为：

BASCOM 〈源文件〉[，〈目标文件〉，〈列表文件〉]/〈开关〉

若在源文件名后跟随一个分号（；），表示后两个文件名缺省。常用的编译开关如表 3-9 所示。

表 3-9　编译开关

开　关	功　　能
/A	在列表文件中列出目标的反汇编清单
/D	允许使用 TRON 语句跟踪程序
/E	允许使用 ON ERROR GOTO 和 RESUME〈n〉语句
/X	允许使用 ON ERROR GOTO 和 RESUME 语句
/N	允许省略不必要的行号
/O	选用 BASCOMG.LIB库生成目标
/R	把数组按行存放
/T	把长度大于 4 的字符串放于盘上，用时再调入

例如：

A>BASCOM DEMO /N, , NUL

A>BASCOM DEMO/D;

都是合法的编译 BASIC 命令行。

（2）编译命令

编译 BASIC 允许使用编译命令来控制源文件和列表文件。编译命令的使用方法是在源程序中使用以 REM$ 开头的编译命令。常用的编译命令如表 3-10 所示。

表 3-10　BASIC 的编译命令

编译命令	功能
$ INCLUDE '文件名'	把指定的文件嵌入源文件
$ LIST+	允许/禁止生成列表文件
$ OCODE+	允许/禁止反汇编列表
$ TITLE '串'	置页标题
$ SUBTITLE '串'	置页副标题
$ LINESIZE:n	置列表设备的行宽
$ PAGESIZE:n	置页长,缺省值为 66
$ SKIP:n	走纸 n 行
$ PAGE	换页
$ PAGEIF:n	如当前页少于 n 行则换页

例如：

```
10 REM $PAGE   { 表示换页 }
20 REM $LINESIZE : 132   { 表示列表文件每行输出 132 个字符 }
```

现对 $INCLUDE 命令作一说明。编译程序在处理到这条命令时，把指定的文件嵌入到 $INCLUDE 命令所在的位置处。被嵌入的程序必须是ASCII码的源程序，它嵌入到某一程序中后，就视为该程序的一个部分。

应当注意，被嵌入程序的起始行号应大于 $INCLUDE 命令所在行的行号。此外，在被嵌入的程序中不允许再使用 $INCLUDE 命令 。

$INCLUE 命令常用来嵌入几个程序模块的公共变量说明程序。例如，模块MENU·BAS 和 PROG1.BAS 都要使用公共变量文件 COMDEF.BAS，三个程序如下：

```
* MENU. BAS
  10 REM MENU.BAS
  20 REM $INCLUDE 'COMDEF.BAS'
   ⋮
1000 CHAIN "PROG1"
* PROG1.BAS
  10 REM PROG1.BAS
  20 REM $INCLUDE 'COMDEF.BAS'
   ⋮
```

```
2000 CHAIN "MENU"
```

```
* COMDEF.BAS
100 REM COMDEF.BAS
110 DIM A(100), B$(200)
120 COMMON I, J, K, A( )
130 COMMON A$, B$( )
```

第四章　UCSD PASCAL 语言及其操作系统

IBM PC 配备有几种不同版本的 PASCAL 语言，它们分别在不同的操作系统支持下运行。UCSD PASCAL 语言是美国加州大学圣地亚哥分校在 1974 年底开发的，此后连同它的操作系统（UCSD P 系统）一起，在许多不同 CPU 的微型计算机上都得到了实现。例如 Intel8080，Z80，Mostek6502/6510，LSI11，TI9900，Intel8088/8086，M68000等。

UCSD PASCAL 语言除了由于实现方面的原因对个别地方有所限制以外，它与Jensen和 Wirth 提出的“标准”PASCAL 语言是兼容的，而且，它对标准 PASCAL 语言进行了许多方面的扩充。例如，增加了长整数和字符串等数据类型以及对它们进行操作的若干内部例行程序，引入了更适合于单用户、交互式环境的文件处理功能，提供了许多在微机上用PASCAL 语言开发大型应用软件的设施，扩充了进行并发程序设计的能力等等。

UCSD P 系统是一个功能齐全、操作方便、可移植性好的单用户、交互式操作系统，它为使用汇编语言和高级语言（如 PASCAL、FORTRAN77、BASIC、MODULA2等）开发系统软件和应用软件提供了一个良好的操作环境。P 系统本身大部分也是用 UCSD PASCAL 语言实现的，层次结构极好，在不同系统、不同机种上进行移植或者扩充都很方便。现在，它已是许多八位和十六位微型计算机的主要操作系统之一了。

本章先介绍 UCSD PASCAL 语言，重点放在与标准 PASCAL 不同的那些扩充成分上，并给出一些实例予以说明。关于标准 PASCAL 语言的基本概念、语法及语义等方面的描述，参考书籍很多，为节省篇幅，本章不作细述。第四节介绍 UCSD P 系统（Ⅳ.0 版），由于它的许多基本概念与 MS-DOS 相类似，所以本章只对它的主要功能和特点侧重进行一些讨论。

第一节　UCSD PASCAL 语言的数据类型及其操作

UCSD PASCAL 语言提供比标准 PASCAL 语言更为丰富的一些数据类型，并扩充了许多内部的过程和函数对它们进行操作处理。本节将把这些数据类型分成标准类型、纯量与子域、构造类型、文件类型、指示字类型以及专用类型等六类进行讨论。

文中凡大写粗体均为UCSD PASCAL 的保留词，小写而有下横线标出的都是 UCSD PASCAL 已预先定义过的标识符。

1．标准数据类型

标准数据类型指的是在 UCSD PASCAL 中已经预先说明的那些数据类型，如 integer（整数），real（实数），long integer（长整数），boolean（布尔）和char（字符）。

（1）整数

① 整数的格式

一个整数的值用一个十进制数字序列来表示，前面可以冠以正负号。整数的范围是－maxint..maxint。maxint是PASCAL预先说明的一个常数，它的值依赖于具体实现，在UCSD PASCAL中是32767。

② 整数的比较操作

两个整数可以进行下列合法的比较操作：＝（等于），＜＞（不等于），＞（大于），＞＝（大于等于），＜（小于），＜＝（小于等于）。

③ 整数的运算

整数前面可以加“＋”号或“－”号。两个整数可以进行＋（加法），－（减法），＊（乘法），**DIV**（整除），**MOD**（整除之余数）五种运算。

④ 关于整数的例行程序

这里的“例行程序”指的是UCSD PASCAL内部预先定义好的函数和过程（也叫内部函数或内部过程）。它们以整数作为自变量，结果也是整数。这样的例行程序有两个：函数abs（取绝对值）和函数sqr（平方）。

（2）实数

① 实数的格式

有两种表示法：科学表示法如12.12E－4，12e4等；普通表示法如123450.0，－12.4，－0.3等。

② 实数的比较操作

与整数一样。但建议不要对两个实数（或一个实数和一个整数）作相等比较。

③ 实数的运算

对实数（包括整数）可进行＋（加），－（减），＊（乘），/（除）四种运算，其结果也是实数。

④ 关于实数的例行程序

PASCAL提供了一组自变量为实数或整数而取值为实数的函数，它们是：abs（取绝对值），sqr（平方），sqrt（开方），sin（正弦），cos（余弦），arctan或atan（反正切），exp（指数），ln（自然对数），log（常用对数）以及pwroften（十的幂次）。最后两个函数是UCSD PASCAL的扩充。log是常用对数，不必解释。pwroften（I）函数的自变量I应为整数，函数的值是十的I次方，如：

```
pwroften(0)=1.0
pwroften(5)=100000.0
```

（3）长整数

长整数是UCSD PASCAL对整数的一个扩充，它们可以用来表示比maxint大或者比－maxint小的整数，当然也能表示在范围－maxint..maxint中的整数。

① 长整数的格式

类型为长整数的常量可以直接在常数说明中给出，如：

```
CONST Rydberg=10973731;
```

而类型为长整数的变量需要用integer〔n〕来说明之，其中n是长度属性，它是一个<=35的无符号整数，表示该长整数可能包含的最大位数。例如：

```
VAR  Big_count : integer〔10〕;
```

指出变量Big_count是一个位数不多于10位的长整数。

② 长整数的比较操作

与整数一样，参加比较的可以是长整数或整数。

③ 长整数的运算

除MOD运算无定义之外，其它运算都与整数一样。运算结果赋值给长整数变量时，若位数超过预先说明的长度n，则产生溢出错误。

④ 长整数的例行程序

长整数本身没有例行程序，函数trunc和str可以把长整数作为自变量，以进行类型转换，这两个函数将在下面讨论。

⑤ 关于长整数的附注

* 长整数类型用在过程首部的参数表中时，下面的表示法是错误的：

```
PROCEDURE Large ( Big_sum : integer〔10〕) ;
```

为说明某参数的类型为长整数，应按下法进行：

```
TYPE Digit10=integer〔10〕;
  :
PROCEDURE Large ( Big_sum : Digit10 ) ;
```

* 长整数不能作为函数的值，在实际使用中解决的办法请参见第三节中的例4—3。

（4）布尔

① 布尔类型的格式

布尔的值或者是true或者是false，false小于true。

② 布尔类型的比较操作

两个布尔类型的操作数可以进行下列的比较操作：=（相等），<>(不等)，<=（蕴涵），>=（被蕴涵），>（不蕴涵），<（不被蕴涵）。

③ 布尔运算

前面所述的整数、实数、长整数的比较操作都会产生布尔类型的值。下面三种运算的操作数只能是布尔类型，其结果也都是布尔类型，即NOT（非），AND（与），OR（或）。

④ 有关的例行程序

函数odd、eof、eoln及unitbusy的值都是布尔类型，这在下面有关部分再作介绍。

⑤ 附注

与标准PASCAL不同，UCSD PASCAL的任何内部过程或函数，都不能输出布尔类型的值。

（5）字符

字符类型用来表示单个的字符，字符常常用来作为集合或者是字符串的元素。UCSD PASCAL 使用的是ASCII 字符集。

① 字符类型的格式

可打印字符放在单引号中表示，如 'a' 、 'B' 、'1' 等，不可打印字符可以借助于内部函数 chr 来表示。

② 字符的比较操作

因为字符集是有序的，所以=，<>，>，>=，<，<=六种比较操作均可对字符进行。

③ 字符的运算

类型为字符的变量可以赋值，但对字符不进行任何运算。

④ 字符类型的例行程序

chr 可以把整数转换为字符，ord 则相反，它把字符转换为整数。pred 和 succ 函数均可用字符作自变量，其值也是一个字符。

（6）类型的转换

程序中不同类型的值可以进行转换。有时转换是自动进行的。例如，整型值赋给实型变量时，自动转换为实型；赋给长整数变量时会自动转换成长整数型。在多数情况下，类型的转换需要依靠系统提供的一些例行程序来完成。这些例行程序是：

① 实数→整数

使用函数 trunc 和 round 来进行，前者只是简单地截去小数部分，后者还进行舍入的处理。如：

```
trunc ( 12.7 ) =12
trunc ( -12.7 ) =-12
round ( 12.7 ) =13
round ( -12.7 ) =-13
```

② 长整数→整数

也使用 trunc 函数来进行，但自变量类型应为长整数。

③ 整数→布尔类型

odd (I) 当 I 为奇数时，函数值为 true，否则为 false。

④ 字符→整数

ord (C) 函数值为字符 C 在 ASCII 字符集中的序号。

⑤ 整数→字符

chr (I) 函数值为 ASCII 字符集中的第 I 个字符。如：

```
chr ( 32 ) =' '
chr ( ord ( 'A' ) ) ='A'
```

⑥ 整数（或长整数）→字符串

str（L，S）

例如，执行 str（12345，S）后，字符串 S 的值为'12345'。

2．纯量和子域

UCSD PASCAL 的纯量和子域的定义及其操作与标准 PASCAL 一样，下面简述之。

（1）纯量

纯量类型是可枚举的值的有序集合，纯量类型的值是用其类型定义中的标识符标记的。

六种关系运算（=，<>，>，>=，<，<=）都可以对纯量进行。

（2）子域

一个类型可以定义为另一纯量类型的子域，只要指定该子域的最小值和最大值。例如，下面类型说明中 Winter 就是纯量 Month 的子域，f_index 是整数的子域。

```
TYPE  Month=(Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct,
              Nov, Dec);
      Winter=Jan..Mar;
VAR   Summer: Jun..Aug;
      tag: Winter;
      f_index: 1..77;
```

注意，标准类型中，字符、整数和布尔都是纯量，但实数和长整数不作为纯量，因此不能用来构造子域。

（3）纯量和子域的例行程序

有三个内部函数可用于对纯量类型和子域类型的值进行操作：

* ord（V）　　纯量 V 的序号
* succ（X）　　纯量或子域类型变量 X 的后继
* pred（X）　　纯量或子域类型变量 X 的前启

3．构造类型

构造类型由其它类型组合而成。刻划一个构造类型特征的是其成分的类型和它的构造方法。这里先介绍数组、串、记录和集合四种构造类型，文件留在后面作专门讨论。

（1）数组

①数组的格式

UCSD PASCAL 规定，数组的成分类型（也叫“基类型”）可以是除文件类型以外的任意类型，维数可以是一维也可以是多维。数组说明中，**ARRAY** 前若冠以保留词 **PACKED**，则表示该数组在内存以紧缩形式存贮。标准 PASCAL 要求紧缩的字符数组至少有两个元素，而 UCSD PASCAL 无此限制。

② 数组的比较操作

UCSD PASCAL 可以对两个大小相同且基类型也相同的（但不是紧缩形式的）数组进行=（相等）和<>（不等）两种比较操作，而这是标准PASCAL不允许的。其余的比较操作，必须对数组的元素逐个地进行。

③ 数组运算

大小相同、基类型也相同的数组可以一次进行赋值，长度相同的紧缩字符数组也可以赋值，这是UCSD PASCAL的扩充。

数组元素的运算依赖于这些元素的基类型。

④ 关于数组的例行程序

UCSD PASCAL为了进行快速的数组处理，提供了四个例行程序。虽然它们主要是对紧缩的字符数组进行处理，但由于实现时并不作类型检查，因此具有一定的通用性，不过使用时必须十分注意。

* fillchar (Destination, Length, Character)

此过程是把字符Character或整数的低八位填入从Destination开始的存贮区中去，共填入Length次。Destination可以是数组，也可以是数组中某一元素。例如，设buf是一个数组 **PACKED ARRAY** 〔0..19〕 **OF** char，则执行fillchar (buf, 10, 'e') 后，数组buf中前十个元素的值均为字母'e'。

* moveleft (Source, Destination, Length)

把Source中的Length个字节传送到Destination中去，传送是从左向右进行的，即按照下标增加的方向进行。例如，设Src:='1234567', Dst:='*******'，则执行过程moveleft (Src, Dst, 5) 之后，Dst中应为'12345**'。

* moveright (Source, Destination, Length)

同moveleft，但传送方向是从右向左进行的。例如，对于上面的数组Src，执行moveright (Src, Src〔3〕, 4) 之后，Src中应为'1231234'。但执行的若是moveleft (Src, Src〔3〕, 4)，则结果为'1231231'。

* scan (Length, 〈部分表达式〉, Source) : integer

这是一个函数，参数中的所谓部分表达式，它由一个"="或"<>"符号再加上一个字符表达式组成。scan函数的功能是：对Source进行扫描，直到满足部分表达式或已扫描完了Length个字符为止（Length为负数时，表示从右向左扫描）；回送给函数的值就是从扫描开始到扫描停止的位移量。

例如：设test:='For he on honey dew hath fed';

则：index:=scan (10, ='h', test〔11〕) ; (index之值为0)

index:=scan (10, ='h', test〔12〕) ; (index之值为9)

index:=scan (−9, ='h', test〔9〕) ; (index之值为−4)

index:=scan (10, <>'h', test〔11〕) ; (index之值为1)

（2）字符串

字符串类型（也称串类型）及其操作是标准PASCAL没有的，这是UCSD PASCAL的扩充。

① 字符串的格式

一个字符串变量的值就是一个字符序列，字符串常数则为一个用单引号括出的字符序列。

字符串的最大长度，如果没有说明时，则为80；如果用方括号给出长度属性，则可达255个字符。例如：

```
heading : string;     （最大长度为80个字符）
graphline : string[200];     （最大长度可达200）
abbrev : string[3];     （最大长度仅为3个字符）
```

注意，串的实际长度是在运行时动态地决定的。

② 字符串的比较操作

字符串是有序的，任何比较操作对字符串均可进行。字符串的排序按通常的字典顺序进行。

③ 字符串的运算

除了可对字符串作赋值之外，通常的运算均不能直接对字符串进行。串中的字符可以用下标来指出，下标的范围只允许从1到该串的实际长度。

④ 有关字符串的例行程序

UCSD PASCAL提供四个函数和两个过程来处理字符串。

* concat（Source1, Source2, …, SourceN）

这是一个函数，其值为一个新的串，它是所有串表达式Source1, Source2, …, SourceN的毗连。例如：

```
concat ( 'All in', concat(' a garden', ' green…' ) );
```

其值为：

```
'All in a garden green…'
```

* copy （Source, Index, Size）

本函数之值是得到串变量Source的一个子串，它从Source[Index]开始，长度为Size个字符。例如：

```
copy ( 'There is a long…', 7, 9 );
```

其值为：

```
'is a long'
```

* delete (Destination, Index, Size)

这是一个过程,它把字符串变量 Destination 中从第Index 个字符起到第 Index+Size-1 个字符止全部删去。设字符串 Dest:='is a long' ，则执行过程

```
delete ( Dest, 5, 5 );
```

之后 Dest 的值为'is a'。

* insert (Source, Destination, Index)

本过程用来把串表达式 Source 插入到串变量 Destination 中去，插入是从Destination〔Index〕开始的。设 Short:=' old' ; Long:=' There are two desks.' ，则执行过程

```
insert ( short, Long, 14 );
```

后 Long 的值改变为:

'There are two old desks."

* length (Source)

函数的值就是字符串表达式 Source 的现行实际长度。

* pos (Pattern, Source)

pos 函数的值就是字符串 Pattern 在串 Source 中第 1 次出现时的位置，例如:
设

```
Long:='There are two desks.';
Short:='re';
```

则

```
pos ( Short, Long )
```

的值等于 4。

注意，若 Short 不在 Long 中出现，则函数值为 0。

(3)集合

① 集合的格式

说明一个类型为集合时可以使用:

TYPE 〈标识符〉=**SET OF** 〈基类型〉

其中基类型必须是纯量或子域类型。集合的值用枚举集合中的元素放在方括号“〔”和“〕”中来表示。例如下面的两个变量均为集合类型

```
ASC=SET OF char;
```

```
Lower_case=SET OF 'a' .. 'z' ;
```

它们的值可以是：

```
ASC:=['A'.. 'Z' , '0' .. '9' ];
Lower_case:=[ 'a' , 'b' , 'e' ];
```

UCSD PASCAL 允许集合中最多只能有 4080 个元素，若集合由整数的子域构成，则不论下界如何，子域的上界不能超过 4079。

② 集合的比较操作

有五种比较操作可对集合进行：=（两个集合相等），<>（不相等），>=（一个集合包含另一个），<=（被包含）以及 IN（属于某集合）。其中比较操作 IN 是对一个元素及一个集合进行的，格式如下：

〈元素〉 IN 〈集合〉

若〈元素〉存在于〈集合〉中，则结果为 true（真），否则为 false（假）。

例如：is_letter:= '4' IN['A' .. 'Z' , 'a' .. 'z']；执行的结果，布尔变量 is_letter之值为false 。

③ 集合的运算

对集合可进行三种运算，如+（集合的并），*（集合的交），以及-（集合的差）。所谓两个集合的差A-B，指的是所有在 A 中但不在 B 中的元素的集合。

无论是集合的比较还是集合的运算，它们都只是在基类型相同的情况下才合法。

④ 集合的例行程序

系统没有提供任何内部函数或过程对集合进行处理，也无法把集合的值按标准的方法进行输出。

（4）记录

① 记录的格式

UCSD PASCAL 的记录类型的定义，除了字段的类型不能是文件之外，它与标准 PASCAL 是完全一致的，严格的定义请读者参考有关资料，下面是一个实例：

```
TYPE
  Complex=RECORD
            real_part,
            imaginary_part : real
          END;
  Student=RECORD
            name : string[30];
            score1, score2 : integer;
            grade : 'A' .. 'F' ;
            repeat : boolean
          END;
```

```
VAR
    Class : ARRAY[1..50] OF Student;
        ⋮
```

一个记录中的元素（称为“字段”）可以用〈记录名〉.〈字段名〉来引用，例如：

```
Class[20].name:='Wang Min';
Class[43].grade:='A';
```

为了使程序更易阅读，可以通过 **WITH** 语句（开域语句）及单纯的〈字段名〉来对记录中的元素进行处理，例如：

```
WITH Class[n] DO
  BEGIN
     name:='Zhang Hua';
     score1:=90;
     score2:=95;
     grade:= 'A' ;
     repeat:=false;
  END;
```

② 记录的比较操作和运算

两个同样类型的记录（且不是紧缩的）可以进行=（等于）和<>（不等于）两种操作，其它比较操作则不能进行。

对记录的整体不能做任何运算，而对其字段所施行的运算，则由它们的类型来决定。

③ 记录变体

一个记录可以有变体成分，从而允许记录中的同一个字段，在不同情况下作为不同类型的变量来处理。这在某些情况下特别方便，读者可以参考第三节中的例 4—7。下面是两个具有变体记录的实例：

```
TYPE  Entry=RECORD
                CASE  head : boolean OF
                  true : ( number : integer ) ;
                  false : ( identify : string);
              END;
      choice= ( addr, bits ) ;
      trix=RECORD
             CASE  choice  OF
               addr : ( locn : integer ) ;
               bits : ( bmap : PACKED ARRAY[0..15] OF boolean ) ;
           END;
```

4 文件类型

文件是PASCAL程序用来与外部环境进行通讯的唯一媒介。UCSD PASCAL有内部文件、外部文件和设备文件三个概念之分。内部文件的概念与标准PASCAL是基本一致的，而后两者，UCSD PASCAL却有许多新的扩充。

（1）内部文件

文件是许多同类型成分的一个串行流。文件类型在程序中可以说明如下：

TYPE 〈标识符〉=**FILE OF** 〈基类型〉

UCSD PASCAL规定文件成分的基类型可以是除文件类型以外的任何其它类型。实际上，UCSD PASCAL规定，任何构造类型的元素均不能为文件类型。例如下面的seismic和enrollment就是两个文件：

```
VAR
  seismic: FILE OF integer;
  enrollment : FILE OF Student;
```

程序内部可以对其成分直接进行处理（如赋值）的文件叫做“内部文件”，简称文件。在UCSD PASCAL中已预先说明了三种文件类型，即行文文件、交互文件和无结构文件。

行文文件类型在内部已定义为：

```
TYPE text=FILE OF char;
```

交互文件也是一种text文件，但当它用内部例行程序read、readln和reset处理时，与一般text文件略有不同。

无结构文件在说明时不必给出文件成分的基类型，例如：

```
VAR  no_structure : FILE
```

是一个无结构文件，这种文件只能由UCSD内部的例程blockread和blockwrite来处理。

在标准PASCAL中，只有两个预先说明了的文件（标准文件）：input和output，它们的类型均为行文文件。而UCSD又增加了一个：keyboard文件，并且规定这三个文件的类型都是交互文件。

（2）外部文件

PASCAL程序与外部环境（键盘、显示器、打印机、磁盘等）的通讯必须通过文件操作来实现。由于操作系统往往把输入输出设备与磁盘上的数据文件统一处理，所以，程序的外部环境都以操作系统概念下的文件形式出现，这就是PASCAL所谓的“外部文件”。

外部文件的名就是本章最后一节所要介绍的UCSD操作系统下的设备（文件）名或磁盘文件名。例如：

```
CONSOLE :   （控制台设备）
SYSTEM :    （控制台设备，但输入无回显）
PRINTER :   （打印机设备）
```

REMIN：　（通讯输入）
REMOUT：（通讯输出）
Mydisk：Sample.Text　（磁盘上的行文文件）
#4：Example.Data　（磁盘上的数据文件）

为了实现UCSD PASCAL程序与外部环境的通讯，程序中必须进行如下的操作：

① 定义内部文件类型，说明内部文件变量；

② 把内部文件与指定的外部文件建立关联（注意文件类型要匹配）；

③ 对（内部）文件进行操作处理（取/存文件成分等），以实现程序与外部环境的通讯。

在某些情况下，①和②在程序中都可以缺省。例如，使用内部过程read和write来进行文件操作时，若其中不指明是对哪一个文件进行操作，则系统就认为是对已预先说明的标准文件input和output（交互文件类型）进行操作，并且把它们与外部文件CONSOLE：（控制台设备）建立关联。所以程序的实际效果就是在控制台上输入或输出数据。

（3）文件操作

PASCAL程序中的文件，概念上是一卷磁带的抽象，它每说明一个文件变量F，系统总是按照其成分的类型，自动引入一个缓冲器变量F^，它正好可以容纳文件中的一个成分，好象是磁带上的一个窗口。程序对文件中数据的处理，总是通过这个窗口（缓冲器变量）来进行的，窗口在磁带上的位置也由程序控制。为了反映窗口相对于文件的位置，PAS CAL提供了两个内部函数。

下面介绍UCSD PASCAL用于处理文件的一组内部函数和过程，其中参数F表示内部文件变量的名，Ext_file表示外部文件名。

① eof（F）

如果窗口已移出文件末端，则函数eof（F）之值为true，此时缓冲器变量F^无定义。

如果F的类型为交互文件，则当程序对文件F读入〈ETX〉字符（即Ctrl—C）时，eof（F）之值成为true。

② eoln（F）

F只能是行文文件或交互文件类型，当程序中对文件F读入字符〈CR〉（回车符）时，eoln之值为true，F^的内容为“空格”。

文件名F缺省时，指标准文件input。

eof为true时，eoln必定为true。

③ reset（F，Ext_file）

把外部文件Ext_file与F建立关联，打开Ext_file文件，若F不是交互文件，则读入文件中第1个成分送入F^；若Ext_file为空文件，则F^无定义且eof（F）为true。

若F为交互文件，虽然Ext_file不空，即eof（F）为false，也不把第1个成分读入

F ^ 。

如果 Ext_file 文件不存在或文件已打开、或设备不联机，则会引起程序运行出错。

如果 Ext_file 为只写文件（如打印机），而 F 的类型又非交互文件，则也引起运行出错。

④ rewrite (F, Ext_file)

本过程主要目的是用来建立一个新的文件。它把外部文件 Ext_file 与内部文件 F 建立关联，若 Ext_file 文件在磁盘上不存在，则用其名建立一个新的文件；若已存在，则建立一个临时文件供程序执行期间进行操作，操作完毕后临时文件可以取代原先已存在的 Ext_file 文件，也可改名，也可丢弃不用，这将取决于下面介绍的 close 过程。

若 Ext_file 指的是脱机设备或已打开的文件，则引起运行出错。

reset 和 rewrite 中的第 2 参数均可缺省，这时，它们等价于

```
reset ( F, 'F' )
rewrite ( F, 'F' )
```

从而与标准 PASCAL 保持兼容。

⑤ close (F 〔, option〕)

本过程的任选部分 option 有四种：normal, lock, purge, crunch。无任选部分时相当于第 1 种任选。

close (F, normal) 用来关闭与 F 关联的由 rewrite 过程所打开的那个磁盘外部文件，删除临时文件，原始文件不受影响。

close (F, lock) 作用同上，但原始文件被删去，临时文件冠以外部文件名被保存。

close (F, purge) 与F关联的外部文件，若为磁盘文件，则删去；若为设备文件，则令其变成脱机状态。

close (F, crunch) 作用与 close (F, lock) 相似，但保存下来的外部文件将截短，get、put、read 或 write 最后一次访问的文件成分将成为文件中的最后一个记录。

注意，PASCAL 程序正常运行完毕时，系统将对所有已打开过的文件执行 close (F, normal) 操作，但这对已关闭的文件毫无影响。

⑥ get (F)

只有当 eof (F) = false 时才能使用本过程。它把文件 F 中的下一个成分取入 F ^ 中，好象窗口在磁带上向前移动了一个记录位置。如果文件已在末端，则使用 get 后，F ^ 中无定义，eof (F) 成为 true。

⑦ put (F)

本过程用来把缓冲器变量 F ^ 中的一个文件成分存入到文件 F 中去。标准 PASCAL 规定，只能在 eof (F) 为 true 时才能使用 put (F)，即新的文件成分只能“添加”到文件尾

端去，而UCSD PASCAL无此限制，允许在文件的任何位置上进行put操作，换言之，UCSD PASCAL允许对文件进行随机的修改。

下面是关于上述文件操作的一个程序片断：

```
VAR Data : FILE OF real;
    S : real;
     ⋮
    S:=0.0;
    reset(Data,'Example.data');
    WHILE NOT eof(Data) DO BEGIN
      S:=Data^+S;
      get(Data)
    END;
```

⑧ read(F, Item1, Item2, …, ItemN)

本过程用来从内部文件F中读取字符，把它转换为一定类型的值，顺序地赋给变量Item1, Item2, …, ItemN。

文件F必须是行文文件类型（即文件的类型应为text, interactive或FILE OF char之一）。如果文件名F缺省，则指的是标准文件input，并且与外部文件CONSOLE：（控制台）建立关联。

变量Item可以是一个或多个。若从文件中读入的数据类型与Item的类型不一致时，发生运行错误。Item的类型只能为整数（或其子域）、长整数、实数、字符（或其子域）或字符串，其它类型均不允许。

UCSD PASCAL的read过程，对于文件F是交互文件类型（interactive）还是标准的text类型，其功能略有不同。对于F是interactive类型时，read(F, ch)，相当于

```
get(F); ch:=F^;
```

而对于文件F为标准的text类型时，read(F, ch)相当于

```
ch:=F^; get(F);
```

⑨ readln(F, Item1, Item2, …, ItemN)

除下面一些具体规定之外，与过程read的功能完全一样。

* 变量Item可以是一个或多个，也可以没有。
* 在所有变量赋值之后，期望读入一个回车字符<CR>，才能使readln执行结束。

因为字符串的值总是以<CR>结束的，所以，从文件中为字符串变量赋值时，必须使用readln过程，并且一次只能有一个Item参数。

⑩ write(F, Value1, Value2, …, ValueN)

本过程把值Value1, Value2, …, ValueN顺序地写到文件F中去，关于文件F的类

型和值 Value 的类型的规定都与 read 过程一样。如果文件名 F 缺省，则指的是标准文件 output，并且与设备文件 CONSOLE：（控制台）建立了关联。值 Value至少需要有一个。

⑪ writeln (F, Value1, Value2, …, ValueN)

与 writeln 一样，但 Value 可以一个也没有，且在输出了所有的值之后，它向文件 F输出一个<CR>字符。

⑫ page (F)

向行文文件 F 写入一个走纸字符（chr (12) ）。

⑬ seek (F, Index)

文件 F 的类型不能是 text、interactive 或 **FILE OF char**，也不能是无结构文件，否则会产生运行错误。

Index 是一个整数。本过程用来使文件窗口移动到 F 文件的第 Index 个记录上。文件F的记录序号从 0 开始，若 Index 小于 0 或大于文件 F 的最大记录序号，则此后的 get (F) 或 put (F) 将会引起 eof (F) = true。

seek 操作只起定位作用，它并未把 F 中的记录读入 F^，为此，seek 后必须执行get (F) 或 put (F)。

UCSD PASCAL 扩充了 seek 内部过程之后，便可以实现对文件的随机存取。

（4）无类型文件

UCSD PASCAL 可以定义某个内部文件的类型为无类型文件，系统不对这种文件引入缓冲器变量，即没有窗口位置的概念。它必须通过 reset 或 rewrite 与某外部文件建立关联，文件成分的读写只能通过下面的两个专用的内部过程来进行。

① blockread (F, Buffer, Count, Relblock)

这是一个函数。它从无类型文件 F 所关联的外部文件中读出 Count 个块（每块 512 字节），送入数组 Buffer（其元素可以是任何类型），Relblock 是开始读出的起始块号，它可以缺省，缺省时表示从现行的块号读起。

函数值是实际读出的块数，正常情况下它应等于 Count，否则，或者是读操作有错，或者是文件已读完。

② blockwrite (F, Buffer, Count, Relblock)

与 blockread 函数的功能完全相似，但执行的是把 Buffer 中的数据向文件 F 所关联的外部文件写入 Count 个块。

5. 指示字类型

前面介绍的数据类型，所对应的变量都是静态变量，即它们的存贮单元是在编译时分配的。变量也可以在运行时进行分配内存，这叫做“动态变量”。动态变量通过指示字值在程

序中被引用。

（1）指示字类型的格式

指示字类型可以说明如下：

TYPE 〈标识符〉＝^〈类型标识符〉；

指示字可以进行赋值，也可以进行＝和＜ ＞两种比较操作。

（2）指示字的应用

UCSD PASCAL 中指示字的概念与标准 PASCAL 没有差别。下面是利用指示字来构造一个链表的程序片断：

```
PROCEDURE Example;
  TYPE
      Linklist=^Node;
      Node=RECORD
             Data : integer;
             Next : Linklist;
             END;
  VAR
     Head, Temp : Linklist;
     Element : integer ;
  BEGIN
     Head:=nil;
     read ( Element ) ;
     WHILE NOT eof DO
       BEGIN
         new ( Temp ) ;
         Temp^.Data:=Element;
         Temp^.Next:=Head;
         Head:=Temp;
         read ( Element )
       END;
  END;
```

例子中的动态变量的类型是 Node，它是一个记录，该动态变量并没有标识符，在程序中是通过指示字变量 Head（和 Temp）对它进行处理的。开始时指示字变量 Head 赋于的值是 nil，表示动态变量尚未分配内存，然后通过内部过程 new 为动态变量申请分配内存，接着为这个变量（记录类型）的两个字段赋值，不断循环，直到输入赋值完毕。从程序可以看出，动态变量的内存分配是在程序执行过程中，用内部过程 new 借助指示字来进行的。

6．专用类型

UCSD PASCAL 还定义了两种专用类型，它们用于处理并发的进程，这是标准PASCAL所没有的。

* 进程标识类型 processid
* 信号灯类型 semaphore

它们的含义及使用请参见第二节。

第二节　UCSD PASCAL 程序的构造

这一节对 UCSD PASCAL 程序的构造，它的各种主要成分，它在存贮器管理及并发程序设计方面的扩充，以及它的各种编译命令进行介绍。与上一节一样，凡与标准 PASCAL 完全一致的部分，这里不再细述，请读者参阅有关的资料。

关于语法的描述，这里将使用扩充的巴科斯范式（EBNF）来进行。例如，**REPEAT**语句（直到语句）的语法可描述如下：

直到语句＝ “**REPEAT**”
〔语句{“；”语句}〕
“**UNTIL**”布尔表达式

其中，凡引号“　”中的成分都应直接出现在程序中，方括号标出的是可有可无的部分，而花括号中的内容则表示可以不出现，也可以出现一次或多次。

1．UCSD PASCAL 程序

（1）程序的轮廓

一个 UCSD PASCAL 程序，其轮廓可以用 EBNF 描述如下：

“**PROGAM**”程序名〔“（”文件名{“，”文件名}“）”〕“；”
〔使用说明部分〕
〔标号说明部分〕
〔常数定义部分〕
〔类型定义部分〕
〔变量说明部分〕
〔例行程序说明部分〕

“**BEGIN**”

〔语句{“；”语句}〕

“**END**”“.”

程序中要用到的每个标识符（名），除了系统已预先说明的以外，都应在程序中按次序进行说明。UCSD PASCAL 仅允许一种情况例外，即下面要介绍的所谓“提前说明”。

程序首部中的文件名在 UCSD PASCAL 中不必给出，即使象标准 PASCAL 一样给出

的话，也被忽略不计。

例行程序可以有三种：过程、函数和进程，它们在下面再介绍。

程序中的使用说明部分，指的是本程序需要使用的其它经过单独编译而得到的 UNIT。

（2）UNIT 和单独编译

UCSD PASCAL 程序中的某一部分或几部分可以单独进行编译，并且被分别“包装”在 UNIT 之中。以后，它们就可以由程序或者别的 UNIT 来使用。

单独编译有许多优点。例如，大型程序不再由于微型机内存容量太小而无法编译，程序修改后不必再把整个程序重新编译一次，一部分程序可以为许多程序所公用等等。

单独编译是通过 UNIT 来进行的，UNIT 的语法描述如下：

“UNIT” 程序单位名 “；”

“INTERFACE”

〔使用说明部分〕
〔常数定义部分〕
〔类型定义部分〕
〔变量说明部分〕
〔过程和函数首部〕

“IMPLEMENTATION”

〔使用说明部分〕
〔标号说明部分〕
〔常数定义部分〕
〔类型定义部分〕
〔变量说明部分〕
〔过程和函数说明部分〕

〔“BEGIN”

〔初始化语句〕
〔“***；”
结束语句〕〕

“END” “.”

一个 UNIT 由接口部分（INTERFACE）和实现部分（IMPLEMENTATION）组成。接口部分所宣布的常数、类型、变量和例行程序都是使用该UNIT 的程序或另一个 UNIT（叫做宿主程序或宿主UNIT）可以直接使用的；实现部分所宣布的常数、类型、变量和例行程序等均局部于该 UNIT，而宿主程序或宿主 UNIT 不可使用。

注意，接口部分的过程或函数只需给出其首部（包括参数表），而在实现部分，它们的首部必须再次给出（但不包括参数表），同时给出其对应的分程序（过程体和函数体）。

实现部分的初始化语句和结束语句都是可选的，它们必须用＊＊＊相互隔开。初始化语句是在宿主程序或宿主 UNIT 开始执行之前就被执行的，而结束语句则在所有宿主程序和宿主 UNIT执行完毕之后才被执行。

UNIT 经过单独编译后得到的目标代码并不能独立运行，它们总是被另外一些程序或 UNIT 所使用。为了使用 UNIT，宿主程序或宿主 UNIT 必须通过“USES 说明”（使用说明）来使用它们。USES 说明的语法是：

“USES”〔“(＊$U” 文件名“＊)”〕 UNIT 名{“,”〔“(＊$U” 文件名“＊)”〕UNIT 名}

其中，文件名指的是 UNIT 目标代码所在的磁盘文件名，若 UNIT 已在系统程序库中，则可不必再指出。USES 说明在宿主程序中必须紧跟在程序首部之后，即在所有其它说明之前。在宿主 UNIT 中，USES 说明需紧跟在 INTERFACE 和 IMPLEMENTATION 之后。

若 USES 说明中使用的几个 UNIT 相互间又有使用关系，例如 UNIT First_level 使用了 UNIT Second_level，则 USES 说明必须按下面给出的次序来写出所使用的 UNIT，否则会发生编译错误。

```
USES  Second_level, First_level;
```

关于 UNIT 的例子，可参见第三节中的例4—11。

2．例行程序

UCSD PACAL 有三种例行程序，调用它们的方式各不相同。过程的调用由语句来进行，该语句就是这个过程的名及参数表（叫做“过程语句”）；函数是在表达式中调用的；而进程则必须使用内部过程 start 来启动。下面对 UCSD PASCAL 的例行程序的一般情况及过程与函数的语法作些介绍，进程将在本节中第5段再专门讨论。

（1）过程说明与函数说明

UCSD PASCAL 的过程说明及函数说明与标准 PASCAL 基本一致，差别仅在于UCSD PASCAL 对于形式参数的类型有一定限制，即形式参数的类型不允许再是过程或者函数。另外，UCSD PASCAL 还限制每个过程或函数的局部变量空间不能超过 16383 个字。

某些情况下，两个过程（或函数）会互相调用，或者有时为了使得程序的可读性更好一些，UCSD PASCAL 可以提前说明某个过程或函数。提前说明的方法是只给出过程或函数的首部（包括参数表），然后用保留词 FORWARD 来代替本应紧接着出现的过程体或函数体（分程序）。此后，在程序的适当位置上，再对该过程和函数作第2次说明。例如：

```
PROCEDURE  grab1 ( X : integer ) ; FORWARD;
PROCEDURE  grab2 ( X : integer ) ;

  { 过程 grab2 的说明部分及过程体 } ;

PROCEDURE  grab1;

  { 过程 grab1 的说明部分及过程体 } ;
```

注意，提前说明的所有过程和函数需集中在一起，它们的第2次说明也要集中，且第2次说明时，过程或函数首部不应再包含参数表。

（2）例行程序段

程序运行的时候，往往并不需要整个程序自始至终全部都放在内存，特别是当它带有许多例行程序的时候。这些例行程序可以在调用时再装入内存，用毕后再被别的例行程序“换出”内存，从而达到在微型机小内存中运行大程序的目的。这种在使用时才装入内存，用毕后又可被其它程序覆盖的例行程序，UCSD PASCAL 中叫做“例行程序段”。

把例行程序宣布为一个独立的“段”的方法，是在程序或 **UNIT** 中的 **PROCEDURE**、**FUNCTION** 或 **PROCESS** 之前冠以保留词 **SEGMENT**。例如：

```
SEGMENT PROCEDURE Initialize;
  BEGIN

    { 过程体 }

  END;
```

把一个例行程序独立成为一个段的做法，并不改变程序的功能，影响的仅仅是程序的执行速度和对存贮器的需求。

UCSD PASCAL规定一个程序最多可以包含255个段。所有独立成段的例行程序都必须在一般的例行程序之前先说明。如果例行程序段要调用一般的例行程序，则后者必须使用前面所说的方法进行提前说明。

例行程序无论是否独立成段，它们都可以包含独立成段的例行程序。下面是例行程序段的一个例子：

```
PROGRAM Gole;
  SEGMENT PROCEDURE St;
    ⋮
    BEGIN
     ⋮
    END;
  PROCEDURE Myndal( Flak: integer ); FORWARD;

  { 这是提前说明的一个过程，因为下面的例行程序段中要用到 }

  SEGMENT FUNCTION Mond( Part, Whole: real ): integer;
    ⋮
    BEGIN
     ⋮
    END;
  PROCEDURE Myndal;
    ⋮
```

```
PROCEDURE Early;
 ⋮
   SEGMENT PROCEDURE Late ( i, j: real );
    ⋮
      BEGIN
       ⋮
      END { Late } ;
    BEGIN
     ⋮
    END { Early } ;
     ⋮
BEGIN
 ⋮
END { Myndal } ;
BEGIN
 ⋮
END { Gole } .
```

（3）外部例行程序

UCSD PASCAL 程序也可以使用汇编语言例行程序，当然这些汇编语言例行程序的编制应该符合 UCSD PASCAL 的调用规程和参数传递规程，且预先已被汇编成目标代码。这样的汇编例行程序就叫做“外部例行程序”。使用外部例行程序的好处有：提高程序中关键部分的执行速度，实现与特殊硬件的低级接口等等。

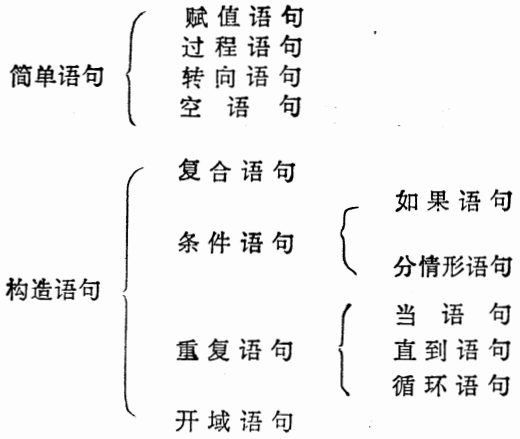
在程序中说明一个例行程序为外部例行程序非常简单，只要给出该例行程序的首部，然后紧跟以保留词**EXTERNAL**即可。例如：

```
PROCEDURE Native_code ( n: integer ) ; EXTERNAL;
FUNCTION Speed ( rush : real; direct : boolean ) : real; EXTERNAL;
```

使用外部例行程序的 UCSD PASCAL 程序，编译后还必须进行连接操作（Link），把汇编程序的目标码与主程序代码进行连接，然后才能正确地在机器上运行。关于连接操作将在第四节操作系统中介绍。

3. 语句

语句用来标记算法的动作，并称为是可执行的。在标准 PASCAL 语言中，语句可以分类如下：



UCSD PASCAL语句的分类、语法和语义几乎与标准 PASCAL 完全一样，只有转向语句和分情形语句略有不同。为了节省篇幅，下面只对转向语句和分情形语句进行介绍。

（1）转向语句

转向语句的语法可以用 EBNF 描述如下：

转向语句＝“**GOTO**”标号

其中标号是范围在 0‥9999 之间的一个整数，它应局部于转向语句所在的那个分程序（例行程序或主程序体）。标号和转向语句必须在同一个分程序之中，这一点比标准PASCAL的规定更加严格。

执行了转向语句之后，程序将从对应的有标号语句处继续运行。有标号语句的语法是：

有标号语句＝标号“：”语句

一个好的 PASCAL 程序一般不应当出现转向语句，否则会破坏程序的模块性。但有时（例如遇到出错情况时）又需要这样做。为此，UCSD PASCAL提供了两个内部过程来允许程序的紧急结束。

＊exit （参数）

这个过程有两种调用方式。当参数为程序名或保留词 **PROGRAM** 时，程序的运行立即结束，控制返回到操作系统；当过程 exit 使用在例行程序中，且参数为例行程序名时，则它将引起该例行程序的结束，关闭所有已打开的文件，并立即返回到调用程序。如果这个例行程序是一个进程，则该进程立即结束。关于 exit 的使用，请参看第三节中例4—5和例 4—11。

＊ halt

调用内部过程 halt 将引起程序立即结束，并给出运行出错，就象在键盘上按下〈break〉键一样。如果该程序是在调试程序 Debugger 控制下运行的，则 halt 使得程序结束，控制返回到Debugger。

（2）分情形语句

分情形语句（**CASE**语句）是条件语句的一种，它的语法为：

分情形语句= “CASE” 表达式 “OF”

常数表 “：” 语句 { “；” 常数表 “：” 语句 } 〔 “；” 〕

“END”

其中，表达式必须计算出一个纯量或子域类型的值，常数表由一个或多个其类型与表达式的值相同的常数组成，相互之间用逗号隔开。一个常数不能在两个以上的常数表中出现。

UCSD PASCAL 规定，分情形语句中至少要有一个语句前面具有常数表。另外，若表达式的值与所有常数表中的常数一个也不匹配时，CASE 语句被跳过，就好象是一条空语句一样。

4. 输入输出与存贮器管理

（1）输入输出操作

UCSD PASCAL 的输入输出操作与标准 PASCAL 一样，都是通过一些内部的过程和函数对文件进行处理实现的。不过，UCSD PASCAL 中文件的概念有了扩充，用来处理各种文件的内部过程和函数也有了增加。这些都已在第一节中介绍过，这里不再重复。

除了通过对文件的处理实现 PASCAL 程序的输入输出操作之外，UCSD PASCAL程序也能直接控制物理设备进行读、写等操作，这是通过调用一组内部过程和函数来实现的，这一组内部过程和函数包括：

* unitclear
* unitread
* unitwrite
* unitstatus
* unitwait
* ioresult

第三节中例4—9是如何使用 unitread 和 unitwrite 的一个实例，供读者参考。其余过程和函数的说明请查阅有关手册，这里不再细述。

（2）存贮器管理

UCSD PASCAL 程序运行的时候，有三个动态数据结构被用来管理存贮器，它们是栈（stack）、堆（heap）和代码池（codepool）。

栈用来存贮静态变量和表达式计算时的中间结果，也用于过程和函数的调用。

堆实际上也是一个“后进先出”的结构，它用来存贮动态变量，下级进程的栈，以及不可重定位的那些代码段。

代码池是一组活动代码段的集合。通常，一个程序或一个 UNIT 被编译后产生一个代码段。若宣布其中的某些例行程序独立成段，则它们的目标代码就由若干段组成。除了程序所对应的主段之外，其它代码段可以换进换出，这些都是在代码池中进行的。

UCSD PASCAL 程序运行时存贮器的状态如图4—1所示。

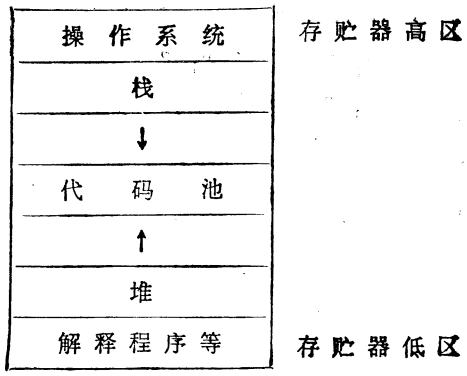


图 4—1 程序运行时内存的布局

UCSD PASCAL 为存贮器管理提供了如下一些内部过程和函数，除了 new 和dispose之外，其它都是标准 PASCAL 所没有的。

* new (Pointer)

参数 Pcinter 的类型是指示字，本过程的功能是在堆中为 Pointer 所指示 的那种类型的变量分配存贮空间，并且，把 Pointer 的值置为该变量的位置。程序中，每个动态变量只有通过 new 申请内存后才能参与运算。

* dispose (Pointer)

把指示字 Pointer 所指着的动态变量 Pointer^ 占用的内存归还给系统，同时把指示字 Pointer的值置为nil (空)。

* varnew (Pointer, Count)

这是一个函数，它为 Pointer 所指示的变量类型分配 Count 个 字 (Count 为 整 数)，若分配成功，函数的值应等于 Count，如果系统的存贮器不够分配，则函数值为 0。

* vardispose (Pointer, Count)

本过程必须与函数 varnew 配对使用，两个参数也必须完全相同。它用来把分配 给动态变量 Pointer^ 的 Count 个字归还给系统。

* mark (Pointer)

本过程用来在堆上做一个标志，它与下面的过程 release 配用。

* release (Pointer)

指示字 Pointer 是指向 Heap 的一个标志。执行本过程后，把堆中自上次 做过该 标志以来所分配出去的空间全部一次归还给堆。mark 和 release 的参数均应为指向整数的指示字。

* memavail

这个函数没有参数，它的值是存贮器中当前尚保留的未分配使用的字数，即栈与代码池之间的字数加上堆与代码池之间的字数之和。由于代码池中可能有某些段能被换出，所以其值可能比存贮器总的自由空间小一些。

* varavail （Segment_list）

参数Segment_list是段名表，本函数的值为一整数。其意义是假定段名表中所有的段都装入内存的话，内存还有多少个字可以使用。

* memlock （Segment_list）
* memswap （Segment_list）

memlock把段名表中指出的那些段锁定在存贮器中，直到执行了过程memswap之后，才允许把它们换出去（实即被覆盖）。

5. 进程管理

UCSD PASCAL具有进程管理的一些设施，因而可以实现并发程序设计。所谓“进程”，是指能够与主程序同时执行的那些过程。由于IBM PC只有一个CPU，这里所说的“同时”执行，实际上是指进程与主程序分享CPU。使用进程在某些情况下是非常有效的，例如中断的处理、外围设备的驱动等。

一个进程被启动以后，它将持续运行直到结束，或者直到它被中断，或者直到它必须等待某些事件的发生。UCSD PASCAL的操作系统，对于具有相同优先权的“就绪”进程，并不做任何“时间片”或“进程切换”之类的控制。

（1）进程说明

进程说明与过程说明非常类似，仅仅首部有所不同。下面是进程首部的EBNF描述：

“PROCESS”进程名〔“(”〔“VAR”〕名表“:”类型名{“;”〔“VAR”〕名表“:”类型名}“)”〕“;”

例如：

PROCESS Example (VAR Result: integer; VAR Pause: semaphore);

一个进程必须在程序中说明，它不能在过程、函数或其它进程中进行说明。

（2）进程的启动

进程不能象过程或函数那样被调用，它要通过一个内部过程start才能启动。一个进程可以被启动多次，它们各自独立运行，需要时也可相互通讯（同步）。下面是内部过程start的说明。

* start （〈Process call〉, Proc_id, Stacksize, Priority）

四个参数中〈Process call〉表示进程名及所需要的实在参数表，其它三个参数都是可选的：可变参数Proc_id的类型是进程标识，系统将为进程的这一次引出而分配给Proc_id一个值（整数），这在程序调试时很有用处；Stacksize是整数类型，用来指出在堆中应为这个

进程的堆栈分配多少个字，如果缺省，则分配 200 字；Priority 也是整数，范围在0··255之内，它用来给出该进程这一次引出的优先权，如果缺省，则为 128。

（3）进程的同步与互斥

当两个以上进程合作完成某项任务时，例如甲进程读磁盘文件，另一个乙进程把它打印出来，则甲、乙两进程必须分别等待某个事件发生后（缓冲器被出空和缓冲器被填满）才能继续执行下去，这种情况就是进程的“同步”。

相反的情况是，甲、乙两个进程可能都要使用控制台，在甲进程使用控制台期间，应设法不让乙进程也去使用控制台，这种情况叫做进程的“互斥”。

进程的同步与互斥实质上是同样的问题。它们可以通过类型为信号灯（semaphore）的变量及有关的一些内部过程来解决。

类型为信号灯（semaphore）的变量由一个队列（队列中是等待这个信号灯变量的进程）和一个在子域 0··maxint 中的计数值所组成。当信号灯变量的值为 0 时，队列中所有进程都必须等待，无法继续执行下去。当该信号灯变量的值不为 0 时，则队列中排在最前面的一个进程就可以继续执行，不必再等待。

信号灯变量不能赋值，也不能象普通类型的变量那样进行运算或处理。它的使用只与下面的四个内部过程有关。

① seminit（Sem, Initial）

参数 Sem 是类型为信号灯的变量，Initial 是 0··maxint 之间的整数。本过程把Sem置成初态，即把与它相关的队列置为空，计数值置为 Initial。本过程通常在主程序中使用。

② signal（Sem）

如果信号灯变量 Sem 的队列中没有进程处于等待状态，则本过程使 Sem 的计数值增加 1；如果 Sem 为 0，且队列中有进程正处于等待状态，则把队列中排在最前面的一个进程释放出来，使之成为“就绪”进程。若该进程的优先权高于正在运行的进程的话，系统还需要进行进程的切换。

③ wait（Sem）

如果Sem 的计数值不为 0，则把它减 1，调用 wait的进程可继续执行下去。否则，Sem的计数值仍保持为 0，调用 wait的进程不能继续运行，它将变成等待状态，进入与Sem相关的队列之中排队等候，直到其它进程使用 signal（Sem）时才有可能被“唤醒”。

④ attach（Sem, Ivec）

本过程使系统的中断信号与信号灯变量建立一定的关联，使得系统中每产生 Ivec 所对应的中断事件时，就自动地调用一次 signal（Sem）过程，从而达到并发地进行中断处理的目的。参数 Ivec 是系统规定的中断向量的值（整数）。为了拆除某中断信号与信号灯变量已建立起来的关联，可以使用 attach（nil, Ivec）来达到目的。

为了具体地说明信号灯变量的使用，下面给出一个程序片断，其功能就是实现本段一开

始所提出的甲进程（读磁盘内容送入缓冲器）与乙进程（把缓冲器内容送出打印）之间的同步。程序中设计了两个信号灯变量，一个用来表示有缓冲器已装满，另一个表示有缓冲器已出空。甲进程的名叫做 Fillbuffer，乙进程叫做 Sendbuffer。缓冲器的数目可以不止一个。关于信号灯变量如何用来实现并发的中断处理，可参考第三节中的例 4—10。

```
PROGRAM Bluff;
  CONST N=｛缓冲器的数目｝;
  VAR buff_full, buff_avail : semaphore;
  PROCESS Fillbuffer;
    BEGIN
      REPEAT
        wait (buff_avail);
          ⋮
          ｛选择一个空缓冲器，并把磁盘读入数据填入｝
          ⋮
        signal (buff_full)
      UNTIL false
    END;
  PROCESS Sendbuffer;
    BEGIN
      REPEAT
        wait (buff_full);
          ⋮
          ｛选择一个满缓冲器，并把其中数据送出打印｝
          ⋮
        signal (buff_avail)
      UNTIL false;
    END;
  BEGIN ｛主程序｝
    seminit (buff_full, 0);
    seminit (buff_avail, N);
    start (Fillbuffer);
    start (Sendbuffer);
      ⋮
  END.
```

6. 编译命令

UCSD PASCAL 的编译程序对源程序的编译需要进行两趟扫描：第 1 趟扫描时会输出显示每个例行程序的名字及行号，每编译一行，屏幕上输出一个点“·”；第 2 趟扫描将给出每个目标代码段的段名，每个点“·”表示一个例行程序。

UCSD PASCAL 编译得到的目标代码并不是 IBM PC 所使用的 8088 CPU 的机器码而是一种“伪码”（P_code），它必须通过解释程序才能边解释、边运行。这种做法虽然使得程序的运行速度明显地降低，但开发出来的软件却具有极好的可移植性。原则上，它们的源程序不必改写，也不用重新编译，其目标代码就可以在使用同样版本 UCSD 操作系统的其它机器上正确运行，即使是不同的 CPU 也没有影响。

为了在编译过程中进行适当的功能选择和控制，UCSD PASCAL 编译系统允许在源程序中插入各种编译命令（Compiler options）。编译命令在源程序中以“伪注解”的形式给出，即：使用{ }或(* *)给出。但其中第 1 个字符必须是美元符号 $。例如：{ $I+ }，(*$Q−*)，{ $I−，Q+ }。下面介绍几种常用的编译命令。

* I（输入输出检查）

正常情况下，编译程序在每个输入输出语句（unitread和unitwrite例外）之后都要生成一段测试程序，用来检查输入输出操作正确与否。使用 I− 命令，则表示不必为输入输出生成测试程序，输入输出操作的正确与否，由程序通过函数 ioresult 的值来判断，并且自行做出处理。I+ 则表示恢复生成测试程序。关于 I 命令的使用实例请参看第三节的例 4—7。

* L（列表）

L+ 命令使编译程序在系统的列表设备（通常为打印机）上输出源程序条文，L− 命令的功能则相反。L 后面若跟以文件名，则表示源程序输出到该文件。

* Q（无编译信息输出）

Q+ 命令使得编译程序在编译期间除了出错信息外，不再输出任何信息。Q− 则相反。

* R（范围检查）

R− 命令表示编译程序不要为数组元素的下标是否出界，子域变量的值是否超出范围，赋值语句类型是否正确等生成检测的代码。这样，目标程序运行可以稍快一些。不过，除非速度的要求很严格，并且源程序又彻底地调试过了，否则，不要使用R− 这条命令。

* R2 或 R4（实数长度）

R2 命令指出实数的长度（内部表示）为32个二进位，而R4 命令则指出实数的长度为64位。本编译命令必须在保留词 PROGRAM 或 UNIT 之前就使用。

* I〈文件名〉（包含文件）

如果由于操作系统中编辑程序缓冲区容量的限制，一个大的源程序无法被包含在一个文件之中，它不得不划分为几个较小的文件存放在盘上，或者源程序中需要嵌入一段现成的程序，这时就可以使用这个命令了。本命令通知编译程序，把〈文件名〉所指出的一个文件（当然是源程序的一个组成部分）从磁盘上读入，嵌入本命令出现之处，对其进行编译，完成后再回到源程序中原处继续编译下去。

* U〈程序库名〉（使用库中的UNIT）

使用说明（ USES ）中被使用的UNIT，如果不在系统程序库 SYSTEM · LIBRARY 中的

话，必须用本命令指出该 UNIT 所在的程序库名或它的目标程序名。

* U（使用操作系统中的UNIT）

U—命令表示用户程序可以使用属于操作系统的一些 UNIT。

编译命令还有一些，这里就不介绍了。最后要指出的是，编译程序缺省的命令状态为：

{ $R+, I-, L-, U+ }

也就是说，如果不特别指出的话，编译程序总是不产生源程序列表输出，不得使用操作系统中的一些 UNIT，对数组元素、子域变量等都要进行范围检查，并对输出操作的结果进行测试。

第三节 UCSD PASCAL程序十二例

本节向读者介绍 12 个 PASCAL 程序设计的实例，它们几乎都是完整的程序。其中大部分能在 IBM PC 机上正确地运行。少数例子对所有 PASCAL 语言都是一致的，但大多数却又具有 UCSD PASCAL 语言专门的一些特点。与此同时，关于程序的功能、运行结果、程序设计的特点等也将作适当的解释。介绍这些程序实例的目的有两个：一是为了具体地描述 UCSD PASCAL 语言的一些特点，二是从软件工程的要求出发，给出 PASCAL 程序的样板。它们在程序的模块化、可读性、可维护性等方面，都可作为用户自己在开发程序时的参考。

例 4—1 分解因子

程序功能：从键盘上读入一个整数，计算出它的质因子。

程序：

```
PROGRAM Factors ;
    { Computes prime factors of an integer read from keyboard }
    VAR
        n, factor : integer ;
    BEGIN
        write ( ' enter number to factor : ' ) ;
        readln ( n ) ;
        factor:=2 ;
        WHILE n>1 DO
          IF n MOD factor = 0
            THEN
              BEGIN
                write ( factor , '' ) ;
                n :=n DIV factor ;
              END
            ELSE factor :=factor+1 ;
        writeln ;
```

```
END.
```

运行结果：

```
enter number to factor : 36
2  2  3  3
enter number to factor : 1719
3  3  191
enter number to factor : 210
2  3  5  7
enter number to factor : 16384
2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

说明：

（1）UCSD PASCAL 程序的程序首部不必指出需要打开的文件。

（2）程序中要适当写一些注释，尤其是在程序（或子程序）开始之处，说明一下程序（或子程序）的功能和作用很有好处。

（3）DIV 是两个整数相除的运算符，其运算结果是截去小数部分而保留的整数部分。MOD 也是适用于整数的运算符，a MOD b就是a除以b后所得的余数。

例4—2　检查字符串是否回文

程序功能：检验一个字符串是否是“回文”，即顺读和倒读都一样的字符串，如MADAM，123321等。

程序：

```
PROGRAM Palin_1 ;
  { Palin_1 tests to see if string s is a palindrome ( reads the same
  forwards and backwards ) . }
  VAR
    s : string ;
    i : inte er ;
    is_palindrome : boolean ;
  BEGIN
    write ( ' enter strin to test : ' ) ;
    readln ( s ) ;
    is_palindrome : =true;
    FOR i: =1 TO length ( s ) DIV 2 DO
      IF s[i]< >s[length ( s ) +1-i]
        THEN is_palindrome : =false ;
    IF is_palindrome
      THEN writeln ( ' a palindrome' )
      ELSE writeln ( ' not a palindrome' ) ;
```

```
END.
```

运行结果：

```
enter string to test : aba
  a palindrome
enter string to test : curious
  not a palindrome
enter string to test : able was I ere I saw elba
  a palindrome
```

说明：

（1）string 是 UCSD PASCAL 的变量类型之一，它用来表示一个字符串。在说明某变量为字符串类型时，可以不指明长度（如同本例），此时最大长度为 80 个字符；也可用方括号指明长度（如，s：string〔20〕；）。允许的最大长度为 255 个字符。

（2）UCSD PASCAL有一组关于字符串的内部函数可供程序设计时使用，length(s) 就是其中之一，它表示字符串 s 的长度。

（3）从键盘上输入一个字符串时，程序中使用语句 readln(s)。由于一个字符串的输入必须以“回车”键作为结束，所以输入语句必须使用 readln 而不能用 read。

（4）变量的名字应尽可能与它的内在含义相符，例如 is_palindrome 就是一个好名字。

例 4—3　阶乘计算

程序功能：计算 1 到 20 的阶乘。

程序：

```
PROGRAM Fact_Test ;
  TYPE
    long_int=integer〔35〕;
  VAR
    n : integer ;
    fact : long_int ;
  PROCEDURE Factorial ( n : integer ; VAR result : long_int ) ;
    { computes the factorial of n and returns in result }
    VAR
      i : integer ;
    BEGIN
      result := 1 ;
      FOR i := 2 TO n DO
        result := result * i ;
    END ; { Factorial }
  BEGIN
```

```
    writeln ('n  n!');
    FOR n:=1 TO 20 DO
      BEGIN
        Factorial(n, fact);
        writeln (n:2,' ', fact);
      END;
  END.
```

运行结果：

```
 n n!
 1 1
 2 2
 3 6
 4 24
 5 120
 6 720
 7 5040
 8 40320
 9 362880
10 3628800
11 39916800
12 479001600
13 6227020800
14 87178291200
15 1307674368000
16 20922789888000
17 355687428096000
18 6402373705728000
19 121645100408832000
20 2432902008176640000
```

说明：

（1）UCSD PASCAL 允许使用长整数，方法是在 integer 后面紧跟一个长度属性，即用方括号指出该整型数的位数。本例中 integer〔35〕表示长度为 35 位的整型数。

（2）函数的值不能为长整数，所以本例设计了一个过程 Factorial，利用其中的变量参数result来存放类型为长整数的阶乘的计算结果。

例 4—4　类型转换

程序**功能**：这是两个子程序，前者为函数，它把类型为字符串的“正整数”转换为整型的正整数，例如 ASCII 码形式的'12345'被转换成为内部表示的 12345；后者为前者的逆过程，但设计成过程的形式。

程序：

```
  FUNCTION Str_to_Int (s: string): integer;
    {converts s to an integer}
    VAR
      i, result: integer;
    BEGIN
```

```
    result:=0;
    FOR i:=1 TO length(s) DO
      result:=result*10+ord(s[i])-ord('0');
    Str_to_Int:=result;
  END; {Str_to_int}
PROCEDURE Int_to_Str(n:integer; VAR s:string);
  {converts integer n to a string returned as s}
  BEGIN
    s:='';
    WHILE n>0 DO
      BEGIN
        s:=concat(' ',s);
        s[1]:=chr(n MOD 10+ord('0'));
        n:=n DIV 10;
      END;
  END; {Int_to_Str}
```

说明：

（1）ord（x）函数的作用是计算出变元 x 在其类型所定义的值的集合里的序号。

（2）UCSD PASCAL的内部函数 concat 用来把几个字符串毗连在一起 得到 一个 新的字符串。本例中，s:=concat(' ',s) 语句的目的是先让字符串 s 左 端扩充一个 虚字 符，然后再通过随后的赋值语句使之更换为某个具体的字符。 直接向字符串 s 添加一 个 字 符的做法，如

```
s:=concat (chr (n MOD 10+ord ('0')),s);
```

是不对的，因为传递给 concat 的参数的类型不正确。

例 4—5　菜单选择

程序功能：

这是UCSD P 系统中使用“点菜单”方式进行人机会话的 一 个样板程序。系统给出一行操作提示信息（一组命令）等待用户输入命令的首字母，然后即转去执行相应的命令解释程序。

程序：

```
PROGRAM Prompt_Test;
  TYPE
    charset=SET OF char;
  FUNCTION Prompt(line: string;
      legal_commands: charset): char;
    {Prompt prints line, then waits for a
     character in legal_commands to be typed.
```

```
        Its uppercase eguivalent is returned. }
    VAR
      ch : char;
    BEGIN
      REPEAT
        write ( line ) ;
        read ( ch ) ;
        writeln;
        IF ch IN ['a' ..'z']
          THEN ch : =chr ( ord ( ch )
            - ord ( 'a' ) + ord ( 'A' ) ) ;
      UNTIL ch IN legal_commands;
      Prompt : = ch;
    END ; { Prompt }
BEGIN
  REPEAT
    CASE Prompt ( 'G ( urgle , W ( hir , S ( plat , Q ( uit : ' ,
        ['G' , 'W' , 'S' , 'Q'] ) OF
      'G' : writeln ( 'gurrggle' ) ;
      'W' : writeln ( 'wwhhhirrr' ) ;
      'S' : writeln ( 'sppplaaat' ) ;
      'Q' : exit ( PROGRAM ) ;
      END ;
  UNTIL false ;
END.
```

运行实例：

```
G ( urgle , W ( hir , S ( plat , Q ( uit : g
gurrggle
G ( urgle , W ( hir , S ( plat , Q ( uit : w
wwhhhirrr
G ( urgle , W ( hir , S ( plat , Q ( uit : g
gurrggle
G ( urgle , W ( hir , S ( plat , Q ( uit : s
sppplaaat
G ( urgle , W ( hir , S ( plat , Q ( uit : g
gurrggle
G ( urgle , W ( hir , S ( plat , Q ( uit : q
```

说明：

（1）类型说明中给出的集合charset是所有ASCII字符的集合。

（2）函数prompt的功能是显示出提示行，等待一个合法的命令字符输入，该命令字符就是prompt的函数值。注意，小写字母将自动转换成对应的大写字母。

（3）主程序中的 **REPEAT** 语句使用 **UNTIL** false 作为结束条件，即永远重复执行其中的 **CASE** 语句。而Q命令的输入，使程序转向执行UCSD PASCAL的内部函数 **exit**（**PROGRAM**），它将使程序立即结束。如果exit的参数为子程序名，则它的作用是结束子程序的运行。

PASCAL语言的输入输出操作指的是从文件取得信息或向文件写入信息。在UCSD P系统中，文件既可以是块结构设备（如磁盘）上的文件，也可以是外围设备（例如控制台或打印机）。UCSD PASCAL语言提供四种输入输出操作：字符输入输出、记录输入输出、块输入输出以及设备输入输出。前两种是标准PASCAL的一部分，后两种则是UCSD PASCAL的扩充，用来实现低级的输入输出操作。下面是四种输入输出操作的概要。

字符输入输出

* 处理的对象是字符串，即操作是对行文文件或字符设备进行的。
* 顺序操作。
* 系统自动提供缓冲区。

记录输入输出

* 处理的是磁盘文件上记录中所存贮的信息。
* 可以是顺序存取，也可随机存取。
* 系统自动提供缓冲。

块输入输出

* 处理对象是无类型文件的字节块（512个字节）。
* 可以顺序存取，也可随机存取。
* 系统不提供缓冲。

对运行速度有要求时使用。

设备输入输出

* 处理对象为一系列的字节。
* 既可以顺序存取，也可随机存取。
* 系统不提供缓冲。
* 直接用来控制物理设备。

下面的例4—6到例4—9是关于输入输出操作的一些实例。

例4—6　　大小写转换

程序功能：把一个大小写混合的行文文件中的所有大写字母转换成小写字母后，产生一个新的行文文件。

程序：

```
PROGRAM lc_filter;
  {This program converts a textfile to lower-case.}
```

```
VAR
   i : integer;
   s : string;
   in_file , out_file : text;
BEGIN
   write ( 'input file : ' ) ;
   readln ( s ) ;
   reset ( in_file, s ) ;
   write ( 'output file : ' ) ;
   readln ( s ) ;
   rewrite ( out_file, s ) ;
   WHILE NOT eof ( in_file ) DO
      BEGIN
         readln ( in_file , s ) ;
         FOR i: = 1 TO length ( s ) DO
            IF s[i] IN ['A'..'Z']
               THEN s[i] : =chr ( ord ( s[i] )
                 -ord ( 'A' )+ ord ( 'a' ) ) ;
         writeln ( out_file, s ) ;
      END;
   close ( out_file , lock ) ;
END.
```

说明：

（1）输入文件名和输出文件名均在运行时由用户给出，它们的类型为 text 文件。UCSD PASCAL中的 text 文件相当于标准 PASCAL 中的 "FILE OF char"，但它既可以是磁盘上的 text 文件，也可以是面向字符的外围设备如控制台、打印机之类。

（2）reset (in_files,s) 语句用于打开文件 s，为它设置一个缓冲区，并把它与内部文件名 in_file 关联起来。rewrite (out_file, s) 语句使已存在的文件打开，不存在时则建立之，并把文件 s 与out-file建立关联，此后程序即对内部文件名 in_file 和 out_file 进行操作，相当方便。而标准 PASCAL 的 reset 和 rewrite 却仅使用第 1 个参数。

（3）UCSD PASCAL的 close 语句用来关闭已打开的文件。

例 4—7　打印文件队列

程序功能：本程序的功能是建立和维护一个打印文件名的队列。当它和另一个程序（打印出队列中文件名所指出的文件，并使该文件名出队）配合时，即可实现排队的打印输出，也就是通常操作系统中所谓的"假脱机"输出功能（SPOOLING）。由于程序较长，打印程序这里就不作介绍。

程序：

```
PROGRAM Print_Queuer;
```

```
TYPE
   charset=SET OF char;
   rec_kind= ( info,alloc,free ) ;
   print_rec=RECORD
      CASE rec_kind OF
        info : ( alloc_tail,free_tail,
          last_block_alloc : integer ) ;
        alloc , free : ( name : string[30]; link : integer ) ;
      END;
VAR
   f : FILE OF print_rec;
   last_block : integer;
FUNCTION Prompt
    ( line : string; legal_commands : charset ) : char;
    { Prompt prints a promptline and returns a
     response character in legal_commands. }
     VAR
        ch : char;
     BEGIN
       REPEAT
         writeln;
         write ( line ) ;
         read ( ch ) ;
         writeln;
         IF ch IN ['a'..'z']
           THEN ch :=chr ( ord ( ch ) - ord ( 'a' ) +ord ( 'A' ) ) ;
       UNTIL ch IN legal_commands;
       prompt:=ch;
     END; { prompt }
PROCEDURE List_Command;
   { Lists the names of files in the queue in
     reverse order ( first file in the queue
     is at the bottom of the list ) . }
   VAR
      i : integer;
   BEGIN
      seek ( f , 0 ) ; get ( f ) ;
      i:= f^ .alloc_tail;
```

```
    writeln;
    writeln ( 'files queued : ' ) ;
    WHILE i < > 0 DO
      BEGIN
        seek ( f, i ) ; get ( f ) ;
        writeln ( '       ', f^ .name ) ;
        i:= f^ .link;
      END;
  END; { List_Command }
PROCEDURE Insert_Command;
  { Inserts a file in the queue. If a free
    record is available, it is used. Otherwise
    a new block is allocated at the end of the
    queue file. }
  VAR
    s : string [30];
    free_block, temp : integer;
  BEGIN
    writeln;
    write ( '          file to insert : ' ) ;
    readln ( s ) ;
    seek ( f, 0 ) ; get ( f ) ;
    IF f^ .free_tail < > 0
      THEN
        BEGIN
          free_block:=f^ .free_tail;
          seek ( f, free_block ) ; get ( f ) ;
          temp:=f^ .link;
          seek ( f, 0 ) ; get ( f ) ;
          f^ .free_tail:=temp;
          seek ( f, 0 ) ; put ( f ) ;
        END
      ELSE
        BEGIN
          last_block:=last_block+1;
          free_block:=last_block;
        END;
    seek ( f, 0 ) ; get ( f ) ;
```

```
        temp :=f^.alloc_tail;
        f^.alloc_tail :=free_block;
        seek (f, 0); put(f);
        f^.name :=s;
        f^.link :=temp;
        seek (f, free_block); put(f);
    END; {Insert_Command}
PROCEDURE Delete_Command;
    {Deletes a file from the queue. If the file
     deleted was in the last block of the queue,
     last_block is decremented, to shorten the
     resulting queue.}
  VAR
    temp, prev, i : integer;
    found : boolean;
    s : string;
  BEGIN
    writeln;
    write ('          delete what file : ');
    readln (s);
    seek (f, 0); get (f);
    prev:=0; i:=f^.alloc_tail;
    found:=false;
    WHILE (i<>0) AND NOT found DO
      BEGIN
        seek (f, i); get (f);
        IF f^.name=s
          THEN found:=true
          ELSE BEGIN prev:=i; i:=f^.link; END;
      END;
    IF found
      THEN
        BEGIN
          temp :=f^.link;
          seek (f, prev); get (f);
          IF prev=0
            THEN f^.alloc_tail:=temp
            ELSE f^.link :=temp;
```

```
            seek (f, prev); put (f);
            IF i=last_block
               THEN last_block :=last_block-1
               ELSE
                 BEGIN
                   seek (f, 0); get (f);
                   temp :=f^.free_tail;
                   f^.free_tail :=i;
                   seek (f, 0); put (f);
                   seek (f, i); get (f);
                   f^.name :='<free>';
                   f^.link :=temp;
                   seek (f, i); put (f);
                 END;
            writeln ('     file deleted')
          END
        ELSE
          BEGIN
            writeln;
            writeln ('     not found');
          END;
    END; { Delete_Command }
PROCEDURE Clear_Command;
    { Empties the queue. }
    BEGIN
      f^.alloc_tail :=0;
      f^.free_tail :=0;
      f^.last_block_alloc :=0;
      seek (f, 0); put (f);
      writeln;
      writeln ('          queue cleared');
    END; { Clear_Command }
PROCEDURE Command;
    VAR
      done : boolean;
    BEGIN
      done :=false;
      REPEAT
```

```
            CASE Prompt ('L(ist, I(nsert, D(elete, C(lear, Q(uit'
                         ,['L', 'I', 'D', 'C', 'Q']) OF
                'L' : List_Command;
                'I' : Insert_Command;
                'D' : Delete_Command;
                'C' : Clear_Command;
                'Q' : done := true;
            END;
        UNTIL done;
    END; { Command }
BEGIN
    { $I- }
    reset (f, 'PRINT.FILES');
    IF ioresult <>0
        THEN
            BEGIN
                rewrite (f, 'PRINT.FILES');
                Clear_Command;
            END;
    seek(f, 0); get(f);
    last_block := f^.last_block_alloc;
    { $I+ }
    Command;
    seek (f, 0); get (f);
    f^.last_block_alloc := last_block;
    seek (f, 0); put (f);
    seek (f, last_block_alloc); get (f);
    close (f, crunch);
END.
```

运行结果：

```
L(ist, I(nsert, D(elete, C(lear, Q(uit      i
   file to insert : QUEUER.LST
L(ist, I(nsert, D(elete, C(lear, Q(uit      i
   file to insert : LINREG.OUT
L(ist, I(nsert, D(elete, C(lear, Q(uit      l
files queued:
   LINREG.OUT
   QUEUER.LST
```

```
L(ist, I(nsert, D(elete, C(lear, Q(uit        i
   file to insert: OWL.TEXT
L(ist, I(nsert, D(elete, C(lear, Q(uit        d
   delete what file : LINREG.OUT
   file deleted
L(ist, I(nsert, D(elete, C(lear, Q(uit        l
files queued :
   OWL.TEXT
   QUEUER.LST
L(ist, I(nsert, D(elete, C(lear, Q(uit        q
```

说明：

（1）程序中全程变量f是一个文件，其中包含着打印文件名的队列（一张先进先出的表）。文件 f 的每一个记录定义为：

```
print_rec=RECORD
   CASE rec_kind OF
      info : (alloc_tail, free_tail,
            last_block_alloc : integer);
      alloc, free : (name : string[30]; link : integer);
   END;
```

这是一个具有变体的记录类型，其标志字段 rec_kind 是一个纯量，每一个记录选用哪一种变体将由 rec_kind 的值来决定。实际上，本例的文件 f 中只有第 1 个记录采用第 1 种变体，即它是 info 记录，其余的记录或者是 alloc 记录（已分配记录）或者是 free 记录（自由记录）。

alloc 记录中的 name 字段用来保存打印文件的名字，link 字段是一个整数，其值为下一个 alloc 记录的序号。free 记录中的name字段没有用处，link 则用来指出下一个 free记录的序号。

info 记录由三个整数组成：alloc_tail 是文件 f 中最后一个 alloc 记录的序号；free_tail 是最后一个 free 记录的序号；last_block_alloc 是文件 f 的最后一个记录的序号，f 文件的组成与结构如图 4—2 所示。

（2）主程序中使用 UCSD PASCAL 编译程序的任选项 {$I−} 和 {$I+}，它分别表示关闭和打开对输入输出操作的检查。因为在执行 reset (f, 'PRINT·FILES') 语句时，若 PRINT.FILES 文件不存在，则 ioresult 函数的值将不等于 0；若不关闭输入输出检查，程序会停止运行。本程序先使用 reset 来打开 PRINT.FILES 文件，若不存在，则再用 rewrite 建立这个文件，并用Clear_Command 过程把文件进行初始化（即把第 1 个记录中的三个字段值均置为0）。这些操作都完成后，再通过 {$I+} 恢复（编译程序的）输入输出检查功能。

（3）Command 过程是程序的主体部分，它完成了大部分文件操作。首先在屏幕上给

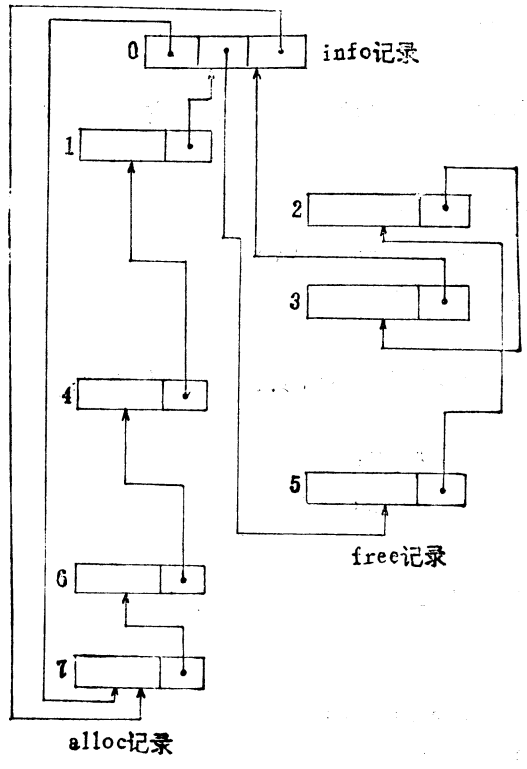


图4—2　文件f的组成与结构

出命令清单，用户输入任一有效的命令字母后，即转向执行对应的命令。这些命令是：

- L　列出队列中所有的打印文件名
- I　把一个打印文件名加入队列
- D　删除队列中某打印文件名
- C　清除队列中所有打印文件名
- Q　退出 Command 过程，返回调用程序

（4）过程 Insert_Command 与 I 命令对应。它把一个打印文件名加入队列，若文件中有自由记录可用，则使用之。否则就使用一个新记录，作为文件的最后一个记录。这个过程使用 UCSD PASCAL 所提供的 seek 内部过程，它的第 1 个参数是文件名，第 2 个参数是记录序号，其功能是实现对记录的随机存取。

（5）过程 Delete_Command 与 D 命令对应。它用来删除队列中的某打印文件名，若被删文件名位于文件中的最后一个记录，则文件减少一个记录，其它情况则仅把该记录释放为自由记录，文件的长度（记录数目）不变。

（6）L命令调用的 List_Command 过程，能把队列中的文件名以反序（先入队者后列出）列表输出，供用户检查队列中还有多少文件尚未被打印输出。

（7）主程序中用全程变量 last_block 来记录文件中最后一个记录的号码，在执行命令 I 和命令D的过程中，last_block 会发生变化。因此，从 Command 过程退出返回主程序后，需根据 last_block 之值修改文件中 info 记录里的 last_block_alloc 字段，然后再读出文件中的 last_block 记录，通过 close （f, crunch）语句把文件 f 中 last_block 之后的所有无用记

录全部删除，从而节省了磁盘空间。

例 4—8　文件比较

程序功能：对两个任意类型的文件进行比较，看它们是否相同。

程序：

```
PROGRAM File_Compare;
    { File_Compare compares two files
    ( of any type ) for equality. }
    TYPE
      block=PACKED ARRAY[0..511] OF char;
    VAR
      a_file, b_file : FILE;
      a_buf, b_buf : block;
      s : string;
      same : boolean;
    BEGIN
      REPEAT
        write ( 'file a: ' );
        readln ( s );
{ $I- } reset ( a_file, s ); { $I+ }
      UNTIL ioresult=0;
      REPEAT
        write ( 'file b: ' );
        readln ( s );
{ $I- } reset ( b_file, s ); { $I+ }
      UNTIL ioresult=0;
      same:=true;
      WHILE same AND NOT eof ( a_file )
                 AND NOT eof ( b_file ) DO
        BEGIN
          same:= ( 1 =blockread ( a_file, a_buf, 1 ) )
            AND ( 1=blockread ( b_file, b_buf, 1 ) );
          IF same
            THEN same:= ( a_buf=b_buf );
        END;
    IF NOT same OR NOT eof ( a_file )
          OR NOT eof ( b_file )
      THEN writeln ( 'files different' )
      ELSE writeln ( 'files same' );
```

```
END.
```

说明：

（1）本例是块输入输出操作的一个示范。因为块输入输出只能处理无类型的文件，所以a_file和b_file这两个文件说明时均无类型；又因为系统不自动提供给块输入输出操作缓冲区，所以程序中自行说明了两个缓冲区：a_buf和b_buf，它们的大小都是512个字节。

（2）使用编译任选项｛$I－｝和｛$I＋｝的目的是用来确保两个参与比较的文件均存在并能正确地打开，否则，文件无法进行比较。

（3）UCSD PASCAL的内部函数blockread用来进行块输入输出操作。其中第1个参数是文件名，第2个参数是缓冲区名，第3个参数是需要读出的块数，函数的值是实际读出的块数。若实际读出块数与需要读出块数不等的话，则必定是操作发生了某种错误。

例4—9　磁盘拷贝

程序功能：把磁盘上0号磁道上的内容拷贝到另一个磁盘上去，0号磁道上的信息，通常情况下是P系统的系统引导程序。

程序：

```
PROGRAM Boot_Copy;
  CONST
    sectors_per_track= 8;
    bytes_per_sector= 512;
  VAR
    i : integer;
    buf : PACKED ARRAY[1..bytes_per_sector] OF char;
  BEGIN
    writeln;
    writeln ( 'Bootstrap Copy Program' ) ;
    writeln;
    writeln ( '         place source disk in drive 4' ) ;
    writeln ( '         and destination disk in drive 5' ) ;
    writeln;
    write ( '         press<return>to continue--' ) ;
    readln;
    FOR i := 1 TO sectors_per_track DO
      BEGIN
        unitread ( 4, buf, 0, i-1, 2 ) ;
        unitwrite ( 5, buf, 0, i-1, 2 ) ;
      END;
    writeln;
    writeln ( '         copy complete' ) ;
  END.
```

说明：

（1）本程序仅仅是设备输入输出操作的一个例子，实际上它并不能成功地在 IBM PC 上拷贝P 系统的引导程序。

（2）用于进行设备输入输出操作的两个内部过程为 unitread 和 unitwrite，它们使用了五个参数，其意义分别为：

* 设备号
* 缓冲区的名字
* 要传送的字节数目（在物理扇区模式时无意义）
* 块的序号或扇区的序号
* 操作模式（“2”表示物理扇区模式）

（3）程序中的常数经常也给予其一定的名字，这样做有许多优点：比如程序易读、易改，当本程序对不同格式的盘进行操作时，仅仅需要修改程序中的常数说明就行了。

例 4—10　　中断处理

程序功能：本例用来说明 UCSD PASCAL 是如何处理中断的。处理中断必须利用 UCSD PASCAL 的并发处理能力，如进程的启动，信号灯的设置等等。本例中的信号灯与时钟中断事件建立了关联，因此，进程 tick-tock 每当时钟中断发生时就被唤醒一次，产生一次输出。整个程序的功能就象一只时钟那样，滴嗒滴嗒地走个不停。

程序：

```
PROGRAM Clock;
  CONST
    clock_vector=8;
  VAR
    s : semaphore;
    pid : processid;
  PROCESS tick_tock;
    BEGIN
      REPEAT
        wait(s);
        writeln('tick');
        wait(s);
        writeln('tock');
      UNTIL false;
    END; {tick_tock}
  BEGIN
    seminit(s, 0);
    attach (s, clock_vector);
    start(tick_tock, pid, 500, 200);
```

```
    REPEAT
    UNTIL false;
END.
```

说明：

（1）在UCSD PASCAL程序中，进程的说明非常类似于过程的说明，但进程并不能被“调用”，它必须通过内部过程start来“启动”。start的第1个参数是被启动的进程名；第2个参数是类型为processid的变量，它由系统赋值并使用；第3个参数是指出该进程要求使用的堆栈的大小；第4个参数是该进程的优先权，其范围为0..255，最高优先权为255。由中断信号所驱动的进程通常具有较高的优先权。

（2）被启动的进程tick_tock开始执行后，遇到内部过程wait(s)，这时必须检查信号灯s之值是否为0。若为0，则进程停止执行，等待信号灯s。由于s已经和时钟中断向量联结在一起（通过主程序中的attach），因此只有产生时钟中断信号时，才能唤醒进程tick_tock继续执行下去。

（3）主程序中的seminit (s, 0)是一个过程，它用来把信号灯s的初值设置为0。

（4）主程序中最后一个语句REPEAT UNTIL false用来使程序不断地“空转”，所以，进程tick_tock和主程序中的这个“空转”进程在系统中并发地进行。

（5）由于UCSD操作系统中时钟中断是被屏蔽的，本程序实际上不能正确地运行。

例4—11　整数堆栈

程序功能：本例是UCSD PASCAL中UNIT的设计与使用的示范。UNIT是实现单独编译的一种手段，它有许多优点。这里所设计的一个UNIT名字为Stack-Ops，其中包括一个过程和一个函数，它们的功能是实现整数堆栈的两个基本操作：Push(n)使整数n压入堆栈，而Pop却使栈顶元素出栈并将其值赋给函数Pop。

程序：

```
UNIT Stack_Ops;
  INTERFACE
    PROCEDURE Push(n : integer);
    FUNCTION Pop : integer;
  IMPLEMENTATION
    CONST
      max=100; {size of stack}
    VAR
      tos : integer;
      stack : ARRAY [1..max] OF integer;
    PROCEDURE Error(message : string);
      {Prints message on stack overflow
         or underflow.}
      BEGIN
        writeln(message);
```

```
        exit( PROGRAM );
    END; { Error }
  PROCEDURE Push { n : integer };
    { Pushes n on stack. }
    BEGIN
      IF tos>=max
        THEN Error ( 'stack overflow' );
      tos:=tos+1;
      stack[tos]:=n;
    END; { Push }
  FUNCTION Pop { :integer };
    { Pops top of stack and returns value. }
    BEGIN
      IF tos<1
        THEN Error ( 'stack underflow' );
      Pop:=stack[tos];
      tos:=tos-1;
    END; { Pop }
BEGIN
  tos:=0;  { initialize stack }
END.
PROGRAM Uses_Unit;
  USES  { $U Stack_Ops.CODE } Stack_Ops;
  BEGIN
    Push ( 4 );
    Push ( 5 );
    Push ( 6 );
    writeln ( Pop, ' ', Pop, ' ', Pop );
  END.
```

说明：

（1）一个 UNIT 由 INTERFACE（接口）及 IMPLEMENTATION（实现）两部分组成。INTERFACE 部分的常数、变量、类型、过程、函数等均为全程所公用，而 IMPLEMENTATON 部分的常数、变量等均局部于该 UNIT。

（2）IMPLEMENTATION 中过程 Push 的参数表和函数 Pop 的类型都不必写出，这里把它们用 { } 括出，仅仅是为了方便程序的阅读。

（3）整数堆栈是用具有 100 个整数元素的一个数组加上一个栈顶指示器构成的。IMPLEMENTATION 中的初始化部分用来把栈顶指示器 tos 置成 0，即每次使用这个 UNIT 时，使堆栈的初态为空。

（4）凡需要使用 Stack_Ops 这个 UNIT 的程序，在程序首部后面紧跟一个USES 说明。编译命令｛$U Stack_Ops.CODE｝表示该 UNIT 所在的文件名，如果 UNIT 已在系统程序库 SYSTEM.LIBRARY 中，则可省去不用编译命令 $U。

例 4—12　内存管理

程序功能：UCSD PASCAL 程序对内存的管理提供了一些设施，除了使用紧缩数组（PACKED ARRAY）、分段子程序、UNIT 等方法来节省内存空间之外，还可以使用一些内部过程来管理内存。本例并不是一个完整的程序，它仅用来说明如何设计一个具有尽可能大的数据缓冲区的程序，而该程序又必须能在不同存贮容量的系统上运行，也就是说，数据缓冲区的大小应是动态可变的。

程序：

```
PROGRAM Myprog;
  CONST
    res_segs='myprog, fileops, pascalio' ;
    slop=2000;
  TYPE
    byte=0..255;
    large_buf=ARRAY[0..32000] OF integer;
  VAR
    buf : ^large_buf;
    buf_size : integer;
  BEGIN
    buf_size :=varavail ( res_segs ) -slop;
    IF varnew ( buf, buf_size ) =0
      THEN Error ( 'problem in allocating buffer' ) ;
      { A buffer of buf_size words has been allocated.
      buf^[0] through buf^[buf_size-1] may now
      be accessed }
    ...
  END.
```

说明：

（1）本程序的意图是：在运行时根据系统内存的大小建立一个尽可能大的数据缓冲区 buf^，该缓冲区最大可存放 32000 个整数。注意，数组 large-buf 必须放在类型说明部分，若放在变量部分，则编译期间就企图对整个缓冲区分配内存，达不到预定的要求。

（2）内部函数 varavail 的参数是一组段名。其中 Myprog 就是这个程序本身，fileops 和 pascalio 是操作系统中关于文件操作及输入输出操作的两个例行程序段，它们都是程序运行时经常用到的段，必须保证需要时能装入内存。varavail 的函数值，就是所有这三段都装入内存之后，内存还留下来的可用空间的字数。slop 是一个常数，它表示程序的其它部

分大约还需2000字的内存空间。所以，计算得到的buf_size，就是程序开始运行时还留下的自由内存空间的大小，它们全部可以用来分配给数据缓冲区。

（3）varnew也是一个UCSD PASCAL的内部函数，它用来为buf所指向的一个数组申请内存，申请的数字由buf_size给出，系统分配给它的字数作为函数的返回值。如果系统无法分配所要求的字数，则返回值为0，表示内存空间不够。本例中，buf_size是经过计算而得到的，应该不会发生这种情况。

（4）此后，可以对数组的元素进行读、写等操作，因为本例中的数组是一个动态变量，所以数组元素应表示为：

```
buf^[0], buf^[1], ……, buf^[buf_size-1]
```

第四节 UCSD PASCAL操作系统

UCSD PASCAL操作系统（又称UCSD P系统或P系统），为在IBM PC上使用PASCAL语言开发规模较大的软件提供了一个良好的支撑环境。实际上，它不仅用于开发PASCAL程序，而且还能支撑汇编、FORTRAN77、编译BASIC、以及MODULA2等语言程序的开发，是一个功能齐全、操作方便、通用性强的单用户、交互式操作系统。本节将介绍UCSD P系统的主要功能及其使用。

1．概述

（1）硬件环境

UCSD P系统是一个磁盘操作系统，它只需要主机、控制台和一个软盘驱动器就可以运行。但它也能支持系统的其它许多扩充设备，例如打印机，通讯控制器，图形显示器，硬盘等等。

P系统的全部系统文件和用户文件均存放在磁盘上。一张在P系统下工作的软盘片，在逻辑上由两部分组成：引导区和文件区（参见图4—3）。引导区中存放的是P系统自举时的引导程序，它只占用盘片上的第0道（或者包括第1道）；文件区中存放了所有各种文件。文件区的单位是“块”（Block），每一块共512个字节，它们从0起顺序编号。其中第0和第1块在引导或诊断时使用；第2—5块为文件目录区，其中可以存放77个文件目录；在盘上具有双份目录时，第6—9块也存放目录。

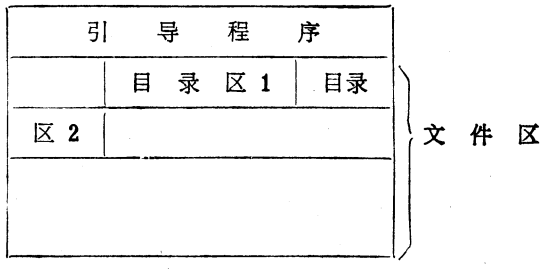


图4—3 UCSD P系统软盘片的布局

（2）命令树

P系统由许多功能模块组成，例如文件管理、行文编辑、汇编程序、编译程序、连接程序、调试程序等等。它们全部采用“点菜单”的方式来进行操作，也就是说，在进入操作系统之后，显示器屏幕上出现一行提示信息，它是一组命令，用户只要在键盘上打入相应命令的第1个字母，系统立即会转去执行该命令所对应的功能模块。进入该模块后，系统又会在屏幕上再次给出一个由命令码所组成的提示行，等待用户输入适当的命令。这种分层次的命令结构叫做命令树。图4—4是P系统命令树的示意图。

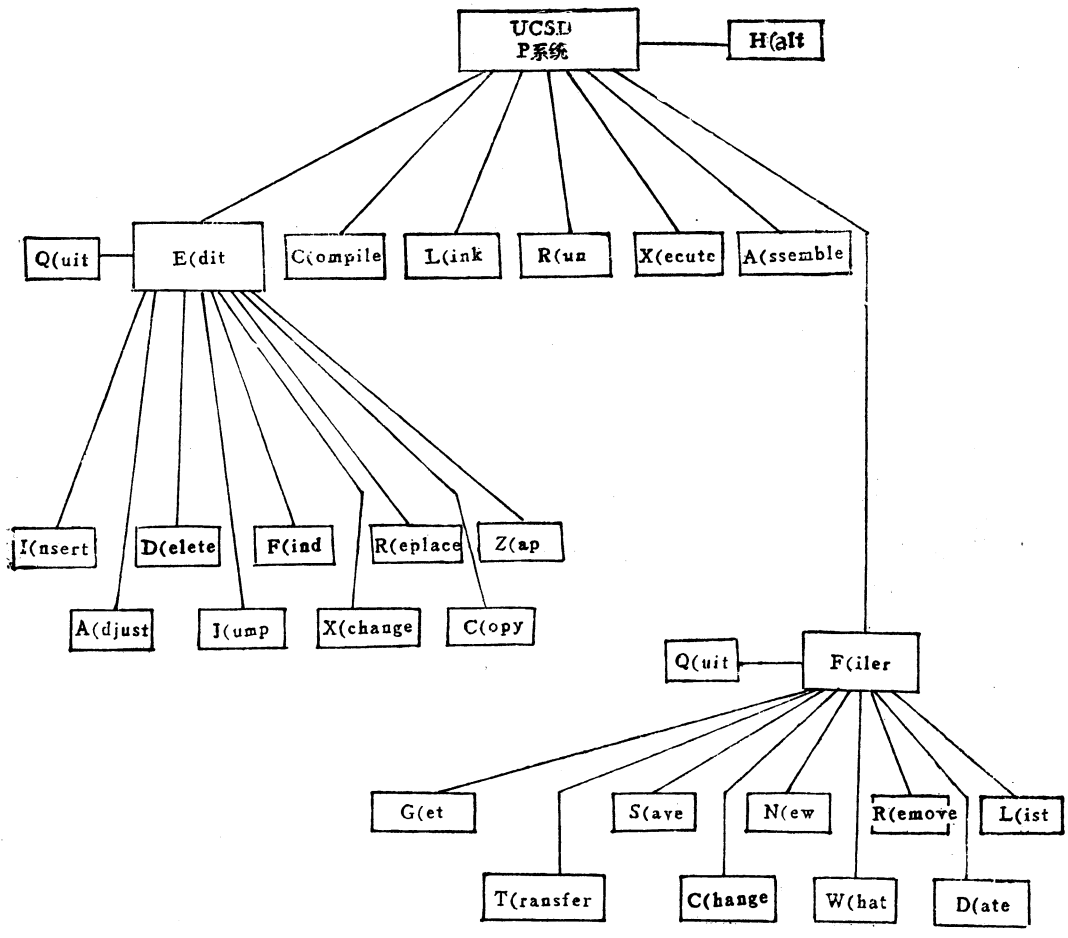


图4—4　UCSD P系统的命令树

（3）文件

UCSD P系统的文件是一组信息的集合。它可以是磁盘上的一组信息，也可以是从外围设备读入或向外围设备送出的一组信息，通常前者称为“磁盘文件”，后者称为“设备文件”。下面如无必要，两者不加区分，并简称“文件”。

UCSD操作系统本身通常也是存贮在磁盘上的，它由若干文件组成，这些文件叫做“系

统文件”。例如：

SYSTEM·PASCAL	（操作系统本身）
SYSTEM·FILER	（文件管理）
SYSTEM·EDITOR	（行文编辑程序）
SYSTEM·LINKER	（连接程序）
SYSTEM·LIBRARY	（系统程序库）
SYSTEM·INTERP	（解释程序）

除了系统文件之外，磁盘上还存放着大量的“用户文件”。用户文件大体分为三类：源程序文件（行文文件），目标代码文件，以及程序运行时需要输入处理或输出的数据文件。这三类文件用文件名中的不同后缀（文件的类型名）来加以区分，其后缀分别是：TEXT、.CODE 和 .DATA。

为了使用方便，磁盘上还有一种所谓“工作文件”，它的作用是用来打草稿。其中文件名约定为 SYSTEM.WRK.TEXT 和 SYSTEM.WRK.CODE，前者可以是一个源程序，后者为目标代码，操作时相当方便。

2. 系统命令

系统命令即 P 系统最外层的一组命令，它是整个系统的主控制。所有的系统命令都在显示器屏幕上的提示中给出，它们是：

Command : E(dit, R(un, F(ile, C(omp, L(ink, X(ecute,
A(ssem, ? 〔IV.03 B3n〕

下面先介绍几个基本的概念，再简单地逐条讨论各个命令，最后通过程序的开发过程来总结各命令之间的相互关系。

（1）基本概念

为了便于介绍各系统命令的功能，先把涉及到的几个基本概念简单说明如下：

* 系统盘　引导出 UCSD P 系统的磁盘叫系统盘。
* 缺省前缀　文件名之前未冠以驻在盘的卷标号时，该文件的前缀就使用缺省前缀。系统初态时，缺省前缀就是系统盘的卷标号。
* 系统库　这是一个名为 SYSTEM.LIBRARY 的文件，其中存放着许多经过单独编译所得到的目标程序代码，它们常常是一些多数用户都经常需要使用的例行程序，所以叫做系统子程序库，简称系统库。
* 用户库　用户各自专用的程序库，它的文件名后缀为 .CODE。往往一个用户会同时使用几个库。
* 库名文件　用户需要使用的程序库的文件名的集合叫库名文件。缺省的库名文件约定为 USERLIB·TEXT。

UCSD PASCAL 程序若需要使用用户程序库时，用户程序库的文件名必须预先存放在库名文件之中，否则，程序运行时就会找不到所需要的程序库。

* 系统输入设备　指用于向操作系统输入信息的设备，系统初态时指的是控制台键盘。

* 系统输出设备　操作系统用来输出信息的设备，系统初态时即为控制台显示器。
* 程序输入设备　指PASCAL语言中标准文件input所关联的物理设备。系统初态时，与系统输入设备相同。
* 程序输出设备　指PASCAL语言中标准文件output所关联的物理设备。系统初态时，与系统输出设备相同。

（2）命令介绍

系统命令共有11个，分别介绍如下：

① E（dit　（源程序编辑）

E命令表示调出编辑程序（SYSTEM.EDITOR）进行源程序的编辑，有关编辑程序的功能与使用将在本节第3段中另行介绍。

② F（ile　（文件管理）

F命令用来调出文件管理程序（SYSTEM.FILER），以便进行对文件的各种处理，它的功能与使用在本节第4段中再作介绍。

③ A（ssem　（汇编）

P系统接收到用户打入的A命令之后，把汇编程序(SYSTEM.ASSMBLER)调出执行。汇编程序对汇编语言源程序进行汇编，产生8088机器码的目标程序。根据使用要求，汇编所产生的目标码程序可以作为PASCAL的外部例行程序，然后用L（连接）命令连接到宿主程序中去，也可以在P系统下独立装入运行。

④ C（omp　（编译）

C命令表示调出编译程序SYSTEM.COMPILER来执行。SYSTEM.COMPILER可以是PASCAL的编译程序，也可以是FORTRAN77、BASIC或者MODULA2的编译程序。当然，不同语言书写的源程序应当用对应的编译程序来进行编译。

如果磁盘中有工作文件SYSTEM.WRK.TEXT存在，则编译程序就对它进行编译，否则就对用户指定的源程序文件进行编译。

编译过程中查出语法错误时，会给出出错信息，并允许用户有三种选择：打入空格键，则编译继续进行下去；打入<Esc>键，则结束编译返回操作系统；打入“E”键，则调出编辑程序，并对源程序进行修改。

⑤ L（ink　（连接）

本命令调出连接程序（SYSTEM·LINKER）来执行。连接程序的功能是把汇编语言编制的外部例行程序连接到高级语言（PASCAL、FORTRAN等）程序中去，也可以把几个单独汇编的程序连接在一起。

连接程序调出执行后，先在屏幕上提问：

Host file?　　{宿主文件？}

用户输入高级语言程序的目标代码文件名之后，连接程序再问：

Output file?　　{连接后生成的文件名？}

用户回答之后，连接程序就开始真正的连接操作。在连接操作期间，它还把所有被连接的外部例行程序名逐一显示出来。

连接操作完成后，它所产生的输出文件就可以被X命令所执行。

⑥ X(ecute　　(执行)

打入X命令后，P系统在显示屏上提问：

Execute what file?　　{执行什么文件？}

用户可以有两种回答：一种是输入想要执行的程序文件名(·CODE文件)，另一种是输入一些命令以改变操作系统的某些状态。

采用第1种方式回答时，系统立即从磁盘上调出该程序来执行。如果该程序尚未经过连接，则系统给出“必须先连接”的警告信息。如果该代码文件不包含程序(仅仅是一些UNIT或外部例行程序)，则系统给出“××文件中没有程序”的警告信息。

第2种回答的格式及含义如下：

L =〈文件名〉　把缺省的库名文件由USERLIB.TEXT改成给出的文件名。

P =〈文件名〉　把缺省的磁盘文件前缀改成给出的文件名。

PI=〈文件名〉　把程序输入设备指定为给出的磁盘文件或设备文件。

PI=〈字符串〉　程序输入来自内存中的便笺式缓冲器，命令中的〈字符串〉将添加到缓冲器中去。缓冲器信息中的逗号“,”当作回车符〈CR〉解释。

PO=〈文件名〉　把程序输出设备指定为给出的磁盘文件或设备文件。

I=〈文件名〉
I=〈字符串〉
O=〈文件名〉　除了改变的是系统输入(输出)设备之外，其余含义均同上面三个命令。

如果PI、PO、I及O命令的等号右面为空，则表示恢复到系统初态的约定。读者不难看出，这几个命令起着类似于MS-DOS中输入输出重定向的作用。

X命令的两种回答可以混合使用，次序是先输入要执行的程序文件名，接着是改变系统状态的一个或几个命令。

⑦ R(un　　(运行)

R命令是上面所介绍过的C(编译)、L(连接)和X(执行)三种命令的组合和自动调用，它的含义可以用图4—5给出的流程来说明，这里就不多加解释。

⑧ U(ser restart　　(用户重启动)

使上一次执行的程序重新再执行一遍。

⑨ I(nit　　(初始化)

使P系统回到初态，但已进行过的重定向命令仍保持有效。系统初始化后，将会从系统盘上寻找SYSTEM·STARTUP文件(这是一个可执行的代码文件)装入内存并立即执行。

⑩ H(alt　　(停止)

使系统停止工作，并立即象系统加电或总清那样，从系统盘上重新进行一次操作系统的引导和装入过程。

⑪ M(on　　(监控)

M命令可以用来建立一个所谓的MONITOR文件，其中记录了操作员在控制台上打入的所有命令及信息。此后，当用X命令来进行系统输入设备的重定向(I=〈MONITOR文件〉)后，操作系统将从该文件中取出一连串预先记录下来的命令自动地执行，这与MS-DOS中

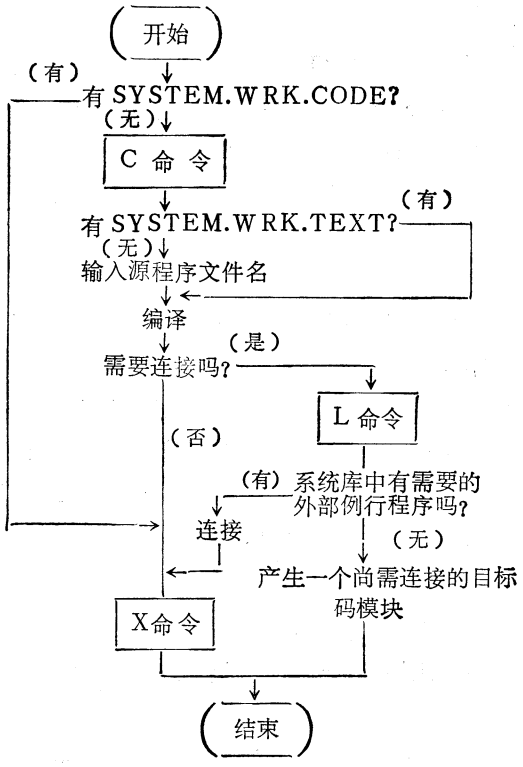


图 4—5　R命令的工作流程

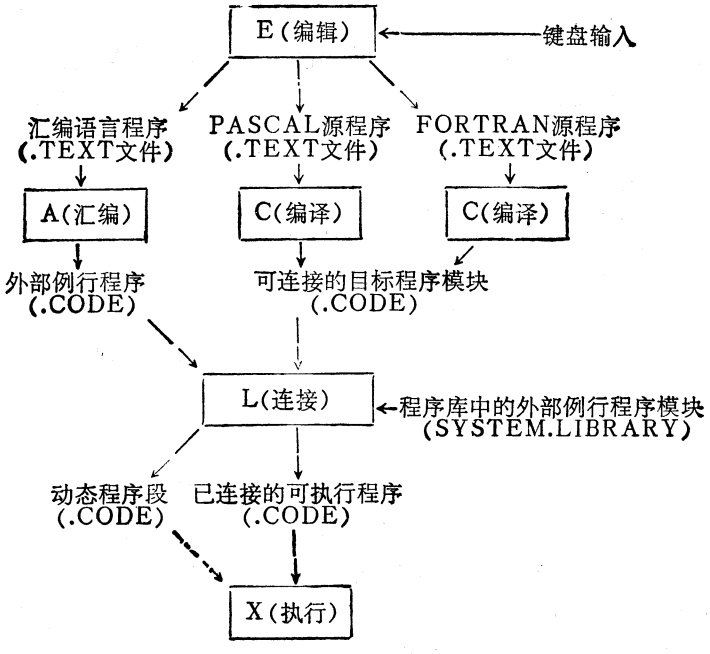


图 4—6　UCSD P系统下应用软件的开发步骤

的批处理功能有点类似。

（3）P系统下应用软件的开发步骤

在UCSD P系统之下，开发应用软件的几个主要步骤都与前面介绍的一些系统命令有关。为更清楚地了解系统命令在软件开发过程中的使用，以及它们与各种文件类型之间的相互关系，现把应用软件开发的步骤用图4—6表示，括号中给出的是文件类型。

3．源程序的编辑

在系统命令级输入E命令之后，操作系统就从磁盘上读出编辑程序(SYSTEM.EDITOR)装入内存，然后把控制转向编辑程序。

UCSD P系统有两种编辑程序，一种是全屏幕编辑程序，另一种是行编辑程序。用户喜欢使用哪一种，就把该程序命名为SYSTEM.EDITOR即可。由于全屏幕编辑程序方便、直观，它既有编辑源程序（包括数据文件）的功能，又有文字处理的功能，所以受到普遍欢迎。下面介绍P系统屏幕编辑程序的主要特点与功能。

（1）基本概念

屏幕编辑程序涉及下列一些基本概念：

- 窗口　在编辑程序控制下，屏幕显示器的第2行到第24行用来显示被编辑的文件。通常，文件远远不止23行，屏幕上看到的仅仅是文件的一部分，所以屏幕上第2行到第23行这个区域叫做编辑程序的窗口。键盘上的“P”字母键可以把窗口在文件上移动一页。
- 光标　光标表示用户在文件中的确切位置，大部分编辑操作都是在光标位置上进行的。光标的移动可以用四个光标键（←，→，↑，↓）或退格键、空格键、横向制表键、回车键来进行。
- 操作方向　有两个方向：向前或向后。它们指出操作（包括光标的移动）是顺序地向前进行，还是倒退着向后进行。现行的操作方向在提示行（第1行）中用第1个字符是＞或＜来表示。改变操作方向通过“+”（或“＞”）键和“−”（或“＜”）键来进行。
- 编辑环境　有两种编辑环境，一种适用于编辑源程序、数据等，另一种适用于编辑信件、报表之类（文字处理）。这里应选用前一种。
- 重复因子　重复因子是命令前面的一个数字，缺省时为“1”。它表示操作应该重复进行的次数，若为“／”时，表示操作重复执行直到文件结束。大部分编辑命令都可使用重复因子。

（2）各种编辑命令

进入编辑程序之后，若磁盘上有工作文件SYSTEM.WRK.TEXT，则把工作文件读入，供用户修改。否则，编辑程序向用户提问需要编辑修改的文件名字，用户回答后，装入该文件进行编辑操作。如果用户仅以回车作回答，则表示需要从头开始编辑一个新文件。

象操作系统一样，编辑程序在屏幕第1行给出一行命令“菜单”，用户只要打入命令的第1个字母，编辑程序就会转去执行相应的编辑命令，编辑命令共15个，下面介绍八个常用的编辑命令。

① I (nsert　　(插入)

从光标所在处插入键盘上输入的信息。

② D (elete　　(删除)

使用光标移动键，把光标所在位置上的字符删去。

③ eX (change　　(改变)

把光标所在位置上的字符改变成键盘上输入的字符。

④ R (eplace　　(替换)

在打入“R”命令的同时，把替换字符串和被替换字符串一起输入，即可实现替换操作。如果每次替换前需要核对一下，则在被替换字符串之前冠以“V”即可。

⑤ F (ind　　(寻找)

从光标所在位置起，寻找目标字符串的第 n 次出现。

⑥ C (opy　　(拷贝)

在光标所处的位置插入一段信息，这段信息可以是磁盘上的一个 .TEXT 文件 (或其中一段)，也可以是上一次使用 D 命令删除后留在缓冲区中的信息。

⑦ A (djust　　(调整)

调整光标所在行的左右位置。

⑧ Q (uit　　(退出)

从编辑程序退出，准备返回到操作系统。在返回操作系统之前，提供用户四种选择：

——修改工作文件并返回操作系统

——作为磁盘文件写入磁盘并返回操作系统

——不作任何文件修改而返回操作系统

——仍回到编辑程序

4. 文件管理

UCSD P 系统的文件管理程序 (SYSTEM·FILER) 功能比较齐全，使用也很方便，只要在系统命令级打入“F”键，文件管理就被调出并执行。

有关P 系统下文件的一些基本概念在概述中已经作过介绍，下面再补充几个文件管理中要用到的概念。

(1) 文件卷　　任何一个输入输出设备都被 P 系统看作为一卷文件，例如打印机，控制台，串行输入，串行输出，磁盘等。除了软盘是可以更换介质的设备，即每张盘片均有其自己的卷名之外，其它每个设备都具有自己固定的卷名，它们是：

设备名称	设备号	卷名
控制台	#1:	CONSOLE:
打印机	#6:	PRINTER:
磁盘机 A	#4:	由用户定义
磁盘机 B	#5:	由用户定义
串行输入	#7:	REMIN:
串行输出	#8:	REMOUT:

(2) 文件名　　每个磁盘文件都有一个自己的名字，名字中最右面一个小数点的后面

是类型名，用来说明该文件的类型。

（3）文件指定　文件管理中需要指出对哪一个或哪一组文件进行操作，这就是“文件指定”。文件指定由两部分组成：卷名和文件名。若为设备文件，则不必指出文件名。卷名也可以用它们的设备号来代替，例如 #1：代表 CONSOLE：，#4：代表磁盘 A。如果文件指定中卷名缺省，则表示为现行磁盘卷。

（4）通用字符（Wildcard）　文件指定中的文件名部分，可以使用两个通用字符“=”和“？”，它们都代表任意的一个字符串，包括空的串在内。使用“？”时，每当对一个文件进行规定的操作（如删除，拷贝等）之前，都会征求用户的意见，然后再决定操作是否进行。

文件管理程序共有 18 个命令，现择要介绍如下：

① L（dir　（列出文件目录）

列出指定的磁盘文件卷中的全部或一部分文件的目录。

② E（xtended list　（列出文件的详细目录）

同 L 命令。但目录信息中除了给出文件更新的日期及文件大小之外，还给出文件在盘上的逻辑地址和文件的类型。

③ C（hange　（文件改名）

改变磁盘文件名或改变文件卷名。

④ R（emove　（文件删除）

删去一个或一组文件。

⑤ T（ransfer　（文件传送）

把一个文件或一组文件、甚至整个一个文件卷传送到另一个文件卷去。如果后者是磁盘，则本命令完成文件的拷贝操作，如果后者是打印机、控制台或串行输出设备，则完成的分别是文件打印、文件显示和文件传送操作。

⑥ M（ake　（建立文件）

用给定的文件名在磁盘上建立一个目录项。文件名后面应跟〔n〕，n 指出文件的大小，单位为“块”（512 字节）。本命令仅仅给要建立的文件起了名，并在盘上保留了空间，它并没有在文件中写入任何内容。

⑦ G（et　（取得工作文件）

取某一个（或两个）文件作为工作文件 SYSTEM.WRK.TEXT 和 SYSTEM.WRK.CODE。

⑧ S（ave　（保存工作文件）

把工作文件冠以给定的文件名存入磁盘，工作文件仍保持不变。

⑨ N（ew　（清除工作文件）

把工作文件清除。

⑩ W（hat　（询问工作文件）

本命令用来了解现行工作文件的原始文件名，以及它是否已被保存过。

⑪ V（olumes　（列出文件卷名）

列出所有联机的文件卷名以及现行磁盘的卷名（即可缺省的文件卷名）。

⑫ P（refix　（改变缺省文件卷名）

把现行可缺省的卷名改成指定的文件卷名。

⑬ K(runch　　(文件移动)

把磁盘上的文件进行移动，使得它们在逻辑位置上相互为邻，紧靠在一起，以便把未使用的块组合成一个大的空白区域。

⑭ Z(ero　(清除文件卷)

把一个磁盘文件卷的全部文件都删除掉，亦即把目录变成空目录，并给该磁盘重新定义一个卷名。

⑮ Q(uit　　(退出)

退出文件管理程序，返回操作系统。

文件管理程序中还有一些其它功能，因不常用，故这里不作介绍。

5，程序库管理

把单独编译的UNIT或汇编生成的各种例行程序组合在一起成为一个·CODE文件，使用时既方便又快速。这种包含多个UNIT和多个外部例行程序的文件就叫做“程序库”。包含各种常用例行程序的面向全体用户的程序库，称为“系统库”，它用专门的文件名SYSTEM·LIBRARY来表示。某些用户专用的、或具有某一类专门功能的程序库叫做“用户库”，如图形子程序库、数学函数库等。

程序库的建立、修改和扩充等是经常需要进行的操作，可通过一个实用程序“LIBRARY”——库管理程序来完成。

LIBRARY需要用系统命令“X”来调用。进入LIBRARY的控制之后，它会向用户提问：

Output file name?　{输出文件名?}

用户回答需要建立、修改或扩充的程序库的文件名(类型应为.CODE)之后，LIBRARY又向用户提问：

Input file name?　{输入文件名?}

用户回答了需要参加到新库中去的例行程序所在的文件名之后，LIBRARY立即读入用户指定的输入文件，然后在屏幕上出现下列画面：

```
Library : N(ew, 0—9(slot-to-slot, E(very, S(elect, C(omp-unit,
        F(ill, ?
Input file? SCREENOPS
    0    u    SCREENOP    921     8
    1    s    SEGSCINI    416     9
    2                             10
    3                             11
    4                             12
    5                             13
    6                             14
    7                             15
Write to what file? NEW.CODE
```

0	8
1	9
2	10
3	11
4	12
5	13
6	14
7	15

其中，屏幕上半部分的内容为输入文件中所包含的各种UNIT（u）、例行程序段（s）、外部汇编程序（a）以及程序（p）在文件中的分布情况(用槽口号码来表示)，下半部分则为输出文件（新程序库）中u、s、a和p的分布情况。屏幕上第1行是LIBRARY的提示行，它给出了用户可以选择进行的各种命令的清单。

下面对LIBRARY所提供的一些主要功能作扼要介绍。

* E(very

把输入文件每一个槽口中的代码段均拷贝进新的程序库中去。若新库中对应槽口中已有内容，则寻找空的槽口进行拷贝；若相同的代码段已存在于新库中，则该代码段不再被拷贝。

* 槽口→槽口的拷贝

打入槽口号码，即可实现对应槽口之间代码段的拷贝。

* S(elect

允许用户有选择地把输入文件中的某槽口内的代码段拷贝到新库中的指定槽口去。

* N(ew

向用户提问另一个新的输入文件名，用户回答后，又可以把新输入文件中的代码段按照上面的操作拷贝到库中去。

* A(bort

使已经进行的库管理操作全部作废，输出文件也不保存到磁盘上去。

* Q(uit

结束库管理操作，等待用户输入版权信息之后即把新库记盘，然后返回操作系统。

6．调试程序

高级语言程序的调试，在开发应用软件时是一个十分重要的问题。UCSD P系统为UCSD PASCAL程序的调试提供了一个工具——调试程序。

调试程序用来对已编译好的可运行的目标程序（伪码程序）进行调试。利用调试程序，用户可以为程序设置断点，要求程序单步执行，显示和修改存贮器内容等。

调试程序可以用系统命令级中的D命令（Debug）来调用。UCSD PASCAL程序中使用的halt内部过程，在运行时如果调试程序处于活动状态，则也会引起程序运行终止，控制转向调试程序。

为了正确地使用调试程序，用户一方面必须打印出编译得到的源程序清单，另一方面还必须熟悉UCSD虚拟机器的系统结构，即伪码的操作码，栈的使用，变量与参数的分配等。

下面用一个十分简单的程序的调试为例，说明调试程序是如何使用的。

设有一UCSD PASCAL 程序，它经过编译后得到的源程序清单如下。程序清单中左面的数字分别表示语句行号、段号、例行程序号:层次、以及需要的存贮器字数，它们在调试时极为有用。

```
Pascal Compiler IV.0 c3.as-2
     Page    1
  1    0    0:d    1   ( $L  #5 : LIST.TEXT )
  2    2    1:d    1   PROGRAM EX1;
  3    2    1:d    1   VAR I, J, K : integer,
  4    2    1:d    4       B1, B2 : boolean;
  5    2    1:0    0   BEGIN
  6    2    1:1    0       I:=1;
  7    2    1:1    3       J:=1;
  8    2    1:1    6       IF K<>1 THEN writeln('whats wrong?');
  9    2     :0    0   END.
End of Compilation.
```

为了调试这个程序，首先用 D 命令进入调试程序，然后可以在 IF 语句的前面设置一个断点。操作方法如下（括号中为用户输入信息）：

```
( BS ) Set break #? ( 0 )  Segname? ( EX1 )  Proc #? ( 1 )  Offset #? ( 6 )
```

{ B 表示进入断点处理，S 表示设置断点，0 表示断点号码为0，断点的位置在EX1 这一段的第 1 个例行程序中偏移量为 6 个字的地方 }

```
( EP )
```

{ 允许在单步执行伪码指令时显示出有关的内部状态 }

```
( R )
```

{ 退出调试程序，但调试程序仍保持活动状态 }

接着，用 X 命令启动EX1 程序进行。当它到达上面所设置的断点时，运行终止，并再次进入调试程序，这时屏幕显示：

```
Hit break  #0 at S=EX1 P#1 O#6
( cd ) S=EX1 P#1 O#6 SLDC1
```

{ 0号断点命中，停止在EX1 段的 1 号过程偏移 6 个字之处，第 2 行表示即将执行的伪码指令。 }

于是，可以用单步方式执行两条伪指令：

```
( cd )  S=EX1  P#1  O#7  SLDC1
( cd )  S=EX1  P#1  O#8  NEQU1
```

{第1次执行的是短装入全程变量1（指的是变量K），第2次执行的是短装入常数1，然后，下面将要执行的是比较指令}

此时可以检查栈中的内容，看看到底是哪两个数将参加比较：

```
(LR)        {列出寄存器内容}
rg mp=EB62 sp=EB82 erec=…
(A) Address? (EB82)        {列出栈指针EB82所指向的内存内容}
a EB82:01 00 C5 14 ……
```

{EB82读出的栈顶元素为常数0001，这是正确的。第2元素14C5是装入的变量K的值，显然不正确}

至此，再查一查源程序，很快会发现错误的原因是没有给变量K赋值。

关于UCSD PASCAL调试程序的详细使用说明，读者可以参阅有关手册，这里不再详细介绍。

第五章　表格处理软件

表格处理软件是近几年来最畅销的微型计算机应用软件之一。它在事务处理中有着广泛的应用。在 IBM PC上运行的表格处理软件有多种，下面介绍三种最常用的表格处理软件：即VISICALC、MULTIPLAN 和 SUPERCALC。

第一节　概　　述

1．表格及其处理

在日常生活和工作中，人们经常要使用表格。例如财会部门的帐册，施工项目的预算计划，商业单位的提货发货清单，管理机构的统计汇总报表，个人使用的亲友通讯录及日常开支记录等，都是最常见、最普通的一些表格。表格与文字、图画一样，也是人们以书面形式表达和记录信息的一种重要方式。

尽管表格的内容和用途各不相同，但不论是表格的形式还是表格的处理过程，却具有许多共同的特点。表 5-1 是一张营业额统计表。

表 5-1　　营业额统计表

1983年度××百货公司营业额统计表 （单位：千元）						
部门	一季度	二季度	三季度	四季度	全年	百分比
服装部	500	480	750	540	2270	26.4
家具部	480	450	410	370	1710	19.9
交电部	200	300	620	710	1830	21.3
食品部	720	600	930	530	2780	32.4
总　计	1900	1830	2710	2150	8590	100.0

表格一般由若干行和若干列组成。表格中的每一个格子存放一项“数据”，它们是表格的最小组成单位，为叙述方便，称它们为“表格单元”，有时也简称为“单元”。

表格单元中的数据主要有两种类型：一种是文字型数据，例如表 5-1 中的部门、服装部、总计、百分比等等，它们对表格中的数值型数据起着定义和说明的作用，所以往往也叫做“标号”。另一种是数值型数据，如表格中的 500、480、32.4 等等。数值型数据又可分成初始数据和结果数据两种。如各部门的一、二、三、四季度的营业额数据：500、480、750、540 等，属于“初始数据”。而全年营业额、百分比、总计等数据，是通过计算得到的，故称为“结果数据”。

结果数据是使用表格中已知数据（初始数据或中间结果数据）按照一定的计算公式计算出来的。例如表 5-1 营业额统计表中，服装部全年营业额及所占公司总营业额的百分比，可分别按下面两个公式算出：

服装部全年营业额＝服装部一季度营业额＋服装部二季度营业额
＋服装部三季度营业额＋服装部四季度营业额

服装部营业额占全公司百分比＝（服装部全年营业额/公司全年营业额）×100%

表格的处理过程也有许多共同之处。一般说来，一张表格的处理过程大致包括如下步骤：

①设计和定义一张表格。

例如确定表格的列数和行数，分配各行各列的用途，规定每一列（或行）的宽度（或行数），在表格中填入标题等等。

②在表格中填入初始数据。

③给出有关的计算公式，并按计算公式算出结果数据填入表中。

④复制、印刷及保存表格。

上述步骤在实际工作过程中往往需要反复进行多次。例如，表格中需要增加一列新的统计数据（如各部门的平均每季度营业额）时，或营业额增长突破 4 位数字需要调整表格宽度时，还有，某部门某季度初始数据出错需要更正时，或者发现某个中间结果数据（如服装部全年营业额）计算有错时，所有这些微小的更动或差错，都需要对表格重新进行处理。显然，当表格比较庞大而计算又相当复杂的时候，一张表格的制作和处理的过程就需要化费大量的时间和人力，而且其中大部分时间是化费在重复性的抄写、整理及计算工作上，因此，工作效率受到了严重的影响。

2．表格处理软件

传统的表格处理是使用铅笔、橡皮、纸张、算盘或计算器等工具来进行的。多年来尽管计算工具不断改进，计算速度也大大加快，但整个制表过程的效率仍然很低，工作量相当可观。微型计算机技术的迅速发展，促使人们考虑如何使用计算机这个智能工具来实现表格处理中各个环节的自动化，扔掉传统的铅笔、橡皮、纸张、计算器等工具，直接在计算机屏幕上完成表格设计、处理和制作的全部操作，从而使大量的办公室人员从表格工作中解放出来，这就是设计开发表格处理软件的实际背景。

表格处理软件是这样一类软件，即在这类软件的作用下，一台微型计算机的内存贮器好象是一大张铺开的表格纸，计算机的键盘相当于铅笔和橡皮，而中央处理机则相当于算盘或计算器，磁盘外存贮器则好象是一大本帐册用来保存已经产生的各式表格。在表格处理的过程中，表格的格式和内容可以通过屏幕显示出来，用户通过键盘能对表格的格式进行设计或者修改；能够向表中填入初始数据；也可输入各种规定的计算公式，经过计算机自动计算后，在表格中填入结果数据；一张表格处理完毕后，可以保存到磁盘存贮器以备下次再用，也可通过打印机在纸上正式打印出来。

第一个表格处理软件是美国 Visicorp 公司于 1979 年 5 月为苹果计算机（Apple Ⅱ）开发的 VISICALC。该软件投入市场后，立即得到了广大用户的好评，销售数量已达几十万份。此后，其它软件公司也相继开发了许多具有不同特点的表格处理软件。例如 Sorcim 公司

的 SUPERCALC，Microsoft 公司的 MULTIPLAN，日本 SORD 公司的 PIPS 等。Visicorp 公司又对 VISICALC 的功能作了不少补充和改进，推出了高级 VISICALC（VISICALC Advanced Version）。几年来，表格处理软件始终是微型计算机市场上最畅销的软件之一。一方面是因为它的通用性好，适用面广，有着极好的使用价值，另一方面是因为它提供给用户使用的是一种面向自然的简易语言，能使任何不懂得计算机程序设计的用户很快地掌握和使用这种软件。

为 IBM PC 微型机所配置的表格处理软件有多种，例如VISICALC、高级VISICALC、MULTIPLAN、SUPERCALC、PEACHCALC 等等。本章首先通过一些实例比较详细地介绍 VISICALC 的功能与操作，然后再扼要地介绍 MULTIPLAN 和 SUPERCALC 所具有的特色。

3．表格处理软件的一些基本概念

在介绍具体的表格处理软件之前，首先介绍几个表格处理软件中常用的基本概念。

* 电子表格　表格处理软件为用户预先设计了一张空白的表格，例如 VISICALC提供的表格是 63 列×254 行；MULTIPLAN 提供的表格是 63 列×255 行；SUPERCALC 的表格是 63 列×254 行。这些表格的载体是存贮器单元而不是普通的纸张，因此称为“电子表格”。

* 窗口　电子表格在存贮器中是不可见的。为了操作的方便，表格处理软件通过显示器屏幕让用户看到正在进行操作处理的表格部分。由于表格很大，屏幕上只能显示出表格的一小部分，所以屏幕仅仅是用户用来观察表格的一个“窗口”，或者说是表格的一个显示区。如果要看到表格的全部内容，则必须把窗口在电子表格上到处移动。有时，为了能同时在屏幕上对比地观察表格中不连续的几个部分，表格处理软件可以在屏幕上定义两个或两个以上的窗口，分别用来观察不同的表格部分。

* 表格单元的标识　电子表格中的每一个格子，叫做一个“表格单元”，有时也简称为一个“单元”(Cell)。每一个表格单元在表格中都有各自的坐标位置，即它所在的列号和行号。不同的表格单元，可以用它们的坐标位置给予标识。例如，A1、B3、A21、AA100、BK1、A254 和 BK254都是 VISICALC 的表格单元的标识。有些表格处理软件，还允许给表格单元命名，因而需要时可以用名字来标识某些表格单元。

* 表格单元的格式　表格单元的格式也叫做表格单元的属性。例如，每个表格单元的宽度，标号或数值在表格单元中的放置位置，数值数据在表格单元内的表示形式（整数形式、两位小数形式、指数形式等）等。VISICALC 提供给用户的表格，在其初态时，每一个单元的格式都相同：表格单元的宽度均为九个字符，标号在表格单元中靠左存放，数值数据在表格单元中靠右存放等。需要时，用户可以改变某些表格单元的格式。

* 当前表格单元　用户在进行表格处理的操作过程中，只能对表格单元一个一个地逐个进行。操作正在进行之中（如填入数据、抹除数据、编辑等）的那个表格单元，就叫做“当前表格单元”或“当前单元”。在显示屏上，当前表格单元采用反视频显示方式进行显示，以便与其它单元相互区别。本书为了排版方便起见，在以后给出的显示输出画面中，将使用下横线来指出当前正在进行操作的表格单元。

* 表格单元的数据与公式　通常，大部分表格单元是用来存放数据的，存放在表格

单元之中的数值型数据也叫做该单元的值。某些表格单元之中的数值数据是初始数据（初值），而有些表格单元中的数值数据是经过计算得到的结果（计算值）。对于那些用于存放计算值的表格单元，预先必须为每个单元指出其相应的计算公式，否则就无法进行计算。必须注意的是，表格软件本身能够为每个表格单元记住输入的原始数据或计算公式，但用户在屏幕上看到的却是每个表格单元中计算出来的结果。

第二节　VISICALC 的操作

1. VISICALC的启动

要在 MS-DOS 下启动 VISICALC，当前盘上至少要有以下两个文件：

VC80.COM　　VISICALC 的引导程序

IBMVC.COM　　VISICALC 的主体程序（隐式文件）

其中 IBMVC.COM 不能用常规方法复制。

启动 VISICALC 的方法是打入：

A> VC80

随后屏幕上显示出 VISICALC 的初态画面，如图 5—1 所示。

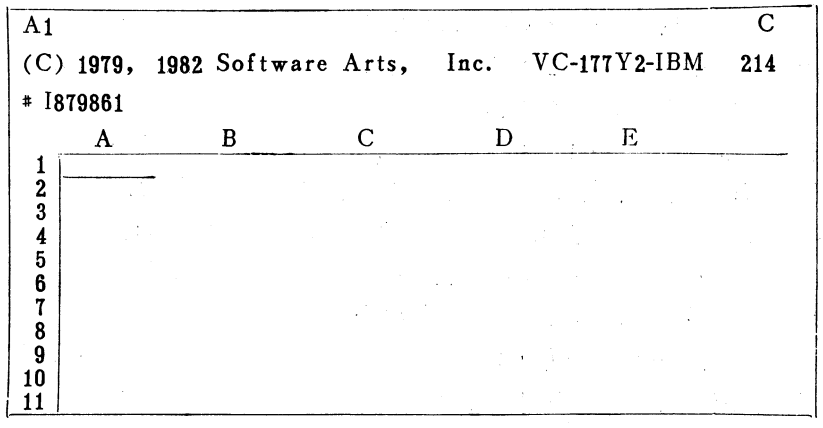


图 5—1　VISICALC的初态画面

初态画面的前三行为 VISICALC 的状态区，用来显示 VISICALC 的工作状态。画面的其余部分为表格显示区（也称为窗口），用来显示 VISICALC 表格中的部分数据。

状态区的第 1 行称为当前单元行（简称为单元行），主要用来显示当前表格单元的位置、类型和内容。单元行的最右部分为方向指示器，用来表示表格单元中数据的重计算方向。当方向指示器为 C 时，表示按列进行计算；为 R 时，表示按行进行计算。

状态区的第 2 行是提示行，主要用来显示操作过程中 VISICALC 对用户的提示信息。

例如，输入数据时，该行就显示出表格单元的类型；而输入命令时，该行又显示出命令清单等。在初态画面中，该行显示的是 VISICALC 的版本信息。提示行的最右部分为存贮器指示器，用来指明可用内存的大小（以 K 为单位）。

状态区的第 3 行为编辑行，用来显示用户输入的数据和公式以及作为编辑命令的编辑区。它在初态画面中显示盘片的登录号。

表格显示区可用来显示 VISICALC 表格中的 21 行×8 列的数据（图 5—1 仅列出了11行×5 列）。每列的初始宽度为九个字符。光标的初始位置为 A1。

2．当前表格单元的定位

VISICALC 是处理表格的软件，它为用户提供一张大型电子表格。用户使用这个表格的方法为：把光标移到指定的表格单元处，填入数据或公式。其中第 1 步称为当前表格单元的定位，它可以通过下列三种方法实现。

（1）步进定位

步进定位通过使用↑ ↓→ ←键移动光标来确定当前表格单元的位置，一次可以移动一列或一行。

如果光标已经移到了屏幕的边界，再要移动光标，会引起屏幕表格区的移动。如果已经移到了表格的边界，再要移动会引起 VISICALC 报警。

（2）参数定位

当前表格单元的另一种定位方法是参数定位，它可以一次把光标定位到任意指定的表格单元处。

参数定位的方法是首先从键盘输入“>”，这时提示行出现下列信息

Go to : Coordinate

询问需要定位的坐标。回答指定表格单元的坐标后，VISICALC 就把光标移到指定的单元处。这样，参数定位的工作就完成了。

（3）光标复位

要从任何位置处把光标移回到初始位置，可以输入〈Home〉键或输入命令：

>A1〈CR〉

3．数据输入

当前表格单元定位以后，就可以向当前单元输入数据了。向表格输送数据的过程类似于人们日常工作中的填表，须逐项地进行。

填入表格的数据可分两种类型，一类是标号（字符串），另一类是数值。在输入这两类数据时，提示行分别用“Label”和“Value”提示。输入完毕后，这两类数据在单元行中以（L）和（V）区分。

在输入过程中，可用〈Esc〉键或〈BS〉键删去刚输入的一个字符，也可用 Ctrl - C 删除整个输入行。

下面分别介绍这两种类型数据的输入方法。

（1）标号的输入

如果输入的第1个字符是双引号或字母，VISICALC就认为输入的数据是标号。对于第1种情况，输入的双引号仅用来说明输入的数据是标号，它本身并不作为数据送入当前单元。VISICALC规定标号的长度不能超过125个字符。

（2）数值的输入

对于输入的非标号数据，VISICALC都认为是数值。这里的数值既包括算术常数，也包括逻辑值（TRUE/FALES）。其中算术常数的精度可达11位有效数字，表示范围约为-10^{61}～10^{61}。

4．公式输入与数据计算

在表格单元中除了可以填入原始数据外，还可填入用来计算结果数据的公式。用户使用电子表格时，可以先设计和定义表格的结构（表格标题和统计栏的计算公式等），再根据不同情况填入不同的原始数据，VISICALC会自动地计算出结果数据。

（1）公式的组成

VISICALC的公式由运算对象（常量和变量）、运算符和函数组成，它仅能处理数值数据。

公式中使用的变量为表格单元的坐标，其值为表格单元的当前值。

在公式中使用的运算符有以下两类：

* 算术运算符　　+，－，*，/，^（乘幂）
* 关系运算符　　>，<，>=，<=，<>

应当注意，如果不加圆括号，算术运算符的优先级是相同的。它们总是以从左到右的顺序计算。例如：

```
1+1*2
```

的结果是4而不是3。

公式中也可以使用函数。函数的参数可以是单个参数，也可以是参数表。其中参数表有下列四种类型：

* 常量和变量，例如：－27，A4等。
* 公式，例如：3+5，3+A1*(A2+A3)等。
* 单元区域，例如：A1…A8，A2…E2等。
* 枚举表，例如：（A1…A8，A4，D2）等。

此外，对于公式中引用的空白单元或标号表格单元，VISICALC认为其值为0。但应注意，不能在表格单元中填入包含自身坐标的公式，否则会引起混乱。

（2）公式的输入

要想在某表格单元中输入一个公式，只要把光标移到该单元处，再输入公式即可。输入的公式必须以数字、小数点、函数引导符“@”和运算符“+”、“－”或“（”开头。例如，公式“A1－B1”应输入为“+A1－B1”。

对于公式中的变量，可以直接输入，也可以采用“形象输入”。所谓形象输入是指使用光标移动键定位到某个单元而输入变量的一种方法。例如，输入公式“+A1-B1”，可先输入“+”，再用光标移动键把光标移到A1，接着输入“-”，再把光标移到B1，输入〈CR〉键后，公式就输入完毕了。

（3）计算方向

在介绍VISICALC的初态画面时曾提到：单元行中的方向指示器为“C”时，表示计算方向为列；为“R”时，计算方向为行。现举例说明方向指示器对重计算结果的影响。

设有一张2×2的表格，其A1单元为原始数据，A2、B1和B2单元的结果数据由下列公式计算：

$$A2=A1+1,\quad B1=A1+A2,\quad B2=A2-B1$$

在A1中填入2，并在其它单元中填入计算公式后，表格的结果如图5—2（a）所示。如果这时方向指示器为“C”，用命令

>A1〈CR〉1〈CR〉

把A1改为1后，计算出的结果如图5—2（b）所示。如果这时的方向指示器为“R”（可用命令“/GOR”实现），计算出的结果则为图5—2（c）所示。

(a) 初始结果

	A	B
1	2	5
2	3	-2

(b) 按列重计算

	A	B
1	1	3
2	2	-1

(c) 按行重计算

	A	B
1	1	4
2	2	-2

图5—2　计算方向对计算结果的影响

5．表格的保存与输出

表格制作好了之后，通常都要从打印机上输出以及保存到盘上供以后使用。这些操作都要使用VISICALC的命令来实现。

表格输出命令的格式如下：

/PP〈右下角坐标〉〈CR〉

用来打印出以当前表格单元为左上角，以命令中的参数为右下角的部分表格。

表格保存命令的格式如下：

/SS〈文件名〉

如果指定的文件名盘上已存在，VISICALC 的提示行则显示出下列信息：

Storage : File exists. Y to replace

它表示盘上有同名文件存在，打入“Y”将更新这个文件。

保存在盘上的表格，可用装入命令装入内存。在装入前，如果内存中已有表格存在，则应首先用清除命令“/CY”清除内存。装入的命令格式为：

/SL〈文件名〉

关于 VISICALC 的其它命令，请详见第五节．这些命令的引导符都为“/”，输入“/”后，提示行会显示出下列命令“菜单”：

Command : B C D E F G I M P R S T V W —

它表示 VISICALC 为用户提供 15 条命令。

第三节　VISICALC 应用举例

在了解了 VISICALC 的基本操作之后，本节再介绍几个 VISICALC 的应用实例。通过这些例子，读者会进一步了解使用 VISICALC 造表的具体方法。其中所使用的一些操作命令，将在第五节中详细给予解释。

1．表格的制作

例 5—1　月销售额、成本和利润表

① 功能：使用 VISICALC 制作下列月销售额、成本和利润表：

销售额	1000元
成　本	600元
利　润	400元

其中成本是销售额的60%，利润则是销售额与成本的差额。

② 操作：首先用命令“/CY”清除内存中的表格，接着输入数据和公式。输入的过程为：

- 在 A1 到 A3 中输入标题：SALES（销售额）、COST（成本）和 GROSS（利润）。
- 用命令“>B1〈CR〉1000〈CR〉”在 B1 中输入原始数据。
- 用命令“↓+B1*0.6〈CR〉”和“↓+B1−B2〈CR〉”在 B2 和 B3 中输入计算公式。

这时屏幕上的画面如图 5—3 所示。

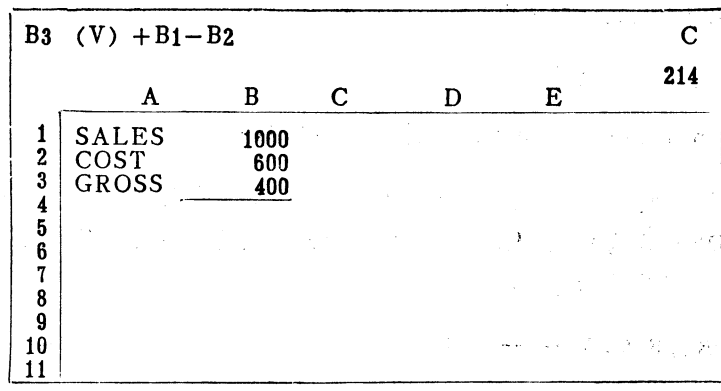


图5—3 月报表结果画面

最后，使用打印命令“‹Home›/PPB3〈CR〉”打印报表。其结果如下：

SALES	1000
COST	600
GROSS	400

2．数据插入与公式的复制

例5—2 季度销售额、成本和利润表

① 功能：把例5—1扩充成一个季度的报表，其中二月和三月的销售额分别为1200元和1500元。

② 操作：首先在第1行处插入一行月份标题行，这可通过使用插行命令“/IR”来完成。其操作命令如下：

>A1〈CR〉/IRMONTH〈CR〉→1〈CR〉→2〈CR〉→3〈CR〉

行标题输入完毕后，可以检查一下B3中的公式。打入“>B3〈CR〉”后，状态区中的单元行显示出：

B3 (V) +B2*.6

可以发现原公式中的B1已自动地调整为B2，从而保证了表格的一致性。

接着，在C2和D2中填入它们的销售额1200和1500，随后在C3、C4、D3和D4中填入计算公式。填入公式可采用直接键盘输入，但这样做输入的键太多且容易出错。为此，可采用VISICALC提供的复制命令“/R”来复制和调整这些公式，其过程如下：

* 打入命令“>B3〈CR〉/R〈CR〉C3.D3”，表示把B3中的公式复制到C3和D3去，这时，状态区中编辑行的信息如下：

B3...B3：C3...D?

● 输入〈CR〉键，编辑行的信息改变成为：

B3：C3...D3：+B2

同时 VISICALC 在提示行上显示出：

Replicate：N=No Change, R=Relative

要求你输入 N（不改变）或 R（改变为相对位置）。这里应输入“R”，以便把 B3 公式中的 B2 调整为 C2、D2 后作为 C3 和 D3 单元的计算公式。同样，也可将 B4 中的计算公式复制到 C4 和 D4 中去，其操作命令为：

>B4〈CR〉/R〈CR〉→···→→〈CR〉RR→→

这时，表格就扩充好了，如图 5—4 所示。

D4 (V) +D2−D3 C

214

	A	B	C	D	E
1	MONTH	1	2	3	
2	SALES	1000	1200	1500	
3	COST	600	720	900	
4	GROSS	400	480	600	
5					
6					
7					
8					
9					
10					
11					

图 5—4 季度报表结果画面

再使用打印命令：

〈Home〉/PPD4〈CR〉

打印输出的结果如下：

MONTH	1	2	3
SALES	1000	1200	1500
COST	600	720	900
GROSS	400	480	600

3．求和函数

例 5—3 季度总计报表

① 功能：对例 5—2 中的表格按季度求销售额、成本和利润的总计。

② 操作：采用 VISICALC 的求和函数 @SUM（ ）来求总计。首先，在 E1 中放入

总计标题“TOTAL”：

>E1〈CR〉TOTAL↓

然后，把季度总销售额的计算公式填入E2：

@SUM（B2.D2）〈CR〉

接着，把这个公式复制到E3和E4中：

/R〈CR〉↓.↓〈CR〉RR　　{复制到E3}

/R〈CR〉↓↓.↓↓〈CR〉RR　　{复制到E4}

这时，使用命令

〈Home〉/PPE4〈CR〉

打印。结果如下：

MONTH	1	2	3	TOTAL
SALES	1000	1200	1500	3700
COST	600	720	900	2220
GROSS	400	480	600	1480

4．格式说明

例5—4　改进的季度总计报表

① 功能：改进例5—3中季度总计报表的格式，使之更加符合使用要求。

② 操作：在前面的几个例子中并没有说明表格的格式，即采用的是VISICALC的缺省格式（列宽为9；标号左靠；数值右靠）。下面采用VISICALC的全局格式说明命令“/G”来修改例5—3中表格的格式。

首先，把表格中金额数据设置成元角分的形式（保留两位小数）。这可以通过打入命令

/GF$

来实现。打印出的报表如下：

MONTH	1.00	2.00	3.00	TOTAL
SALES	1000.00	1200.00	1500.00	3700.00
COST	600.00	720.00	900.00	2220.00
GROSS	400.00	480.00	600.00	1480.00

可是第1行的月份也被取了两位小数，所以还得用局部格式命令“/F”改回来：

>B1〈CR〉/FI→/FI→/FI

其次，再把表中的字符串改为在表格单元中向右靠齐存放：

→/FR〈Home〉/FR↓/FR↓/FR↓/FR〈Home〉

这时屏幕如图 5—5 所示。可见，改进后的表格更加符合实用的要求了。

```
A1   /FR  (L)  MONTH                                          C
                                                             214
        A          B          C          D          E
1
2     MONTH          1          2          3      TOTAL
3     SALES    1000.00    1200.00    1500.00    3700.00
4      COST     600.00     720.00     900.00    2220.00
5     GROSS     400.00     480.00     600.00    1480.00
6
7
8
9
10
11
```

图 5—5　改进的季度总计表

5. 百分比

例 5—5　百分比报表

① 功能：在例 5—4 的表格中再增加一行，用来计算每月的销售额占全季度的百分比。

② 操作：在 A5 中输入标题“S/T%”，在 B5、C5 和 D5 中填入百分比计算公式。其过程如下：

```
>A5〈CR〉/FRS/T%〈CR〉          {输入标题}
→(B2/E2)*100〈CR〉             {在B5中输入公式}
/R〈CR〉C5·D5〈CR〉RN           {把公式复制到C5和D5中去}
```

操作结束所产生的表格如图 5—6 所示。

```
35    (V)  (B2/E2)*100                                         C
                                                             214
        A          B          C          D          E
1
2     MONTH          1          2          3      TOTAL
3     SALES    1000.00    1200.00    1500.00    3700.00
4      COST     600.00     720.00     900.00    2220.00
5     GROSS     400.00     480.00     600.00    1480.00
6       S/T%      27.03      32.43      40.54
7
8
9
10
11
12
```

图 5—6　百分比表格的结果画面

6. 制表

例5—6 作出百分比表格的报表

① **功能**：作出百分比报表并画出简单的条形图。

② **操作**：首先，把列宽放大为12，并加入间隔线：

```
/GC12〈CR〉          {放大列宽}
〈Home〉/IR/IR↓↓↓/IR↓↓/IR↓↓/IR↓↓/IR/IR     {插入七行空行}
>B1〈CR〉AN EXAMPLE→OF TABLE〈CR〉     {写入表格标题}
>A2〈CR〉/--〈CR〉     {在A2中填满"-"}
/R〈CR〉B2.E2〈CR〉          {在第2行中填满"-"}
/R.E2〈CR〉A4〈CR〉/R.E2〈CR〉A6〈CR〉/R.E2〈CR〉A8〈CR〉     {复制}
>A10〈CR〉/-=〈CR〉/R〈CR〉B10.E10〈CR〉     {在第10行填满"="}
>B11〈CR〉"        BAR→" GRAPH OF SA→LES〈CR〉     {写条形图标题}
```

然后，画条形图：

```
>A13〈CR〉/FI1↓/FI2↓/FI3〈CR〉          {写月份}
>B13〈CR〉/F* /R〈CR〉B14.B15〈CR〉          {把B13～B15定义为"*"格式}
+B12/4↓+C12/4↓+D12/4〈CR〉          {作条形图}
```

最后，把这张表记盘保存并打印出来：

```
/SS VCEX10〈CR〉          {把表格以VCEX10为名存盘}
〈Home〉/PPE15〈CR〉     {打印报表}
```

打印出的报表如下：

AN EXAMPLE OF TABLE

MONTH	1	2	3	TOTAL
SALES	1000.00	1200.00	1500.00	3700.00
COST	600.00	720.00	900.00	2220.00
GROSS	400.00	480.00	600.00	1480.00

BAR GRAPH OF SALES

```
S/T%          27.03      32.43      40.54
   1 ******
   2 ********
   3 **********
```

第四节 VISICALC 的函数

VISICALC 提供了一组标准函数，用以实现事务处理中的常用计算。这组函数可用于建立表格单元中的计算公式，其引用格式如下：

@〈函数名〉(〈参数〉)

对于不同的函数，参数可为单一的常量或变量，也可为一个参数表（区域或枚举表），甚至可以没有参数。此后，用“V”表示单一参数，用“L”表示参数表。

下面按功能分四组介绍 VISICALC 的标准函数。

1．商用和统计函数

（1）@SUM（L）

求和函数，用以把参数表中所有值型元素的值相加。例如，输入“@SUM（A1·A 8，B1，B3）”表示要计算 A1...A8，B1，B3的和。在输入“A1·A 8”后，屏幕会显示出“A1...A8”。

（2）@MIN（L）

最小值函数，用来求得参数表中的最小值。

（3）@MAX（L）

最大值函数。

（4）@COUNT（L）

计数函数，用来计算参数表中值型元素的个数。

（5）@AVERAGE（L）

平均值函数，其值由公式

@AVERAGE（L）=@SUM（L）/@COUNT（L）

得出。

（6）@NPV（V，L）

该函数称为现得净利函数（Net Present Value function），常用来计算固定利率和固定偿还条件下，银行所获得的净利。它的计算公式为：

$$@NPV(V,L)=\sum_{i=1}^{n} Li/(1+V)^{i}$$

例如，某厂向银行借款 4000 元，准备分五个月还清，每月还款 1200 元。又设利率为 10%，则银行获得净利可用下列方法算出。

首先填入利率 A10=0.1，再在 A11 ··A15 中都填入 1200，接着输入

>A16〈CR〉@NPV（A10，A11·A15）

这时 A16 的值为 4548.944，这个数字再减去借出的本金 4000 元，得净利 548.94 元。

2．算术函数

（1）@ABS（V）

绝对值函数，求出 V 的绝对值。

（2）@INT（V）

取整函数，求出 V 的整数部分。

（3）@SQRT（V）

平方根函数。

（4）@EXP（V）

指数函数，其底为 e。

（5）对数函数

该组函数有两个：

@LOG10（V）	以 10 为底的常用对数
@LN（V）	以 e 为底的自然对数

（6）三角函数

VISICALC 使用的三角函数的角度单位均为弧度。以下是 VISICALC 的三角函数：

@SIN（V）	正弦函数
@COS（V）	余弦函数
@TAN（V）	正切函数
@ASIN（V）	反正弦函数
@ACOS（V）	反余弦函数
@ATAN（V）	反正切函数

（7）@PI

圆周率函数，其值为 3.1415926536。

3．逻辑函数

（1）逻辑常量函数

该组逻辑函数用来返回指定的逻辑值，共有以下两个：

@TRUE	返回逻辑值“TRUE”
@FALSE	返回逻辑值“FALSE”

（2）逻辑运算函数

该组函数可以用来完成逻辑运算“与”、“或”和“非”。它们分别为：

@AND（V1，V2）	“与”函数
@OR（V1，V2）	“或”函数
@NOT（V）	“非”函数

（3）条件函数

条件函数的格式为：

@IF（V1，V2，V3）

表示 V1 为真时，函数值为 V2，否则为 V3。

（4）判错函数

该组函数有两个，分别用来判别指定的单元是否出错或是否不定，其含义为：

@ ISERROR（V）　　如 V 的值为“ERROR”，则函数值为“TRUE”，否则为“FALSE”。

@ISNA（V）　　如 V 的值为“NA”，则函数值为“TRUE”，否则为“FALSE”。

4．辅助函数

（1）选择函数

选择函数的格式为：

@CHOOSE（V，L）

它根据 V 的取整值，在参数表 L 中选出第 V 个元素作为函数值。当 V 的值超出参数表的范围时，表达式的值为“NA”。例如，当 A10 为 4 时，函数 @CHOOSE（A10，A11．A15）的值为 A14 的值。

（2）查找函数

该函数的格式为：

@LOOKUP（V，L）

它在区域 L 中查找与 V 相等或小于但最接近于 V 的元素，作为匹配元素。如区域 L 以列的方式排列，函数值为匹配元素右边元素的值；如果 L 以行的方式排列，函数值就为匹配元素下边的元素的值。

（3）出错值产生函数

该组函数可用来模拟产生一个错误值，其含义如下：

@ERROR　　函数值为“ERROR”。

@NA　　函数值为“NA”。

5．应用举例

例 5—7　画出在区间〔0，π〕上的正弦函数曲线

操作过程如下：

```
／CY                          {清除表格}
／GC30〈CR〉                   {取列宽为 30}
0 ↓ @PI/20＋A1〈CR〉           {赋初值：A1＝0，A2＝π/20}
／R〈CR〉A3·A21〈CR〉R          {A1…A21 置为 0～π，步长为 π/20}
```

```
→30 * @SIN ( A2 )〈CR〉   { 在 B2 中输入公式 }
/F*                      { 定义 B2 为 * 格式 }
/R〈CR〉B3.B21〈CR〉R       { 定义 B3～B21 也为 * 格式 }
〈Home〉/PPB21〈CR〉       { 打印 }
```

命令执行的结果如下：

```
              0
.15707963268  * * * *
.31415926536  * * * * * * * * *
.47123889804  * * * * * * * * * * * * *
.62831853072  * * * * * * * * * * * * * * * * *
 .7853981634  * * * * * * * * * * * * * * * * * * * * *
.94247779608  * * * * * * * * * * * * * * * * * * * * * * * *
1.0995574287  * * * * * * * * * * * * * * * * * * * * * * * * *
1.2566370613  * * * * * * * * * * * * * * * * * * * * * * * * * *
1.4137166939  * * * * * * * * * * * * * * * * * * * * * * * * * *
1.5707963265  * * * * * * * * * * * * * * * * * * * * * * * * * *
1.7278759591  * * * * * * * * * * * * * * * * * * * * * * * * * *
1.8849555917  * * * * * * * * * * * * * * * * * * * * * * * * *
2.0420352243  * * * * * * * * * * * * * * * * * * * * * * *
2.1991148569  * * * * * * * * * * * * * * * * * * * * * *
2.3561944895  * * * * * * * * * * * * * * * * * * *
2.5132741221  * * * * * * * * * * * * * * * * *
2.6703537547  * * * * * * * * * * * * *
2.8274333873  * * * * * * * * *
2.9845130199  * * * *
3.1415926525
```

例 5—8　从三角函数表中查出 x 的正弦值

假设 A2…A22 为自变量 x 的值，B2…B22 为 SIN (x)，C2 为查找变量，D2 为查找结果。查表的过程如下：

```
>C2〈CR〉@ NA〈CR〉      { C2 中暂不给查找变量 }
→@ LOOKUP ( C2, A2.A22 )〈CR〉←      { 在 D2 中定义查找函数 }
@ PI/4〈CR〉      { 查找 x=π/4 时的正弦值，结果 D2=0.7071068 }
@ PI/8〈CR〉→     { SIN ( π/8 ) =0.3090170 }
```

这时屏幕上显示出的结果如图 5—7 所示。

例 5—9　开发票

设发票的格式为：第 1 行是发票标题；第 2 行是发票的品名、数量、单价和金额等四个小标题；第 3—7 行中的品名和数量由用户填入．品名使用 0 到 999 之间的编号表示；第 11

—17 行是品名的编号和单价；第 3 — 7 行的单价栏的计算公式为@LOOKUP（Ai,A11...A17），金额栏的公式为 +Bi * Ci（其中 i 分别为 3—7）；第 9 行合计使用公式 @SUM（D3...D7）算出。在开发票的整个计算中都采用手动方式（／GRM），凡表示钱的单元都精确到分（／F$）。发票的结构如图 5—8 所示。

D2 (V) @LOOKUP(C2,A2...A22) C 214

	A	B	C	D	E
1	X[0,@PI]	SIN(X)	X	@LOOKUP	
2	0	0	.3926991	.3090170	
3	.1570796	.1564345			
4	.3141593	.3090170			
5	.4712389	.4539905			
6	.6283185	.5877853			
7	.7853982	.7071068			
8	.9424778	.8090170			
9	1.099557	.8910065			
10	1.256637	.9510565			
11	1.413717	.9876883			
12	1.570796	1			
13	1.727876	.9876883			
14	1.884955	.9510565			
15	2.042035	.8910065			
16	2.199115	.8090170			
17	2.356194	.7071068			
18	2.513274	.5877853			
19	2.670354	.4539905			
20	2.827433	.3090170			
21	2.984513	.1564345			

图 5—7 查三角函数表

C3 /F$ (V) @LOOKUP(A3,A11...A17) C 214

	A	B	C	D	E
1		BILL			
2	Part	Quantity	Unit	Price	
3			0.00	0.00	
4			0.00	0.00	
5			0.00	0.00	
6			0.00	0.00	
7			0.00	0.00	
8					
9			**Total**	0.00	
10					
11	0	0.00			
12	1	1.35			
13	2	.65			
14	3	2.00			
15	4	1.84			
16	5	4.99			
17	999	0.00			
18					
19					
20					
21					

图 5—8 发票的结构

需要时可以把这个发票结构用命令

/SS bill〈CR〉

保存到盘上，在使用时再调出。

如果某用户购买了3号商品5件，4号商品1件，2号商品2件。营业员填写后的发票如图5—9所示。

B5 (V) 2 C 214

	A	B	C	D	E
1		BILL			
2	Part	Quantity	Unit	Price	
3	3	5	0.00	0.00	
4	4	1	0.00	0.00	
5	2	2	0.00	0.00	
6			0.00	0.00	
7			0.00	0.00	
8					
9			**Total**	0.00	
10					
11	0	0.00			
12	1	1.35			
13	2	.65			
14	3	2.00			
15	4	1.84			
16	5	4.99			
17	999	0.00			
18					
19					
20					
21					

图5—9　填写后的发票

随后，只要输入一个“！”（重计算）以及输入命令

〈Home〉/PPD9〈CR〉

就可以打印出下面的发票：

BILL

Part	Quantity	Unit	Price
3	5	2.00	10.00
4	1	1.84	1.84
2	2	0.65	1.30
		0.00	0.00
		0.00	0.00
		Total	13.14

第五节　VISICALC 的操作命令

VISICALC 的命令由斜杠“/”和一个字母（或“－”）组成。命令使用的形式为会话式，在输入“/”之后，VISICALC 的提示行会显示出下列命令“菜单”：

Command: B C D E F G I M P R S T V W －

这时任意输入一条命令，VISICALC 都会在提示行给出该命令的提示信息或给出“子命令菜单”。例如，输入“F”命令，提示行如下：

Format : D G I L R $ *

其中Format 是“F”命令的全名，冒号后的“菜单”为子命令的清单。

VISICALC 的命令采用层次结构，它的所有命令组成一棵命令树，如图 5—10 所示。

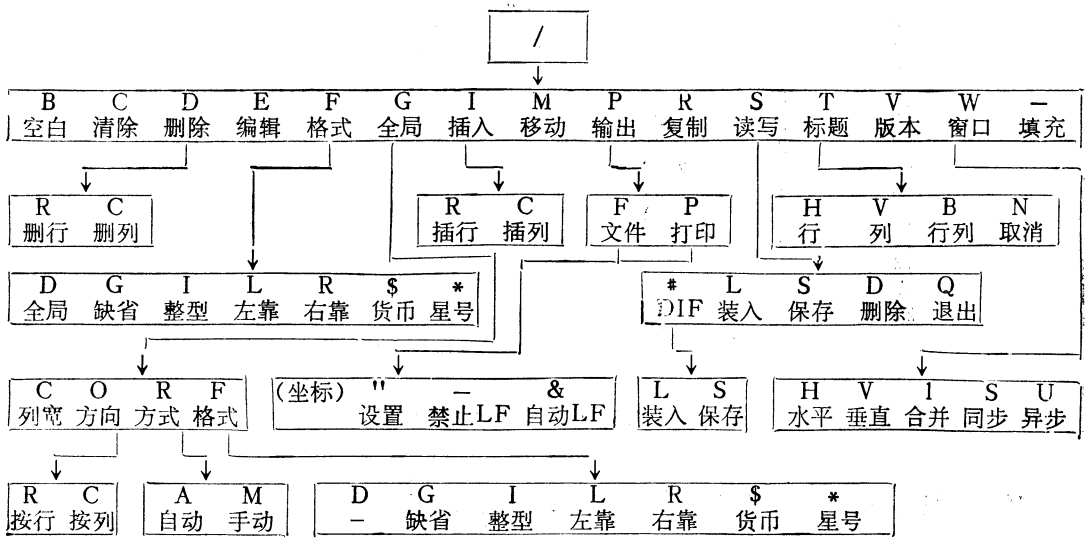


图 5—10　VISICALC的命令树

下面以字母为序，逐条介绍 VISICALC 的命令，供读者随时查阅。

1．填空白命令Blank

（1）　功能：清除指定表格单元的值。

（2）　提示：Blank

（3）　使用：

〈CR〉　　清除当前单元，光标不动。

→ ← ↑ ↓　　清除当前单元并移动光标。

其 它 键　　退出 Blank 命令状态。

2．清除命令 Clear

（1） 功能：清除全部表格。

（2） 提示：Clear：Type Y to confirm

（3） 使用：打入“Y”就可以把表格清除为初态。

3．删除命令 Delete

（1） 功能：删除一行或一列。

（2） 提示：Delete：R C

（3） 子命令

① R：删除光标所在的行。

② C：删除光标所在的列。

（4） 说明：行或列删除后，VISICALC 会自动调整公式中的单元变量，以保持表格的一致性。当公式调整发生困难时，就利用出错函数进行填补。例如：

A1=1，A2=+A1+1

A2 的值为 2，如删去第 1 行，原来的 A 2 就变为 A1，其值@ERROR+1=“ERROR”

4．编辑命令 Edit

（1） 功能：编辑当前表格单元。

（2） 提示：〔EDIT〕：Label　　{ 或〔EDIT〕：Value }

（3） 编辑键

〈Esc〉或〈BS〉	删除光标前的一个字符。
Ctrl C	撤消编辑命令。
〈CR〉	结束编辑，编辑结果送当前单元。
其它键	插入到光标处的位置上。

5．格式命令 Format

（1） 功能：设置当前表格单元的数据格式。其缺省格式：宽度为 9 个字符；标号左边对齐，数值右边对齐；数值的有效数字一般为 8 位。

（2） 提示：Format：D G I L R $ *

（3） 子命令

① D：设置当前单元为全局格式。

② G：设置当前单元为缺省格式。

③ I：设置当前单元为整型格式。这种格式的值是通过舍入取整得到的。如发生溢出，则填入八个大于号“>”。

④ L：数值靠左存放，其最高位为符号位。

⑤ R：标号靠右存放。

⑥ $：保留两位小数，主要用于表示货币的数量。

⑦ *：表示格子中填入单元值所对应的若干个星号，星号的个数通过对单元值取整得到。这种格式常用来作简单的条形图。

（4） 例子

例5—10 画条形图

设A1...A4是四种产品占总产值的百分比，分别为20%，40%，30%和10%。定义B1...B4的格式为“/F*”，表示按其内容在单元中输出若干个*号，B1…B4的内容为Ai/10（i=1，2，3，4），这样B1...B4就构成了一幅条形图。

操作过程如下：

```
/CY20↓40↓30↓10〈CR〉                    {送初值}
>B1〈CR〉/F* /R〈CR〉B2.B4〈CR〉           {定义格式}
+A1/10〈CR〉/R〈CR〉B2.B4〈CR〉R           {作条形图}
>A1〈CR〉/PPB4〈CR〉                      {打印结果}
```

打印出的结果如下：

```
20 * *
40 * * * *
30 * * *
10 *
```

6. 全局命令 Global

（1） 功能：定义表格的全局格式。

（2） 提示：Global：C O R F

（3） 子命令

① C：定义单元的宽度。这条子命令的提示信息为：

Column width {请输入列的宽度}

这时可根据需要输入3到77之间的一个数，用以设置列宽。

② O：定义计算顺序方向指示器。其提示信息为：

Reeval： R C

其中，R表示按行，C则表示按列。使用这条命令更改方向指示器后，仅在重新计算时才生效。

③ R：设置单元值的重计算方式。提示行中的信息为：

Recalc：A M

其中A表示自动重计算，M表示手动重计算（使用“!”重计算）。其缺省值为自动重计算。在表格的调试过程中，为了方便起见，可以把值的重计算方式改为手动方式。

④ F：定义全局单元格式。其提示信息为：

Format：D G I L R $ *

其中各子命令的含义同格式命令Format。任一表格单元显示数据的过程如图5—11所示。

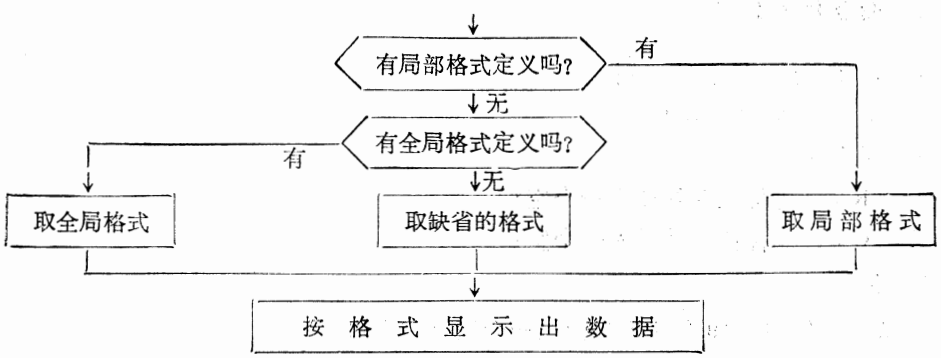


图5—11　显示数据的过程

（4）　例子

例 5—11　　打印出九九表

对于九九表来说，由于缺省的列数 9 太宽，故把它改为 3 。其操作过程如下：

```
/CY/GC3〈CR〉1→1+←〈CR〉/R〈CR〉C1.I1〈CR〉R      { 九九表的第 1 行 }
←↓1+↑〈CR〉/R〈CR〉A3.A9〈CR〉R      { 九九表的第 1 列 }
→+←*↑〈CR〉/R〈CR〉C2.I2〈CR〉NR      { 九九表的第 2 行 }
/R〈CR〉B3.B9〈CR〉RN      { 九九表的第 2 列 }
```

同样可以生成九九表的第 3 列到第 9 列，这样九九表就制成了。再使用命令

```
〈Home〉/PPJ10〈CR〉
```

可以打印出九九表：

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

从表面看，制作九九表的过程较为麻烦，但只要改动 A1 的值，就可以得到另一张乘法表。例如，执行下列命令：

```
〈Home〉/GC4〈CR〉11〈CR〉/PPJ10〈CR〉
```

可以得到一张11～19的乘法表：

11	12	13	14	15	16	17	18	19
12	144	156	168	180	192	204	216	228
13	156	169	182	195	208	221	234	247
14	168	182	196	210	224	238	252	266
15	180	195	210	225	240	255	270	285
16	192	208	224	240	256	272	288	304
17	204	221	238	255	272	289	306	323
18	216	234	252	270	288	306	324	342
19	228	247	266	285	304	323	342	361

7. 插入命令 Insert

（1） 功能：插入一行或一列。

（2） 提示：Insert：R C

（3） 子命令

① R：在光标位置处插入一个空行。

② C：在光标位置处插入一个空列。

（4） 例子

例 5—12 在 11～19 乘法表中插入一个空行和空列

操作过程：

>E5〈CR〉/IR/IC

结果如图 5—12 所示。

E5 C 214

	A	B	C	D	E	F	G	H	I	J	K
1	11	12	13	14		15	16	17	18	19	
2	12	144	156	168		180	192	204	216	228	
3	13	156	169	182		195	208	221	234	247	
4	14	168	182	196		210	224	238	252	266	
5											
6	15	180	195	210		225	240	255	270	285	
7	16	192	208	224		240	256	272	288	304	
8	17	204	221	238		255	272	289	306	323	
9	18	216	234	252		270	288	306	324	342	
10	19	228	247	266		285	304	323	342	361	
11											

图 5—12 插入行和列

8. 移动命令 Move

（1） 功能：把表格的一行或一列移到表格的另一行或另一列的位置上。

（2） 提示：Move：From…To

（3） 使用：先给出要移动的行（列），再给出目的行（列）。

（4） 例子

例 5—13 移动例 5—12 中的乘法表

先用命令

>K1〈CR〉@SUM（F1.J1）

在 K1 中输入一个求和公式，则屏幕上的画面如图 5—13 (a)所示。

接着移动第2列，其过程如下：

```
>B1〈CR〉          {把光标移到第2列}
/M·H1〈CR〉        {把第2列移到第7列}
```

移动后的画面如图5—13(b)所示。其中，原来的B列移到G列的位置，而原来的C～G列均左移1列。光标位置保持不动。

K1 (V) @SUM(F1...J1) C 214

	A	B	C	D	E	F	G	H	I	J	K
1	11	12	13	14		15	16	17	18	19	85
2	12	144	156	168		180	192	204	216	228	
3	13	156	169	182		195	208	221	234	247	
4	14	168	182	196		210	224	238	252	266	
5											
6	15	180	195	210		225	240	255	270	285	
7	16	192	208	224		240	256	272	288	304	
8	17	204	221	238		255	272	289	306	323	
9	18	216	234	252		270	288	306	324	342	
10	19	228	247	266		285	304	323	342	361	
11											

(a) 移动前的画面

B1 (V) 1+G1 C 214

	A	B	C	D	E	F	G	H	I	J	K
1	11	13	14		15	16	12	17	18	19	97
2	12	156	168		180	192	144	204	216	228	
3	13	169	182		195	208	156	221	234	247	
4	14	182	196		210	224	168	238	252	266	
5											
6	15	195	210		225	240	180	255	270	285	
7	16	208	224		240	256	192	272	288	304	
8	17	221	238		255	272	204	289	306	323	
9	18	234	252		270	288	216	306	324	342	
10	19	247	266		285	304	228	323	342	361	
11											

(b) 移动后的画面

图5—13 乘法表的移动

9. 输出命令 Print

（1） 功能：把制作的表格从打印机输出或写到磁盘文件中。

（2） 提示：Print：File，Printer

（3） 子命令

① F：表示把设计的表格按打印格式写到盘上去。提示为：

Print：File name　　{请求用户输入文件名}

这时就可以输入文件名了。在输入文件名时，不必输入类型名，VISICALC会自动加上.PRF。输入文件名后，提示行显示出：

Print：Lower right，"setup，—，&

一般仅要输入欲输出的表格的右下角坐标即可。也可以更改输出的格式：

" 更改打印机的控制字符

— 禁止输出回车换行符

& 自动换行（在回车符后加上换行符）

② P：从打印机上输出表格。其提示信息为：

Print：Lower right，"setup，—，&

提示行的各项含义同F子命令。

（4）例子

例5—14　以小号字打印九九表

如果使用命令

/PPJ11〈CR〉

打印出的九九表如下：

9＊9 Table

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

那么使用命令：

/PP"^H0F〈CR〉J11〈CR〉　　{^H0F表示把打印机改成小号字方式}

打印出的九九表如下：

9＊9 Table

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

此外，较常用的打印机控制字符还有以下几个：

^CL　换页。

^C/　等价于^H0F，表示小号字。

^EE　加重字体（印标题用）。

10. 复制命令 Replicate

（1） 功能：复制表格单元中的常量或公式。

（2） 提示：Replicate : Source range or ENTER

（3） 使用方法：

在打入 /R 后，编辑行上显示出当前单元的坐标，如不是从当前单元开始复制，则打入〈Esc〉键。输入了源范围之后，输入一个冒号（或〈CR〉键），屏幕上提示行显示出：

Replicate : Target range

这时，可以输入目的范围。源范围和目的范围都必须是一行或一列中相邻的单元，且两者为行或为列须一致，即不允许把行复制到列，也不允许把列复制到行。

如果复制的是公式，VISICALC 还会提问是否要对公式中变量进行调整。这种调整是依照目的范围决定的。提示行显示出：

Replicate : N＝No Change，　R＝Relative

如回答“N”表示不作调整，回答“R”表示作调整。

（4） 例子

例 5—15　打印一张 21～30 的乘法表

设屏幕上已有如图 5—14 所示的一张九九表

A1 (V) 1　　C 214

	A	B	C	D	E	F	G	H	I	J	K
1	1	2	3	4	5	6	7	8	9		
2	2	4	6	8	10	12	14	16	18		
3	3	6	9	12	15	18	21	24	27		
4	4	8	12	16	20	24	28	32	36		
5	5	10	15	20	25	30	35	40	45		
6	6	12	18	24	30	36	42	48	54		
7	7	14	21	28	35	42	49	56	63		
8	8	16	24	32	40	48	56	64	72		
9	9	18	27	36	45	54	63	72	81		
10											
11											

图 5—14　九九表

首先，把九九表扩充为10×10乘法表。其过程如下：

```
>A2〈CR〉/R      { 这时提示行为. Replicate : Source range or ENTER }
.I2〈CR〉        { 提示行: Replicate : Target range }
                { 输入行: A2...I2 : }
A10〈CR〉        { 提示行: Replicate : N=No Change , R=Relative }
R               { 把1+A1改为1+A9 }
RN              { 在复制B2到B10时把公式改为+A10 * B1 }
```

重复这一步直至把 I2 复制到 I10 为止。使用命令

```
>I10〈CR〉
```

后显示出的结果如图 5—15 所示。

I10 (V) +A10*I1　　C 214

	A	B	C	D	E	F	G	H	I	J	K
1	1	2	3	4	5	6	7	8	9		
2	2	4	6	8	10	12	14	16	18		
3	3	6	9	12	15	18	21	24	27		
4	4	8	12	16	20	24	28	32	36		
5	5	10	15	20	25	30	35	40	45		
6	6	12	18	24	30	36	42	48	54		
7	7	14	21	28	35	42	49	56	63		
8	8	16	24	32	40	48	59	64	72		
9	9	18	27	36	45	54	63	72	81		
10	10	20	30	40	50	60	70	80	90		
11											

图 5—15　　10×10表

使用类似的方法，可以把第 B 列复制到第 J 列：

```
>B1〈CR〉/R .B10〈CR〉J1〈CR〉R
NR NR NR NR NR NR NR NR NR
```

这样就制成了10×10表。在此基础上使用下面的命令：

```
〈Home〉21〈CR〉/PPK11
```

即可打印出21～30的乘法表：

21	22	23	24	25	26	27	28	29	30
22	484	506	528	550	572	594	616	638	660
23	506	529	552	575	598	621	644	667	690
24	528	552	576	600	624	648	672	696	720
25	550	575	600	625	650	675	700	725	750
26	572	598	624	650	676	702	728	754	780
27	594	621	648	675	702	729	756	783	810
28	616	644	672	700	728	756	784	812	840
29	638	667	696	725	754	783	812	841	870
30	660	690	720	750	780	810	840	870	900

11. 读写命令 Storage

（1） 功能：进行磁盘操作，保存和装入表格文件，删除磁盘文件等。

（2） 提示：Storage：L S D Q #

（3） 子命令

① L：从磁盘上装入保存在盘上的表格文件，其类型名为.VC。提示行为：

Storage：File to Load

这时，可直接输入文件名，也可以使用→键逐个显示盘上的表格文件，直至显示出所需要文件，再打〈CR〉即可。装入表格的未使用部分不覆盖已有表格，为了避免混淆，通常在这条命令前使用/CY命令清除内存中的当前表格。

② S：把内存中的表格以文件的形式保存到盘上。提示行的信息为：

Storage：File for Saving

这时可输入文件名。如果给出的文件名盘上已经存在，则提示行显示出：

Storage：File exists. Y to replace

如输入“Y”就可以替换盘上的文件。在这条子命令中也可以使用→键查看盘上的内容。

③ D：删除盘上的文件。要删除文件的文件名可直接输入，也可用→键选择。

④ Q：退出 VISICALC 返回操作系统，键入“Y”后生效。

⑤ #：读写数据转换文件（其类型名为.DIF）。这种数据转换文件主要用于 VISICALC 与其它高级语言交换数据。这条子命令的提示行如下：

Data：Save Load

这时如输入“S”表示写盘，输入“L”表示读盘。

（4） 例子

例 5—16 VISICALC 的三类文件

首先制作一个 5×5 表，如图 5—16 所示。

A1 C 214

	A	B	C	D	E	F	G	H	I	J	K
1	--	5*5	Table								
2	1	2	3	4	5						
3	2	4	6	8	10						
4	3	6	9	12	15						
5	4	8	12	16	20						
6	5	10	15	20	25						
7											
8											
9											
10											
11											

图 5—16 5×5表

再以四种形式把这张表格记盘：

/SSS5X5〈CR〉 {以 S5X5.VC 为名记盘}

```
/PFP5X5〈CR〉F7〈CR〉          { 以 P5X5.PRF 为名记盘 }
/S#SD5X5〈CR〉F7〈CR〉R        { 以 D5X5.DIF 为名按行记盘 }
/S#SD5X5C〈CR〉F3〈CR〉C       { 以 D5X5C.DIF 为名，按列存放表格的三行 }
>F7〈CR〉/S#LD5X5C〈CR〉R      { 在F7处按行装入D5X5C文件，结果如下图 }
```

F7

	A	B	C	D	E	F	G	H	I	J	K
1		5*5	Table								
2	1	2	3	4	5						
3	2	4	6	8	10						
4	3	6	9	12	15						
5	4	8	12	16	20						
6	5	10	15	20	25						
7							1	2			
8					5*		2	4			
9					5Ta		3	6			
10					ble		4	8			
11							5	10			

图5—17　按列存放表

接着使用命令“/SQY”退出 VISICALC，再使用 MS-DOS 的 TYPE 命令打印出这几个文件。其过程和结果如下：

```
A> TYPE S5X5.VC
>E6：+A6*E2
>D6：+A6*D2
>C6：+A6*C2
>B6：+A6*B2
>A6：1+A5
>E5：+A5*E2
>D5：+A5*D2
>C5：+A5*C2
>B5：+A5*B2
>A5：1+A4
>E4：+A4*E2
>D4：+A4*D2
>C4：+A4*C2
>B4：+A4*B2
>A4：1+A3
>E3：+A3*E2
>D3：+A3*D2
>C3：+A3*C2
>B3：+A3*B2
>A3：1+A2
>E2：1+D2
>D2：1+C2
>C2：1+B2
>B2：1+A2
>A2：1
>D1："ble
>C1："5 Ta
>B1："  5*
```

表格单元的内容

```
/W1
/GOC
/GRA                    } 格式说明及指针位置的指定
/GC4
/X>A1：>A1：
A> TYPE P5X5.PRF
```

5 * 5 Table

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

数据转换文件，由文件头标、文件内容和文件尾标三部分组成，现把打印文件D5X5.DIF得出的结果给出如下（由于排版原因，分成自左至右八列）：

	文件头标	文件内容(1～6列)						文件尾标
	TABLE	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0	-1,0
	0,1	BOT	BOT	BOT	BOT	BOT	BOT	EOD
	" "	1,0	1,0	1,0	1,0	1,0	1,0	
行数	VECTORS	" "	" 5*"	"5 Ta"	"ble "	" "	" "	
行数	0,7	0,1	0,2	0,3	0,4	0,5	1,0	
行数	" "	V	V	V	V	V	" "	
列数	TUPLES	0,2	0,4	0,6	0,8	0,10	1,0	
列数	0,6	V	V	V	V	V	" "	
列数	" "	0,3	0,6	0,9	0,12	0,15	1,0	
数据	DATA	V	V	V	V	V	" "	
数据	0,0	0,4	0,8	0,12	0,16	0,20	1,0	
数据	" "	V	V	V	V	V	" "	
		0,5	0,10	0,15	0,20	0,25	1,0	
		V	V	V	V	V	""	
		1,0	1,0	1,0	1,0	1,0	1,0	
		" "	" "	" "	" "	" "	" "	

如果数据交换文件是按列存放的，其结果与文件D5X5.DIF中数据的行列顺序相反。

12．标题命令Title

（1） 功能：设置标题，标题行或列不随屏幕的表格区移动而移动。

（2） 提示：Title：H V B N

（3） 子命令

① H：设置光标所在行之上的所有行为标题区（包括光标所在的行）。

② V：设置光标所在列左边的所有列为标题区（包括光标所在的列）。

③ B：为H和V子命令的综合，光标的左上部为标题区。

④ N：删除标题区。

（4） 说明：被定义为标题区的表格单元，不能使用光标移动键定位，但可使用参数

定位法（“>”命令）定位。

13．版本命令Version

该命令可用来显示VISICALC的版本号。其状态区显示出的信息与VISICALC的初始画面相同。

14．窗口命令Window

（1） 功能：把屏幕的表格区分成两个窗口，以便对照两张表格。

（2） 提示：Window：H V 1 S U

（3） 子命令

① H：以光标所在的行为界把表格区分成两个窗口。其中光标所在的窗口为主窗口，它可以自由移动，而另一个窗口为辅窗口，它固定不动。要把光标从一个窗口移到另一个窗口，可以输入分号“；”。

② V：以光标所在的列为界把表格区分成两个窗口。

③ 1：把两个窗口合成一个窗口。

④ S：同步移动，即在水平（或垂直）移动一个窗口时，另一个窗口也同步移动。

⑤ U：取消同步移动。

（4） 例子

例5—17 设置窗口

设屏幕上有一张九九表，打入命令

>A5〈CR〉/WH

后，表格区以A5为界水平地分成了两个窗口。再输入命令

/GC8〈CR〉

把主窗口的列宽改为8，然后移动光标把第E列移出主窗口，这时屏幕如图5—18所示。

K4　　C 214

	F	G	H	I	J	K
1	6	7	8	9		
2	12	14	16	18		
3	18	21	24	27		
4	24	28	32	36		

	A	B	C	D	E	F	G	H	I	J	K
5	5	10	15	20	25	30	35	40	45		
6	6	12	18	24	30	36	42	48	54		
7	7	14	21	28	35	42	49	56	63		
8	8	16	24	32	40	48	56	64	72		
9	9	18	27	36	45	54	63	72	81		
10											
11											

图 5—18　两个窗口

15．重复填充命令 —

（1） 功能：使用指定的字符串填充整个当前单元。

（2） 提示：Label：Repeating

第六节 MULTIPLAN 简介

MULTIPLAN 是另一个使用较为广泛的表格处理软件，它的功能比 VISICALC 强，使用起来也比 VISICALC 方便。

与 VISICALC 相比，MULTIPLAN 主要增加了以下几项功能：

* 可以对表格单元或表格区域命名，以便按名引用。
* 可以显示表格单元中的公式。
* 允许对表格按列排序。
* 可提供多窗口显示。
* 提供较强的联机求助功能，用户可随时查阅联机手册。
* 允许引用外部表格。

1. MULTIPLAN 的操作

（1）启动

MULTIPLAN 的启动命令是：

A>MP80

输入这条命令后，屏幕上显示出 MULTIPLAN 的初态画面如图 5—19 所示。

```
#1      1       2       3       4       5       6       7
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
COMMAND: Alpha Blank Copy Delete Edit Format Goto Help Insert Lock Move
         Name Options Print Quit Sort Transfer Value Window Xternal
Select option or type command letter
R1C1                                    100% Free NL Multiplan: TEMP
```

图 5—19 MULTIPLAN 的初态画面

初态画面的最末四行为状态区。状态区的前两行为命令“菜单”；第3行为提示行；第4行为当前单元行。

初态画面的其余部分为表格显示区。表格显示区左上角的“#1”表示窗口号。MULTIPLAN的行列标号均为数字，一屏可以显示20行×7列表格。每个单元的缺省格式为：宽度为10；数值向右对齐；标号向左对齐。

MULTIPLAN使用命令的方法与VISICALC不完全相同。它提供两种方法使用命令，第1种是直接输入命令的第1个字母，第2种是使用功能键〈F9〉和〈F10〉选择需要的命令并打入〈CR〉键。

此外，MULTIPLAN还具有较强的联机求助功能。这可以通过选择“HELP”命令实现，也可以随时使用Alt-H键来查询联机手册。

（2）当前单元的定位

MULTIPLAN允许使用的当前单元的定位键如表5-2所示。

表5-2　MULTIPLAN的定位键与功能键

定位键	功能	功能键	功能
↑	上移一个单元	‹Home›	移到窗口的左上角
↓	下移一个单元	‹F1›	移到下一窗口
→	右移一个单元	‹F2›	移到下一非保护单元
←	左移一个单元	‹F3›	绝对坐标
PgUp	上移一页(20行)	‹F4›	重计算
PgDn	下移一页(20行)	‹F7›	左移一个单词
Ctrl-→	右移一页(7列)	‹F8›	右移一个单词
Ctrl-←	左移一页(7列)	‹F9›	左移一个字符或一个菜单项目
‹End›	把光标移到表格右下角	‹F10›	右移一个字符或一个菜单项目
Ctrl-PgUp	把光标移到R1 C1	‹Tab›	移到下一字段

MULIPLAN也可以使用GOTO命令进行参数定位，其中参数可以是单元坐标、单元名或窗口号。

（3）数据和公式的输入

MULTIPLAN采用命令V和A来区分输入的是数值公式还是字符串。其中数值的表示范围约为-10^{62}～10^{62}，精度为14位有效数字。

MULTIPLAN的变量有以下三种形式：

* 绝对坐标：例如，R1C1，R1，R1：5等。
* 相对坐标：相对于当前单元的坐标。例如，R〔+2〕C〔-1〕表示当前单元的左列下两行的单元。
* 单元名称：是用命名命令（NAME）对绝对坐标起的名字。

其中绝对坐标又具有以下六种形式：

RnCm　单个表格单元（第n行第m列上的单元）。

Rn　第n行。

Cn　　第 n 列。

Rn：m　第 n 行到第 m 行间的区域。

Cn：m　第 n 列到第 m 列间的区域。

Rn：Cm　矩形区域。

此外，MULTIPLAN 的运算符比 VISICALC 增加了两个：

%　百分比，等效于／100。

&　字符串并置运算，例如“A”&“BC”的结果为“ABC”。

其中第 2 个算符“&”为一个字符串运算符（MULTIPLAN 的公式中允许出现字符串数据）。除此以外，MULTIPLAN 还提供了一组字符串函数。

2. MULTIPLAN 的函数

MULTIPLAN 中使用 ALPHA 和 VALUE 命令区分字符串和公式，所以函数前不必带引导符。而 VISICALC 需使用@作为函数引导符。

在下文中，“L”表示参数表，“V”表示单一值型参数，“T”表示字符串参数。

（1）商用和统计函数

① 求和函数：SUM（L）

② 最大值、最小值函数：MAX（L），MIN（L）

③ 计数函数：COUNT（L）

④ 平均值函数：AVERAGE（L）

⑤ 现得净利函数：NPV（V，L）

⑥ 标准方差函数：STDEV（L）

⑦ 迭代△函数：DELTA（ ）

这个函数用来设置迭代的出口条件，当两次迭代值的差小于设定的值时，结束迭代。

⑧ 迭代次数函数：ITERCNT（ ）

这个函数可设置迭代的次数。

（2）算术函数

① 绝对值函数：ABS（V）

② 取整函数：INT（V）

③ 平方根函数：SQRT（V）

④ 指数函数：EXP（V）

⑤ 对数函数：LOG10（V），LN（V）

⑥ 三角函数：SIN（V），COS（V），TAN（V），ATAN（V）

⑦ 圆周率函数：PI（ ）

⑧ 取模函数：MOD（V1，V2）

⑨ 舍入函数：ROUND（V1，V2）

对 V1 舍入，保留 V2 位小数。

⑩ 符号函数：SIGN（V）

（3）逻辑函数

① 逻辑常量函数：TRUE（ ），FALSE（ ）

② 逻辑运算函数：AND（V1，V2），OR（V1，V2），NOT（V）

③ 条件函数：IF（V1，V2，V3）

④ 判错函数：ISERROR（V），ISNA（V）

（4）字符串函数

① 货币串函数：DOLLAR（V）

该函数把V转换成字符串并加上“$”。

② 舍入串函数：FIXED（V1，V2）

把V1转换为带V2位小数的字符串。

③ 长度函数：LEN（T）

用来计算字符串T的长度。

④ 子字符串函数：MID（T，V1，V2）

从字符串T的第V1个字符开始取出V2个字符组成一个子字符串。

⑤ 重复函数：REPT（T，V）

把字符串T重复V次，组成一个新字符串。

⑥ 取值函数：VALUE（T）

把字符串形式的数值转换成数值形式。

（5）辅助函数

① 选择函数：

* INDEX（L，V）　　取L中的第V个元素的值为函数值。
* INDEX（L，V1，V2）　取区域L中第V1行V2列元素的值为函数值。

② 查找函数：LOOKUP（V，L）

③ 不定值产生函数：NA（ ）

④ 行列号函数：

* COLUMN（ ）　　当前表格单元的列号。
* ROW（ ）　　当前表格单元的行号。

3. MULTIPLAN的命令

MULTIPLAN启动后，在状态区显示出命令“菜单”，可用〈F9〉和〈F10〉移动光标选择命令，也可直接输入命令的第1个字母。

下面按字母为序介绍MULTIPLAN的所有命令。

（1）字符串模式命令 ALPHA

该命令用来说明输入的数据为字符串。如果使用光标移动键结束输入或编辑，MULTIPLAN则根据输入的第1个字符决定数据的类型。第1个字符为0～9，＋，－（和“为值型，其它字符则为字符串型。

（2）填空白命令 BLANK

用来清除非保护的单元或区域。

（3）复制命令 COPY

这条命令用来复制一个表格单元或一个表格区域。如果使用的是相对坐标，MULTIPLAN会自动调整坐标参数。复制命令有以下三种形式：

* COPY RIGHT　行内的复制。
* COPY DOWN　列内的复制。
* COPY FROM　区域复制。

（4）删除命令 DELETE

删除表格的若干行或若干列：

* DELETE ROWS　　删行。
* DELETE COLUMNS　删列。

（5）编辑命令 EDIT

把当前单元的内容显示在编辑行上，用户编辑修改后，用〈CR〉键存入当前单元。

（6）格式命令 FORMAT

格式命令用来设置表格的格式，它有以下六种形式：

* FORMAT DEFAULT CELLS　定义全局格式。
* FORMAT CELLS　定义局部格式。
* FORMAT WIDTH　定义列宽（3～32 个字符）。
* FORMAT DEFAULT　定义为缺省格式。
* FORMAT DEFAULT WIDTH　定义为缺省列宽。
* FORMAT OPTIONS　如选“comma”表示数值每3位加一个逗号；如选“formulas”表示显示公式。

其中前两式可定义表格单元中数据的位置和格式，它们的含义分别见表 5-3 和表5-4。

表 5-3　位置参数

参数	含　义	参数	含　义
Def	全局格式	Left	向左对齐
Ctr	中心对齐	Right	向右对齐
Gen	缺省格式	—	无作用

表 5-4　格式参数

参数	含　义	参数	含　义
Def	全局格式	Fix	定点格式
Gen	缺省格式	$	货币格式
Cont	连续存放（可伸入下一空单元）	*	“*”串格式
Exp	指数格式	%	百分数格式
Int	整型格式	—	无作用

（7）转移命令 GOTO

该命令用来按指定参数定位当前单元，它有下列三种形式：

* GOTO NAME　　按名转移。
* GOTO ROW-COL　按坐标转移。
* GOTO WINDOW　转移到指定窗口。

（8）插入命令 INSERT

用来插入若干行或若干列。

（9）保护命令 LOCK

该命令用来对表格单元加保护，使它的内容不能被修改或删除，它有下列两种格式：

* LOCK CELLS　　对单元加/去保护。
* LOCK FORMULAS　对公式加保护。

（10）移动命令 MOVE

可用来移动若干行或若干列。

（11）命名命令 NAME

该命令用来对一个单元或单元区域命名，它的参数有：

define name：〈名〉

to refer to：〈区域〉

其中名可由多达 32 个字符的字符串组成，这个字符串的第 1 个字符须为字母。这里区域可为单元、行列和矩形区域。

（12）选择命令 OPTIONS

该命令可用来选择 MULTIPLAN 的一些特殊功能，它有以下几个参数：

* recalc（重新计算）　选“NO”表示采用手动重计算（按〈F4〉），否则为自动重计算。
* mute（报警）　是否停止报警的声音。
* iteration（迭代）　计算迭代公式的值，直到指定的条件满足为止。其缺省条件为：

DELTA（ ）＜0.001

（13）输出命令 PRINT

把表格从打印机输出或输出到指定的文件中。它有四条子命令：

* PRINT PRINTER　打印表格。
* PRINT FILE　表格写盘。
* PRINT MARGINS　设置打印格式，其参数如下：

left：n　左边框（缺省值为 5）

top：n　页间隔（缺省值为 6 行）

print width：n　行宽（缺省值为 70）

print length：n　页长（缺省值为 66）

* PRINT OPTION　　设置打印参数，它的参数有以下几项：

area：〈范围〉 打印的表格范围（缺省值为整个表格）。

setup：选择打印机并设置打印机参数(控制键用“^”表示；〈Esc〉键用 ^〔表示。例如，〈Esc〉G 表示为 ^〔G）。

formulas： 打印单元中的公式。

row-col numbers： 选择打印时是否打印行列号。

（14）退出命令 QUIT

退出 MULTIPLAN 返回操作系统。

（15）排序命令 SORT

这条命令可按指定的关键列对表格进行排序，其参数如下：

by column：〈关键列号〉

between rows：〈始行〉 and：〈末行〉

order：〈顺序〉 {“>”表示增序；“<”表示减序}

（16）传送命令 TRANSFER

这条命令主要用来保存、装入、删除和重命名表格文件。它分以下几个子命令：

* TRANSFER LOAD　　装入指定的表格文件。
* TRANSFER SAVE　　把内存中的表格保存到盘上。
* TRANSFER CLEAR　　清除整个内存中的表格。
* TRANSFER DELETE　　删除盘上指定的表格文件。
* TRANSFER OPTION　　设置传送方式。可设置的格式有以下几种：

Normal	MULTIPLAN 表格文件方式
Symbolic	ASCII 码文件方式（SYLK 文件）
Other	VISICALC 表格文件格式

* TRANSFER RENAME 重命名。

（17）值命令 VALUE

表示输入的数据是数值。

（18）窗口命令 WINDOW

在表格区上开若干个窗口。这条命令下的子命令有这样几条：

* WINDOW SPLIT HORIZONTAL　横向开窗口。
* WINDOW SPLIT VERTICAL　纵向开窗口。
* WINDOW SPLIT TITLES　把当前窗口劈为两个或四个同步窗口。
* WINDOW BORDER　在窗口上加边框线，如已有边框线，则抹去边框线.
* WINDOW CLOSE　删除指定的窗口。
* WINDOW LINK　连接两个窗口为同步移动或异步移动。
* WINDOW PAINT　在窗口里着色。

（19）外部命令 XTERNAL

这条命令用来引用外部表格，它的子命令有：

* EXTERNAL COPY　从外部表格中取出数据复制到当前表格中。其参数如下：

  from sheet：〈文件名〉

  name：〈源区域名〉

  to：〈目的区域〉

  linked：〈关联否〉

* EXTERNAL LIST　列出当前表格使用的外部表格区域及向别的表格提供的外部区域。

* EXTERNAL USE　为外部表格取名，以便在EXTERNAL COPY子命令中直接引用。这条子命令的参数如下：

  filename：〈文件名〉

  instead of：〈外部表格名〉

4．应用举例

下面通过几个例子具体地说明MULTIPLAN的使用方法，并对前面介绍的MULTIPLAN的功能作一些补充。

例5—18　九九表

① 功能：制作一份九九表。

② 操作：操作过程及其说明如下：

TCY　{清除表格}

FDW4〈CR〉　{定义列宽为4}

V1↓1+R〔−1〕〈CR〉　{在R1C1中填入1；在R2C1中填入公式“1+R〔−1〕”}

CD7〈CR〉　{把R2C1中的公式在列内复制七次}

〈Home〉→V1+C〔−1〕〈CR〉CR7〈CR〉　{制作九九表的第1行}

在上面制作九九表的第1列和第1行的过程中，V命令表示输入的数据是数值。另外，为了方便地复制公式，引用单元时采用相对坐标。例如，R〔−1〕表示当前单元行号减1的单元。接着填入九九表的第2列和第2、3行：

↓VC1*R1〈CR〉CD7〈CR〉　{九九表的第2列}

CR7〈CR〉　{制作第2行}

↓CR7〈CR〉　{制作第3行}

用同样的方法可制作4～9行。这时屏幕上显示出的九九表如下：

#1	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81
10									

最后，显示表格单元中的公式。其过程为：打入命令

FO

状态区显示出：

FORMAT OPTIONS comma : Yes (No) formulas : Yes (No)

使用〈Tab〉和〈F9〉把光标移到“formulas :”后的“Yes”处，打入〈CR〉键。这时，MULTIPLAN 会自动调整列宽并显示表格单元中的公式。其结果如下：

#1	1	2	3	…	8	9
1	1	1+C[−1]	1+C[−1]	…	1+C[−1]	1+C[−1]
2	1+R[−1]	C1*R1	C1*R1	…	C1*R1	C1*R1
3	1+R[−1]	C1*R1	C1*R1	…	C1*R1	C1*R1
4	1+R[−1]	C1*R1	C1*R1	…	C1*R1	C1*R1
5	1+R[−1]	C1*R1	C1*R1	…	C1*R1	C1*R1
6	1+R[−1]	C1*R1	C1*R1	…	C1*R1	C1*R1
7	1+R[−1]	C1*R1	C1*R1	…	C1*R1	C1*R1
8	1+R[−1]	C1*R1	C1*R1	…	C1*R1	C1*R1
9	1+R[−1]	C1*R1	C1*R1	…	C1*R1	C1*R1
10						

例 5—19　制作 28～32 的乘法表

① 功能：制作九九表，在九九表的基础上制作 28～32 的乘法表。

② 操作：

首先用简单的方法重新制作九九表：

TCY　{ 清除表格 }

V1↓1+R[−1]〈CR〉CD7〈CR〉　{ 填入第 1 列 }

〈Home〉→V1+C[−1]〈CR〉CR7〈CR〉　{ 填入第 1 行 }

↓VC1*R1〈CR〉　{ 在 R2C2 中填入公式 }

CF〈Tab〉〈F8〉：R9C9〈CR〉　{ 把 R2C2 复制到 R2C2：R9C9 区域中 }

FDW4〈CR〉　{ 定义整个表格的列宽为 4 }

FW6〈Tab〉5〈Tab〉9〈CR〉　{ 5～9 列的宽度置为 6 }

WS〈CR〉5〈CR〉　{ 以第 5 行为界把表格区分成两个窗口 }

〈F1〉WSV5〈CR〉　{ 从第 5 列处把 1 号窗口分成两个窗口 }

〈F1〉〈F1〉WSV5〈CR〉　{ 把第 2 号窗口从第 5 列处分成两个窗口 }

〈F1〉〈Home〉28〈CR〉　{ 在 R1C1 中填入 28 }

经过上述操作得到的四个窗口的显示结果分别如下：

#1	1	2	3	4
1	28	29	30	31
2	29	841	870	899
3	30	870	900	930
4	31	899	930	961

#3	5	6	7	8	9
1	32	33	34	35	36
2	928	957	986	1015	1044
3	960	990	1020	1050	1080
4	992	1023	1054	1085	1116

#2	1	2	3	4
5	32	928	960	992
6	33	957	990	####
7	34	986	########	
8	35	##########		
9	36	##########		
10				

#4	5	6	7	8	9
5	1024	1056	1088	1120	1152
6	1056	1089	1122	1155	1188
7	1088	1122	1156	1190	1224
8	1120	1155	1190	1225	1260
9	1152	1188	1224	1260	1296
10					

例 5—20　三角函数表

① 功能：以步长 π／20 编制 0～π／2 的三角函数表。三角函数表有以下几项：

x　sinx　cosx　tgx　ctgx

② 操作：

首先，在第 1 行中输入标题："Triangular Table"。为了一次输入标题，把 R1C2、R1C3 和 R1C4 定义为连续格式（可延伸到下一格显示）。其过程如下：

FCR1C2：R1C4〈Tab〉〈Tab〉

再用功能键〈F10〉选择"Cont"格式，输入〈CR〉。这样，格式就定义好了。

→A〈CR〉　Triangular Table〈CR〉　{输入标题}

然后，在第 2 行的 C1～C5 中以对中方式输入小标题：

FCR2C1：R2C5

再用〈Tab〉键移到 alignment 字段，选择格式 Ctr 并输入〈CR〉，然后光标移到 R2C1 处再打入：

Ax→sinx→cosx→tgx→ctgx〈CR〉　{输入小标题}

接着，使用命令 NAME 把 C 2 和 C 3 分别命名为 sinx 和 cosx：

Nsinx〈Tab〉C 2〈CR〉Ncosx〈Tab〉C 3〈CR〉

以上各项准备好后，可以正式向三角函数表中输入内容：

GR3〈Tab〉1〈CR〉V0↓+PI( )/20+R[−1]〈CR〉C D9〈CR〉　{输入自变量}

#1	1	2	3	4	5
1			Triangular Table		
2	x	sinx	cosx	tgx	ctgx
3	0	0	1	0	#DIV/0!
4	0.1570796	0.1564345	0.9876883	0.1583844	6.3137515
5	0.3141593	0.309017	0.9510565	0.3249197	3.0776835
6	0.4712389	0.4539905	0.8910065	0.5095254	1.9626105
7	0.6283185	0.5877853	0.809017	0.7265425	1.3763819
8	0.7853982	0.7071068	0.7071068	1	1
9	0.9424778	0.809017	0.5877853	1.3763819	0.7265425
10	1.0995574	0.8910065	0.4539905	1.9626105	0.5095254
11	1.2566371	0.9510565	0.309017	3.0776835	0.3249197
12	1.4137167	0.9876883	0.1564345	6.3137515	0.1583844
13	1.5707963	1	0	#NUM!	0

```
→↑Vsin(C1)〈CR〉CD10〈CR〉        {填入 sinx}
→Vcos(C1)〈CR〉CD10〈CR〉         {填入 cosx}
→Vtan(C1)〈CR〉CD10〈CR〉         {填入 tgx}
→Vcosx/sinx〈CR〉CD10〈CR〉       {填入 ctgx}
```

屏幕上显示出的三角函数表如上页末所示。其中，“#NUM!”和“#DIV/0!”都是出错告警信号，分别表示数值溢出和除数为0。

最后，用“FC”命令把R3C1：R13C5中的单元格式定义为6位小数的定点格式（Fix-format code），并用打印命令输出6位三角函数表。结果如下：

Triangular Table

x	sinx	cosx	tgx	ctgx
0.000000	0.000000	1.000000	0.000000	#DIV/0!
0.157080	0.156434	0.987688	0.158384	6.313752
0.314159	0.309017	0.951057	0.324920	3.077684
0.471239	0.453990	0.891007	0.509525	1.962611
0.628319	0.587785	0.809017	0.726543	1.376382
0.785398	0.707107	0.707107	1.000000	1.000000
0.942478	0.809017	0.587785	1.376382	0.726543
1.099557	0.891007	0.453990	1.962611	0.509525
1.256637	0.951057	0.309017	3.077684	0.324920
1.413717	0.987688	0.156434	6.313752	0.158384
1.570796	1.000000	0.000000	#NUM!	0.000000

例5—21　计算职工的工资

① 功能：设屏幕上已有下列职员表：

#1

	1	2	3	4	5
1	Dept	Name	Salary		
2	Cau-a	Adept	103.8		
3	Qen-b	Bdept	74.3		
4	Shen-c	Adept	58.7		
5	Li-d	Bdept	80.76		
6	Zhou-e	Cdept	60		
7	Wang-f	Cdept	73.4		
8	Fong-g	Adept	68.9		
9					
10					

要求计算各部门工资的小计以及全体职工工资的总计。

② 操作：首先用SORT命令对表格按部门进行排序。然后在各部门分界处插入小计行，最后求出总计。操作的结果如下：

Salary Table

Dept	Name	Salary	Total
Cau-a	Adept	$103.80	
Fong-g	Adept	$68.90	
Shen-c	Adept	$58.70	
		* Adept *	$231.40
Li-d	Bdept	$80.76	
Qen-b	Bdept	$74.30	
		* Bdept *	$155.06
Wang-f	Cdept	$73.40	
Zhou-e	Cdept	$60.00	
		* Cdept *	$133.40
		* * Total * *	$519.86

例 5—22 计算利润

① 功能：设净利润是总利润减去 10% 净利润税，求净利润。

② 操作：

在 R1C1～R1C4 中填入下列数据和公式，在输入以前首先执行 Options 命令，选择迭代状态并指明测试迭代单元为R1C1。

```
R1C1：ITERCNT( ) =2
R1C2：$1000.00      {总利润}
R1C3：$100.00       {税 C[-1]*10%}
R1C4：$900.00       {净利润 C[-2]-C[-1]}
```

接着把税（R1C3）改为净利润的 10%（C[+1]*10%），这样 R1C3 和 R1C4 构成迭代式。经过两次迭代计算，直至它们满足 R1C1 中的停止条件为止。计算后的结果如下：

```
R1C1：TRUE
R1C2：$1000.00
R1C3：$90.90
R1C4：$909.10
```

第七节 SUPERCALC 简介

这里介绍的 SUPERCALC 是 SORCIM 公司 1982 年发行的 SUPERCALC 1.10 版。它也要在 MS-DOS 的支持下运行。

与 VISICALC 相比，SUPERCALC 具有以下几个特点：

* 表格可以用矩形区域来引用。
* 可以显示表格单元中的公式。

* 有联机求助功能。
* 允许对表格区域加保护。
* 可定义不同宽度的列。

下面仅对 SUPERCALC 的操作方法作一简单说明，并扼要地列出 SUPERCALC 的函数和命令，以供查阅。

1. 操　作

欲启动 SUPERCALC，可以打入：

A > SC

这时屏幕上显示 SUPERCALC 的版本信息和下列提示信息：

Enter "？" for HELP or "return" to start.

打入〈CR〉键后，屏幕上显示出类似图 5—20 的初态画面。

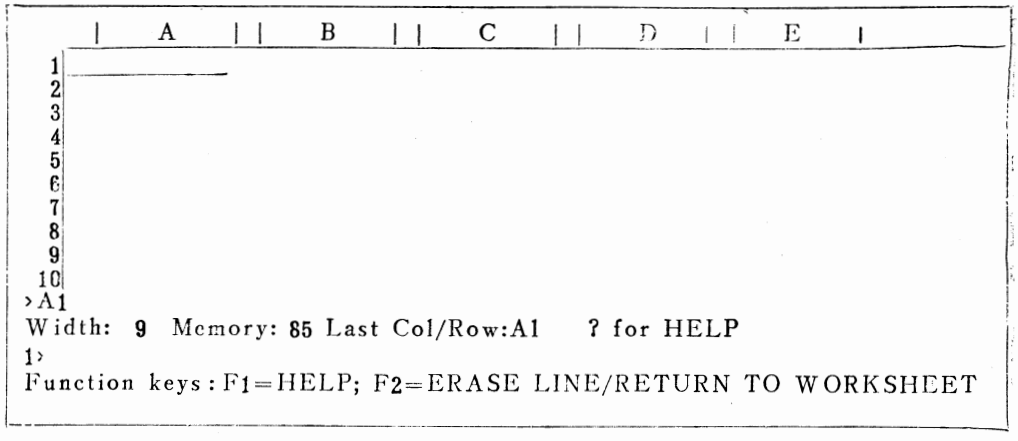


图 5—23　SUPERCALC 的初态画面

初态画面中的最末四行为 SUPERCALC 的状态区。其中第 1 行为当前单元行，当前单元坐标前的"＞"表示光标移动方向。第 2 行为提示行，其中"width：n"表示当前单元的宽度为n；"Memory：n"表示可用的内存（K字节数）；"Last Col/Row：××"表示表格右下角的坐标；"?"等效于〈F1〉键（求助键）。第 3 行为编辑行，其中"n＞"表示编辑光标处在编辑行中的位置。第 4 行用来说明〈F1〉为求助键；〈F2〉为删行键。

SUPERCALC 启动之后，就可输入数据或公式了。在输入的第 1 个字符中，下列字符有特殊的含义：

" 字符串引导符。
/ 命令引导符。
= 参数定位引导符（类似 VISICAL的"＞"）。
! 手动重计算命令符。

；　把当前单元指针从一个窗口移到另一个窗口。

，　填充字符串直至非空单元为止（类似 VISICALC的/－命令）。

在输入和编辑数据或公式时，可使用下列编辑键：

〈Esc〉　复制当前单元的坐标。

〈Ins〉　插入字符。

〈Del〉　删除字符。

〈F2〉　删行。

SUPERCALC 能处理的数据范围：字符串为 110 个字符；数值量为 12 位有效数字，范围约为 $-10^{62}\sim10^{62}$。

在输入公式时，常要引用表格区域。SUPERCALC 的表格区域主要有以下五种形式：

* 一个单元，例如：A15，J10 等。
* 一行，由行号表示，例如：1，254 等。
* 一列，由列号表示，例如：B，BK 等。
* 矩形区域，由左上角坐标和右下角坐标表示，例如：A1：C4 等。
* 用户定义的整个表格，用“ALL”表示。

2．函数与命令

（1）SUPERCALC 的函数

① 商用和统计函数

* SUM(L)　求和函数
* MIN(L)　最小值函数
* MAX(L)　最大值函数
* COUNT(L)　计数函数
* AVERAGE(L)　平均值函数
* NPV(V，L)　现得净利函数

② 算术函数

* ABS(V)　绝对值函数
* INT(V)　取整函数
* SQRT(V)　平方根函数
* EXP(V)　指数函数
* LOG10(V)，LN(V)　对数函数
* SIN(V)，ASIN(V)，COS(V)，ACOS(V)，TAN(V)，ATAN(V)　三角函数

③ 逻辑函数

* AND（V1，V2）　与函数
* OR（V1，V2）　或函数
* NOT（V）　非函数
* IF（V1，V2，V3）　条件函数

④ 辅助函数

* LOOKUP (V , L)　　查找函数
* ERROR , NA　　模拟出错函数

(2) 命令

① 填空白命令/Blank

② 对等复制命令/Copy

一对一地复制一个表格区域。

③ 删除命令/Delete

删除表格的行、列或删除文件。

④ 编辑命令/Edit

⑤ 格式命令/Format

该命令用来定义表格的格式，其子命令有以下几条：

* Entry　定义单元或矩形区域的格式。
* Row　定义指定行的格式。
* Column　定义指定列的格式。
* Global　定义全局格式。

这几条子命令可定义的格式如表 5-5 所示。

表 5-5　表格单元的格式类型

格　式	含　义	格　式	含　义	格　式	含　义
0-125	定义列宽	I	整型格式	D	缺省格式
E	指数格式	$	货币格式	R,L	数值右、左靠
G	普通格式	*	"*" 串格式	TR,TL	字符串右、左靠

⑥ 全局命令/Global

这条命令可定义表格的全局格式，它有以下几条子命令：

* Row-wise　按列计算。
* Column-wise　按行计算。
* Automatic-Recaculation　自动重计算。
* Manual Recaculation　手动重计算。
* Tab　能/不能选择空单元和加保护单元。
* Borders　显示/不显示行列号。
* Formula Display　显示/不显示单元中的公式。
* Next　在输入后，移动/不移光标。

⑦ 插入命令/Insert

⑧ 装入命令/Load

⑨ 移动命令/Move

⑩ 输出命令/Output

⑪ 保护命令/Protect

设置表格的保护区域。

⑫ 退出命令/Quit

⑬ 复制命令/Replicate

一对多复制，可把一个单元复制到多个单元，也可把一行(或列)复制成一个区域。

⑭ 保存命令/Save

该命令用来保存内存中的表格，它有以下三种参数：

* All　　保存整个表格（类型名为 .CAL）。
* Value　　仅保存各单元的值。
* Part　　保存部分表格。

⑮ 标题命令/Title

⑯ 去保护命令/Unprotect

该命令用来撤消用 /P 命令设置的保护区域。

⑰ 窗口命令/Window

⑱ 清除命令/Zap

这条命令用来清除整个表格。

⑲ 接收命令/Xexecute

可用来接收从文件中输入的命令和数据。

第六章　数据库管理系统dBASE Ⅱ及其应用

早期的电子计算机主要应用于数值计算领域。在数值计算中，数据的类型和结构都比较简单，程序设计的重点主要放在算法的表达方面。随着计算机技术的发展，计算机应用领域不断扩大，计算机在数据处理领域得到了广泛的应用。数据处理问题的特点是数据量很大，数据类型多，结构复杂，对数据的贮存、检索、分类、统计等处理要求较高。

为了满足数据处理的要求，把数据从附属于程序的做法改变为数据与程序相互独立，对数据加以组织和管理，使之能为许多不同的程序所共享，这就是“数据库系统”的基本出发点。

数据库系统由四个部分所组成：即经过组织的、可供多方面使用的数据集合——数据库；支撑数据贮存和数据操作的计算机系统；介于数据库和应用程序之间的数据库管理系统；提供给用户使用的各种数据库应用程序。

本章介绍在IBM PC机上使用的一个小型关系式数据库管理系统dBASE Ⅰ。通过介绍，使读者对数据库管理系统的功能及数据库应用程序的开发有一个概括的了解。

第一节　概　　述

dBASE Ⅰ是一个在多种微型计算机上运行的、受到用户普遍欢迎的小型关系式数据库管理系统，在国外享有“大众数据库”之称。它的前身是用FORTRAN语言为大型机编写的，后来被移植到微型机上，用Intel 8080汇编语言进行了改写，经过不断的改进，才演变成目前的版本。

1. dBASE Ⅰ的启动

在MS-DOS操作系统下运行的dBASE Ⅰ 2.4版共有三个文件组成：

DBASE·COM	常驻内存的公用模块。
DBASEOVR·COM	可覆盖模块，需要时才被调入内存。
DBASEMSG·TXT	求援说明，用来联机地提供给用户一些操作信息。

把包含上述文件的盘片插入驱动器中，在键盘上输入命令：

A>DBASE<CR>

屏幕上将显示出：

```
Copyright (C) 1982 RSP Inc.
*** dBASE II/86 Ver 2.4                    1 July 1983
.
```

其中第1行和第2行是dBASE Ⅰ的签到信息，第3行的“·”是dBASE Ⅰ的提示符，它

表示此时 IBM PC 已处于 dBASE Ⅰ 的控制之下，用户可以用 dBASE Ⅰ 提供的命令语句进行各种操作，或者执行由 dBASE Ⅰ 命令语句所编制成的数据库应用程序。

在向 dBASE Ⅰ 输入语句时，如果有语法错误，dBASE Ⅰ 会立即指出，并询问用户是否要改正。若用户作了改正，则语句就立即执行；若用户回答不进行改正，则表示语句作废。例如：

```
·LIST FILES ON b: *.bas
* * *  SYNTAX ERROR  * * *
                ?
LIST FILES ON b:*.bas
CORRECT AND RETRY (Y/N)? Y
CHANGE FROM:*
CHANGE TO :LIKE *
LIST FILES ON b: LIKE *.bas
MORE CORRECTIONS (Y/N)? N
:
```

其中，输入的语句的含义是列出B盘上的所有BASIC程序文件，但语句中缺少关键字LIKE，经过改正后即可正常执行。

用户在操作过程中如果忘记了某些语句的语法或它的功能，可以使用 HELP 语句来取得系统的帮助。例如，当需要了解 IF 语句时，可以打入 HELP IF 语句，系统将会显示出：

```
· help if
› IF -- in command file, IF structure permits conditional execution of commands. ELSE clause
        is optional.
 Syntax: IF <exp>                          Example: IF STATE='CA'
            <any statements>                            DO INSTATE (CMD file)
         [ELSE                                       ELSE
            <any statements>]                           DO OUTSTATE (CMD file)
         ENDIF                                       ENDIF
```

其中前两行扼要地介绍了 IF 语句的功能，然后给出了 IF 语句的语法以及使用的例子。

2. 数据库文件

数据库管理系统 dBASE Ⅰ 的主要处理对象是以表格形式组织起来的数据，它们以文件的形式存放在数据库中，即所谓“数据库文件”。一个数据库文件相当于日常生活中的一张表格，以下面的职工情况一览表为例（表6-1），可以看出，一张表格至少包含下列三个成分：

* 表的名称　用来区分各种不同的表。
* 表的格式　指出共有多少栏、每栏的标题及宽度等。

表 6-1　　职工情况一览表

	姓　　名	年　龄	工　　资	部　　门	课题负责人
1	赵　a	29	86.50	A	周　e
2	钱　b	42	59.40	B	李　d
3	孙　c	22	53.40		
4	李　d	30	100.76	B	王　f
5	周　e	35	96.40	A	王　f
6	王　f	50	160.30	C	

* 表的内容　即表中所填入的数据，也叫做每一栏的值。

因此，dBASE Ⅱ 的每一个数据库文件也至少包含有三个不可缺少的部分：文件名、文件结构以及文件的内容。

文件名与表名相对应。文件结构指的是表的格式。在计算机内部，表的格式只能以描述的形式而不能以直观的形式给出。例如，表 6-1 的结构描述可以给出如下：

栏序号	栏标题	栏宽度
1	姓　　名	10
2	年　　龄	2
3	工　　资	6
4	部　　门	6
5	课题负责人	10

在 dBASE Ⅱ 中，表格的每一栏叫做一个“字段”（FLD），栏标题叫做“字段名”（NAME），栏宽度叫做“字段宽度”（WIDTH）。为了数据处理的需要，dBASE Ⅱ 还在结构描述中增加两个内容：字段的值的类型（字符串型 C、数值型 N、逻辑型 L），以及数值型字段时小数部分的位数（DEC）。下面是表 6-1 所对应的数据库文件的结构描述：

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	010	
002	AGE	N	002	
003	SALA	N	006	002
004	DNAME	C	006	
005	MANAG	C	010	

文件内容就是表格中所填入的数据。表格是以“行”为单位的，每一行是一个整体，dBASE Ⅱ 把它称为一个“记录”（RECORD）。所以文件内容实际上就是一组记录，每个记录都有一个编号（记录序号或记录号）表示它在表中的位置。表 6-1 作为一个数据库文件时，其文件的内容为：

00001	Cau-a	29	86.50	Adept	Zhou-e
00002	Qen-b	42	59.40	Bdept	Li-d

00003	Shen-c	22	53.40		
00004	Li-d	30	100.76	Bdept	Wang-f
00005	Zhou-e	35	96.40	Adept	Wang-f
00006	Wang-f	50	160.30	Cdept	

不难理解，在一个数据库文件中，文件内容仅仅是数据本身，而它的结构才是数据的定义和解释。

3. dBASE Ⅰ的功能

作为一个小型的数据库管理系统，dBASE Ⅰ向用户提供下列六个方面的功能：

（1）数据的定义

dBASE Ⅰ允许用户在创建一个数据库文件时定义它的文件结构，也允许对已有的数据库文件的结构进行修改，需要时还能生成数据库文件的结构描述文件，从而为数据的定义带来很大的灵活性。

（2）数据的输入和更新

向数据库文件输入数据是以记录作为单位的。dBASE Ⅰ提供了一组语句用来向数据库文件添加记录、插入记录、删除记录、替换记录中某些字段的内容等等。同时，dBASE Ⅰ还允许数据库文件与正文文件相互进行转换，这样，就为dBASE Ⅰ应用程序与其它高级语言程序相互进行通讯创造了条件。

（3）数据的操作

dBASE Ⅰ允许用户对数据库文件中的数据进行各种常用的操作，例如，检索、排序、统计、求和等等。操作比较方便、灵活。

（4）数据的输出

数据操作的结果，可以在屏幕上显示出来，也可以作为新的文件存贮在数据库中，或者通过报表的形式在打印机上打印输出。

（5）应用程序的开发

dBASE Ⅰ向用户提供了一组控制语句，它能把上述各种用于数据处理的语句组织起来构成一个应用程序，满足用户的各种特定需要。因此，dBASE Ⅰ是一个自含式的数据库管理系统，它可以独立于其它高级语言而开发出各种应用程序。

（6）其它辅助功能

例如文件管理功能、存贮变量的操作与处理、设备控制等等。

4. 命令语句

dBASE Ⅰ向用户提供了一组内容丰富的命令语句（以下简称语句），它的全部功能均体现在这些语句上。与BASIC语言有点类似，这些语句都是由dBASE Ⅰ解释执行的，它们既可会话式地立即执行，也可以组织成一个程序（叫做dBASE Ⅰ应用程序）连续地执行。

dBASE Ⅱ 的语句接近于英语，它以动词作为语句的开始，表示该语句的操作类别，宾语就是它的操作对象（有时可以缺省），再加上一些短语作为其修饰或补充。例如：

DELETE FILE〈文件名〉 {删除所指出的文件}

DELETE〔〈范围〉〕〔FOR〈表达式〉〕{删除指定范围中所有满足给定条件的记录}

DISPLAY〔〈范围〉〕〔〈字段名表〉〕〔FOR〈表达式〉〕〔OFF〕 {显示输出指定范围中满足条件的所有记录，但仅限于指定的那些字段的内容}

应当注意，相同的动词常常由于宾语不同而构成不同的语句，它们的功能有时差别很大。语句中用〔 〕标出的部分表示是句子中的任选成分，缺省时的含义各语句都有规定。

dBASE Ⅱ 语句中常用的几个成分简单解释如下：

* 〈范围〉 用以指出该语句的有效作用范围，它或者缺省，或者可以取下面三种值之一：

ALL 表示对文件中所有记录都要进行规定的操作。

NEXT n 表示对文件中从当前记录开始的 n 个记录都进行规定的操作。

RECORD n 表示只对文件中第 n 个记录进行操作。

* FOR〈表达式〉 这是一个条件子句，其中的表达式应为逻辑表达式。当语句中使用FOR子句时，表示应对所有满足规定条件（即，使得表达式取值为“真”）的记录进行该语句所指出的操作。

* WHILE〈表达式〉 这也是一个条件子句，用于控制 DO 循环的执行，条件不满足时，DO 循环结束。

* FROM〈文件名〉 表示该语句操作时所需要使用的数据来自指定的文件。

* TO〈文件名〉 表示该语句操作过程中所产生的数据送到所指出的文件中去。

其它一些子句因为不常用，这里暂不作说明，以后出现时再作解释。

语句中子句出现的次序可以任意。语句的最大长度为 254 个字符，一个较长的语句需要占用多行时，每行以“；”和〈CR〉结束。为了减短语句的长度，语句中的任何关键词均只需要打入起始的四个字母，dBASE Ⅱ 就能正确识别。

大部分语句的操作是针对一个数据库文件进行的，该文件称为“当前文件”或“正在使用中的文件”，对它进行操作之前，必须先用“使用语句”（USE〈文件名〉）把它打开。此后，语句中就不再指出。

5．表达式

与通常的程序设计语言一样，表达式也是构成 dBASE Ⅱ 语句的一个重要组成部分。dBASE Ⅱ 的表达式由下列五个成分组成：

* 运算符
* 常数
* 存贮器变量
* 数据库文件中的变量（即字段名）
* 函数

根据表达式取值的类型，它又可以分为算术表达式、逻辑表达式和字符串表达式三种。现将所使用的运算符以优先级为序，列于表 6-2。

表 6-2　dBASE Ⅰ 的运算符

算术运算符	逻辑运算符	字符串运算符
圆括号( ) 函数 单目+，－ *，/ +，－ 关系运算符(<，>，=，#，<=和>=)	.NOT. .AND. .OR.	圆括号 ( ) 函 数 关系运算符(<，>，=，#，<=和>=) $(子字符串运算符) +，－(并置)

特别要指出的是子字符串运算符 $ 和并置运算符+，－。A$B 仅当 A 为 B 的子字符串时才为真，否则为假。并置运算符“－”与“+”的区别是“－”要压缩尾部的空格符。例如：

```
. ? 'ABCD ' +'EDGH'
ABCD EDGH
. ? 'ABCD ' -'EDGH'
ABCDEDGH
```

这里的“？”是输出语句的关键字，这条语句用以计算表达式的值并输出结果，输出时先回车换行。另一条输出语句“？？”类同于“？”，但输出时不回车换行。

dBASE Ⅰ 表达式中的运算对象有四种，记录中的字段值、常数、已定义的存贮变量以及函数。其中存贮变量在使用前，一般须用 STORE 语句赋值。STORE 语句的格式为：

STORE 〈表达式〉 TO 〈存贮变量名〉

其中存贮变量名由用户自行定义，同时可使用的存贮变量不能超过 64 个。使用 STORE 语句赋了值的存贮变量可用 LIST MEMORY语句列出。例如：

```
. STORE 123.467 TO X
 123.467
. STORE "example of string"  TO Y
example of string
. LIST MEMORY

X              (N)       123.467
Y              (C)    example of string
** TOTAL **        02 VARIABLES USED    00025 BYTES USED
```

存贮变量取值的另一种方法是使用输入语句，dBASE Ⅰ 的输入语句的格式为：

INPUT 〔“〈提示〉”〕 TO 〈存贮变量〉

ACCEPT 〔“〈提示〉”〕 TO 〈存贮变量〉

其中 INPUT 语句用来输入数值、逻辑值和字符串。而 ACCEPT 则仅用来输入字符串，对于输入的字符串可用单引号、双引号和方括号括起来，也可直接输入。例如：

```
. ACCEPT "ENTER PERSONS NAME" TO NAM
ENTER PERSONS NAME: Li-d
. INPUT TO X
: 3
  3
```

另外，还可使用存贮变量文件. MEM 保存内存中的存贮变量，以便使可用的存贮变量超过 64 个。存贮变量文件的写和读语句为：

SAVE TO 〈文件名〉

RESTORE FROM 〈文件名〉

其中 SAVE 为写语句，RESTORE 为读语句。例如：

```
. LIST MEMO
X               (N)       3
Y               (C)    example of string
NAM             (C)    Li-d
** TOTAL **          03 VARIABLES USED 00030 BYTES USED
. SAVE TO memfile
. RELEASE ALL
. LIST MEMO
** TOTAL **          00 VARIABLES USED 00000 BYTES USED
. RESTORE FROM memfile
. LIST MEMO
X               (N)       3
Y               (C)    example of string
NAM             (C)    Li-d
** TOTAL **          03 VARIABLES USED 00030 BYTES USED
```

其中“释放语句”（RELEASE ALL）表示系统收回已使用的全部存贮变量。

下面再简单地介绍 dBASE Ⅱ的函数，以供使用时查阅。

① “求当前记录序号”函数 #

例如：

```
. ? #
  4                {表示当前记录的序号为 4}
. SKIP             {当前记录指针加 1}
. ? #
5
```

②“测试删除标记”函数 *

该函数用以测试当前记录是否已删除。如已删除，函数值为真，否则为假。例如：

```
. USE emp
. GOTO BOTTOM
. ? *
 F
. DELETE
0031 DELETION(S)
. ? *
.T.
```

其中 GOTO BOTTOM 表示定位到最后一个记录，DELETE 表示删除当前记录。

③“测试文件存在”函数 FILE( )

该函数的格式为：

FILE(“〈文件名〉”)

当测试的文件存在时，函数值为真，否则为假。例如：

```
. ? FILE( "emp" )
.T.
```

④“测试文件结束”函数 EOF

该函数可用于判定文件是否已处理完毕。例如：

```
. USE emp
. GOTO BOTTOM
. ? EOF
.F.
. SKIP
. ? EOF
.T.
```

其中 SKIP 表示跳过一个记录。

⑤“表达式类型”函数 TYPE( )

函数的格式为：

TYPE(〈表达式〉)

函数的结果视表达式为数值、逻辑值或字符串型，而分别为“N”，“L”或“C”。例如：

```
. STORE 123 TO X
  123
. ? TYPE(X)
N
. ? TYPE(DATE)
U
. ? TYPE(DATE())
```

```
C
.
```

其中，U 表示 DATE 是没有定义的表达式，因为日期函数是 DATE( )。

⑥“取整”函数 INT( )

例如：

```
. ? INT(123.487)
  123
```

⑦“字符串长度”函数 LEN( )

例如：

```
. USE emp
. STORE name TO S
  Cau-a
. ? LEN(S)
10
```

其中第 2 个语句把 name 字段的值送入 S，因为 name 的宽度为 10，所以函数 LEN(S) 的值也是10。

⑧“小写改大写”函数 !( )

例如：

```
. ? ! ("abc")
. ABC
```

⑨“截去尾部空格”函数 TRIM( )

例如：

```
. USE emp
. ? LEN(name)
  10
. STORE TRIM(name) TO S
Cau-a
. ? LEN(S)
  5
```

⑩“取子字符串”函数$( )

该函数的格式为：

$(〈字符串〉,〈起始位置〉,〈长度〉)

其作用是在给定的字符串中取出一个子字符串。

例如：

```
. STORE 3 TO m
  3
. STORE 3 TO n
  3
. ? $ ("abcdefghi", m, n)
cde
. ? $ ("abcdefghi", 6, 7)
fghi
```

⑪“查找子字符串”函数@(　　)

这个函数的格式为：

@(〈子字符串〉,〈字符串〉)

如果字符串中包含给定的子字符串，则返回其在字符串中的位置，否则返回 0。

例如：

```
. ? @( "abc" , "abcdefghi" )
  1
. ? @( "def" , "abcdefghi" )
  4
. ? @( "xyz" , "abcdefghi" )
  0
```

⑫“数值量转换成字符串”函数 STR(　　)

这个函数的格式为：

STR(〈算术表达式〉,〈宽度〉,〔〈小数位数〉〕)

其功能是把数值量转换成对应的字符串。例如：

```
. ? STR (123.487, 9, 3)
  123.487
```

⑬“字符串转换成数值量”函数 VAL(　　)

例如：

```
. ? VAL ( "123" )
  123
. ? VAL ( "123 abc" )
  123
. ? VAL ("-123.456")
  -123
```

VAL 仅能转换整型数，对于实型量，可用宏定义符“&”来转换。

例如：

```
. STORE "123.456" TO X
. ? 14+& X
137.456
```

⑭ “ASCII 码转换成字符” 函数 CHR ( )

例如：

```
. ? "abcd" +CHR ( 13 ) +CHR ( 10 ) + "----"
a b c d
- - - -
```

⑮ “字符转换成 ASC II 码” 函数 RANK ( )

该函数的格式为：

RANK (〈字符串〉)

其函数值为字符串中第 1 个字符的 ASCII 码。

例如：

```
. ? RANK ( "0123" )
 48
```

⑯ “取日期” 函数 DATE ( )

该函数用来显示当天的日期，这个日期由系统或 SET DATE TO 语句设置。

例如：

```
. SET DATE TO 5 30 84
. ? DATE ( )
05/30/84
. STORE DATE ( ) TO DATETMP
. ? DATETMP
05/30/84
```

6. 文件类型及文件管理

前面介绍的数据库文件是数据库中最基本的、也是数量最多的一种文件。除此之外，dBASE Ⅱ 在进行数据的管理和操作的过程中，还会使用或生成其它一些类型的文件，现简单介绍如下（括号中为文件的类型名）：

（1）数据库文件（·DBF）

这是数据库中最基本的文件，用户所需要的数据均预先存贮在数据库文件中，数据库主要就是由许许多多的数据库文件所组成。每个数据库文件包含一系列的记录，每个记录由若干字段组成，它们的格式都相同。记录中的字段个数、字段名、字段类型及字段的宽度等信息就是该数据库文件的结构。

（2）正文文件（·TXT）

这是 MS-DOS 可以用 TYPE 命令打印出来的正文文件，它全部由可打印的 ASCII 字符组成。dBASE Ⅰ可以通过一定的语句使 ·DBF 文件和 ·TXT 文件相互进行转换。因此，这种文件可以用作高级语言程序和 dBASE Ⅰ之间的接口。

（3）索引文件（.NDX）

这是用于进行快速数据操作的一个辅助文件，它由“索引语句”（INDEX）所生成，文件中只包含经过排序处理的字段名以及它们的指针。

（4）报表格式文件（·FRM）

这也是一个辅助文件，它用来指出 dBASE Ⅰ产生报表输出时所应该遵循的格式要求。·FRM 文件是在使用“报表语句”（REPORT）时生成的，用户可以通过某些语句（如 MODIFY COMMAND）对它进行修改。

（5）应用程序文件（·PRG）

应用程序文件也叫做“命令文件”，因为它由一连串的命令语句所组成，用户可以通过“执行语句”（DO）来启动该应用程序的执行。·PRG 文件可以使用 dBASE Ⅰ中的 MODIFY COMMAND 语句来创建和编辑，也可以通过 MS-DOS 的行编辑程序或文字处理程序来编辑生成。

（6）存贮变量文件（·MEM）

尽管 dBASE Ⅰ操作处理的主要对象是数据库文件，但为了某些操作的需要，有时还必须使用一些存贮单元（变量）来存放常数、计算结果、替换字符串等，但变量的数目不能超过 64 个。当实际需要使用更多的变量时，可以通过“保存语句”（SAVE）把现行变量存放到磁盘上，这就是“存贮变量文件”，然后用“释放语句”（RELEASE）释放已使用过的内存单元，这样，用户就可以定义和使用更多的变量了。需要时，“存贮变量文件”可以用“恢复语句”（RESTORE）恢复。

（7）格式文件（·FMT）

·FMT文件由“格式输出语句”（@）及注释语句（*）所组成。

类似于通常的磁盘操作系统，dBASE Ⅰ对于上述各种文件提供了一些简单的管理功能，如列出文件目录，文件重命名，删除文件等。对应的 dBASE Ⅰ语句介绍如下：

* 显示文件目录（LIST FILES或DISPLAY FILES）

语句格式：

LIST（或DISPLAY）FILES〔ON〈驱动器号〉〕〔LIKE〈文件引用名〉〕

使用 LIST FILES和 DISPLAY FILES 语句的差别在于：后者在显示了一屏信息之后会暂停，等待用户任意敲入一键后再继续显示。

* 文件重命名（RENAME）

语句格式：

RENAME〈老文件名〉TO〈新文件名〉

若新文件名中未给出文件的类型名时，则 dBASE Ⅰ将把·DBF 作为新文件的类型名。

* 删除文件（DELETE FILE）

语句格式：

DELETE FILE〈文件名〉

它的作用是在磁盘上删去指定的文件。

上述三个语句对MS-DOS下的所有磁盘文件也都是同样有效的。至于其它一些文件管理操作，如文件的建立、复制、打印输出等，dBASE Ⅰ都有其特定的处理方法，它们在下面各节再分别进行介绍。

第二节　数据库文件的创建

1. 文件结构的描述

用户使用dBASE Ⅰ的主要任务之一是建立数据库文件，数据库文件由文件的结构描述和文件的具体内容两部分组成。要创建一个数据库文件，必须对文件结构进行描述。

文件的结构需要通过“创建语句”（CREATE）来进行描述。在描述过程中，必须向dBASE Ⅱ提供以下信息：

* 数据库文件的文件名
* 每个字段的字段名
* 每个字段的类型
* 每个字段的宽度

数据库文件的文件名与MS-DOS一样，也由1～8个字符组成。文件的类型名则规定为·DBF，它由dBASE Ⅰ自动加在文件名后登录于文件目录中。因此，用语句QUIT退出dBASE Ⅰ返回操作系统后，使用DIR命令显示文件目录时，凡类型名为·DBF的文件，一般都是数据库文件。

结构描述中的字段名相当于表格中每栏的标题，它由不多于10个的字母、数字或冒号组成，但必须以字母开头。例如：

FIELD：NAME
A123456789
A：B：C：D
ABCD：

都是合法的字段名。但下面的字段名都是不合法的：

A23 45	（中间出现了空格符）
ABC,DEF	（不允许使用逗号）
：ABCD	（冒号不允许使用在字段名的开头）

字段的类型用来说明字段值的性质。在dBASE Ⅰ中，允许使用以下三种类型：

* 字符串类型（C）　它的值为不含有控制字符的ASCII字符串。
* 数值类型（N）　其值为整型数或实型数，对实型数还应指明其小数部分的位数。
* 逻辑类型（L）　其值只有“真”（用T，t，Y或y表示）和“假”（用F，

f，N或n表示）两个。

字段的宽度是指字段在结构中所占字节的个数。对于数值型字段，其宽度还应包括小数点和符号位（正数的符号位不必计算在内）。字段的排列顺序以结构描述中的字段序号表示。

2．数据库文件的直接创建

创建一个新的数据库文件，意味着向dBASE Ⅰ定义该文件的名字并描述该文件的结构，这可以使用CREATE语句来完成。这条语句有两种使用形式，一种是直接通过键盘输入来描述文件的结构，另一种是把指定的文件结构描述信息预先作为一个固定结构的文件存放在盘上，再用CREATE语句间接地对文件结构进行描述。

第1种形式的CREATE语句，其格式为：

CREATE〔〈文件名〉〕

如果CREATE后不打入文件名，进入CREATE状态后，dBASE Ⅰ会首先提问文件名。下面是使用CREATE语句描述职员文件结构的过程：

```
·CREATE
ENTER FILENAME：emp
ENTER RECORD STRUCTURE AS FOLLOWS：
  FIELD    NAME, TYPE, WIDTH, DECIMAL PLACES
  001      name, c, 10
  002      age, n, 2
  003      sala, n, 6, 2
  004      dname, c, 6
  005      manag, c10
BAD WIDTH FIELD
  005      manag, c, 10
  006      〈CR〉
INPUT DATA NOW? N
```

打入“CREATE”以后，dBASE Ⅰ首先要求用户输入数据库文件的文件名。输入后，屏幕上又显示出两行提示信息告诉用户：请按给定的次序输入文件的结构描述。接着就可以逐行输入文件的结构描述了。在本例中，字段序号为005的行，由于在字段宽度和类型的描述中没有按规定用逗号隔开，所以dBASE Ⅰ给出了出错信息，须重新输入一次。输入完毕后，打入〈CR〉键，dBASE Ⅰ会立即询问是否要向文件输入数据。如果回答是（Y），dBASE Ⅰ将立即出现输入数据的提示画面，用户就可以从键盘上输入该文件的数据。如果回答否（N），则返回dBASE Ⅰ的提示符状态。

在文件的结构描述完毕之后，随时都可用LIST STRUCTURE语句或用DISPLAY STRUCTURE语句显示文件的结构描述。但在使用该语句以前，必须先用USE语句指出进行操作的是哪一个文件。其过程为：

```
. USE emp
. LIST STRUCTURE
STRUCTURE FOR FILE:  A:EMP          .DBF
NUMBER OF RECORDS:   00000
DATE OF LAST UPDATE: 05/20/84
PRIMARY USE DATABASE
FLD        NAME          TYPE WIDTH   DEC
001        NAME          C    010
002        AGE           N    002
003        SALA          N    006     002
004        DNAME         C    006
005        MANAG         C    010
** TOTAL **              00035
```

这里的 LIST 若用 DISPLAY 代替，效果是一样的，所不同的仅在于 DISPLAY 一次只显示一屏，输入任何键后，继续显示下一屏。

显示出的清单中的最末行为记录的总长度，它比实际的字段宽度的总和多一个字节，这个字节是 dBASE Ⅰ内部使用的，用于标记该记录是否已被删除。一般把这个字节称为“删除标记”。

dBASE Ⅰ为当前文件划分了两个区，一个是主区（primary），另一个是辅区（secondary）。如不特别说明，使用的都是主区。例如，上面使用 LIST STRUTURE 列出的清单中，“PRIMARY USE DATABASE”就表示当前使用的区是主区。

为了方便用户，dBASE Ⅰ允许用户使用功能键〈F1〉——〈F10〉加快语句的输入速度。这十个功能键的预定义如表 6-3 所示。这些功能键也可由用户自行定义，定义所使用的语句为：

SET F〈n〉 TO 〈'字符串'〉

其中 n 为 1～10。例如：

```
. SET F1 TO "CREATE"
```

表示把〈F1〉键定义为“CREATE”。

表 6-3　dBASE Ⅰ的功能键

功能键	功能	功能键	功能
〈F1〉	HELP;	〈F6〉	LIST STATUS;
〈F2〉	DISP;	〈F7〉	LIST MEMO;
〈F3〉	LIST;	〈F8〉	CREATE;
〈F4〉	LIST FILES;	〈F9〉	APPEND;
〈F5〉	LIST STRU;	〈F10〉	EDIT *;

3．文件结构的修改

对于一个已经创建的数据库文件，如果要修改它的文件结构，可以使用语句：

MODIFY STRUCTURE

使用该语句时，dBASE Ⅰ会给出警告性的提示："MODIFY ERASE ALL DATA RECORD…PROCEED？（Y／N）"（修改结构描述将擦去文件的所有数据记录…继续进行吗？）。如果回答"是"（Y），屏幕上会出现供用户编辑的画面，用户可用编辑键对文件的结构描述进行全屏幕编辑。如果输入"否"（N），则返回 dBASE Ⅰ的提示符状态。

使用全屏幕编辑修改文件的结构时，其过程是先用光标移动键（表 6-4 所示）把光标移至需要修改处，再用 MODIFY 编辑键（表 6-5 所示）进行修改，修改完毕后，使用 ^ W 键退出。

表 6-4　光标移动键

光标移动键	功能
^E 或 ^A, ↑	光标上移一行（一个字段）
^X 或 ^F, ↓	光标下移一行（一个字段）
^S 或 ←	光标左移
^D 或 →	光标右移
^G	删除光标处的一个字符
^Y	删除光标以右部分
^V	进入/退出插入方式

表 6-5　MODIFY 编辑键

编辑键	功能
^T	删除光标所在的行
^N	在光标处插入一个空行
^C或‹PgUp›	画面向上滚动半屏
^R或‹PgDn›	画面向下滚动半屏
^W	把修改的结果记盘并返回
^Q	所有修改的结果作废并返回

作为例子可先把 emp 文件的结构复制到文件 empnew 中去。具体做法如下：

```
. USE emp
. COPY TO empnew STRUCTURE
. LIST STRU
STRUCTURE FOR FILE: A: EMP        . DBF
NUMBER OF RECORDS: 00006
DATE OF LAST UPDATE: 05/27/84
```

```
PRIMARY USE DATABASE
FLD         NAME          TYPE  WIDTH   DEC
001         NAME          C     010
002         AGE           N     002
003         SALA          N     006     002
004         DNAME         C     006
005         MANAG         C     010
**TOTAL**                       00035
```

其中第2条复制语句仅复制老文件的结构到新文件中。下面两条语句对 empnew 的结构进行修改。

```
. USE empnew
. MODIFY STRU
MODIFY ERASE ALL DATA RECORD ··· PROCEED? (Y/N)?
```

打入“Y”后，屏幕上出现的画面如下：

```
                NAME      TYP   LEN   DEC
FIELD 01    : NAME        C     010   000   :
FIELD 02    : AGE         N     002   000   :
FIELD 03    : SALA        N     006   002   :
FIELD 04    : DNAME       C     006   000   :
FIELD 05    : MANAG       C     010   000   :
FIELD 06    :                               :
FIELD 07    :                               :
FIELD 08    :                               :
FIELD 09    :                               :
FIELD 10    :                               :
FIELD 11    :                               :
FIELD 12    :                               :
    ⋮
```

于是可使用光标移动键和编辑键进行编辑，把光标移至03号字段的描述行，输入一个 Ctrl N 键，屏幕上的画面如下：

```
                NAME      TYP   LEN   DEC
FIELD 01    : NAME        C     010   000   :
FIELD 02    : AGE         N     002   000   :
FIELD 03    :                               :
FIELD 04    : SALA        N     006   002   :
FIELD 05    : DNAME       C     006   000   :
```

```
FIELD 06   : MANAG      C     010     000   :
FIELD 07   :                                :
FIELD 08   :                                :
FIELD 09   :                                :
FIELD 10   :                                :
FIELD 11   :                                :
FIELD 12   :                                :
    :
```

这时，可插入字段 SEX 的描述。用同样的方法，在 05 号字段处插入一个 FILL 字段的描述，再把 06 号字段的宽度增加两个字节。修改后的画面如下：

```
                NAME      TYP   LEN    DEC
FIELD 01   : NAME        C     010     000   :
FIELD 02   : AGE         N     002     000   :
FIELD 03   : SEX         C       1           :
FIELD 04   : SALA        N     006     002   :
FIELD 05   : FILL        C       2           :
FIELD 06   : DNAME       C     008     000   :
FIELD 07   : MANAG       C     010     000   :
FIELD 08   :                                 :
FIELD 09   :                                 :
FIELD 10   :                                 :
FIELD 11   :                                 :
FIELD 12   :                                 :
    :
```

输入 Ctrl-W 返回 dBASE Ⅱ 提示符状态后，列出的修改后的文件结构如下：

```
. LIST STRU
STRUCTURE FOR FILE:    A:EMPNEW .DBF
NUMBER OF RECORDS:     00000
DATE OF LAST UPDATE:   05/27/84
PRIMARY USE DATABASE
FLD          NAME          TYPE    WIDTH    DEC
001          NAME            C       010
002          AGE             N       002
003          SEX             C       001
004          SALA            N       006     002
005          FILL            C       002
006          DNAME           C       008
```

```
007          MANAG              C       010
**TOTAL**                               00049
```

如果既要修改文件结构，又要保留文件内容，则必须采取间接的办法。例如，若要把文件 emp 的结构修改成文件 empnew 的结构，又要求把原有的文件内容放入到 empnew 中去，并设 emp 中已有下列六个记录：

```
00001    Cau-a      29    86.50  Adept    Zhou-e
00002    Qen-b      42    59.40  Bdept    Li-d
00003    Shen-c     22    53.40
00004    Li-d       30   100.76  Bdept    Wang-f
00005    Zhou-e     35    96.40  Adept    Wang-f
00006    Wang-f     50   160.30  Cdept
```

修改的步骤为：

第 1 步，将emp 的结构复制到文件 empnew中去：

```
. USE  emp
. COPY  TO  empnew  STRU
```

第 2 步，对 empnew进行结构修改：

```
. USE  empnew
. MODIFY  STRUCTURE
```

第 3 步，把 emp 中的记录复制到 empnew 中去，通过添加语句（APPEND）可以实现。

APPEND 语句的格式为：

APPEND FROM〈文件名〉

这里的文件名是添加记录的来源。第 3 步的操作过程如下：

```
. LIST  STRU
STRUCTURE FOR FILE: A:EMPNEW .DBF
NUMBER OF RECORDS: 00000
DATE OF LAST UPDATE: 05/27/84
PRIMARY USE DATABASE
FLD     NAME      TYPE WIDTH  DEC
001     NAME      C    010
002     AGE       N    002
003     SEX       C    001
004     SALA      N    006    002
005     FILL      C    002
006     DNAME     C    008
007     MANAG     C    010
```

```
** TOTAL **    00040
. APPEND FROM emp
00006 RECORDS ADDED
. LIST
00001    Cai-a     29     86.50    Adept    Zhou-e
00002    Qen-b     42     59.40    Bdept    Li-d
00003    Shen-c    22     53.40
00004    Li-d      30    100.76    Bdept    Wang-f
00005    Zhou-e    35     96.40    Adept    Wang-f
00006    Wang-     50    160.30    Cdept
```

最后的 LIST 语句用来列出数据库文件 empnew 的所有记录。

APPEND FROM 语句在对记录进行编排复制时，仅要求字段的名和类型相同，而与字段的先后顺序以及宽度无关。当类型不同或宽度变窄发生溢出时，dBASE Ⅱ 会自动处理并给出提示信息。此外，APPEND FROM 语句还可带 FOR 子句，有条件地选取记录进行编排复制。例如：

```
. APPEND FROM emp FOR sala>100
```

表示仅选择工资高于 100 元的职员记录，复制到 empnew 中去。

对于文件 empnew，新增字段的值可用第三节将要介绍的编辑语句 EDIT 或替换语句 REPLACE 进行输入。

最后一步是把文件名 empnew 改成 emp，这需要执行下列两个语句：

```
. DELETE FILE emp
. RENAME empnew TO emp
```

4. 数据库文件的间接创建

通过前面的介绍可以发现，每个数据库文件的结构描述也都是一张表，表的格式完全相同，仅仅是表的行数和表中的内容不同而已。因为一张表就对应着一个数据库文件，因此任意一个数据库文件的结构描述都可以转换成为一个数据库文件，故可把它称为“结构描述文件”。为某个数据库文件生成它的结构描述文件可使用语句：

```
COPY TO 〈文件名〉 STRUCTURE EXTENDED
```

例如，使用语句

```
. USE emp
. COPY TO empstru STRU EXTE
```

之后，文件 empstru 的结构和内容如下：

```
. USE empstru
  LIST STRU
STRUCTURE FOR FILE: A:EMPSTRU .DBF
NUMBER OF RECORDS: 00005
```

```
DATE OF LAST UPDATE: 05/27/84
PRIMARY USE DATABASE
FLD    NAME       TYPE WIDTH    DEC
001    FIELD:NAME  C       010
002    FIELD:TYPE  C       001
003    FIELD:LEN   N       003
004    FIELD:DEC   N       003
** TOTAL **          00018
. LIST
00001    NAME      C    10    0
00002    AGE       N     2    0
00003    SALA      N     6    2
00004    DNAME     C     6    0
00005    MANAG     D    10    0
```

显然，任何结构描述文件的结构都是相同的。所以，如果想要建立一个新的结构描述文件，就不必更改文件的结构，仅需修改文件的内容。

有了结构描述文件，就可使用语句

CREATE〈文件名〉FROM〈结构描述文件名〉EXTENDED

来间接地创建一个数据库文件了。下面举例说明间接创建数据库文件的过程。

首先，删除 empstru 文件中的第 5 个记录（删除语句详见第三节）。记录值删除后，并未从文件中抹去，只是打上了删除标志“*”，表示是已删记录，将不再参加文件操作。随后，根据文件 empstru 间接地创建新文件 empa 的结构。最后，把老文件emp 中的记录按相应的字段编排后添加到新的文件 empa 中去。该例的操作过程及其结果如下：

```
. DELETE RECORD 5
00001 DELETION(S)
. LIST
00001    NAME      C    10    0
00002    AGE       N     2    0
00003    SALA      N     6    2
00004    DNAME     C     6    0
00005  *MANAG      C    10    0
. CREATE empa FROM empstru EXTENDED
. USE empa
. LIST STRU
STRUCTURE FOR FILE:     A:EMPA      .DBF
NUMBER OF RECORDS:      00000
DATE OF LAST UPDATE:    05/27/84
PRIMARY USE  DATABASE
FLD    NAME     TYPE    WIDTH    DEC
001    NAME      C        010
002    AGE       N        002
003    SALA      N        006      002
004    DNAME     C        006
** TOTAL **             00025
```

```
. APPEND FROM emp
00006 RECORDS ADDED
. LIST
00001    Cau-a      29    86.50    Adept
00002    Qen-b      42    59.40    Bdept
00003    Shen-c     22    53.40
00004    Li-d       30    100.76   Bdept
00005    Zhou-e     35    96.40    Adept
00006    Wang-f     50    160.30   Cdept
```

上面主要介绍了如何创建一个新的数据库文件，使用的主要语句有：

① 数据库文件创建语句

* CREATE〔〈文件名〉〕 直接创建数据库文件。
* CREATE〈文件名〉FROM〈结构描述文件名〉EXTENDED

间接地根据结构描述文件的内容创建一个文件。

② 显示文件的结构

* LIST STRUCTURE
* DISP STRUCTURE

③ 列目录

* LIST FILES 列出数据库文件的目录。

④ 修改文件的结构

* MODIFY STRUCTURE

修改当前文件的结构，修改时将破坏文件的全部内容。

⑤ 复制文件的结构描述

* COPY TO〈文件名〉STRUCTURE

建立一个与当前文件结构相同的新文件。

* COPY TO〈结构描述文件名〉STRUCTURE EXTENDED

建立一个描述当前文件结构的结构描述文件。

⑥ 向当前文件添加记录

* APPEND FROM〈文件名〉〔〈FOR 子句〉〕

从指定文件中，取出满足〈FOR 子句〉的记录，经过编排后添加到当前文件中去。如果〈FOR 子句〉缺省，则取当前文件的全部记录。

⑦ 选择当前使用的一个数据库文件

* USE〈文件名〉

dBASE Ⅱ的某些限制如下：

① 记录长度不允许超过 1000 个字节。

② 记录结构中，字段的个数不超过 32 个。

③ 字符串类型的字段宽度不超过 254 个字节，数值类型字段的精度不超过十位有效数字（数值的范围约为10^{-63}～10^{63}之间）。

④ 一个文件的内容不允许超过 65535 个记录。

第三节　数据库文件的数据输入与更新

数据库文件创建之后，就可以向数据库文件输入数据了。所谓输入数据，是指向数据库文件输入一个个记录，就象一行行地填表似的。

1．记录的添加

向数据库文件添加记录需要使用添加语句，其格式为：

APPEND

它的功能是把用户随后输入的记录依次添加到当前数据库文件的末尾上去。

第二节所介绍的 CREATE 语句，在直接创建数据库文件的过程中，当 dBASE Ⅱ 询问是否要输入数据时，如回答“是”（Y），dBASE Ⅱ 做的处理也完全类似于执行上述的 APPEND 语句。

APPEND 语句一执行，屏幕上就按字段的排列顺序显示出当前文件的结构。这时，用户就可以使用光标移动键定位并输入数据。在 APPEND 语句的执行过程中，可以使用的控制键如下：

Ctrl-W　把输入的记录添加到文件中去，并准备添加下一个记录。该键等价于光标在最末字段处所输入的〈CR〉键。

Ctrl-Q　当前输入的记录作废，返回 dBASE Ⅱ 的提示符状态。

〈CR〉　光标处于第 1 字段第 1 个字符位置时，用来返回 dBASE Ⅱ 提示符状态；处于最末字段时，用来把输入的记录添加到文件中去，并准备添加下一个记录。

例如，下列两条语句

```
. USE emp
. APPEND
```

执行之后，屏幕上出现的画面是：

```
RECORD # 00007
NAME      :          :
AGE       :   :
SALA      :       :
DNAME     :       :
MANAG     :          :
```

其中记录序号（RECORD # 00007）表示当前要添加的记录是文件的第 7 个记录。冒号框住的部分表示字段的宽度。如果键盘输入的值能够填满字段，光标会自动移到下一个字段。如果字段值短于字段描述的宽度，可用〈CR〉键把光标移到下一个字段。最末字段填满或输入〈CR〉键结束输入之后，就会又显示出添加下一记录的画面。若在第 1 个字段开始位置就输入〈CR〉键，则表示所有记录均添加完毕，并返回 dBASE Ⅱ 提示符状态。

向当前文件添加一个空白记录的语句为：

APPEND BLANK

这里的空白记录占据一个记录的位置并占有一个记录序号。空白记录中数值型、字符串型和逻辑型的字段的值分别为“0”、空和“.F.”。

2. 记录的置换

数据库文件中已有的记录，可用“置换语句”更换其中的数据。置换语句的格式为：

REPLACE〔〈范围〉〕〈字段名〉WITH〈表达式〉{，〈字段名〉WITH〈表达式〉}〔FOR〈表达式〉〕

其中范围的缺省值为当前记录。

使用置换语句修改记录值，可以避免在应用程序的执行过程中，介入用户的键盘输入。例如，文件 emp 中已有以下六个记录值：

```
. LIST
00001    Cau-a      29    86.50    Adept    Zhou-e
00002    Qen-b      42    59.40    Bdept    Li-d
00003    Shen-c     22    53.40
00004    Li-d       30   100.76    Bdept    Wang-f
00005    Zhou-e     35    96.40    Adept    Wang-f
00006    Wang-f     50   160.30    Cdept
```

执行下列语句：

```
. APPEND BLANK
. REPLACE name WITH "Fong-g", age WITH 45. Sala WITH 47.30, dname WITH "Cdept",
        manag WITH "Li-d"
```

将添加一个记录值，添加后的结果是：

```
. LIST
00001    Cau-a      29    86.50    Adept    Zhou-e
00002    Qen-b      42    59.40    Bdept    Li-d
00003    Shen-c     22    53.40
00004    Li-d       30   100.76    Bdept    Wang-f
00005    Zhou-e     35    96.40    Adept    Wang-f
00006    Wang-f     50   160.30    Cdept
00007    Fong-g     45    47.30    Cdept    Li-d
```

置换语句中的范围若为 ALL，则表示对整个文件进行置换。例如，执行语句

```
. USE emp
. REPLACE ALL age WITH age+1
```

之后，文件的内容如下：

```
. LIST
00001    Cau-a      31    86.50    Adept    Zhou-e
00002    Qen-b      43    59.40    Bdept    Li-d
```

00003	Shen-c	23	53.40		
00004	Li-d	31	100.76	Bdept	Wang-f
00005	Zhou-e	36	96.40	Adept	Wang-f
00006	Wang-f	51	160.30	Cdept	
00007	Fong-g	46	47.30	Cdept	Li-d

如果替代的范围指定为 RECORD n，则表示置换操作仅对记录序号为 n 的记录进行。而范围为 NEXT n，则表示对从当前记录开始的 n 个记录进行置换。

置换语句中还可选用“FOR子句”来给出置换的条件。例如，对职员文件 emp 中工资低于 100 元的职员每人增加 10%，实现过程如下：

```
· USE emp
· REPLACE ALL sala WITH sala * 1.1 FOR sala <100
```

文件的内容被修改为：

```
· LIST
00001   Cau-a    30   95.15   Adept   Zhou-e
00002   Qen-b    43   65.34   Bdept   Li-d
00003   Shen-c   23   58.74
00004   Li-d     31   100.76  Bdept   Wang-f
00005   Zhou-e   36   106.04  Adept   Wang-f
00006   Wang-f   51   160.30  Cdept
00007   Fong-g   46   52.03   Cdept   Li-d
```

3. 记录的定位与插入

用APPEND 语句仅能在文件的末尾添加一些记录，如果想要把某个记录插入到文件中间，则应使用“插入语句”。在插入之前，先使用“定位语句”把当前记录定位到要插入的位置处。

定位用定位语句 LOCATE 来完成，它的格式为：

LOCATE 〔〈范围〉〕〔FOR〈表达式〉〕

这条语句用来查找满足条件的第 1 个记录，此后若使用 CONTINUE 语句，则表示继续查找满足条件的下一个记录。对于每个查找到的记录，dBASE Ⅰ 给出以下信息：

RECORD：〈当前记录序号〉

如果文件中找不到满足条件的记录，将给出信息：“END OF FILE”（文件结束）。

例如：

```
· USE emp
· LOCATE RECORD 2
· DISPLAY
00002   Qen-b    43   65·34   Bdept   Li-d
```

下面是带条件的定位语句及 CONTINUE 语句的使用例子：

```
· LOCATE FOR Sala>100
RECORD: 00004
```

```
. DISPLAY name, sala
00004      Li-d                100.76
. CONTINUE
RECORD: 00005
. CONTINUE
RECORD: 00006
. CONTINUE
END OF FILE ENCOUNTERED
```

除了使用定位语句 LOCATE以外，还可用 GOTO 语句和 SKIP 语句对记录进行 定位。GOTO语句主要有以下三种形式：

* n 或GOTO RECORD n　把第 n 号记录定位为当前记录。
* GOTO TOP　把文件中的第 1 个记录作为当前记录。
* GOTO BOTTOM　把文件的最末一个记录作为当前记录。

SKIP 语句的格式为：

SKIP〈表达式〉

表示把当前记录的指针相对移动若干个记录。

当前记录定位完毕后，就可用插入语句插入新的记录了。插入语句具有四种格式：

* INSERT　表示在当前记录后面插入一个新记录，屏幕上显示出供 用户插入记录的输入画面。
* INSERT BEFORE　表示在当前记录之 前插入一个记录，屏幕上显示出供用户插入记录的输入画面。
* INSERT BLANK　表示在当前记录之后插入一个空白记录。
* INSERT BEFORE BLANK　表示在当前记录之前插入一个空白记录。

4．记录的删除与恢复

输入文件内容后，不可避免地要产生文件更新的问题。所谓更新是指添加新记录、删除已有记录和修改已有记录。添加新记录的方法已经介绍过，这里先介绍记录的删除和恢复。

“删除语句”DELETE 可用于删除一个或多个记录，它的格式为：

DELETE〔〈范围〉〕〔FOR〈表达式〉〕

其中〈范围〉与 LOCATE 语句中一样，它有三种选择：

* ALL　表示文件的全部记录。
* RECORD n　表示序号为 n 的记录。
* NEXT n　表示从当前记录算起的 n 个记录。

DELETE语句执行之后，dBASE Ⅱ 并未真正地把记录抹去，仅仅在记录的删除标记位置上打上了一个星号。所以被删除的记录需要时还可用“恢复 语 句”（RECALL）进 行恢复。RECALL 语句的格式为：

RECALL〔〈范围〉〕〔FOR〈表达式〉〕

下面是删除和恢复记录的一个例子：

```
. USE emp
. LIST
00001     Cau-a      30    95.15   Adept    Zhou-e
00002     Qen-b      43    65.34   Bdept    Li-d
00003     Shen-c     23    58.74
00004     Li-d       31   100.76   Bdept    Wang-f
00005     Zhou-e     36   106.04   Adept    Wang-f
00006     Wang-f     51   160.30   Cdept
00007     Fong-g     46    52.03   Cdept    Li-d
. DELETE FOR sala >100
00003 DELETION(S)
. LIST
00001     Cau-a      30    95.15   Adept    Zhou-e
00002     Qen-b      43    65.34   Bdept    Li-d
00003     Shen-c     23    58.74
00004    *Li-d       31   100.76   Bdept    Wang-f
00005    *Zhou-e     36   106.04   Adept    Wang-f
00006    *Wang-f     51   160.30   Cdept
00007     Fong-g     46    52.03   Cdept    Li-d
. RECALL FOR "a"$name
00001 RECALL(S)
. LIST
00001     Cau-a      30    95.15   Adept    Zhou-e
00002     Qen-b      43    65.34   Bdept    Li-d
00003     Shen-c     23    58.74
00004    *Li-d       31   100.76   Bdept    Wang-f
00005    *Zhou-e     36   106.04   Adept    Wang-f
00006     Wang-f     51   160.30   Cdept
00007     Fong-g     46    52.03   Cdept    Li-d
. RECALL ALL
00002 RECALL(S)
. LIST
00001     Cau-a      30    95.15   Adept    Zhou-e
00002     Qen-b      43    65.34   Bdept    Li-d
00003     Shen-c     23    58.74
00004     Li-d       31   100.76   Bdept    Wang-f
00005     Zhou-e     36   106.04   Adept    Wang-f
00006     Wang-f     51   160.30   Cdept
00007     Fong-g     46    52.03   Cdept    Li-d
```

被删除了的记录只有在执行了“整理文件内容”语句（PACK）之后，才真正从文件中被抹除，并把整个文件的记录序号重新进行调整。

5．记录的修改

在向数据库文件输入数据的过程中，除了要插入和删除记录外，有时还要对记录进行修改。修改记录可以使用前面介绍过的 REPLACE 语句，也可使用 EDIT 语句、CHANGE 语

句和 BROWSE 语句，下面逐条介绍。

编辑语句的格式为：

EDIT 〔〈记录序号〉〕

其中记录序号的缺省值为当前记录。输入 EDIT 语句之后，dBASE Ⅱ 把指定的记录按字段分行显示在屏幕上，以供用户修改。修改完毕后，可以使用表 6-6 所示的编辑键进行处理。

表 6-6 EDIT 编辑键

编辑键	功能
^C	把当前记录记入文件，准备修改下一记录
^R	把当前记录记入文件，准备修改上一记录
^W	把当前记录记入文件并退出 EDIT
^Q	撤消对当前记录的修改并退出 EDIT
^U	改变当前记录的删除标记，即把删除标记从“*”改为空格或从空格改为“*”

CHANGE 语句用来有选择地修改指定的字段值，它的格式为：

CHANGE〔〈范围〉〕FIELD 〈字段名表〉〔FOR〈表达式〉〕

其中〈字段名表〉为要修改的字段名的集合。这条语句输入后，dBASE Ⅱ会逐次提问要修改的字段，直至文件中没有满足条件的记录为止。

BROWSE 语句是一条全屏幕、全文件的编辑语句。它除了可以使用表 6-4 的光标移动键控制光标外，还可使用表 6-7 列出的编辑键进行操作。

表 6-7 BROWSE 编辑键

编辑键	功能
^W	保存修改的结果并退出BROWSE
^Q	撤消所作的修改并退出BROWSE
^B	画面左移一个字段
^Z	画面右移一个字段
^C	写入当前记录，当前记录指针加1
^R	写入当前记录，当前记录指针减1
^U	改变记录的删除标记

下面举一个使用 BROWSE 语句修改记录的例子。执行下列语句

```
. USE emp
. BROWSE
```

之后，屏幕上出现如下画面：

```
RECORD # : 00001
NAME ------ AG SALA-- DNAME--MANAG-----
Cau-a       30  95.15  Aedpt   Zhou-e
Qen-b       43  65.34  Bdept   Li-d
Shen-c      23  58.74  Cdept   Li-d
Li-d        31 100.76  Bdept   Wang-f
Zhou-e      36 106.04  Adept   Wang-f
Wang-f      51 160.30  Cdept
Fong-g      46  52.03  Cdept   Li-d
```

打入 ^B 后，屏幕上的数据左移一个字段，这时屏幕上的内容为：

```
RECORD # : 00001
AG SALA-- DNAME--MANAG-----
30  95.15  Adept   Zhou-e
43  65.34  Bdept   Li-d
23  58.74
31 100.76  Bdept   Wang-f
36 106.04  Adept   Wang-f
51 160.30  Cdept
46  52.03  Cdept   Li-d
```

然后可用全屏幕编辑键对整个文件进行编辑和修改。

BROWSE 语句是一条十分方便的编辑语句，它的缺点是没有插入记录的功能。补救的办法是先用 INSERT BLANK 语句插入空白记录，再用 BROWSE 语句填入记录的值。尽管如此，BROWSE 语句仍是一条值得推荐的编辑语句。

6．从正文文件向数据库文件输入数据

向数据库文件输入数据的另一种方法是从正文文件输入数据。这种方法常用于 dBASE Ⅰ与其它高级语言交换数据，即数据库管理系统可以使用高级语言建立的正文文件或为高级语言提供正文文件。

虽然 dBASE Ⅰ并没有直接提供与高级语言的接口，但只要适当地组织正文文件，就可以借助于它来沟通 dBASE Ⅰ和高级语言之间数据交换的渠道。

dBASE Ⅰ提供了较强的复制功能，可以把数据库文件中的数据复制到正文文件中去，也可以把正文文件复制到数据库文件中去。前者用 COPY 语句实现，后者则用 APPEND 语句实现。

有两种格式的 COPY 语句可以使用。第 1 种格式为：

COPY TO〈文件名〉SDF

其中 SDF 表示产生的文件是一个 ASCII 码文件（正文文件），它的类型名为 ·TXT。这个 ·TXT 文件的内容为当前数据库文件的内容，它的格式须严格地根据各字段的定义宽度对

齐，且每个记录以回车换行符结束。·TXT 文件可用操作系统的 TYPE 命令打印出来。例如：

```
. LIST
00001    Cau-a      30    95.15    Adept    Zhou-e
00002    Qen-b      43    65.34    Bdept    Li-d
00003    Shen-c     23    58.74
00004    Li-d       31   100.76    Bdept    Wang-f
00005    Zhou-e     36   106.04    Adept    Wang-f
00006    Wang-f     51   160.30    Cdept
00007    Fong-g     46    52.03    Cdept    Li-d
. COPY TO empdat SDF
00007 RECORDS COPIED
. QUIT
*** END RUN        dBASE II ***
A>TYPE empdat.txt
Cau-a      30  95.15Adept  Zhou-e
Qen-b      43  65.34Bdept  Li-d
Shen-c     23  58.74
Li-d       31 100.76Bdept  Wang-f
Zhou-e     36 106.04Adept  Wang-f
Wang-f     51 160.30Cdept
Fong-g     46  52.03Cdept  Li-d
```

COPY 语句的第 2 种格式为：

COPY TO〈文件名〉DELIMITED

所生成的 ASCII 码文件为紧凑格式，即不按字段宽度而用分隔符来区分记录中的各字段。对于字符串型字段，还要用单引号括起来。

例如：

```
. LIST
00001    Cau-a      30    95.15    Adept    Zhou-e
00002    Qen-b      43    65.34    Bdept    Li-d
00003    Shen-c     23    58.74
00004    Li-d       31   100.76    Bdept    Wang-f
00005    Zhou-e     36   106.04    Adept    Wang-f
00006    Wang-f     51   160.30    Cdept
00007    Fong-g     46    52.03    Cdept    Li-d
. COPY TO empdat1 DELIMITED
00007 RECORDS COPIED
. QUIT
*** END RUN        dBASE II        ***
```

```
A>TYPE empdat1.txt
'Cau-a', 30, 95.15, 'Adept', 'Zhou-e'
'Qen-b', 43, 65.34, 'Bdept', 'Li-d'
'Shen-c', 23, 58.74, ' ', '
'Li-d', 31, 100.76, 'Bdept', 'Wang-f'
'Zhou-e', 36, 106.04, 'Adept', 'Wang-f'
'Wang-f', 51, 160.30, 'Cdept', ' '
'Fong-g', 46, 52.03, 'Cdept', 'Li-d'
```

注意，用上述两条语句生成的文件是以物理记录（扇区）为单位分配盘空间的。即每个记录的长度总是 512 字节的倍数。

与 COPY 语句相对应，APPEND 语句也有两种格式：

APPEND FROM〈文件名〉SDF

APPEND FROM〈文件名〉DELIMITED

它们可以把相应的 .TXT 文件添加到数据库文件中去。APPEND 语句的使用方法如下例：

```
. USE emp
. COPY TO empt1 STRUCTURE
. USE empt1
  APPEND FROM empdat SDF
00007 RECORDS ADDED
. APPEND FROM empdat1 DELIMITED
00007 RECORDS ADDED
. LIST
00001   Cau-a      30   95.15   Adept   Zhou-e
00002   Qen-b      43   65.34   Bdept   Li-d
00003   Shen-c     23   58.74
00004   Li-d       31   100.76  Bdept   Wang-f
00005   Zhou-e     36   106.04  Adept   Wang-f
00006   Wang-f     51   160.30  Cdept
00007   Fong-g     46   52.03   Cdept   Li-d
00008   Cau-a      30   95.15   Adept   Zhou-e
00009   Qen-b      43   65.34   Bdept   Li-d
00010   Shen-c     23   58.74
00011   Li-d       31   100.76  Bdept   Wang-f
00012   Zhou-e     36   106.04  Adept   Wang-f
00013   Wang-f     51   160.30  Cdept
00014   Fong-g     46   52.03   Cdept   Li-d
```

那么，使用高级语言或行编辑程序建立的类似格式的正文文件，能否直接使用 APPEND 语句添加到数据库文件中去呢？一般地说这是有困难的。其原因是用这些方法建立的文件很难确保其长度为 512 个字节的倍数。不过，若把这种文件使用 dBASE Ⅱ 的语句

MODIFY COMMAND〈文件名〉

编辑一次（即使不作任何修改），文件的长度就会自动地调整为 512 个字节的倍数，然后再用 APPEND 语句便可获得成功。

遗憾的是经过 MODIFY COMMAND 语句编辑过的文件，其中内码大于 128 的字符都

已作了处理，这对 dBASE Ⅱ 的汉字化十分不利。为了解决上述问题，可用 BASIC 语言编制一个输入程序，专门为 dBASE Ⅱ 准备数据。例 6—1 就是这样的一种情况。

例 6—1　　dBASE Ⅱ 的数据输入程序

①功能：假设 IBM PC 的串行接口（RS232C）上接有一台汉字终端。程序把从汉字终端上输入的汉字编码建立成一个汉字正文文件，以提供给 dBASE Ⅱ 的数据库文件使用。这样在完全不改变 dBASE Ⅱ 的前提之下，实现了 dBASE Ⅱ 部分功能的汉字化。

②程序：

```
 10  PRINT "INPUT FILE NAME IS INF·DAT"
 30  OPEN "COM1：1200, N, 8, 1, CS, DS, CD, LF" AS #1
 50  OPEN "INF·DAT" FOR OUTPUT AS #2
 60  L=0
 70  PRINT #1, CHR$(13) ":" ;
 80  GOSUB 150
 90  GOTO 70
100  END
110  IF EOF(1) THEN 110
120  B$=INPUT$(LOC(1), #1)
130  K=LEN(B$)
140  RETURN
150  A$= ""
160  J=1
170  GOSUB 110
180  IF B$=CHR$(127) THEN 240
190  IF B$=CHR$(13) THEN 270
200  IF B$=CHR$(10) THEN 290
210  A$=A$+B$ : J=J+K : L=L+K
220  PRINT #1, B$;
230  GOTO 170
240  IF J=1 THEN 170
250  PRINT #1,CHR$(27) "d" ;:J=J-1:L=L-1:A$=LEFT$(A$, J-1)
260  GOTO 170
270  PRINT #2, A$ : L=L+2
280  RETURN
290  X=511-(L MOD 512)
300  FOR Y=1 TO X : PRINT #2, CHR$(32) ; : NEXT
310  CLOSE : END
```

③说明：30 行把接于串行口的汉字终端打开。50 行把数据文件打开。60—90 行输入数据并记盘。110—140 行子程序的作用是从汉字终端上输入一个字符、或一个汉字到字符

串变量B$中去，如果输入的是字符，则K为1，否则K为2。150—230行的子程序表示从汉字终端输入的ASCII码字符或汉字，若不为回车、换行或退格符的话，则把输入的字符串变量B$联结到A$中去。240—260行为退格符处理。270—280行为回车符处理，用来向数据文件中写入一行数据。290—310行为换行符处理，把文件以空格填足512字节的倍数，并关闭文件。

上面介绍了如何向数据库文件输入数据，以及如何对数据库文件进行更新（添加、删除和修改）。使用的主要语句有：

① 添加记录

- APPEND　直接通过键盘向数据库文件添加记录。
- APPEND BLANK　向数据库文件添加一个空白记录。
- APPEND FROM 〈数据库文件名〉〔FOR〈表达式〉〕 从指定的数据库文件中取记录值，按当前文件的结构进行排，添加到当前文件中。
- APPEND FROM〈正文文件名〉SDF | DELIMITED　从正文文件中取出记录添加到数据库文件中去。如选用SDF则表示每个记录按记录长度进行匹配，而选用DELIMITED则表示按分隔符“,”匹配字段值。

② 置换语句

- REPLACE〔〈范围〉〕〈字段名〉WITH〈表达式〉{,〈字段名〉WITH〈表达式〉}〔FOR〈表达式〉〕 置换满足条件的记录中的字段值。其中范围的含义如下：

ALL	所有满足〈FOR 子句〉条件的记录
RECORD n	序号为n的记录
NEXT n	从当前记录开始的n个记录
缺省	当前记录

③ 删除与恢复记录

- DELETE〔〈范围〉〕〔FOR〈表达式〉〕 对满足条件的记录标以删除标记。
- RECALL〔〈范围〉〕〔FOR〈表达式〉〕 对满足条件的记录取消删除标记。

④ 当前记录定位

- LOCATE〔〈范围〉〕〔FOR〈表达式〉〕 把满足条件的第1个记录作为当前记录。
- CONTINUE　把满足条件的下一个记录作为当前记录（须与LOCATE语句联用，并要求LOCATE和CONTINUE语句之间没有执行变更当前记录指针的命令）。
- GOTO RECORD n | TOP | BOTTOM　把指定的记录作为当前记录，其中TOP和BOTTOM分别表示把文件首记录和末记录作为当前记录。
- SKIP〈表达式〉　把当前记录的指针相对移动若干行。

⑤ 插入记录

* INSERT〔BEFORE〕 直接从键盘上插入一个记录。
* INSERT〔BEFORE〕BLANK 插入一个空白记录。

⑥ 记录的修改和编辑

* EDIT〈记录序号〉 全屏幕编辑一个记录。
* BROWSE 全屏幕编辑整个文件。
* CHANGE〔〈范围〉〕FIELD〈字段名表〉〔FOR〈表达式〉〕 有选择地修改一些字段的值。这里的字段名表可定义为：

〈字段名表〉::=〈字段名〉|〈字段名表〉,〈字段名〉

第四节 数据库文件的操作——检索、排序和统计

数据库建立之后，用户就可以使用了。用户使用数据库的主要形式是检索，即从数据库中取出满足用户要求的数据。检索出的结果，既可按需要显示出来，也可以存放于数据库中。后一种方法常用于需要多次检索同一批数据的情况。

用户除了需要检索数据外，还常常需要对检索出来的数据进行统计。下面介绍的是dBASE Ⅰ的检索和统计功能。

1. 检索操作

关系数据库的任何检索操作的实现，都是由三种基本检索运算组合而成的。这三种基本运算是选择（Select）、投影（Project）和联结（Join）。

选择的对象是一张表格。选择运算就是从已知的表格中选出满足条件的记录组成一张新的表格。例如，工资高于100元的职员表，实际上就是从职员表中选取满足条件——“工资高于100元”——的所有职员记录组成的一张新表。

投影则是从一张表格中抽取几列重新组成一张新表的运算。若需要一张仅由部门、姓名和工资三项组成的职员表，则对普通的职员表施行投影运算就可得到。

联结的对象需要涉及多张表格。对于两张表格而言，联结运算把表格1的记录和表格2的记录组合起来构成一张新的表格。例如表6-8中的（a）是职员工资表，（b）是职员通讯地址表，（c）是（a）与（b）联结的结果。

表6-8 表格的联结运算

(a)

姓 名	年龄	工 资	部 门
赵 a	29	86.50	A
钱 b	42	59.40	B
李 d	30	100.76	B
周 e	35	96.40	A
王 f	50	160.30	C

(b)

姓 名	性别	地 址	电 话
赵 a	女	ND8-201	1234
钱 b	男	NG1-312	
孙 c	男	ND2-434	3478
李 d	女	NH1-436	7780
王 f	女	NK1-123	4321

(c)

姓 名	年龄	工 资	部门	姓 名	性别	地 址	电 话
赵 a	29	86.50	A	赵 a	女	ND8-201	1234
赵 a	29	86.50	A	钱 b	男	NG1-312	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
赵 a	29	86.50	A	王 f	女	NK1-123	4321
钱 b	42	59.40	B	赵 a	女	ND8-201	1234
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
钱 b	42	59.40	B	王 f	女	NK1-123	4321
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
王 f	50	160.30	C	赵 a	女	ND8-201	1234
⋮	⋮	⋮	⋮	⋮	⋮	⋮	…
王 f	50	160.30	C	王 f	女	NK1-123	4321

(d)

姓 名	工 资	地 址	电 话
赵 a	86.50	ND8-201	1234
钱 b	59.40	NG1-312	

显然这样的联结结果意义并不大，实际的联结运算往往是联结、选择和投影的复合操作。表中（d）是工资在100元以下的职员通信地址表。它是从（a）和（b）中按照条件：

（表（a）·姓名＝表（b）·姓名）·AND·（表（a）·工资＜100）

取（a）中的姓名和工资两个字段，（b）中的地址和电话两个字段联结得到的结果。

dBASE Ⅰ用于检索的基本语句为：

LIST〔〈范围〉〕〔〈字段表〉〕〔FOR〈表达式〉〕〔OFF〕

或　　DISPLAY〔〈范围〉〕〔〈字段表〉〕〔FOR〈表达式〉〕〔OFF〕

其中范围是指搜索数据的范围，FOR子句是检索的条件，字段表列出的是要作投影的字段名，OFF则表示显示检索结果时不带记录序号。

LIST语句和DISPLAY语句的区别在于：DISPLAY语句是按屏显示，且在缺省FOR子句的情况下，缺省的“范围”为当前记录，而LIST语句是连续显示，且在缺省FOR子句时缺省“范围”为ALL。

为说明该语句的使用方法，下面举几个例子。

例6—2　选择emp文件的全部记录

```
. USE emp
. LIST
00001    Cau-a      30    95.15    Adept    Zhou-e
00002    Qen-b      43    65.34    Bdept    Li-d
00003    Shen-c     23    58.74
00004    Li-d       31   100.76    Bdept    Wang-f
00005    Zhou-e     36   116.04    Adept    Wang-f
```

```
00006      Wang-f      51    160.30    Cdept
00007      Fong-g      46     52.03    Cdept      Li-d
```

例 6—3　检索出工资高于 100 元的职员记录

```
. DISPLAY FOR sala>100
00004      Li-d        31    100.76    Bdept      Wang-f
00005      Zhou-e      36    106.04    Adept      Wang-f
00006      Wang-f      51    160.30    Cdept
```

例 6—4　检索出工资高于 100 元且属于 A 部门的职员记录

```
. DISPLAY FOR sala>100.AND.dname= "Adept"
00005      Zhou-e      36    106.04    Adept      Wang-f
. DISPLAY FOR sala>100.AND."A"$dname
00005      Zhou-e      36    106.04    Adept      Wang-f
```

说明：本例中的两条语句的效果相同。这里的"A"$dname 仅当字段 dname 的值包含字符 "A" 时才为真。

例 6—5　检索出属于A部门的所有记录，但只需列出姓名、年龄和工资三项。

```
. DISPLAY name,age,sala FOR dname='Adept'
00001      Cau-a       30    95.15
00005      Zhou-e      36    106.04
```

说明：这是一个投影加选择的例子。

例 6—6　从职员表的最后四个记录中，检索出课题负责人是 "Wang-f" 的职员的姓名和年龄。

```
. USE emp
. LIST
00001      Cau-a       30    95.15     Adept      Zhou-e
00002      Qen-b       43    65.34     Bdept      Li-d
00003      Shen-c      23    58.74
00004      Li-d        31    100.76    Bdept      Wang-f
00005      Zhou-e      36    106.04    Adept      Wang-f
00006      Wang-f      51    160.30    Cdept
00007      Fong-g      46    52.03     Cdept      Li-d
. GOTO BOTTOM
. SKIP -3
RECORD: 00004
. DISP NEXT 4 name, age FOR manag= "Wang-f"
00004      Li-d        31
00005      Zhou-e      36
```

如果希望把检索出的结果作为数据库文件保存，则必须使用 COPY 语句检索。COPY 语句的格式为：

COPY TO 〈文件名〉〔〈范围〉〕〔FIELD 〈字段表〉〕〔FOR 〈表达式〉〕

其中各项的含义同 LIST 语句。比如，把例 6—5 的结果保存于数据库的过程为：

```
. USE emp
. COPY TO emp1 FIELD name, age, sala FOR "A" $dname
00002 RECORDS COPIED
. USE emp1
. DISP STRU
STRUCTURE FOR FILE:     A:EMP1    .DBF
NUMBER OF RECORDS:      00002
DATE OF LAST UPDATE: 05/30/84
PRIMARY USE DATABASE
FLD        NAME        TYPE  WIDTH  DEC
001        NAME          C     010
002        AGE           N     002
003        SALA          N     006     002
** TOTAL **                     00019
. LIST
00001      Cau-a          30       95.15
00002      Zhou-e         36      106.04
. LIST OFF
Cau-a          30    95.15
Zhou-e         36   106.04
```

dBASE Ⅱ 的联结运算是通过 JOIN 语句实现的。它只能对两个文件进行联结，而三个以上的文件联结，须经过多次两两联结才能完成。在联结文件以前，必须把要联结的两个文件分别使用 SELECT 语句放入主区与辅区。SELECT 语句的格式为：

SELECT PRIMARY | SECONDARY

其中 PRIMARY 表示主区，SECONDARY 则表示辅区。再用联结语句 JOIN 进行联结。JOIN 语句的格式为：

JOIN TO〈文件名〉FOR〈表达式〉〔FIELD〈字段名表〉〕

必须注意，只有回到主区后才能顺利地进行联结。下面举两个例子说明联结文件的过程。

例 6—7　按条件联结 emp 和 empaddr

① 功能：设已存在数据库文件 emp 和 empaddr，它们的内容和结构如下：

```
. USE emp
. LIST
00001      Cau-a          30    93.80    Cdept     Zhou-e
00002      Qen-b          43    84.30    Adept     Li-d
00003      Shen-c         23    68.74    Adept     Wang-f
00004      Li-d           40    90.76    Bdept     Wang-f
00005      Zhou-e         36    70.00    Adept     Wang-f
00006      Wang-f         51   160.30    Cdept
00007      Fong-g         41    68.90    Cdept     Li-d
00008      Jang-h         43    68.90    Cdept     Zhang-k
00009      Qu-i           44    73.90    Cdept     Zhang-k
00010      Zhang-k        45    78.50    Cdept
00011      Hua-j          28    47.30    Cdept     Zhang-k
. LIST STRU
```

```
STRUCTURE FOR FILE:    A:EMP          .DBF
NUMBER OF RECORDS:     00011
DATE OF LAST UPDATE:   05/30/84
PRIMARY USE DATABASE
FLD        NAME        TYPE WIDTH    DEC
001        NAME        C    010
002        AGE         N    002
003        SALA        N    006      002
004        DNAME       C    006
005        MANAG       C    010
** TOTAL **            00035
. USE empaddr
. LIST
00001   Cau-a        f  A01   ND  8-201          34651-1234
00002   Qen-b        m  B02   NG  1-312          34663-023
00003   Shen-c       m  A03   ND  2-434          34655-3478
00004   Li-d         f  B05   NH  1-436          34651-7780
00005   Zhou-e       m  A10   NH  7-7-213        42315-1234
00006   Wang-f       f  C01   NK  1-4-123        43412-103
00007   Fong-g       m  C05   ND  7-345          34651-4423
00008   Jang-h       m  C10   BJ  16-3-201       34651-1234
00009   Qu-i         m  C15   Erdiaxian4        34655-4567
00010   Zhang-k      m  C50   BJXR13-301        34651-1234
00011   Hua-j        m  M12   ND  8-403          34655-403
. LIST STRU
STRUCTURE FOR FILE:    A: EMPADDR .DBF
NUMBER OF RECORDS:     00011
DATE OF LAST UPDATE:   05/30/84
PRIMARY USE DATABASE
FLD          NAME       TYPE  WIDTH    DEC
001          NAME       C     010
002          SEX        C     001
003          CODE       C     003
004          ADDR       C     010
005          TEL        C     010
** TOTAL **             00035
```

本例要求把文件 emp 和 empaddr 联结起来，生成一个新的数据库文件 empj1，联结的条件是姓名相同的记录，联结后的文件要求保留文件 emp 中的姓名、年龄和工资以及文件 empaddr 中的职员号和性别。

② 实现：

```
. USE emp
. SELECT SECONDARY
. USE empaddr
. SELECT PRIMARY
```

```
. JOIN TO empj1 FOR name=S.name FIELD S.code, name, age, S.sex, sala
. USE empj1
. LIST STRU
STRUCTURE FOR FILE:     A: EMPJ1        .DBF
NUMBER OF RECORDS:      00011
DATE OF LAST UPDATE:    00/00/00
PRIMARY USE DATABASE
FLD       NAME        TYPE WIDTH  DEC
001       CODE          C    003
002       NAME          C    010
003       AGE           N    002
004       SEX           C    001
005       SALA          N    006    002
** TOTAL **            00023
. LIST
00001   A01   Cau-a      30  f   93.80
00002   B02   Qen-b      43  m   84.30
00003   A03   Shen-c     23  m   68.74
00004   B05   Li-d       40  f   90.76
00005   A10   Zhou-e     36  m   70.00
00006   C01   Wang-f     51  f  160.30
00007   C05   Fong-g     41  m   68.90
00008   C10   Jang-h     43  m   68.90
00009   C15   Qu-i       44  m   73.90
00010   C50   Zhang-k    45  m   78.50
00011   M12   Hua-j      28  m   47.30
```

③ 说明：在联结操作中，结果文件会涉及到主区和辅区中的文件。这时，一般都使用限定符加以区分。主区的限定符为“P·”，辅区的限定符为“S·”，限定符的缺省值为SELECT语句设定的区。

例 6—8　检索职员文件中工资低于课题负责人工资 60% 的职员

① 分析：这实质上是一个联结问题，因为单用主区无法表达检索条件。为了解决这个问题，可把 emp 同时放于主区和辅区中，然后进行联结运算生成结果文件jt1。

② 实现：

```
. USE emp
. LIST
00001   Cau-a      30   93.80   Cdept   Zhou-e
00002   Qen-b      43   84.30   Adept   Li-d
00003   Shen-c     23   68.74   Adept   Wang-f
00004   Li-d       40   90.76   Bdept   Wang-f
00005   Zhou-e     36   70.00   Adept   Wang-f
00006   Wang-f     51  160.30   Cdept
00007   Fong-g     41   68.90   Cdept   Li-d
00008   Jang-h     43   68.90   Cdept   Zhang-k
00009   Qu-i       44   73.90   Cdept   Zhang-k
```

```
00010      Zhang-k        45    78.50   Cdept
00011      Hua-j          28    47.30   Cdept      Zhang-k
. SELECT SECONDARY
. USE emp
. SELECT PRIMARY
. JOIN TO jt1 FOR manag=s.name.AND. sala<s.sala*0.6;
        FIELD name, age, sala, dname, manag
. USE jt1
. LIST
00001      Shen-c         23    68.74   Adept      Wang-f
00002      Li-d           40    90.76   Bdept      Wang-f
00003      Zhou-e         36    70.00   Adept      Wang-f
. LIST STRU
STRUCTURE FOR FILE:    A:JT1    .DBF
NUMBER OF RECORDS:     00003
DATE OF LAST UPDATE:   00/00/00
PRIMARY USE DATABASE
FLD      NAME       TYPE   WIDTH    DEC
001      NAME       C      010
002      AGE        N      002
003      SALA       N      006      002
004      DNAME      C      006
005      MANAG      C      010
** TOTAL **          00035
```

2. 排序和索引

排序是把指定的文件按某个字段值的大小进行重排，这个字段被称为“关键字段”。序可按关键字段的值由小到大（增序）进行，也可从大到小（减序）进行。重排的结果产一个新的文件。排序语句的格式为：

SORT ON〈字段名〉TO〈文件名〉〔ASCENDING | DESCENDING〕

其中任选项〔ASCENDING | DESCENDING〕为排序的方式，ASCE 表示增序，DESC 表示减序，缺省值为增序。例如，把职员的文件按姓名进行排序，操作如下：

```
. USE emp
. LIST
00001      Cau-a          30    93.80   Cdept      Zhou-e
00002      Qen-b          43    84.30   Adept      Li-d
00003      Shen-c         23    68.74   Adept      Wang-f
00004      Li-d           40    90.76   Bdept      Wang-f
00005      Zhou-e         36    70.00   Adept      Wang-f
00006      Wang-f         51   160.30   Cdept
00007      Fong-g         41    68.90   Cdept      Li-d
00008      Jang-h         43    68.90   Cdept      Zhang-k
```

00009	Qu-i	44	73.90	Cdelp	Zhang-k
00010	Zhang-k	45	78.50	Cdept	
00011	Hua-j	28	47.30	Cdept	Zhang-k

```
. SORT ON name TO empsort
SORT COMPLETE
```

然后用 LIST 语句把排序后得到的新文件内容列出检查：

```
. USE empsort
. LIST
```

00001	Cau-a	30	93.80	Cdept	Zhou-e
00002	Fong-g	41	68.90	Cdept	Li-d
00003	Hua-j	28	47.30	Cdept	Zhang-k
00004	Jang-h	43	68.90	Cdept	Zhang-k
00005	Li-d	40	90.76	Bdept	Wang-f
00006	Qen-b	43	84.30	Adept	Li-d
00007	Qu-i	44	73.90	Cdept	Zhang-k
00008	Shen-c	23	68.74	Adept	Wang-f
00009	Wang-f	51	160.30	Cdept	
00010	Zhang-k	45	78.50	Cdept	
00011	Zhou-e	36	70.00	Adept	Wang-f

如果需要对职员文件按所属部门进行排序，则操作如下：

```
. USE emp
.SORT ON dname TO empsort1
.USE empsort1
.LIST
```

00001	Qen-b	43	84.30	Adept	Li-d
00002	Shen-c	23	68.74	Adept	Wang-f
00003	Zhou-e	36	70.00	Adept	Wang-f
00004	Li-d	40	90.76	Bdept	Wang-
00005	Cau-a	30	93.80	Cdept	Zhou-e
00006	Fong-g	41	68.90	Cdept	Li-d
00007	Hua-j	28	47.30	Cdept	Zhang-k
00008	Jang-h	43	68.90	Cdept	Zhang-k
00009	Qu-i	44	73.90	Cdept	Zhang-k
00010	Wang-f	51	160.30	Cdept	
00011	Zhang-k	45	78.50	Cdept	

在 dBASE Ⅰ 的排序操作中，仅允许使用一个关键字段。若想对多个字段进行排序，必

须建立多个文件，这样就造成了数据的大量冗余。为克服排序的这一缺陷，dBASE Ⅰ提供了一种索引操作。执行索引操作之后，dBASE Ⅰ将建立一个索引文件（·NDX），此后它将配合原数据库文件进行各种操作。

索引文件由关键字段及其在原文件中的记录序号所组成，它的结构示意如下：

关键字段	指 针
Cau a	00001
Fong g	00007
Hua j	00011
Jang h	00008
Li d	00004
Qen b	00002
Qu i	00009
Shen c	00003
Wang f	00006
Zhang k	00010
Zhou e	00005

但应注意，真正的索引文件并不是一个 ASCII 码文件。

使用索引文件的好处是：在记录数目较大的时候，索引文件比排序文件小得多，可大大地节省盘空间；索引文件的关键字段可以是几个字段的联结；使用带索引的 USE 语句等效于使用排序文件；由于数据没有多大冗余，数据的一致性问题比较好解决。对已经做了索引的数据库文件，需要更新时，可使用带索引的 USE 语句。这样，对当前文件的任何更新，dBASE Ⅰ都会对 USE 中列出的索引文件自动地进行更新，从而保证了数据的一致性。

索引语句的格式为：

INDEX ON〈字段表达式〉TO〈索引文件名〉

它用以建立索引文件。而带索引的 USE语句的格式为：

USE〈文件名〉INDEX〈索引文件〉{，〈索引文件〉}

例如：

```
. USE emp
. INDEX ON name TO empndx1
. USE emp INDEX empndx1
. LIST
00001    Cau-a      30    93.80    Cdept    Zhou-e
00007    Fong-g     41    68.90    Cdept    Li-d
00011    Hua-j      28    47.30    Cdept    Zhang-k
00008    Jang-h     43    68.90    Cdept    Zhang-k
00004    Li-d       40    90.76    Bdept    Wang-f
00002    Qen-b      43    84.30    Adept    Li-d
```

00009	Qu-i	44	73.90	Cdept	Zhang-k
00003	Shen-c	23	68.74	Adept	Wang-f
00006	Wang-f	51	160.30	Cdept	
00010	Zhang-k	45	78.50	Cdept	
00005	Zhou-e	36	70.00	Adept	Wang-f

下面举几个使用索引文件进行排序操作的例子。

例 6—9 按先部门后姓名对职员文件进行排序

```
. USE emp
. INDEX ON dname+name TO empndx2
. USE emp INDEX empndx2
. LIST
```

00002	Qen-b	43	84.30	Adept	Li-d
00003	Shen-c	23	68.74	Adept	Wang-f
00005	Zhou-e	36	70.00	Adept	Wang-f
00004	Li-d	40	90.76	Bdept	Wang-f
00001	Cau-a	30	93.80	Cdept	Zhou-e
00007	Fong-g	41	68.90	Cdept	Li-d
00011	Hua-j	28	47.30	Cdept	Zhang-k
00008	Jang-h	43	68.90	Cdept	Zhang-k
00009	Qu-i	44	73.90	Cdept	Zhang-k
00006	Wang-f	51	160.30	Cdept	
00010	Zhang-k	45	78.50	Cdept	

说明：列表输出的顺序是先按部门，同一部门又按姓名为序的。

例 6—10 对 emp 文件添加一个记录

如果使用下面的语句进行操作：

```
. USE emp
. APPEND
  ⋮                    {在 emp 中添加第 12 个记录}
. LIST
```

00001	Cau-a	30	93.80	Cdept	Zhou-e
00002	Qen-b	43	84.30	Adept	Li-d
00003	Shen-c	23	68.74	Adept	Wang-f
00004	Li-d	40	90.76	Bdept	Wang-f
00005	Zhou-e	36	70.00	Adept	Wang-f
00006	Wang-f	51	160.30	Cdept	
00007	Fong-g	41	68.90	Cdept	Li-d
00008	Jang-h	43	68.90	Cdept	Zhang-k

00009	Qu-i	44	73.90	Cqept	Zhang-k
00010	Zhang-k	45	78.50	Cdept	
00011	Hua-j	28	47.30	Cdept	Zhang-k
00012	Add-name	45	80.50		

然后执行语句：

```
. USE emp INDEX empndx2, empndx1
. LIST
```

00002	Qen-b	43	84.30	Adept	Li-d
00003	Shen-c	23	68.74	Adept	Wang-f
00005	Zhou-e	36	70.00	Adept	Wang-f
00004	Li-d	40	90.76	Bdept	Wang-f
00001	Cau-a	30	93.80	Cdept	Zhou-e
00007	Fong-g	41	68.90	Cdept	Li-d
00011	Hua-j	28	47.30	Cdept	Zhang-k
00008	Jang-h	43	68.90	Cdept	Zhang-k
00009	Qu-i	44	73.90	Cdept	Zhang-k
00006	Wang-f	51	160.30	Cdept	
00010	Zhang-k	45	78.50	Cdept	

可以看出 emp 文件添加了新记录之后，根据原来的文件内容所生成的索引 empndx1、empndx2 并没有作相应的调整，所以添加的新记录没有被列出。为了保证索引文件和更新的文件内容保持一致，可执行“重编索引”（REINDEX）语句来更新索引文件：

```
. REINDEX
REINDEXING INDEX FILE - A : EMPNDX2 .NDX
00012 RECORDS INDEXED
REINDEXING INDEX FILE - A : EMPNDX1 .NDX
00012 RECORDS INDEXED
. LIST
```

00012	Add-name	45	80.50		
00002	Qen-b	43	84.30	Adept	Li-d
00003	Shen-c	23	68.74	Adept	Wang-f
00005	Zhou-e	36	70.00	Adept	Wang-f
00004	Li-d	40	90.76	Bdept	Wang-f
00001	Cau-a	30	93.80	Cdept	Zhou-e
00007	Fong-g	41	68.90	Cdept	Li-d
00011	Hua-j	28	47.30	Cdept	Zhang-k
00008	Jang-h	43	68.90	Cdept	Zhang-k
00009	Qu-i	44	73.90	Cdept	Zhang-k
00006	Wang-f	51	160.30	Cdept	
00010	Zhang-k	45	78.50	Cdept	

另一种更方便的做法是在 emp 文件添加新记录（或执行任何文件更新操作）之前，使

用带索引文件的 USE 语句来打开 emp 文件。这样，emp 文件的任何更新，都会使 USE 语句中列出的索引文件也自动地得到更新。因此，本例的做法是：

```
. USE emp INDEX empndx2, empndx1
. APPEND
      ⋮      {在 emp 文件中添加第 12 个记录}
. LIST
```

最后所列出的文件内容，将与上面得到的结果完全一致。

3．统计操作

dBASE Ⅰ 的统计功能比较简单，一般只能直接进行计数和求和，在此过程中，也可利用第一节所介绍的运算符和函数协助进行计算。

计数语句的格式为：

COUNT〔〈范围〉〕〔**FOR** 〈表达式〉〕〔**TO** 〈存贮变量〉〕

它的最简单的应用是统计文件中的记录个数。例如：

```
. USE emp
. LIST
00001    Cau-a      30    93.80   Cdept    Zhou-e
00002    Qen-b      43    84.30   Adept    Li-d
00003    Shen-c     23    68.74   Adept    Wang-f
00004    Li-d       40    90.76   Bdept    Wang-f
00005    Zhou-e     36    70.00   Adept    Wang-f
00006    Wang-f     51   160.30   Cdept
00007    Fong-g     41    68.90   Cdept    Li-d
00008    Jang-h     43    68.90   Cdept    Zhang-k
00009    Qu-i       44    73.90   Cdept    Zhang-k
00010    Zhang-k    45    78.50   Cdept
00011    Hua-j      28    47.30   Cdept    Zhang-k
. COUNT
COUNT=00011
```

下面举几个使用 **COUNT** 语句进行统计的例子。

例 6—11　统计文件 emp 中工资低于 100 元的职员所占的比例

① 功能：设存贮变量X存放职员总人数，Y存放工资低于 100 元的职员人数，求百分比。

② 实现：

```
· COUNT TO X
COUNT=00011
· COUNT FOR Sala <100 TO Y
COUNT=00010
· ? Y * 100/X
  90
```

③ 说明：dBASE Ⅰ 在决定计算结果的小数位数时，根据运算对象中小数位数最多的量

来决定。所以，如果希望保留小数后1位，则将计算百分比的语句改为下式来计算：

```
. ? (Y/X+0.0)*100
  90.9
```

例 6—12　统计属于 A 部门的职员人数

```
. COUNT ALL FOR "A" $ dname
COUNT=00003
```

求和操作仅能对数值型字段进行，求和语句的格式为：

SUM〈字段表达式〉{，〈字段表达式〉}　〔TO〈存贮变量〉〕〔〈范围〉〕〔FOR〈表达式〉〕

其中字段表达式是指运算对象含字段名的表达式。下面是使用 SUM 语句的几个例子。

例 6—13　求职员的总工资和平均工资

```
. USE emp
. SUM sala
  905.40
```

求平均工资的操作为：

```
. SUM sala/X          {总工资除以总人数}
  82.30
```

式中的X是例 6—11 计算出的总人数。这里平均工资的计算公式为：

平均工资＝职员 1 的工资 /X＋…＋职员 n 的工资 /X

例 6—14　求 C 部门的职员平均工资和平均年龄

```
. COUNT FOR "C" $dname TO cx
COUNT=00007
. SUM sala/cx, age/cx FOR "C" $dname TO salas, ages
 84.51 40
. ? salas
     84.51
. ? ages
       40
```

其中，存贮变量 salas 中为平均工资，ages 中为平均年龄。

对于已排序或已做了索引的文件，可以对文件按关键字段进行累计，这在统计工作中非常有用。

按关键字段累计仅能对数值字段进行，累计语句的格式为：

TOTAL ON〈关键字段〉TO〈文件名〉〔FIELD〈字段名表〉〕〔FOR〈表达式〉〕

例如，要统计各部门的工资小计，可按下列过程进行。首先建立一个文件 TOTALF1，以供存放累计结果。文件的结构：

```
. LIST STRU
STRUCTURE FOR FILE: A: TOTALF1 .DBF
NUMBER OF RECORDS: 00000
```

```
DATE OF LAST UPDATE: 05/20/84
PRIMARY USE DATABASE
FLD        NAME       TYPE WIDTH  DEC
001        DNAME      C    006
002        SALA       N    010    002
**TOTAL**                  00017
```

然后使用 TOTAL 语句把各部门的累计结果存入文件 TOTALF1：

```
. USE emp INDEX empndx2
. TOTAL ON dname TO totalf1
00004 RECORDS COPIED
. LIST
00012   Add-name   45   80.50
00002   Qen-b      43   84.30   Adept   Li-d
00003   Shen-c     23   68.74   Adept   Wang-f
00005   Zhou-e     36   70.00   Adept   Wang-f
00004   Li-d       40   90.76   Bdept   Wang-f
00001   Cau-a      30   93.80   Cdept   Zhou-e
00007   Fong-g     41   68.90   Cdept   Li-d
00011   Hua-j      28   47.30   Cdept   Zhang-k
00008   Jang-h     43   68.90   Cdept   Zhang-k
00009   Qu-i       44   73.90   Cdept   Zhang-k
00006   Wang-f     51   160.30  Cdept
00010   Zhang-k    45   78.50   Cdept
. USE totalf1
. LIST
00001                  80.50
00002   Adept          223.04
00003   Bdept          90.76
00004   Cdept          591.60
```

如果结果文件不预先建立，也可用任选项 FIELD 后的字段名表自动生成一个文件。但对上例，由于 sala 字段过小（N，6，2），不宜使用这种方法。

4．按关键字段对文件进行更新

所谓按关键字段对文件内容进行更新，是指根据其它文件的内容对当前文件进行更新。它所使用的语句是“更新”语句（UPDATE），其格式如下：

UPDATE FROM〈文件名〉ON〈关键字段〉〔ADD〈字段名表〉〕〔REPLACE〈字段名表〉〕

其中 ADD 后列出的字段名必须为数值字段，它们用来与原文件的对应字段作相加运算。REPLACE 后列出的字段名为进行置换操作的字段。

注意，在更新以前必须对所涉及的两个文件进行排序或做索引。按文件内容进行更新的方法可从下例看出。

例 6—15　职员表的更新

① 功能：假设由于体制改革，某些职员的工资和所属部门作了如表 6-9 所示的调整。

设需要调整的职员表格已被作为数据库文件 empaj 建立，现要利用文件 empaj 对文件 emp 进行更新。

表 6-9　职员调整表

姓　　名	增加的工资	调往部门
Shen-c	20.00	Bdept
Qen-b	10.00	Bdept
Wang-f	−10.00	Apept
Add-name	—	Adept

② 实现过程：

假设文件 empaj 已经建立，并已输入数据，它的结构和内容为：

```
. USE empaj
. LIST STRU
STRUCTURE FOR FILE: A:EMPAJ .DBF
NUMBER OF RECORDS: 00004
DATE OF LAST UPDATE: 05/30/84
PRIMARY USE DATABASE
FLD     NAME      TYPE WIDTH    DEC
001     NAME      C    010
002     SALA      N    006      002
003     DNAME     C    006
** TOTAL **            00023
. LIST
00001   Shen-c          20.00   Bdept
00002   Qen-b           10.00   Bdept
00003   Wang-f          -10.00  Adept
00004   Add-name          0.00  Adept
```

emp 文件的内容为：

```
. USE emp
. LIST
00001   Cau-a    30   93.80   Cdept   Zhou-e
00002   Qen-b    43   84.30   Adept   Li-d
00003   Shen-c   23   68.74   Adept   Wang-f
00004   Li-d     40   90.76   Bdept   Wang-f
00005   Zhou-e   36   70.00   Adept   Wang-f
00006   Wang-f   51   160.30  Cdept
00007   Fong-g   41   68.90   Cdept   Li-d
00008   Jang-h   43   68.90   Cdept   Zhang-k
00009   Qu-i     44   73.90   Cdept   Zhang-k
00010   Zhang-k  45   78.50   Cdept
00011   Hua-j    28   47.30   Cdept   Zhang-k
00012   Add-name 45   80.50
```

然后，对 empaj 按关键字段 name 进行排序，排序的结果存放在文件 empajf 中。

```
. USE empaj
. SORT ON name TO empajf
SORT COMPLETE
. USE empajf
. LIST
00001    Add-name      0.00    Adept
00002    Qen-b        10.00    Bdept
00003    Sen-c        20.00    Bdept
00004    Wang-f      -10.00    Adept
```

通过索引文件 empndx1 来使用 emp 文件：

```
. USE emp INDEX empndx1
. LIST
00012    Add-name     45    80.50
00001    Cau-a        30    93.80    Cdept    Zhou-e
00007    Fong-g       41    68.90    Cdept    Li-d
00011    Hua-j        28    47.30    Cdept    Zhang-k
00008    Jang-h       43    68.90    Cdept    Zhang-k
00004    Li-d         40    90.76    Bdept    Wang-f
00002    Qen-b        43    84.30    Adept    Li-d
00009    Qu-i         44    73.90    Cdept    Zhang-k
00003    Shen-c       23    68.74    Adept    Wang-f
00006    Wang-f       51   160.30    Cdept
00010    Zhang-k      45    78.50    Cdept
00005    Zhou-e       36    70.00    Adept    Wang-f
```

最后，更新文件并列出更新后的结果：

```
. UPDATE ON name FROM empajf ADD sala REPLACE dname
. LIST
00012    Add-name     45    80.50    Adept
00001    Cau-a        30    93.80    Cdept    Zhou-e
00007    Fong-g       41    68.90    Cdept    Li-d
00011    Hua-j        28    47.30    Cdept    Zhang-k
00008    Jang-h       43    68.90    Cdept    Zhang-k
00004    Li-d         40    90.76    Bdept    Warg-f
00002    Qen-b        43    94.30    Bdept    Li-d
00009    Qu-i         44    73.90    Cdept    Zhang-k
00003    Shen-c       23    68.74    Adept    Wang-f
00006    Wang-f       51   150.30    Adept
00010    Zhang-k      45    78.50    Cdept
00005    Zhou-e       36    70.00    Adept    Wang-f
```

上面介绍了对数据库文件中的数据进行的各种操作，如检索、统计、排序、更新等。本节使用的主要语句有：

① 检索

* DISPLAY(LIST)〔〈范围〉〕〔〈字段表〉〕〔FOR〈表达式〉〕〔OFF〕　显示输出检索的结果。
* COPY TO〈文件名〉〔〈范围〉〕〔FIELD〈字段表〉〕〔FOR〈表达式〉〕　把检索的结果以文件的形式保存起来。
* JOIN TO〈文件名〉〔FOR〈表达式〉〕〔FIELD〈字段表〉〕　联结两个文件，生成一个新文件。

② 统计

* COUNT〔〈范围〉〕〔FOR〈表达式〉〕〔TO〈存贮变量〉〕　对文件的内容计数。
* SUM〈字段表达式〉〔〈范围〉〕〔TO〈存贮变量〉〕〔FOR〈表达式〉〕　对数值型字段值作纵向求和。

③ 排序与索引

* SORT ON〈关键字段〉TO〈文件名〉〔ASCENDING | DESCENDING〕　按关键字段进行排序。
* INDEX ON〈关键字段〉TO〈索引文件名〉　建立索引文件，其中关键字段可以是若干个字段的联合。
* USE〈文件名〉INDEX〈索引文件名表〉　按给定的索引使用文件。文件的排序仅按索引文件表中的第一个索引文件进行。

④ TOTAL 和 UPDATE 语句

* TOTAL TO〈文件名〉ON〈关键字段〉〔FIELD〈字段表〉〕　按关键字段累计。
* UPDATE FROM〈文件名〉ON〈关键字段〉〔ADD〈字段表〉〕〔REPLACE〈字段表〉〕　按文件内容更新。

第五节　报表生成

1. 报表语句

到目前为止，有关数据库文件的输出都是通过 LIST 和 DISPLAY 语句实现的。显然，这种格式并不十分理想。

为了把数据库文件中的内容以更符合用户要求的格式报告给用户，dBASE Ⅰ提供了一条“报表语句”（REPORT），它既用来生成一种描述用户对报表格式要求的文件，称为“报表格式文件”（·FRM），又用来产生报表输出。

在使用报表语句时，如果指定的报表格式文件不存在，则 dBASE Ⅰ会采用会话方式生成报表格式文件。报表格式文件生成之后，dBASE Ⅰ就自动地根据该报表格式的要求，从当前数据库文件中取出有关数据，生成报表并显示或打印出来。

报表语句的语法为：

REPORT〔FORM〈报表格式文件〉〕〔〈范围〉〕〔TO PRINT〕〔FOR〈表达式〉〕

其中〈报表格式文件〉缺省时，用户给出的报表格式用完之后不保存，下次使用时必须重新生

成。TO PRINT 表示报表在打印机上输出。

dBASE Ⅱ 在生成报表格式文件的过程中，它将与用户进行一连串的会话。它的第1个问题是：

ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH

其中 M 表示报表的左边框位置，L 表示每页的行数，W 表示行宽。这三个参数的缺省值分别为5，57和80。接着，dBASE Ⅱ 询问是否要有报表标题：

PAGE HEADING? (Y/N)

如果回答“是”（Y），dBASE Ⅱ 会请求输入标题。随后，dBASE Ⅱ 将询问是否要隔行输出，对数值字段是否要作累计以及是否要按关键字段分部累计。最后，dBASE Ⅱ 还要询问各字段在报表上的具体位置。

下面将通过两个实例进一步说明报表的构造方法。

2. 实　例

例 6—16　对文件 emp 加上栏标题输出，并计算出总工资。

首先描述报表的一般格式：

```
.USE emp
.REPORT FORM rex1
ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH<CR>
PAGE HEADING? (Y/N) Y
ENTER PAGE HEADING: An Example of Report
DOUBLE SPACE REPORT? (Y/N) N
ARE TOTALS REQUIRED ? (Y/N) Y
SUBTOTALS IN REPORT? (Y/N) N
```

再设计具体字段在报表中的位置，并给出各栏的栏名：

```
COL    WIDTH, CONTENTS
001    10, name
ENTER HEADING: Name
002    3, age
ENTER HEADING: Age
ARE TOTALS REQUIRED? (Y/N) N
003    10, sala
ENTER HEADING: Salary
ARE TOTALS REQUIRED? (Y/N) Y
004    8, dname
ENTER HEADING: Dept
005    10, manag
ENTER HEADING:Manager
006    <CR>
```

这样，报表格式文件就建立完毕，它的文件名为 rex1.FRM，内容如下：

```
Y                          { 需要打印报表标题 }
An Exampie of Report       { 报表标题 }
N                          { 不必隔行输出 }
Y                          { 进行累计 }
N                          { 不必分部累计 }
10, name                   { 第1栏宽度为10，内容为 name 的值 }
Name                       { 第1栏栏目为Name }
3, age                     {第2栏为 Age，不求累计}
Age
N
10, sala                   { 第3栏为 Salary，求累计 }
Salary
Y
8, dname                   {第4栏}
Dept
10, manag                  {第5栏}
Manager
```

报表格式文件建立之后，dBASE Ⅱ 会立即从显示器上显示出生成的报表。这个报表也可用 ^P 键从打印机上输出。印出的结果如下：

```
PAGE NO. 00001
01/01/80

                    An Example of Report

Name      Age    Salary   Dept    Manager
Cau-a      30     93.80   Cdept   Zhou-e
Qen-b      43     84.30   Adept   Li-d
Shen-c     23     68.74   Adept   Wang-f
Li-d       40     90.76   Bdept   Wang-f
Zhou-e     36     70.00   Adept   Wang-f
Wang-f     51    160.30   Cdept
Fong-g     41     68.90   Cdept   Li-d
Jang-h     43     68.90   Cdept   Zhang-k
Qu-i       44     73.90   Cdept   Zhang-k
Zhang-k    45     78.50   Cdept
Hua-j      28     47.30   Cdept   Zhang-k
Add-name   45     80.50
** TOTAL **
                  985.90
```

为了对照，使用 LIST 语句列出文件 emp的内容为：

```
. LIST
00001    Cau-a      30    93.80   Cdept   Zhou-e
00002    Qen-b      43    84.30   Adept   Li-d
00003    Shen-c     23    68.74   Adept   Wang-f
00004    Li-d       40    90.76   Bdept   Wang-f
00005    Zhou-e     36    70.00   Adept   Wang-f
```

00006	Wang-f	51	160.30	Cdept	
00007	Fong-g	41	68.90	Cdept	Li-d
00008	Jang-h	43	68.90	Cdept	Zhang-k
00009	Qu-i	44	73.90	Cdept	Zhang-k
00010	Zhang-k	45	78.50	Cdept	
00011	Hua-i	28	47.30	Cdept	Zhang-k
00012	Add-name	45	80.50		

仔细观察输出的报表，便可以发现它还有以下几处不够令人满意：

* 报表标题过于靠右，须修改报表的行宽（W 改为 60）。另外，还需使用DATE（ ）函数更改当前的日期。
* 每栏的名称须与本栏的数据对齐（字符串型靠左对齐，数值型靠右对齐）。这可通过在输入的标题前加上“<”（向左靠齐）或“>”（向右靠齐）来实现。
* sala 和 dname 两个字段的栏靠得太近，需插入几个空格。

为了达到上述要求，可以重建 rex1·FRM，也可以用 MODIFY COMMAND 语句直接对 rex1·FRM 进行修改。现将文件 rex1·FRM 修改成：

```
W=60                    { 行宽由80改为60 }
Y
An Example of Report
N
Y
N
10, name
<Name                   { 标题向左靠齐 }
3, age
Age
N
10, sala
>Salary                 { 标题向右靠齐 }
Y
10, "     "+dname       { 插入五个空格 }
>Department
10, "     "+manag
>Manager
```

改进后的报表可用下列语句列出：

```
. REPORT FORM rex1
```

```
PAGE NO. 00001
05/30/84
                    An Example of Report
Name      Age     Salary  Department      Manager
Cau-a     30      93.80   Cdept           Zhou-e
Qen-b     43      84.30   Adept           Li-d
Shen-c    23      68.74   Adept           Wang-f
```

```
Li-d          43     90.76     Bdept     Wang-f
Zhou-e        36     70.00     Adept     Wang-f
Wang-f        51    160.30     Cdept
Fong-g        41     68.90     Cdept     Li-d
Jang-h        43     68.90     Cdept     Zhang-k
Qu-i          44     73.90     Cdept     Zhang-k
Zhang-k       45     78.50     Cdept
Hua-j         28     47.30     Cdept     Zhang-k
Add-name      45     80.50
**TOTAL**
                    985.90
```

例 6—17　按部门求工资小计，并求工资总额。此外，还希望增加一项按年龄的工资补贴。计算方法是：（年龄－30）×2。最后打出实发工资数。

由于报表中涉及到按部门累计，所以在使用 USE 打开 emp 文件时，要给出按单位排序的索引文件 empndx2：

```
. USE emp INDEX empndx2
. REPORT FORM rex2 TO PRINT
```

接着使用对话方式建立报表格式文件 rex2·FRM：

```
ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH <CR>
PAGE HEADING? (Y/N)Y
ENTER PAGE HEADING: Example of Salary Report
DOUBLE SPACE REPORT? (Y/N)N
ARE TOTALS REQUIRED? (Y/N)Y
SUBTOTALS IN REPORT? (Y/N)Y
ENTER SUBTOTALS FIELD:dname
SUMMARY REPORT ONLY? (Y/N)N
EJECT PAGE AFTER SUBTOTALS? (Y/N)N
ENTER SUBTOTAL HEADING:Dept.Total
```

然后逐项输入各栏的描述：

```
COL     WIDTH, CONTENTS
001     10, name
ENTER HEADING:Name
002     3, age
ENTER HEADING:Age
ARE TOTALS REQUIRED? (Y/N)N
003     10, sala
ENTER HEADING: >Salary
ARE TOTALS REQUIRED? (Y/N) Y
004     8, (age-30) * 2.00
ENTER HEADING:Sala+
ARE TOTALS REQUIRED? (Y/N)Y
```

```
005       10, sala+(age-30)*2.00
ENTER HEADING:Real.Sala
ARE TOTALS REQUIRED? (Y/N)Y
006       12, dname
ENTER HEADING:Department
007       10, manag
ENTER HEADING:Manager
008       <CR>
```

于是，按此格式输出的报表为：

PAGE NO. 00001
05/30/84

Example of Salary Report

Name	Age	Salary	Sala+	Real.Sala	Department	Manager
*Dept.Total						
Add-name	45	80.50	30.00	110.50		
SUBTOTAL						
		80.50	30.00	110.50		
*Dept.Total Adept						
Qen-b	43	84.30	26.00	110.30	Adept	Li-d
Shen-c	23	68.74	−14.00	54.74	Adept	Wang-f
Zhou-e	36	70.00	12.00	82.00	Adept	Wang-f
SUBTOTAL						
		223.04	24.00	247.04		
*Dept.Total Bdept						
Li-d	40	90.76	20.00	110.76	Bdept	Wang-f
SUBTOTAL						
		90.76	20.00	110.76		
*Dept.Total Cdept						
Cau-a	30	93.80	0.00	93.80	Cdept	Zhou-e
Fong-g	41	68.90	22.00	90.90	Cdept	Li-d
Hua-j	28	47.30	−4.00	43.30	Cdept	Zhang-k
Jang-h	43	68.90	26.00	94.90	Cdept	Zhang-k
Qu-i	44	73.90	28.00	101.90	Cdept	Zhang-k
Wang-f	51	160.30	42.00	202.30	Cdept	
Zhang-k	45	78.50	30.00	108.50	Cdept	
SUBTOTAL						
		591.60	144.00	735.60		
TOTAL						
		985.90	218.00	1203.90		

如果首先把 rex2·FRM 复制成 rex3·FRM，再使用下列语句

```
·MODIFY COMMAND rex3.FRM
```

把报表格式文件改成：

```
m=1
y
Example of Summary Report
n
y
y
dname
y                                  { 只打印累计数字 }
n
Dept.Total; ----------
10, name
Name
3, age
Age
n
10, sala
>Salary
y
8, (age-30)*2.00
>Sala+
y
10, sala+(age-30)*2.00
Real.Sala
y
12, dname
Department
10, manag
Manager
```

接着使用语句

```
.REPORT FORM rex3 TO PRINT
```

产生的报表如下：

```
PAGE NO. 00001
05/30/84
                    Example of Summary Report
   Name    Age  Salary  Sala+   Real.Sala  Department  Manager
*          Dept.Total
**SUBTOTAL**
                   80.50   30.00   110.50
*          Dept.Total              Adept
**SUBTOTAL**
                  223.04   24.00   247.04
*          Dept.Total              Bdept
**SUBTOTAL**
```

```
                    90.76   20.00   110.76
*          Dept.Total          Cdept
* *SUBTOTAL* *
                    591.60  144.00  735.60
* *TOTAL* *
                    985.90  218.00 1203.90
```

上述 dBASE Ⅱ 的报表功能不很灵活，但使用起来也还方便。报表语句的格式为：

REPORT〔FORM〈报表格式文件〉〕〔〈范围〉〕〔TO PRINT〕〔FOR〈表达式〉〕

它既可用来生成报表格式文件，又可用来输出报表。其它数据库管理系统，如 INGRES 和 INFORMIX，都是分两条语句来实现的。

第六节　dBASE Ⅱ 的应用程序

在前几节介绍 dBASE Ⅱ 语句的基本功能时，采用的都是对话方式。本节将介绍 dBASE Ⅱ 语句的另一种使用方式——应用程序方式。这种方式的实现过程是先把完成一个特定任务的有关语句组合成一个应用程序，再使用 DO 语句执行这个程序以完成指定的任务。应用程序方式的优点是可以重复执行，也便于调试和修改。

1．建立应用程序文件

建立和修改应用程序文件可用 MS-DOS 下的行编辑程序 EDLIN 来完成，也可用 dBASE Ⅱ 提供的 MODIFY COMMAND 语句来完成。这个文件的类型名·PRG 在使用第 2 种方法时可省略。下面就介绍使用 MODIFY COMMAND 语句建立和修改应用程序文件的方法。

MODIFY COMMAND 语句的格式为：

MODIFY COMMAND〈应用程序文件名〉

使用这条语句后，屏幕上显示出供编辑用的画面。接着可以使用编辑键对文件进行编辑（编辑键同 MODIFY STRUCTURE语句，参见第二节中表 6-4 和 6-5 ）。

应用程序文件建立之后，可以使用“执行语句”

DO〈文件名〉

来启动执行，也可在进入 dBASE Ⅱ 的时候立即执行。例如，在操作系统下打入：

```
A > DBASE FMODI
```

表示调入 dBASE Ⅱ 并立即执行应用程序 FMODI·PRG。

2．应用程序的结构

在应用程序中，除了可以使用前几节介绍的各种数据库操作语句外，还可以使用一些程序控制语句。主要有以下几种语句。

（1）条件语句

IF〈表达式〉

〈语句串〉

〔ELSE

〈语句串〉〕

ENDIF

（2）循环语句

DO WHILE〈表达式〉

〈语句串〉

〔LOOP〕

〈语句串〉

ENDDO

（3）情况语句

DO CASE

CASE〈表达式 1〉

〈语句串 1〉

CASE〈表达式 2〉

〈语句串 2〉

⋮

CASE〈表达式 n〉

〈语句串 n〉

OTHERWISE

〈语句串〉

ENDCASE

（4）过　程

dBASE Ⅱ 的过程是一个以 RETURN 语句结尾的应用程序文件。它的调用语句为：

DO〈文件名〉

这条语句可用在程序的任何位置，但嵌套必须正确。例如：

```
DO CASE
    CASE choice= "1"
      DO insfile
    CASE choice= "2"
      DO delfile
    OTHERWISE
      RETURN
ENDCASE
```

（5）停止语句

dBASE Ⅰ的应用程序中可以使用的停止语句有下列三条：

* QUIT　　停止程序执行并返回操作系统。
* CANCEL　　停止程序执行并返回 dBASE Ⅰ的提示符状态。
* RETURN　　停止程序执行并返回调用程序，如无调用程序，就返回 dBASE Ⅰ的提示符状态。

下面介绍几个简单的应用程序，一方面是补充前几节介绍的内容，另一方面也可供读者编制自己的应用程序时参考。

3．应用程序举例

例 6—18　　修改数据库文件的结构

① 功能：从第二节可知，要修改文件的结构且保持文件的内容，是一件较麻烦的事。如若在操作中稍不当心就会破坏整个文件。为了避免操作失误，可以编制一个应用程序来完成修改文件结构的操作。

② 程序：FMODI·PRG

```
NOTE--THIS PROGRAM MODIFY DATABASE FILE STRUCTURE
* *
SET TALK OFF
ERASE
? "* * * * * * * * * * * *"
? "*   MODIFY FILE   *"
? "* * * * * * * * * * * *"
?
?
ACCEPT  "ENTER MODIFY FILE NAME" TO FNAME
IF FILE( "&FNAME" )
USE &FNAME
COPY TO WFILE STRUCTURE
USE WFILE
MODIFY STRUCTURE
APPEND FROM &FNAME
COPY TO &FNAME
DELETE FILE WFILE
ENDIF
SET TALK ON
RETURN
```

③ 说明：程序中的第 1 行和第 2 行为注释语句 NOTE 和 *，另一条注释语句是 REMARK。在程序执行时，REMARK 语句要显示注释的内容。第 3 行的 SET TALK OFF 语句用来改变系统状态，执行这条语句之后，执行的语句不在屏幕上显示出来。第 4 行的 ERASE 语句表示清除屏幕。第 5— 第 9 行的 ？ 语句显示程序标题。第 10 行的 ACCEPT 语句用来输入要修改的文件名。第 11 行中的&FNAME 表示变量 FNAME 的内容，例如：

```
·STORE "DELETE RECORD" TO X
·&X 5
```

等价于

```
· DELETE RECORD 5
```

程序中的第11到第19行为一个条件语句，如果给定的文件存在，就修改文件的结构（参阅第二节）。最后，在程序结束以前恢复dBASE Ⅱ的会话环境，并用RETURN语句返回dBASE Ⅱ。

例6—19 职员数据库文件的维护

① 功能：设文件emp.DBF的结构和内容如下：

```
· emp
· LIST STRU
STRUCTURE FOR FILE: A:EMP  .DBF
NUMBER OF RECORDS : 00012
DATE OF LAST UPDATE: 05/31/84
PRIMARY USE DATABASE
FLD    NAME   TYPE  WIDTH  DEC
001    NAME     C     010
002    AGE      N     002
003    SALA     N     006    002
004    DNAME    C     006
005    MANAG    C     010
** TOTAL **          00035
· LIST
00001    Cau-a        30    93.80   Cdept   Zhou-e
00002    Qen-b        43    84.30   Adept   Li-d
00003    Shen-c       23    68.74   Adept   Wang-f
00004    Li-d         40    90.76   Bdept   Wang-f
00005    Zhou-e       36    70.00   Adept   Wang-f
00006    Wang-f       51   160.30   Cdept
00007    Fong-g       41    68.90   Cdept   Li-d
00008    Jang-h       43    68.90   Cdept   Zhang-k
00009    Qu-i         44    73.90   Cdept   Zhang-k
00010    Zhang-k      45    78.50   Cdeqt
00011    Hua-j        28    47.30   Cdept   Zhang-k
00012    Add-name     45    80.50   Adept
```

磁盘上还有一个为实现记录的添加、插入和删除而设计的画面格式文件empfmt·FMT。这个文件的内容为：

```
@ 6,5    say "Record # :"
@ 6,16   say  rno
@ 8,5    say "name     " get name
@ 10,5   say "age      " get age
```

```
@ 12,5    say "sala      " get sala
@ 14,5    say "dname     " get dname
@ 16,5    say "manag     " get manag
```

其中语句@为格式输出语句，其格式为：

@〈坐标〉〔SAY〈表达式〉〔USING "格式"〕〕〔GET〈变量〉〔PICTURE "格式"〕〕

其中格式类同于BASIC语言的格式串（#，9表示数字，A表示字母）。

此外，盘上还有按dname作的索引文件empndx2·NDX以及基于empndx2·NDX编制的报表格式文件rex2·FRM和rex3·FRM。其中rex2·FRM用于打印按部门求累计的工资报表，rex3·FRM用于打印仅由累计和总计组成的摘要报表。rex2·FRM的内容如下：

```
m=1
y
Example of Salary Report
n
y
y
dname
n
n
Dept.Total
10,name
Name
3,age
Age
n
10,sala
>Salary
y
8,(age-30)*2.00
>Sala+
y
10,sala+(age-30)*2.00
Real.Sala
y
12,dname
Department
10,manag
Manager
```

rex3·FRM文件的内容为：

```
m=1
y
Example of Summary Report
n
```

```
y
y
dname
y
n
Dept.Total; ------------------
10,name
Name
3,age
Age
n
10,sala
>Salary
y
8,(age-30)*2.00
>Sala+
y
10,sala+(age-30)*2.00
Real.Sala
y
12,dname
Department
10,manag
Manager
```

在上述文件已存在的情况下，可编制一个应用程序文件 menu·PRG。这个程序可以采用“菜单”方式完成下列几个功能：

- 对 emp 文件增加记录
- 对emp 文件插入记录
- 对 emp 文件删除记录
- 修改 emp 文件中的记录
- 打印工资报表
- 打印工资报表摘要
- 返回 dBASE Ⅰ 提示符状态

② 程序：

```
NOTE--EXAMPLE 2
**
**
SET TALK OFF
DO WHILE T
ERASE
?
?
```

```
? '**********************'
? '*          MENU      *'
? '*    1. ADD RECORD   *'
? '*    2. INSERT RECORD *'
? '*    3. DELETE RECORD *'
? '*    4. MODIFY RECORD *'
? '*    5. SALARY REPORT *'
? '*    6. SUMMARY REPORT *'
? '*    7. EXIT          *'
? '**********************'
? 'enter desired action'
WAIT TO action
USE emp
IF action= '7'
  SET TALK ON
  CANCEL
ENDIF
IF action= '1'
   APPEND BLANK
   STORE # TO RNO
   SET FORMAT TO empfmt
   READ
ENDIF
IF action= '2'
   ACCEPT 'RECORD #' TO RNO
   LOCATE RECORD &RNO
   INSERT BLANK
   STORE # TO RNO
   SET FORMAT TO empfmt
   READ
ENDIF
IF action= '3'
   ACCEPT 'enter delete condition' TO exp
   DELETE FOR &exp
ENDIF
IF action= '4'
   ACCEPT 'enter record #' TO RNO
   LOCATE RECORD &RNO
   SET FORMAT TO empfmt
   READ
ENDIF
IF action= '5'
   USE emp INDEX empndx2
   REPORT FORM rex2 TO PRINT
ENDIF
IF action= '6'
   USE emp index empndx2
   REPORT FORM rex3 TO PRINT
```

```
ENDIF
INDEX ON dname+name TO empndx2
ENDDO
RETURN
```

③ 说明：

* 程序的主体用 DO WHILE T 和ENDDO 括起来。由于给定的条件永真，程序会一直循环下去。程序仅当用户选择功能 7 时，才执行 CANCEL 语句强行结束。

* 程序中用？语句输出的方框为功能“菜单”。“菜单”的选择号由 WAIT 语句送入存贮变量 action。

* 功能 1 用来对 emp 添加一个记录。它首先添加一个空白记录，再把当前记录号送入存贮变量 RNO（STORE #TO RNO）。接着把用户输入的记录值存入文件。下面就是执行功能 1 的过程。假设 emp 文件中已有12 个记录，在菜单状态下打入 1：

```
**************************
*          MENU          *
*     1. ADD RECORD      *
*     2. INSERT RECORD   *
*     3. DELETE RECORD   *
*     4. MODIFY RECORD   *
*     5. SALARY REPORT   *
*     6. SUMMARY REPORT  *
*     7. EXIT            *
**************************
enter desired action
WAITING 1
```

随后，屏幕上出现下列画面：

```
Record  # :        13
name      :        :
age       :   :
sala      :       :
dname     :       :
manag     :         :
```

这时用户就可以输入第 13 号记录的记录值，用户输入以下数据：

```
Record # :          13
name        :Ma-x   :
age         :50:
sala        :100.00:
dname       :Edept :
manag       :Li-d   :
```

输入完毕之后，程序返回“菜单”状态。下面打入“7”返回 dBASE Ⅱ 提示符状态：

```
* * * * * * * * * * * * * * * * * * * * * * *
*                  MENU                       *
*          1. ADD RECORD                      *
*          2. INSERT RECORD                   *
*          3. DELETE RECORD                   *
*          4. MODIFY RECORD                   *
*          5. SALARY REPORT                   *
*          6. SUMMARY REPORT                  *
*          7. EXIT                            *
* * * * * * * * * * * * * * * * * * * * * * *
enter desired action
WAITING 7
```

再用 DISPLAY 语句检查输入的记录是否正确：

```
. DISPLAY
00013      Ma-x          50    100.00    Edept     Li-d
```

* 功能 2 基本类似于功能 1，所不同的仅在于输入的新记录可以插在原文件的任意位置。
* 功能 3 使用宏定义符“&”把用户输入的表达式填到 DELETE FOR 之后，形成了一个完整的删除语句。执行这条语句就可以删除指定的记录。
* 功能 4 用来修改指定的记录。
* 功能 5 和功能 6 用来打印报表。它们输出的结果类似于例 6—17。

例 6—20　车票预订系统

① 功能：本例是会务管理系统的一个组成部分，它用来为会议代表预订返程车票。其功能有：车票登记、修改、票价结算、分类统计和派车调度等。

② 程序：该系统只使用一个数据库文件，称为 TRAINF，它的文件结构打印如下：

```
USE TRAINF
.LIST STRU
STRUCTURE FOR FILE:    A:TRAINF  .DBF
NUMBER OF RECORDS:     00012
DATE OF LAST UPDATE:   07/30/84
PRIMARY USE DATABASE
FLD     NAME   TYPE WIDTH  DEC
001     NO     C    003
002     NM     C    008
003     ED     C    010
004     SQ     C    004
005     TM     C    010
006     TY     C    002
007     NU     N    002
008     AC     N    006     002
009     CO     N    006     002
010     BK     N    006     002
011     CM     C    010
```

```
** TOTAL **                    00068
```

其中字段名的意义作如下约定：

NO —代表编号
NM—代表的房号
ED —终点站名
SQ —车次
TM —发车时间
TY —车票等级
NU —车票数量
AC —预付款
CO —票价
BK —应退款数目
CM —备注

应用程序的主模块 JMENU·PRG 如下：

```
NOTE-THIS IS A MAIN MODULE
**
**
SET TALK OFF
USE TRAINF
DO WHILE T
ERASE
?
?
?
?
?
? "          * * * * * * * * * * * * * * * * * * * * * * *"
? "          *                                             *"
? "          *                main menu                    *"
? "          *                --------                     *"
? "          *                1. record                    *"
? "          *                2. modify                    *"
? "          *                3. compute                   *"
? "          *                4. report                    *"
? "          *                5. exit                      *"
? "          * * * * * * * * * * * * * * * * * * * * * * *"
?
? " enter desired action"
WAIT TO action
```

```
IF action= "1"
   DO jmenu1
ENDIF
IF action= "2"
   DO jmenu2
ENDIF
IF action= "3"
   REPLACE ALL BK WITH ( ac-nu * co )
ENDIF
IF action= "4"
   DO jmenu4
ENDIF
IF action= "5"
   SET TALK ON
   CANCEL
ENDIF
LOOP
ENDDO
RETURN
```

由主模块调用的子模块有三个，它们是JMENU1·PRG、JMENU2·PRG和JMENU4·PRG。

* JMENU1·PRG主要完成订票的登记工作：

```
NOTE—THIS IS MODULE1
*
DO WHILE T
ERASE
?
? "==== input data===="
?
ACCEPT "enter NO" TO wno
IF wno= "  "
   RETURN
ENDIF
?
ACCEPT "enter NM" TO wnm
ACCEPT "enter ED" TO wed
ACCEPT "enter SQ" TO wsq
ACCEPT "enter TM" TO wtm
```

```
ACCEPT "enter TY" TO wty
INPUT  "enter NU" TO wnu
INPUT  "enter AC" TO wac
INPUT  "enter CO" TO wco
ACCEPT "enter CM" TO wcm
?
INPUT "Are all field-correct? (y/n)" TO wwd
IF .NOT.wwd
LOOP
ENDIF
USE trainf INEDX noindex, nmindex, esttdex
APPEND BLANK
REPLACE NO WITH wno,NM WITH wnm,ED WITH wed,TM WITH wtm, ;
        SQ WITH wsq, NU WITH wnu, AC WITH wac,CO WITH wco, ;
        TY WITH wty, CM WITH wcm
ENDDO
```

其中 USE trainf INDEX noindex, nmindex, esttdex 是为保证数据库文件和索引文件更新时的一致性而使用的命令语句，行末的分号表示在屏幕上另起一行。

* JMENU2·PRG 修改已登记的订票：

```
NOTE—MODIFY MODULE
DO WHILE T
ERASE
?
? "=======modify record======="
? "if code= <cr> return to main menu"
ACCEPT "enter your NO" TO wno
USE trainf INDEX noindex, nmindex, esttdex
FIND &wno
EDIT #
IF wno <> " "
LOOP
ENDIF
RETURN
ENDDO
```

* JMENU4·PRG 用于打印报表。按房号打印的报表便于财会人员按房间结算，按终点站、车次、时间、车票等级打印的报表便于购票人员使用，也便于安排送代表去车站的交通车。

```
NOTE—REPORT
**
**
DO WHILE T
ERASE
?
?
? "          * * * * * * * * * * * * * * * * * * * * * * * * * "
? "          *                  report menu                    * "
? "          *        1. index                                 * "
? "          *        2. report by NM                          * "
? "          *        3. report by ED+SQ+TM+TY                 * "
? "          *        5. exit                                  * "
? "          * * * * * * * * * * * * * * * * * * * * * * * * * "
? " enter desired action"
WAIT TO act
IF act="1"
INDEX ON nm TO nmindex
INDEX ON no TO noindex
INDEX ON ed+sq+tm+ty TO esttdex
ENDIF
IF act="2"
USE trainf INDEX nmindex
REPORT FORM nmrp TO PRINT
ENDIF
IF act="3"
USE trainf INDEX esttdex
REPORT FORM esttrp TO PRINT
ENDIF
IF act<> "5"
LOOP
ENDIF
RETURN
ENDDO
```

③ 说明：下面是选择主菜单1后进行车票登记的画面。冒号后面是输入的内容：代表编号100，住在5—506房间，返程回上海，乘46次火车，时间是5月26日13：37，要求软座（T2），订票1张，预付15元，实际价格是12.50元。当确认后输入“是”（y），订票要求便进入文件。然后再输入下一个代表的订票要求。如果代表编号取空白，则表示登

记结束．返回主菜单。下面是代表100订票的画面：

```
enter NO : 100
enter NM : 5-596
enter ED: SHANG HAI
enter SQ:46
enter TM:5/26/13:37
enter TY:T2
enrer NU:1
enter AC:15
enter CO:12.5
enter CM:
Are all field-correct? (y/n): Y
```

假定代表要把预定的软座（T2）改为硬座，票价是7.50元，可使用主菜单的第2项，调用修改模块。修改的画面如下：

```
======== modify record ========
if code=<cr> return to main menu
enter your NO:100
RECORD # 00012
NO          :100:
NM          :5-596:
ED          :SHANG HAI :
SQ          :46 :
TM          :5/26/13:37:
TY          :T2:
NU          : 1:
AC          : 15.00:
CO          : 12.50:
BK          :  0.00:
CM          :              :
```

修改后的画面为：

```
RECORD # 00012
NO          :100:
NM          :5-506  :
ED          :SHANG HAI :
SQ          :46 :
TM          :5/26/13:37:
TY          :T3:
NU          : 1:
AC          : 15.00 :
CO          :  7.50:
BK          :  0.00:
CM          :              :
```

计算票价（主菜单的第3个项目）后，选择主菜单的第4个项目，打印报表。下面是按房号

和终点站名、车次、时间和等级所生成的报表，同时也列出了订票数据库文件的内容和两个报表描述文件的清单，供参考。

PAGE NO. 00001
07/30/84

REPORT A

Code	Room No.	Station	Seq.	Date-Time	Ty.	Nu.	Regis.	Price	Regre.
* Room No. 5—201									
016	5—201	WU XI	311	5/17/ 4:56	T2	1	15.00	6.50	8.50
* * SUBTOTAL * *									
						1	15.00	6.50	8.50
* Room No. 5—312									
099	5—312	SHANG HAI	46	5/27/13:37	T1	1	19.70	12.50	7.20
042	5—312	WU HU	330	5/30/ 6:40	T1	1	19.00	5.90	13.10
083	5—312	GUANG ZHOU	236	5/17/ 9:45	F1	1	90.00	75.00	15.00
* * SUBTOTAL * *									
						3	128.70	93.40	35.30
* Room No. 5—442									
123	5—442	SU ZOU	301	5/22/ 6:25	T3	1	10.00	4.60	5.40
* * SUBTOTAL * *									
						1	10.00	4.60	5.40
* Room No. 5—501									
002	5—501	WU XI	311	5/25/ 4:56	T2	4	90.00	6.00	66.00
125	5—501	WU XI	311	5/21/ 4:36	T3	1	5.00	4.60	0.40
079	5—501	BEI JING	296	5/17/10:00	F1	1	55.00	50.00	5.00
* * SUBTOTAL * *									
						6	150.00	60.60	71.40
* Room No. 5—503									
013	5—503	XU ZOU	126	5/15/ 7:50	T3	3	15.00	4.30	2.10
* * SUBTOTAL * *									
						3	15.00	4.30	2.10
* Room No. 5—506									
074	5—506	SU ZOU	301	5/22/ 6:25	T3	1	10.00	4.90	5.10
150	5—506	SHANG HAI	46	5/26/13:37	T3	1	10.00	7.50	2.50
100	5—506	SHANG HAI	46	5/26/13:37	T3	1	15.00	7.50	7.50
* * SUBTOTAL * *									
						3	35.00	19.90	15.10
* * TOTAL * *									
						17	353.70	189.30	137.80

REPORT B

Code	Room No.	Station	Seq.	Date-Time	Ty.	Nu.	Regis.	Price	Regre.
* Destination. BEI JING									
079	5-501	BEI JING	296	5/17/10 :00	F1	1	55.00	50.00	5.00
* * SUBTOTAL * *									
						1	55.00	50.00	5.00
* Destination. GUANG ZHOU									
083	5-312	GUANG ZHOU	236	5/17/ 9:45	F1	1	90.00	75.00	15.00
* * SUBTOTAL * *									
						1	90.00	75.00	15.00
* Destination. SHANG HAI									
150	5-506	SHANG HAI	46	5/26/13:37	T3	1	10.00	7.50	2.50
100	5-506	SHANG HAI	46	5/26/13:37	T3	1	15.00	7.50	7.50
099	5-312	SHANG HAI	46	5/27/13:37	T1	1	19.70	12.50	7.20
* * SUBTOTAL * *									
						3	44.70	27.50	17.20
* Destination. SU ZOU									
074	5-506	SU ZOU	301	5/22/ 6:25	T3	1	10.00	4.90	5.10
123	5-442	SU ZOU	301	5/22/ 6:25	T3	1	10.00	4.60	5.40
* * SUBTOTAL * *									
						2	20.00	9.50	10.50
* Destination. WU HU									
042	5-312	WU HU	330	5/30/ 6:40	T1	1	19.00	5.90	13.10
* * SUBTOTAL * *									
						1	19.00	5.90	13.10
* Destination. WU XI									
016	5-201	WU XI	311	5/17/ 4:56	T2	1	15.00	6.50	8.50
125	5-501	WU XI	311	5/21/ 4:36	T3	1	5.00	4.60	0.40
002	5-501	WU XI	311	5/25/ 4:56	T2	4	90.00	6.00	66.00
* * SUBTOTAL * *									
						6	110.00	17.10	74.90
* Destination. XU ZOU									
013	5-503	XU ZOU	126	5/15/ 7:50	T3	3	15.00	4.30	2.10
* * SUBTOTAL * *									
						3	15.00	4.30	2.10
* * TOTAL * *									
						17	353.70	189.30	137.80

USE TRAINF

· LIST

00001	074	5-506	SU ZOU	301	5/22/ 6:25	T3 1	10.00	4.90	5.10
00002	102	5-501	WU XI	311	5/25/ 4:56	T2 4	90.00	6.00	66.00
00003	125	5-511	WU XI	311	5/21/ 4:36	T3 1	5.00	4.60	0.40
00004	099	5-312	SHANG HAI	46	5/27/13:37	T1 1	19.70	12.50	7.20
00005	013	5-503	XU ZOU	126	5/15/ 7:50	T3 3	15.00	4.30	2.10
00006	016	5-201	WU XI	311	5/17/ 4:56	T2 1	15.00	6.50	8.50
00007	042	5-312	WU HU	330	5/30/ 6:40	T1 1	19.00	5.90	13.10
00008	079	5-501	BEI JING	296	5/17/10:00	F1 1	55.00	50.00	5.00
00009	183	5-312	GUANG ZHOU	236	5/17/ 9:45	F1 1	90.00	75.00	15.00
00010	123	5-442	SU ZOU	301	5/22/ 6:25	T3 1	10.00	4.60	5.40
00011	150	5-506	SHANG HAI	46	5/26/13:37	T3 1	10.00	7.50	2.50
00012	100	5-506	SHANG HAI	46	5/26/13:37	T3 1	15.00	7.50	0.00

```
A>TYPE NMRP.FRM

Y
REPORT A
N
Y
Y
NM
N
N
Room No.
4, no
Code
8, nm
Room No.
10,ed
Station
4, sq
Seq.
10, tm
Date-Time
3, ty
Ty.
3, nu
Nu.
y
```

```
A>TYPE ESTTRP.FRM
m=3
Y
REPORT B
N
Y
Y
ED
N
N
Destination.
4, no
Code
8, nm
Room No.
10, ed
Station
4, sq
Seq.
10, tm
Date-Time
3, ty
Ty.
3, nu
Nu.
y
```

```
6, ac                    6, ac
Regis.                   Regis.
y                        y
6, co                    6. co
Price                    Price
y                        y
6, bk                    6, bk
Regre.                   Regre.
y                        y
```

例 6—21 BASIC 程序与数据库应用程序的连接

① 功能：假设一个 BASIC 程序计算出 a 和 b 两个结果之后，需要使用 dBASE Ⅱ 的应用程序，统计出数据库文件 emp（职员文件）中年龄在 a 和 b 之间的职员的人数 X，然后 BASIC 程序再继续进行后续的处理。

② 实现：本例的关键是 BASIC 程序和数据库应用程序的连接。为此，BASIC 程序把 a、b 两个值放入文本文件 INTXT·TXT 中。然后，dBASE Ⅱ 应用程序从 INTXT·TXT 中取得 a、b 之值，对 emp 文件进行检索和统计，统计结果再转换成文本文件 OUTTXT·TXT。最后，BASIC 程序可从 OUTTXT·TXT 中取得统计结果 X并继续执行下去。

为了便于说明问题，BASIC程序中 a、b 的计算过程简化为从键盘上输入，对统计结果 X 的后续处理简化为把 X 之值显示输出。

下面是完成这两部分功能的 BASIC 程序。程序的第一部分（BASPGM·BAS）为：

```
 10 PRINT "Please input a,b" : INPUT A, B
 20 GOSUB 100
 30 SYSTEM
100 OPEN "intxt.txt" FOR OUTPUT AS #1
110 PRINT #1, STR$(A)
120 PRINT #1, STR$(B)
130 FOR S=1 TO 512 : PRINT #1, CHR$(32) ; : NEXT
140 CLOSE : RETURN
```

其中子程序 100 用来建立文本文件 INTXT·TXT，语句 130 的作用是使文件长度凑满 512个字节。程序的第二部分（BASPGMA·BAS）为：

```
 40 GOSUB 200
 50 PRINT "a<age<b : " ; X
 60 END
200 OPEN "outtxt.txt" FOR INPUT AS #2
210 INPUT #2, X
220 CLOSE : RETURN
```

它从 dBASE Ⅱ 产生的 OUTTXT·TXT 文件中读入X，然后把它显示输出。

dBASE Ⅱ 的应用程序为了与 BASIC 程序生成的 INTXT·TXT 文件接口，它预先创建了一个数据库文件 FACE，其结构如下：

```
. LIST STRU
STRUCTURE FOR FILE:   A:FACE   .DBF
NUMBER OF RECORDS     00000
DATE OF LAST UPDATE   07/30/84
PRIMARY USE DATABASE
FLD       NAME       TYPE    WIDTH    DEC
001       VAL          N       005
** TOTAL **                    00006
```

数据库应用程序的文件名是 COUNTPG·PRG，它的程序如下：

```
use face
delete all
pack
append from intxt.txt delimited
go top
store val to a
skip
store val to b
use emp
copy to empt for age> a.and.age< b
use empt
count to x
use face
delete all
pack
append blank
replace val with x
copy to outtxt.txt delimited
quit
```

于是，在 MS-DOS 操作系统下，建立一个批命令文件（BASDB·BAT）：

```
basic baspgm
dbase countpg
basic baspgma
```

整个过程便能自动地连续执行了。

假设 emp 数据库文件的内容为：

00001	Cau-a	30	103.80	Adept	Zhou-e
00002	Qen-b	43	74.30	Bdept	Li-d
00003	Shen-c	23	68.74	Adept	Wang-f
00004	Li-d	40	90.76	Bdept	Wang-f

00005	Zhou-e	36	70.00	Adept	Wang-f
00006	Wang-f	51	160.30	Cdept	
00007	Fong-g	41	68.90	Cdept	Li-d
00008	Jang-x	43	68.90	Cdept	Zhang-m
00009	Qu-k	44	73.90	Cdept	Zhang-m
00010	Zhang-m	45	78.50	Cdept	
00011	Li-h	28	47.30	Cdept	Zhang-m

下面是执行批命令文件 BASDB·BAT 的结果：

```
Please input a, b
? 25, 45
a<age<b : 9
OK
```

它表示 BASIC 程序给 dBASE Ⅱ 应用程序的年龄界限是 25 岁～45 岁，而从应用程序取回的统计数据是 9 人。

第七章　集成软件 LOTUS 1-2-3

第五章和第六章已分别介绍了两类在微型机上流行的应用软件——表格处理软件和数据库管理系统。虽然这两类软件都是以事务管理为应用对象的，但它们的实际应用场合并不完全相同。表格软件主要应用于处理带有计算功能的商业报表，而数据库管理系统则多用于对大量的数据进行组织、管理和检索。

那么能否使一个应用软件兼有上述两类软件的功能或兼有更多应用软件的功能呢？这就是集成软件的设计目标。所谓集成软件是把若干种应用软件“集成”起来组成的一种新型应用软件，这种新型的应用软件一般都兼有被集成的各种应用软件的功能和优点。

本章所介绍的集成软件 Lotus 1-2-3 是美国 Lotus Development 公司于 1982 年为 IBM PC 微型机开发的，它是表格软件、数据库管理系统和统计图表软件的功能集成体。Lotus 1-2-3 也简称为 1-2-3，它可以进行各种表格处理，可以从数据库中检索出数据，也可以求出统计结果或作出统计图。

第一节　概　述

1．1-2-3 的主要功能与特点

1-2-3 中的“1”代表表格处理，“2”代表数据库管理，“3”代表商用统计图绘制。它既具有表格软件的简明性，又能象数据库管理系统一样方便地检索数据，同时还能绘制多种多样的高质量的商用统计图。

1-2-3 主要有以下几个特点：

① 使用方法简单，易于初学者学习。

② 处理速度比普通的表格软件快。

③ 1-2-3 的命令采用层次结构，用户输入命令的方式为点“菜单”式。

④ 一次可处理的数据量较大，可达 2048×256 项。

⑤ 提供较强的联机求助功能和会话式教学系统。

1-2-3 具有三大功能，现分别予以叙述。

（1）表格处理

1-2-3 可以完成普通表格软件的所有功能，例如可以定义表格的格式、输入数据和公式、打印表格和保存表格。

1-2-3 的表格处理有较强的统计计算能力，除了可进行四则运算外，它还提供下列六类函数：

* 商用函数，例如 NPV、FV、PV 等。
* 统计函数，例如 COUNT、SUM、AVG、STD 等。
* 算术函数，例如三角函数、指数函数、对数函数等。
* 逻辑函数，例如与、或、非函数等。
* 日期函数，例如 DAY、MONTH、YEAR 等。
* 辅助函数，例如 CHOOSE、HLOOKUP 等。

（2）数据库管理

1-2-3可用来收集和组织一定数量的数据，并可对这些数据进行查询和统计。1-2-3的数据库文件由若干记录组成，每个记录占表格中的一行，每行又由若干字段（表格单元）组成。表格在作为数据库文件使用时，它的第 1 行为记录说明行，用来指出各字段名，其余各行为记录，故允许有 2047 个记录，每个记录最多可有 256 个字段。

1-2-3 可以完成下列数据库管理任务：

* 按关键字段对记录进行排序。
* 可进行查找、选择和删除等检索操作。
* 可制作频度分布等预测表。
* 对数据进行统计计算（DCOUNT、DSUM、DAVG、DSTD 等）。

（3）商用统计图

1-2-3 的商用统计图绘制系统可以把表格中的数据以商用统计图的形式绘制出来。1-2-3 的绘图系统使用简单，绘制的图形质量较高，这是1-2-3的主要特点之一。

1-2-3 的绘图系统主要可执行以下几个功能：

* 绘制多种类型的统计图，例如条形图、叠置条形图、扇形图、折线图和XY图。
* 画出二维坐标轴和标题。
* 图形的比例变换和旋转变换。
* 使用的绘图设备有图形显示器、点阵打印机和绘图仪。

2. 1-2-3 的管理系统

Lotus 1-2-3 通过一个名为 LOTUS 的管理系统对分布在三张盘上的五组应用程序进行管理和调度。管理系统 LOTUS 与 1-2-3 本身一样，也采用“菜单”方式工作，使用起来十分方便。

（1）1-2-3 系统的组成

1-2-3 的整个系统由分布在四张盘上的六组程序组成，这四张盘及其内容如下：

#1（系统盘）：管理系统、1-2-3、磁盘管理和文件管理系统

#2（图形打印盘）：图形打印程序

#3（实用程序盘）：文件转换程序

#4（教学程序盘）：1-2-3 的教学程序

这些应用程序之间的关系如图 7—1 所示。

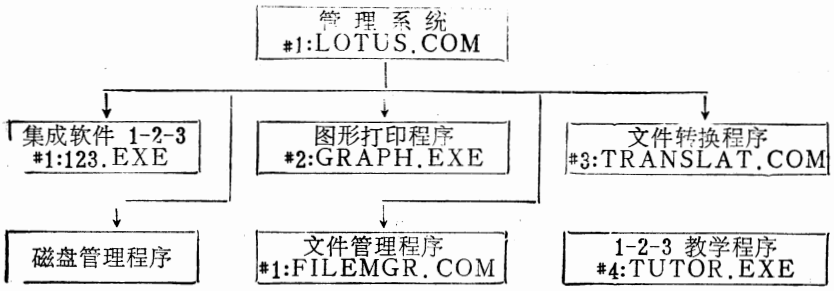


图7-1 1-2-3的组成

其中1-2-3的教学程序是独立的，它不受管理系统LOTUS的控制。这个程序主要用于辅导1-2-3的初学者学习使用1-2-3，它可以在MS DOS的支持下单独运行。

（2）管理系统的使用

在MS-DOS下，通过打入

A〉LOTUS

启动1-2-3的管理系统，屏幕上显示出下列“菜单”：

1-2-3 File—Mgr Disk—Mgr GRAPH Translate PC—DOS

这时如要选择执行某个程序有两种方法，其一是把光标移到所选程序名上并打入一个〈CR〉键，其二是输入程序名的第1个字母。

被选的程序执行完毕后，用Q命令返回管理系统。此外，在使用管理系统的过程中，还随时可以按下〈F1〉键（求助功能键），系统会给出有关的操作提示。

（3）磁盘管理（Disk—Mgr）

在选择了“Disk—Mgr”之后，管理系统就会执行磁盘管理程序。它可以完成下列功能：

- Disk—Copy　复制A盘到B盘。
- Compare　比较A盘和B盘。
- Prepare　初始化B盘。
- Status　查看指定盘的空间。

（4）文件管理（File—Mgr）

在管理系统下选择“File—Mgr”之后，1-2-3首先要求输入源盘和目的盘的盘符，接着提供下列命令供用户选择：

- Copy　复制文件。
- Erase　删除源盘上指定的文件。
- Rename　文件更名。
- Archive　以新文件名建立指定文件的备份文件。

* Disk—Drive　　变更源盘。
* Sort　　对文件目录排序列表。
* Quit　　返回管理系统 LOTUS。

3．图形打印程序

图形打印程序主要用来打印由集成软件 1-2-3 生成的商用统计图文件（类型名为·PIC），也可用来设置打印机参数。

图形打印程序主要有以下几条命令：

* Select　　选择要打印的统计图文件。
* Go　　打印所选的统计图文件。
* Options　　选择打印参数。
* Configure　　设置硬件环境参数。
* Align　　设置打印机页首位置。
* Page　　走纸到下一页页首。
* Quit　　退出。

4．文件类型的转换

1-2-3 为了和高级语言或其它应用软件（如 dBASE Ⅱ 和 VISICALC）交换数据，提供了一个文件转换程序。这个程序可处理以下几类文件：

* 1-2-3 的表格文件（·WKS）
* VISICALC 表格文件（·VC）
* dBASE Ⅱ数据库文件（·DBF）
* 数据交换格式文件（·DIF）

在进入文件转换程序后，屏幕上显示下列“菜单”供用户转换不同类型的文件：

* VC　to WKS
* DIF　to WKS
* WKS　to DIF
* DBF　to WKS
* WKS to DBF
* Quit

第二节　1-2-3 的操作提要

1．1-2-3 的启动

1-2-3对硬件的要求并不高，它可以在有图形显示器的系统下运行，也可以在只有字符显示器的系统下运行。1-2-3的启动方法有两种，其一是先启动管理系统 LOTUS，再打入一个〈CR〉键选择 1-2-3；另一种方法是在 MS-DOS 下直接打入：

```
A>123
```

这时屏幕上显示出 1-2-3 的版本信息。输入任意键后，屏幕上出现 1-2-3 的初态画面，如图 7-2 所示。

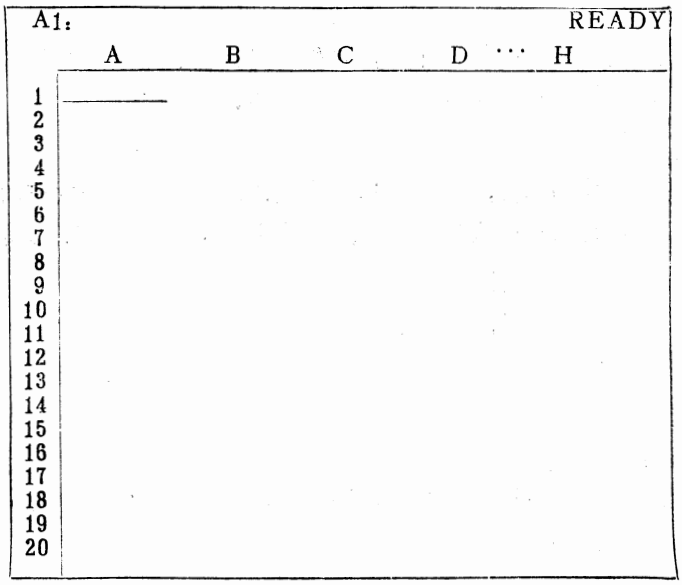


图 7—2　1-2-3 的初态画面

初态画面的前三行为 1-2-3 的状态区。其中第 1 行为当前单元行，这行的最右边为方式指示器，用来表示 1-2-3 的工作方式为 READY（齐备）、EDIT（编辑）、WAIT（等待）和 MENU（菜单）等。状态区的第 2 行为编辑行，用来显示编辑的数据或公式，有时也用来显示“菜单”和提示信息。状态区的最末行用来显示被选命令的参数或子命令。

画面的最后一行为出错信息行，用来显示 1-2-3 的出错信息。这行的最右边为键盘状态指示器，用来显示〈NumLock〉、〈ScrollLock〉和〈CapsLock〉等键按下与否的状态。

画面其余部分为 1-2-3 的表格显示区，用来显示表格中的 20 行 ×8 列表格单元（图 7—2中仅给出了 5 列）。表格区中光标所在的位置为当前单元，用下划线表示。

1-2-3的表格可达 2048 行 ×256 列，它的引用单位为表格区域。1-2-3 的表格区域可有以下几种形式：

* 单个表格单元，例如A1，IV2048等。
* 一行或一列的部分，例如A1..A10，A1..B1等。
* 矩形表格区，例如 B5..E9 等。

此外，在使用 1-2-3 的过程中，随时可用〈F1〉键进行联机求助，使用户在任何情况下都不致于陷入困境。其它功能键的作用请参见第四节中的表7-2。

2．表格单元的定位

1-2-3共有三种光标，第 1 种是命令选择光标，第 2 种是编辑行中的编辑光标，第 3 种

则是表示当前单元的单元光标。通过移动单元光标可任意指定当前单元。下面分别介绍表格单元定位的几种方法。

（1）步进定位

步进定位是指使用光标移动键↑、↓、←和→直接移动光标的方法，它一次可移动一行或一列。

如果光标已处在表格显示区的边缘，再要移动光标会引起表格显示区的上下或左右移动。如已移到表格的边缘，再要移动则会引起警告声。

（2）页式定位

如果希望大幅度地移动光标，则可采用页式定位。页式定位一次可向上、下、左或右移动一页。这里所说的页是指一个表格显示区。页式定位可通过打入下列键来完成：

〈PgUp〉	上移一页
〈PgDn〉	下移一页
|←(〈Shift-Tab〉)	左移一页
→|(〈Tab〉)	右移一页

（3）按地址定位

如果已经知道要定位的地址，可采用按地址定位法进行定位。这种定位方法的格式如下：

〈F5〉〈单元坐标〉〈CR〉

其中〈F5〉为功能键。各功能键的含义请参见表7-2。

另外，如果要定位的单元为A1，可简单地输入〈Home〉键，代替输入〈F5〉A1〈CR〉。

（4）按内容定位

1-2-3的表格可分为若干个区，若区与区之间有空行相互隔开，则可使用〈End〉键加光标移动键，把光标按照指定方向在区域边界上移动。例如，图7—3中A列有两个区，则从A2开始按下三次“〈End〉↓”后，光标的位置为A8。

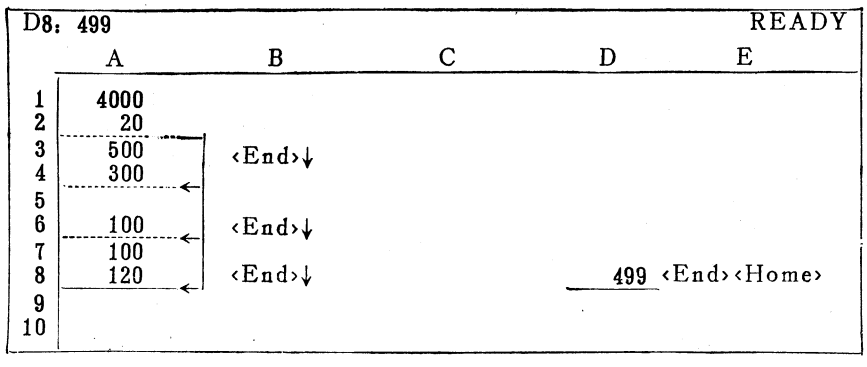


图7—3　‹End›键的使用

如果要把光标移到用户定义的表格的右下角，可按下〈End〉〈Home〉两个键来实现（如

图7—3所示）。

为了在屏幕上同时看到几个表格区域，1-2-3 允许用户在表格显示区上开设多个窗口。按下〈F6〉键，可以把光标从一个窗口移到另一个窗口。

（5）窗口的步进移动

按下〈ScrollLock〉键之后，再按下光标移动键会引起表格显示区的移动。一次只移一行或一列，且并不改变当前单元的位置。

3．数据输入

启动了1-2-3之后，就可以开始向表格输入数据和公式。输入数据是使用1-2-3的主要任务之一，它的实现方法有以下三种：

① 从键盘直接输入。

② 利用／Copy 命令从已存在的表格区域中复制数据。

③ 从磁盘上读入由 1-2-3 或其它软件建立的数据文件。

下面着重介绍第 1 种方式，其它两种方式的使用方法将另行叙述。

（1）数据的输入

输入数据的方法是：把光标移到指定的单元处，直接从键盘输入数据，数据输入完毕后，输入〈CR〉键或光标移动键。1-2-3 根据输入的第 1 个字符确定输入数据的类型。

如果输入的第 1 个字符是下列字符之一：

0..9，＋，－，．，（，@，＃和$

1-2-3就认为输入的是数值或公式。1-2-3表示的数值其精度可多达 15 位有效数字，其范围约为$-10^{99}\sim10^{99}$。

此外，1-2-3还有两类特殊的数值量。当输入的公式值不定时（如除数为0），就取值“NA”（不定）。当公式的值为非法时，其值为“ERR”，表示公式出错。

可输入的另一类数据是字符串或标号。输入标号时，通常都用前导符打头。1-2-3 的标号前导符有以下四个：

'	向左对齐
"	向右对齐
^	中心对齐
\	填满整个单元

当输入的标号为向左对齐格式且以字母开头时，可省去前导符，直接输入这个标号。

（2）数据的编辑

在输入数据的过程中，随时可用〈BS〉键删去刚输入的错误字符，或用〈Esc〉键删去一行。

对已输入的单元，则用编辑键进行修改。修改的过程是：按下〈F2〉键进入编辑状态，编辑行出现当前单元中的内容，再使用编辑键进行修改。在修改过程中，可用下列键移动编辑光标：

←	光标左移一个字符
→	光标右移一个字符

→|　　光标右移五个字符

|←　　光标左移五个字符

〈Home〉　光标移到行首

〈End〉　光标移到行末

如要插入字符，则将光标移到欲插入处，直接输入即可。如要删除字符可用下列键来完成：

〈BS〉　删除光标前的一个字符

〈Del〉　删除光标处的一个字符

〈Esc〉　删除一行

在编辑结束后，可输入一个〈CR〉键返回“READY”状态。

（3）输入后的处理

在输入数据之后，1-2-3首先检查输入的内容是否正确，如不正确就给出警告，并自动进入编辑状态，用光标指出出错位置，要求用户修改。

如果输入的数据是正确的，1-2-3通常会对表格自动地进行重新计算。若这时的重新计算方式已置为手动，则输入〈F9〉后，才进行重计算。

重计算之后，1-2-3更新显示屏幕，并返回“READY”状态。

第三节　1-2-3 的公式与函数

1-2-3的公式是一串由运算符或函数连接起来的数值常量或变量。其中变量可以是表格单元的坐标，也可以是一个表格区域或一个区域名。

1．表格单元的坐标

在公式中一般都使用坐标来引用表格单元。1-2-3共有三种形式的表格单元坐标，它们是：

- 相对坐标：通常使用的坐标都是相对坐标，这种坐标在公式复制时会自动地进行调整。例如“A1”和“P78”都是相对坐标。
- 绝对坐标：绝对坐标在公式复制时不作调整，它们在行列号前加“$”来表示。例如“$A$1”和“$P$78”都是绝对坐标。
- 混合坐标：混合坐标是介于相对坐标和绝对坐标之间的一种坐标。它的行坐标是绝对的而列坐标是相对的或者反之。例如“$A1”和“P$78”都是混合坐标。

表格单元的坐标可直接输入，也可采用形象输入。在采用形象输入时，可用〈F4〉键更改坐标的性质。

2．表格区域

在第一节中已经介绍过，1-2-3的表格区域可以有三种形式，当时使用的表格区域是相对区域。同样，绝对区域和混合区域也有三种形式：

① 单个表格单元，例如A1，$IV2048等。

② 行或列的部分，例如A1..A10，A$1..B$1等。

③ 矩形区，例如B5..E9等。

另外，1-2-3还允许对区域命名，在使用时按名引用。对区域命名可采用／RNC命令来完成。例如，要把区域A1..A10命名为“DEPOSITS”，可打入命令：

／RNCDEPOSITS〈CR〉A1.A10〈CR〉

1-2-3的区域名由1～15个字符组成，但必须以字母开头。

3．运算符

1-2-3的运算符分为三类，即算术运算符、关系运算符和逻辑运算符。在公式的计算过程中，依照其优先级大小顺序进行计算。运算符的优先级可用加圆括号的方法改变。

表7-1以运算符的优先级为序列出了1-2-3的所有运算符。

表7-1　1-2-3的运算符及其优先级

运　算　符	功　能	优先级
（，）	提高运算符的优先级	8
^	乘　幂	7
－，＋	单目取负、取正运算	6
＊，／	乘、除	5
＋，－	加、减	4
=,<,<= >,>=,<>	关系运算符	3
#NOT#	逻辑非	2
#AND#,#OR#	逻辑与、逻辑或	1

4．函数

1-2-3的函数，其格式如下：

@〈函数名〉(〈参数〉{，〈参数〉})

其中参数可以是单个常量、变量或公式，也可以是一个区域。1-2-3的函数可在公式的任意位置处出现。

下面分类介绍1-2-3的函数。在介绍中，使用V表示单一参数，使用L表示区域参数。

（1）商用函数

① 偿还函数@PMT(V1，V2，V3)

设V1为贷款本金，V2为利率（复利），V3为分期偿还的期数，则@PMT函数用来计算每期应偿还的金额。@PMT函数的计算公式为：

$$@PMT = 贷款本金 \times \frac{利率}{1-(1+利率)^{-n}}$$

其中 n 为偿还期数。

例如，某工厂为了扩大再生产，向银行贷款 1000 万元，准备分 12 个月还清。设月利率为 0.1（复利），问每月应偿还多少万元？

	A	B
1	Principal	1000
2	Interest	.1
3	Term	12

利用@PMT 函数计算如下：

@PMT（1000，0.1，12）＝146.76332

@PMT（B1，B2，B3）＝146.76332

所以，每月应偿还 146.76332 万元。

② 可贷款函数@PV（V1，V2，V3

与@PMT 函数相反，@PV 函数用来计算固定偿还能力下可贷款的总数。函数中的 V1 为每期的偿还能力，V2 为利率（复利），V3 为偿还期数。该函数的计算公式为：

$$@PV = 每期偿还额 \times \frac{1-(1+利率)^{-n}}{利率}$$

其中 n 为偿还期数。

例如，某工厂每月的偿还能力为 100 万元，准备贷一笔款，利率为 0.1（复利），分12个月还清。问可贷款多少万元？

可贷款的总数可直接利用 @PV 函数算出：

@PV（100，0.1，12 ＝681.36918 （万元）

③ 可得款函数@FV（V1，V2，V3）

设每期存款 V1 元，利率为 V2（复利），共存款 V3 期，那么期满后共可得款（连本带利）多少呢？这可利用@FV函数算出，该函数的计算公式为：

$$@FV = 每期存款额 \times \frac{(1+利率)^{n}-1}{利率}$$

其中 n 为存款期数。

例如，某单位进行零存整取，每月存入 1000 元，利率为 0.1（复利），共存 12 个月，年底可得款：

@FV（1000，0.1，12）＝21384.28（元）

④ 现得净利函数@NPV（V，L）

设向银行贷款 I 元，以后逐月按计划偿还 L1、L2、…、Ln元，设利率为 V，则 n 个月后贷款与还款的差额为：

$$Y = \sum_{i=1}^{n} L_i \times (1+V)^{n-i} - I \times (1+V)^{n}$$

等式两边同除（1+V）^{n}并移项得：

$$Y/(1+V)^n+I=\sum_{i=1}^{n} L_i/(1+V)^i$$

上式中的等号右边即为@NPV函数的计算公式。从等式左边可以看出@NPV的值为银行收回的本金和货款时即可得到的净利之和。

在@NPV函数的参数中，V为存放利率的表格单元，L为逐月还款的数量所在的区域，这个区域只能是一行或一列的一部分。

例如，某工厂向银行借贷1000元，随后的四个月中每个月各还款400元，利率为10.5%。产生的@NPV值为1254.34元。银行得到的净利为：

1254.34－1000＝254.34（元）

⑤ 利率估算函数 @IRR（V，L）

该函数可根据借款和还款的数量和利率估计值来计算利率，以保证收支平衡。@IRR函数的参数V为利率估计值，L中存放借款和存款数。

例如，某顾客向银行借款1000元，计划分四个月还清，这四个月的还款预计分别为500元、400元、200元和100元，现要求计算出利率使借还款平衡。

先在A1··A6中输入借还款数，再在B2中填入利率估计值0.14，最后在B6中填入公式“@IRR（B2，A2·A6）”。经过多次迭代，计算出利率，如图7—4所示。

B6: @IRR(B2,A2..A6) READY

	A	B	C	D	E
1	Payments	Guess			
2	-1000	0.14			
3	500				
4	400				
5	200				
6	100	0.102212			
7					
8					
9					
10					

图 7—4　利率估算

（2）统计函数

① 求和函数　@SUM（L）

该函数对列出的表格单元进行求和，求和时空白单元略过，标号单元的值理解为0。

② 计数函数　@COUNT（L）

该函数对所列出的所有非空白表格单元进行计算。

③ 平均值函数　@AVG（L）

该函数可用来求平均值，由下式算出：

@AVG（L）＝@SUM（L）／@COUNT（L）

④ 方差函数 @VAR (L)

该函数用来对所列出的各项求方差，它的计算公式为：

$$@VAR=\frac{n\times\sum x^2-(\sum x)^2}{n^2}$$

其中 n=@COUNT (L)

⑤ 标准偏差函数 @STD (L)

该函数可用来计算标准偏差，它的计算公式为：

@STD (L) =@SQRT (@VAR (L))

⑥ 最大值函数 @MAX (L)

⑦ 最小值函数 @MIN (L)

(3) 算术函数

① 绝对值函数 @ABS (V)

该函数用来取 V 的绝对值。

② 平方根函数 @SQRT (V)

@SQRT (V) 用来求 V 的平方根。

③ 指数和对数函数

该组函数有以下三个：

* @EXP (V)　e 的 V 次方，V 大于 230 时，值为 "ERR"。
* @LOG (V)　常用对数函数。
* @LN (V)　自然对数函数。

④ 三角函数

1-2-3 提供的三角函数和反三角函数共有下列七个：

* @SIN (V)　正弦函数。
* @ASIN (V)　反正弦函数。
* @COS (V)　余弦函数。
* @ACOS (V)　反余弦函数。
* @TAN (V)　正切函数。
* @ATAN (V)　反正切函数，V 取值范围为 $-\pi/2\sim\pi/2$。
* @ATAN2 (V1, V2)　V2／V1 的反正切，取值范围为 $-\pi\sim\pi$。

⑤ 圆周率函数 @PI

该函数的值为：3.14159265358。

⑥ 取整函数 @INT (V)

取 V 的整数部分。

⑦ 舍入函数 @ROUND (V1, V2)

对 V1 进行舍入，小数点后保留 V2 位。

⑧ 取模函数 @MOD (V1, V2)

对 V1 进行模 V2 计算，计算公式为：

@MOD (V1, V2) =V1− (V2×@INT (V1／V2))

⑨ 随机数生成函数 @RAND

产生 0～1 间的随机数。

（4）逻辑函数

① 逻辑常量函数

* @TRUE　值为“TRUE”。
* @FALSE　值为“FALSE”。

② 判错函数

该组函数可用来判断表格单元的值是否为“NA”或“ERR”，它有以下两个函数：

* ISNA (V)　如果 V 的值为“NA”，则函数值为真。
* ISERR (V)　如果 V 的值为“ERR”，则函数值为真。

③ 条件函数 @IF (V1, V2, V3)

如果 V1 为真，函数取 V2 的值，否则取 V3的值。

（5）日期函数

该组函数可用来计算从 1900 年 1 月 1 日以来的日期，下文中的“天数”表示从 1900 年 1 月 1 日起到指定日期的总天数。这组函数有以下五个：

* @TODAY　计算当天的天数（从 1900 年 1 月 1 日算起）。
* @DATE (年、月、日)　计算初始日期到指定日期的天数。
* @YEAR (天数)　给出指定天数的年号。
* @MONTH (天数)　给出指定天数的月份。
* @DAY (天数)　给出指定天数的日期。

（6）辅助函数

① 选择函数 @CHOOSE (V, V0, V1, …, Vn)

该函数取 V0…Vn 中的第 V 个值为函数值。

例如：

@CHOOSE (3, 42.6, −4.2, B1, @NA, 3.4) =NA

② 按列查表函数 @VLOOKUP (V1, L, V2)

这个函数通过查自变量来取对应函数表的值，查表的方式为按列进行。参数表中的 V1 为待查的自变量，L 为函数表区域，V2 为函数值列与自变量列的间距。下面举例说明使用@VLOOKUP 函数查表的方法。

例7—1　查函数表

设已存在一张函数表，如图 7—5 所示。现利用 @VLOOKUP 函数查表，打入

@VLOOKUP (3, A1·C4, 1)〈F9〉

表示查自变量为 3 时的函数值，函数值从第 B 列中查出。这时屏幕上显示出结果：“14.42”。

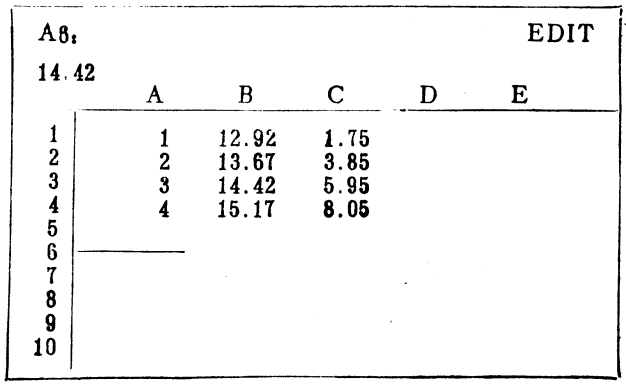


图 7—5　查函数表

用同样方法可以查出：

@VLOOKUP (2, A1·C4, 2) =3.85

@VLOOKUP (2, A1·C4, 1) =13.67

③ 按行查表函数 @HLOOKUP (V1, L, V2)

这个函数类似于 @VLOOKUP 函数，也用来查表，但它的函数表是按行存放的，查找也按行进行。

④ 出错模拟函数

* @NA　产生一个“NA”值。
* @ERR　产生一个“ERR”值。

(7) 数据库统计函数

数据库统计函数的格式为：

@〈函数名〉(L1, V, L2)

其中 L1 为函数的输入区域，用来表示数据库文件的数据区；V 表示参加计算的数据位于输入区域的第 V+1 列；L2 为条件区域，用来表明参加计算的数据所应满足的条件。

数据库统计函数的计算公式类似于普通的统计函数，这里不再给出。下面仅列出所有 1-2-3的数据库统计函数及其含义：

@DCOUNT (L1, V, L2)	计数函数
@SUM (L1, V, L2)	求和函数
@DAVG (L1, V, L2)	平均值函数
@DSTD (L1, V, L2)	标准差函数
@DVAR (L1, V, L2)	方差函数
@DMAX (L1, V, L2)	最大值函数
@DMIN (L1, V, L2)	最小值函数

例7—2　库存数据库文件的统计

设已存在一个小型的库存数据库文件，如图 7—6 所示。现要对这个数据库文件中的数

据作统计。

F2: (T)+A2>1001 READY

	A	B	C	D	E	F
1	Item	Quantity	Type		Type	Item
2	1001	2	A		A	+A2>1001
3	1002	4	B			
4	1003	8	C			
5	1004	9	B			
6	1005	1	C			
7	1006	0	B			
8	1007	15	A			
9	1008	11	C			
10	1009	7	A			
11	1010	6	A			
12						

图7—6　库存数据库文件

首先命名输入区域和条件区域。其操作过程如下：

/RNCDBAREA〈CR〉A1.C11〈CR〉　{输入区域 A1..C11 命名为DBAREA}

/RNCCRIT1〈CR〉E1.E2〈CR〉　{把 E1..E2 作为第 1 个条件区 CRIT1}

/RNCCRIT2〈CR〉F1.F2〈CR〉　{把 F1..F2 作为第 2 个条件区 CRIT2}

接着使用数据库统计函数对数据库中的数据进行统计。满足条件1（类型为A）的项号（Item）平均值为：

@DAVG(DBAREA, 0, CRIT1)
=@AVG(1001, 1007, 1009, 1010)
=1006.75

满足条件2（A2>1001）的数量（Quantity）最大值为：

@DMAX(DBAREA, 1, CRIT2)
=@MAX(4, 8, 9, 1, 0, 15, 11, 7, 6)
=15

第四节　1-2-3 的命令树

1. 1-2-3 命令系统的功能与特点

1-2-3 具有较强的命令系统，它可以完成下列功能：

- 复制、移动和删除表格中的数据。
- 内存的工作区与磁盘交换数据。
- 打印报表。
- 绘制商用统计图。
- 管理数据库文件。

1-2-3 的命令系统设计得较为合理，它既使用方便又功能齐全，使用的方法也有多种。1-2-3 的命令系统主要有以下特点：

* 命令系统采用多层次的分层结构，每个层次都使用“菜单”结构。
* 使用命令的方式为交互方式，并为用户提供缺省回答。
* 在使用“菜单”时，可直接输入命令的第 1 个字母，也可以移动光标进行选择。

要启动命令系统，则打入斜杠“／”，这时状态区的第 2 和第 3 行显示如下信息：

Worksheet Range Copy Move File Print Graph Data Quit
Global, Insert, Delete, Column-Width, Erase, Titles, Window, Status

其中第 2 行为主命令“菜单”，第 3 行为光标所指定的命令的参数提示或子命令“菜单”。用户可使用两种方法选择命令，其一是输入命令名的第 1 个字母，其二是移动光标到要选择的命令处并打入〈CR〉键。为叙述方便，下文中主要使用第 1 种方式。例如，／WG 表示／Worksheet Global 命令。

2．宏命令

在使用 1-2-3 命令的过程中，常会遇到多次使用同样的一串命令，1-2-3 能象 dBASE Ⅱ 的应用程序一样，对这些命令一次定义后多次使用，1-2-3 提供的宏命令就是用来完成这一功能的。

表 7-2　宏命令中的特殊键

特殊键	宏命令的表示	含　义
‹CR›	～	回车换行键
↑	{Up}	光标上移
↓	{Down}	光标下移
←	{Left}	光标左移
→	{Right}	光标右移
‹Home›	{Home}	光标回初始位置
‹End›	{End}	光标至行末
‹PgDn›	{PgDn}	光标下移一页
‹PgUp›	{PgUp}	光标上移一行
‹Esc›	{Esc}	终止当前命令
‹Del›	{Del}	删除键
‹BS›	{Bs}	退格键
‹F2›	{Edit}	进入编辑状态
‹F3›	{Name}	显示区域名“菜单”
‹F4›	{Abs}	更改坐标的绝对性
‹F5›	{GoTo}	参数定位
‹F6›	{Window}	窗口定位
‹F7›	{Query}	重新检索
‹F8›	{Table}	重计算数据表
‹F9›	{Calc}	重新计算公式
‹F10›	{Graph}	重新作图

宏命令由一串 1-2-3 的命令及参数组成，它由用户自行定义，并以一个字母为名，在使用时只要按下〈Alt〉键并打入该字母即可。

（1）宏命令的定义

要定义宏命令，首先在表格中开辟一个宏命令区，并以标号的形式输入 1-2-3 的命令串。在命令串中出现的特殊键，使用表 7-2 的对应符号或字符串表示。

宏命令区可由一行或多行命令串组成，在命令串输入完毕后，用 /RN 命令命名。宏命令名由一个反斜杠和一个字母组成，例如

\A，\C，\Z

都是合法的宏命令名。如果宏命令命名为\0，则表示该宏命令为自动启动宏命令，当装入该表格后，会立即执行这条宏命令。但这个宏命令定义后不会立刻执行，须下次装入后才会自动执行。如要执行，可直接用 Alt-0 调用。

例如，要在第 1 行中写入 LOTUS 公司的名称，通常的操作过程如下：

〈F5〉A1〈CR〉Lotus Development Corp〈CR〉

这个过程写成宏命令，则为：

{GoTo} A1~Lotus Development Corp~

设这行宏命令填在 A10 中，则可输入命令：

/RNC\A〈CR〉A10·A10〈CR〉

把这个宏命令命名为\A。

（2）宏命令的执行和终止

对已定义的宏命令可随时引用，调用的方法是按下〈Alt〉键后再打入命名的字母即可。例如上面定义的宏命令“\A”，可打入“Alt-A”来调用。

对已定义的宏命令，可进行单步执行。单步执行多用于对宏命令进行调试。打入 Alt—〈F1〉后，1-2-3 就进入单步状态，这时执行宏命令，每做一步都会产生暂停。要退出单步状态，再输入一次 Alt-〈F1〉。

如果希望在宏命令的执行过程中退出，仅需打入 Ctrl-Break 即可。

此外，在宏命令中还可用{？}引起宏命令的执行暂停，用户可从键盘输入数据，输入完毕后宏命令将继续执行。也可输入 Ctrl-Break 终止宏命令的执行。

（3）宏命令的控制结构

1-2-3 提供了一条/X 命令用来控制宏命令中命令的执行顺序。这条/X 命令仅能用于宏命令中，不能直接从键盘输入执行。

/X命令主要有以下几条子命令：

* /XI〈条件〉~…　　条件子命令（IF…THEN）。
* /XG〈位置〉~　　转移子命令（GOTO）。
* /XC〈位置〉~　　转子子命令（GOSUB）。

- /XR　返回主程序（RETURN）。
- /XQ　终止宏命令的执行（QUIT）。
- /XM〈位置〉～　处理用户定义的“菜单”。
- /XL〈数据〉～〈位置〉　在状态区显示〈数据〉，并从键盘上输入一个标号到指定位置。
- /XN〈数据〉～〈位置〉　在状态区显示〈数据〉，并从键盘上输入数值到指定位置。

3. 1-2-3命令树

下面逐条介绍1-2-3命令的功能，并给出命令的结构。对于各命令下的子命令只作简单的说明，读者若要深入了解这些子命令的详细含义，请参阅1-2-3的用户手册。关于1-2-3命令的具体使用方法将在第五节中结合例子进行介绍。

（1）表格管理命令（/Worksheet）

① 命令的功能

/W命令主要用来定义表格的格式、划分标题区和窗口、对表格进行大幅度的编辑以及修改缺省的设备参数。/W命令是1-2-3中使用频率最高的命令。

如果不用/W命令定义表格的格式，1-2-3将取缺省格式作为表格的格式。1-2-3的缺省格式为：

- 数值量为自然格式；标号的缺省前导符为（’），表示向左靠齐。
- 列宽为9（可定义的范围为1～72）。
- 重计算的方式为自动。
- 禁止对表格单元加保护。

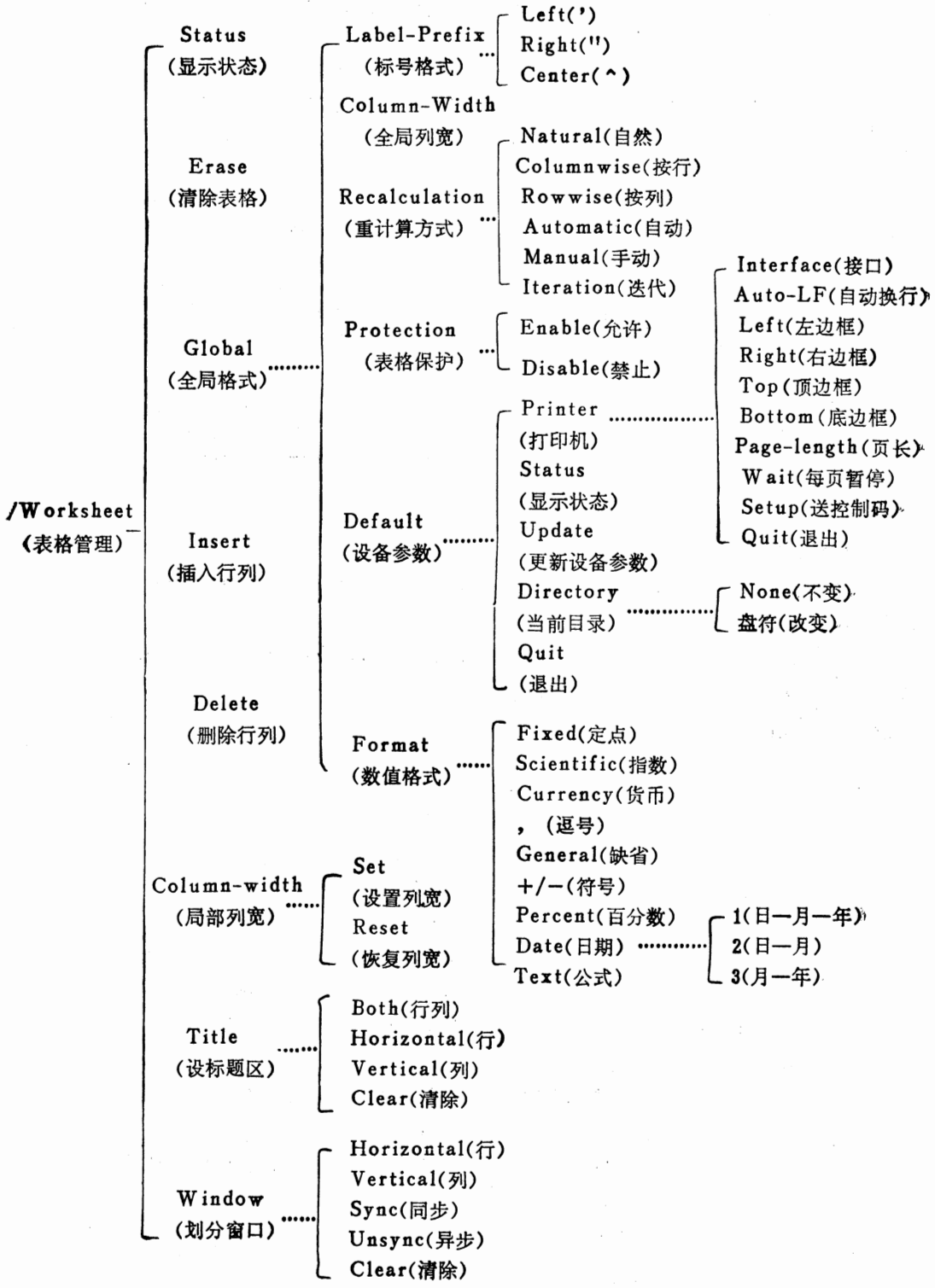
② 命令的结构（见下页）

③ 说明

- /WT命令用来把光标以上的若干行、光标以左的若干列或两者指定为标题区标题区固定于屏幕的表格显示区中。
- /WW命令可用来在表格显示区上开窗口，以同时显示表格的几个区域。其子命令“S”表示移动光标时，几个窗口同步地移动。
- /WGF命令用来定义表格单元的全局数值格式，可定义的各类格式及其含义如表7-3所示。

表7-3　数值量的格式

格　式	含　义	格　式	含　义
Fixed	定点数（小数位数可为0～15）	+/-	符号格式，输出指定个数的“+”或“-”
Scientific	科学记数格式(指数格式)		
Currency	货币格式($××，×××.××)	Percent	百分数格式,数据乘100并加百分号
,	逗号格式(××，×××.××)	Text	显示表格单元中的公式
General	缺省格式(自然格式)	Date	日期格式


/Worksheet
(表格管理)
Status
(显示状态)
Erase
(清除表格)
Global
(全局格式)
Insert
(插入行列)
Delete
(删除行列)
Column-width
(局部列宽)
Title
(设标题区)
Window
(划分窗口)
Label-Prefix
(标号格式)
Left(')
Right(")
Center(^)
Column-Width
(全局列宽)
Recalculation
(重计算方式)
Natural(自然)
Columnwise(按行)
Rowwise(按列)
Automatic(自动)
Manual(手动)
Iteration(迭代)
Protection
(表格保护)
Enable(允许)
Disable(禁止)
Default
(设备参数)
Printer
(打印机)
Interface(接口)
Auto-LF(自动换行)
Left(左边框)
Right(右边框)
Top(顶边框)
Bottom(底边框)
Page-length(页长)
Wait(每页暂停)
Setup(送控制码)
Quit(退出)
Status
(显示状态)
Update
(更新设备参数)
Directory
(当前目录)
None(不变)
盘符(改变)
Quit
(退出)
Format
(数值格式)
Fixed(定点)
Scientific(指数)
Currency(货币)
, (逗号)
General(缺省)
+/-(符号)
Percent(百分数)
Date(日期)
1(日—月—年)
2(日—月)
3(月—年)
Text(公式)
Set
(设置列宽)
Reset
(恢复列宽)
Both(行列)
Horizontal(行)
Vertical(列)
Clear(清除)
Horizontal(行)
Vertical(列)
Sync(同步)
Unsync(异步)
Clear(清除)

* /WGD命令可以根据机器的硬件设置对1-2-3进行重组，建立一个文件123.CNF，用来表示设备的配置状态。1-2-3的缺省设备状态如表7-4所示。

表7-4　1-2-3的缺省设备参数

项　　目	缺省值	含　　义
Interface	Parallel	打印机的接口为并行口
Auto Line-Feed	No	不自动加<LF>符
Left margin	4	左边框为第4列
Right margin	76	右边框为第76列
Top margin	2	上边框为第2行
Bottom margin	2	页末空2行
Page length	66	页长为66行
Pause at end of page	No	每页输出后不暂停
Setup string	(none)	打印机控制字符串为空
Directory	B:\	当前目录为B盘的根目录

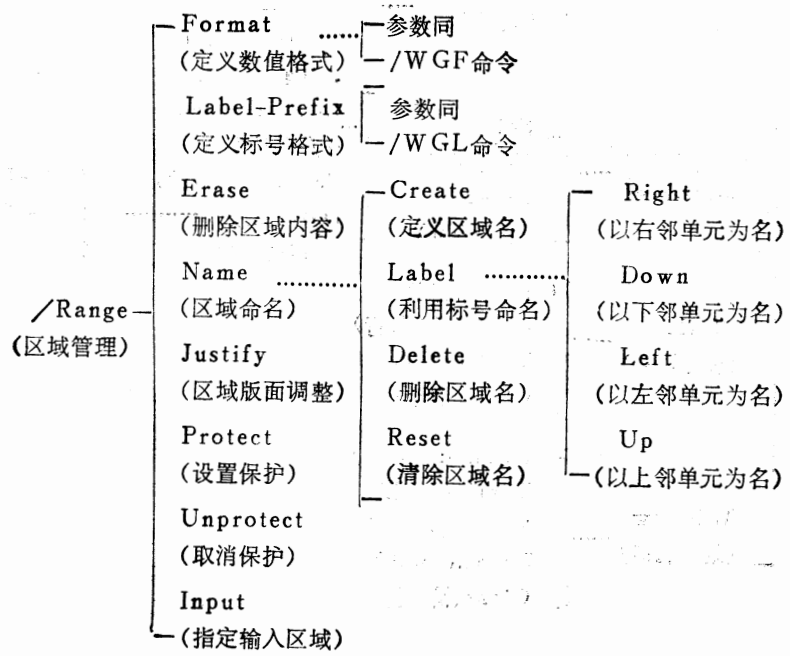
(2) 区域管理命令(/Range)

① 命令的功能

/R命令可用来处理一个表格区域，例如给表格区域命名、定义局部数据格式和删除单元内容等。

/R命令是十分有用的，它处理的区域可作为函数的参数，也可作为表格的一部分打印和存盘，还可以作为数据库的查询条件区域。

② 命令的结构



③ 说明

* /RF 命令的参数同 /WGF 命令基本一样，只是多了一种参数“Reset”（恢复为全局格式）。
* /RNL 命令用来以指定相邻单元中的标号为名命名表格单元。其参数 R、D、L 和 U 表示相邻单元的位置。
* /RNR 命令用来删除所有区域名。
* /RJ 命令对指定区域中的字符串，以单词为单位进行版面调整。
* /RI 命令定义一个输入区域，在这个区域中，光标仅能在非保护单元间移动。

（3）区域复制命令（/Copy）

该命令可用于复制指定区域中的数据或公式。该命令下没有子命令。使用这条命令复制区域时，仅需给出源区域范围和目的区域范围，1-2-3 就会自动地进行复制。

在复制公式的过程中，1-2-3 会自动调整公式中使用的相对坐标和混合坐标，调整时根据目的区域的地址进行。

（4）区域移动命令（/Move）

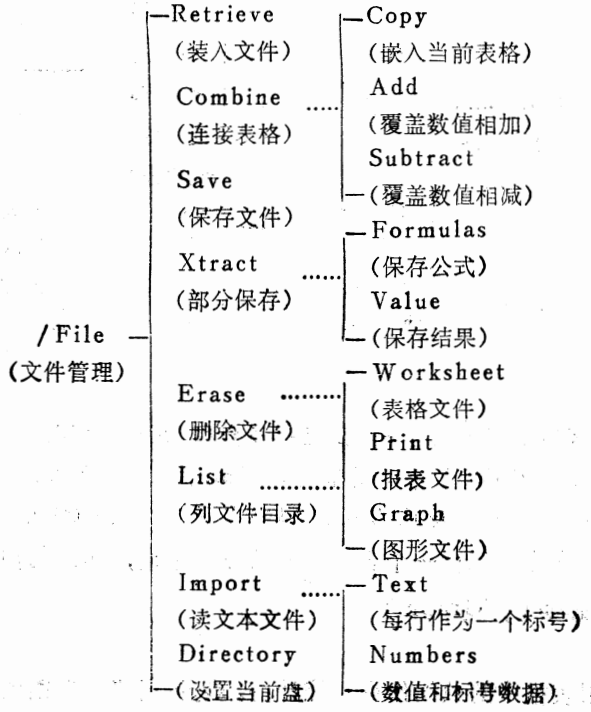
/M 命令的功能为移动一块表格区域，移动后区域中的公式会自动调整，且源区域将被清除。在使用/M 命令时，源区域和目的区域大小和形状应该一致。/M命令下没有子命令。

（5）文件管理命令（/File）

① 命令的功能

这条命令主要负责表格与磁盘交换数据。它可以保存、装入、连接和删除表格文件，也可以列目录和更改当前盘。

② 命令的结构



③ 说明

* /FCC 命令把指定的文件直接嵌入当前表格，原来对应单元的内容被替换；/FCA 表示把指定表格文件与当前表格对应数值单元相加后嵌入；/FCS表示对应单元相减后嵌入。

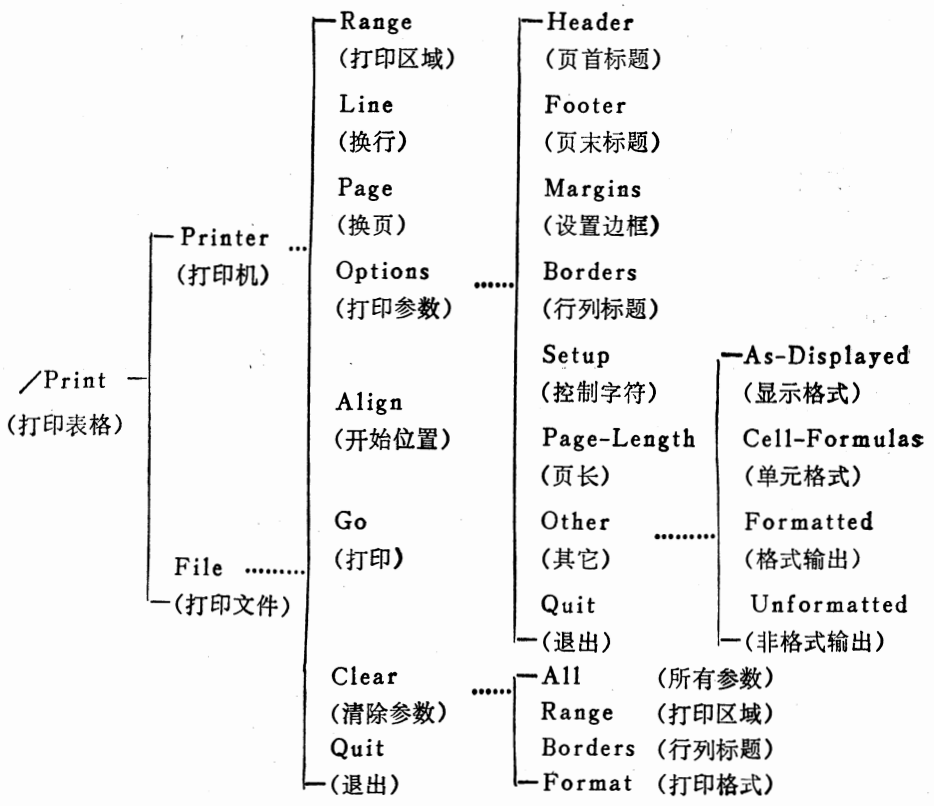
* /FI 命令可读入文本文件（类型名为·PRN）。/FIT 以行为单位读入标号替代原表格中的内容；/FIN 以单元为单位读入数值或标号数据，标号须用双引号括起来。

（6）打印表格命令（/Print）

① 命令的功能

/P 命令用来生成报表并在打印机上打印出来，或以可打印文件的形式保存在磁盘上。

② 命令的结构



③ 说明

* 如果打印区域宽于或长于一页，1-2-3会自动分页打印。
* /PPOS 和 /PFOS 命令可用来设置打印机状态（字体和行距等）。在这条命令中，控制码用“\”及其 ASCII 码来表示，例如 Ctrl-O 表示为 \015。
* /PPOOC 命令以每行一个表格单元的方式输出，输出的内容类似于当前单元行。

（7）作统计图命令（/Graph）

① 命令的功能

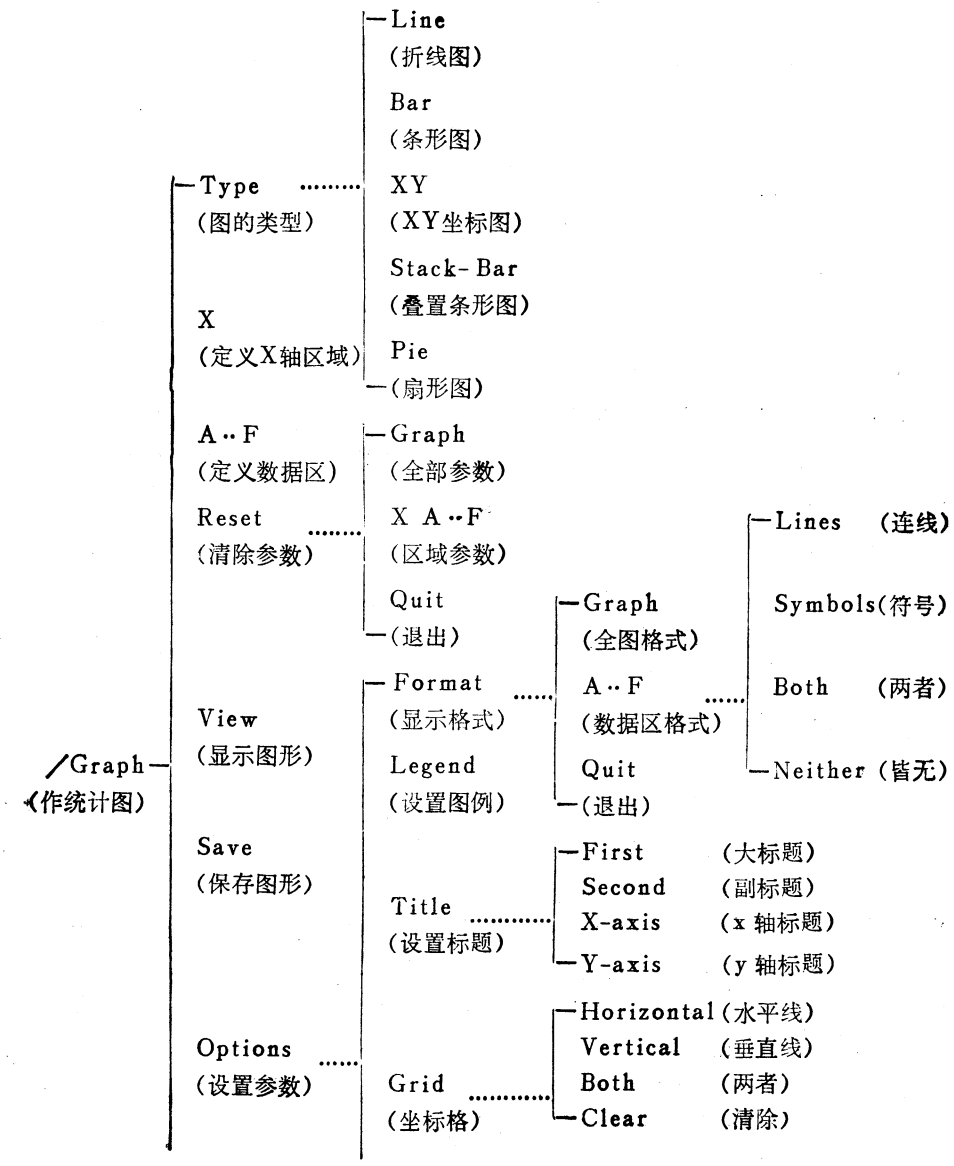
这条命令可利用表格中的数据作商用统计图。它可用来设置图的类型、图的参数和该图

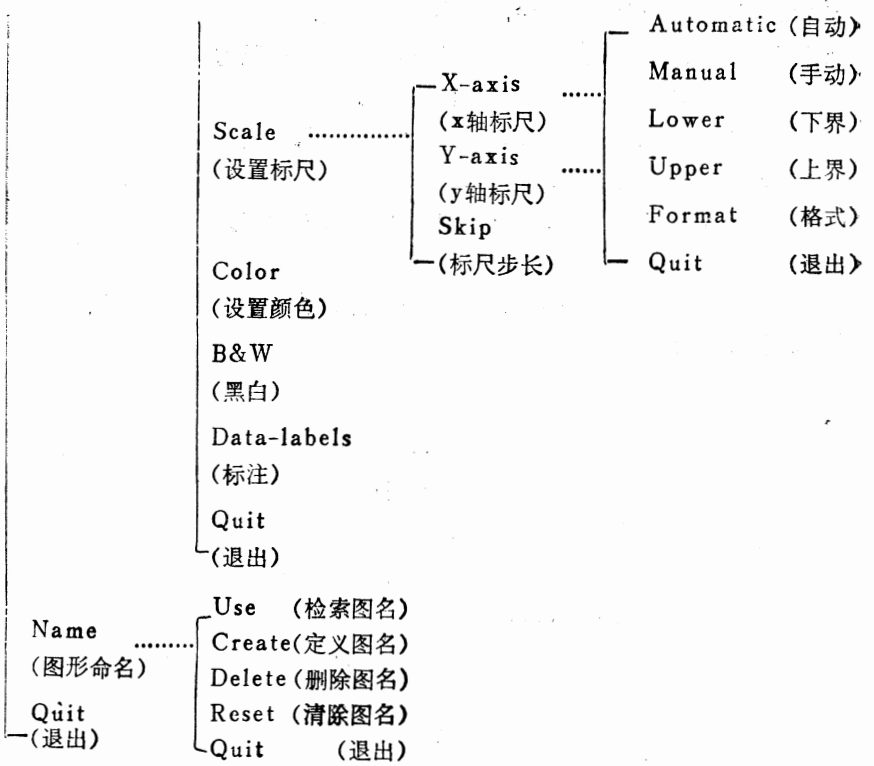
所表示的区域，显示和保存欲绘制的商用统计图。

使用 1-2-3 的绘图系统绘制商用统计图的过程一般为：

* 选择图的类型并设置图的有关参数。
* 设置 X 轴及最多六组统计图的数据区域。
* 从显示器输出图形。
* 修改数据并利用〈F10〉重新显示所作的统计图。
* 把图形文件保存到盘上。
* 返回管理系统 LOTUS 并调用图形打印程序打印图形文件。

② 命令的结构





③ 说明

* /GX 命令用来指明表示 X 轴的数据区域。

* /GOD 命令可把指定的区域中的内容作为数据区域（A～F）的标注。

（8）数据处理命令（/Data）

① 命令的功能

/D 命令主要用来完成 1-2-3 的数据库管理功能。它可对数据库文件中的数据进行排序和检索，也可以利用数据库文件中的数据建立预测表，来对任意变化的条件进行“灵敏”分析。

② 命令的结构（见下页）

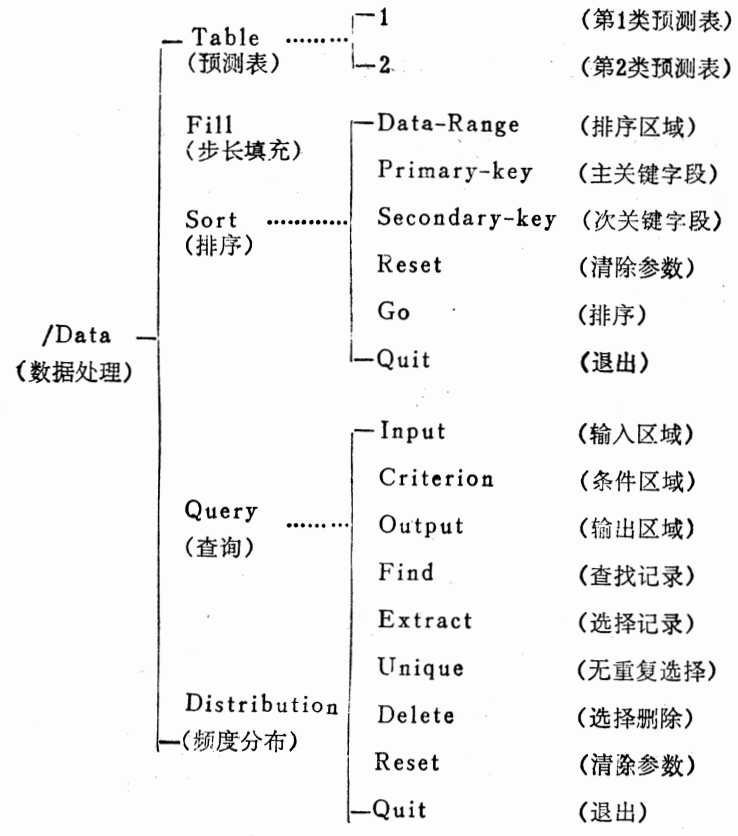
③ 说明

* 1-2-3的数据库文件由一块表格区域组成，这个区域的第1行为记录中各字段的字段名，其余各行为该数据库文件的记录。每个记录又分为若干字段，每个字段占用一个表格单元。

* 定义字段类型、输入数据和修改数据均借用表格处理的相应命令来完成。例如，要在数据库文件中插入一个记录，则可用 /WIR 命令插入一个空行，再填入记录值即可。

* 数据库文件的分类和排序操作由 /DS 命令来完成。排序的过程通常为：指定数据库文件所在的区域（Data—Range），选择排序的主次关键字段，执行排序（Go）。

* 检索命令 /DQ 则用来按条件对数据库中的记录进行选择。1-2-3 提供了四种检索操作：查询、选择、无重复选择和选择删除。它们一般都涉及三个区域：存放数据库文件的输入区域、存放检索条件的条件区域和存放检索结果的输出区域。



* 条件区域中的每个单元可存放一个检索条件，同一行中的条件为“与”的关系，同一列中的条件为“或”的关系。

* 1-2-3数据库文件的统计操作由第三节中介绍的数据库统计函数来完成。这组函数类似于1-2-3表格公式中使用的统计函数，但其运算对象由数据库文件中选出。

* 1-2-3数据库的另一个特点是可以方便地建立预测表（／DT命令）。预测表常用于作灵敏性分析，即对某一公式或若干公式中的自变量（输入变量）进行试探，并分析公式值的变化情况。例如，可以利用预测表来分析和预测利率变化对某工厂可贷款数额的影响。

* 1-2-3的预测表分为两类，第1类预测表（／DT1）的输入变量有一个，而相关的公式可有多个；第2类预测表（／DT2）的输入变量有两个，而相关的公式仅有一个。

* ／D还有两条辅助子命令。一条是步长填充命令／DF，它常用于为／DT命令准备输入变量的值。另一条是频度分布命令／DD，它是数据库统计函数的补充，用以增强1-2-3的统计能力。

（9）退出命令（／Quit）

这条命令用来退出1-2-3，返回管理系统LOTUS或返回MS-DOS。仅当打入“Y”后，／Q命令才生效。

第五节　1-2-3 应用举例

前面几节主要介绍了 1-2-3 的基本内容，本节则给出 12 个关于 1-2-3 的应用实例。通过这些例子，一方面是对前面的内容作些补充，另一方面也是为读者在使用 1-2-3 时提供示范。

1．表格的设计与制作

例7—3　年度收支平衡表

（1）功能　设计一张一年中四个季度的收支平衡表。

（2）命令介绍

① 表格管理命令／W

／W 命令主要用来设置表格的格式，它的子命令有以下几个：

* Global　定义表格的全局格式。
* Insert　插入若干行或若干列，打入“C”表示插入空列，打入“R”表示插入空行。
* Delete　删除表格中的若干行或若干列，打入“C”表示删除指定的列．打入“R”表示删除指定的行。
* Column—Width　定义当前单元所在列的列宽。其子命令“S〈列宽〉”用来设置指定的列宽；“R”用来把当前列恢复成全局列宽。
* Erase　清除整个表格，打入“Y”后才有效。
* Titles　设置表格的标题区。
* Windcw　在表格区中开窗口。
* Status　显示表格的全局格式。

② 复制区域命令／Copy

这条命令可用来复制一个指定的表格区域。输入“／C”后，1-2-3 要求用户输入源区域的左上角和右下角坐标，并要求输入目的区域的左上角坐标。输入完毕后，1-2-3 立即就进行复制。1-2-3允许源区域与目的区域重叠。

／C 命令还可复制区域中的公式。为了对复制的公式进行调整，1-2-3 引入了相对坐标和绝对坐标的概念。1-2-3 把普通的坐标都认为是相对坐标，在复制时自动地进行调整。绝对坐标则用行列坐标前加“$”表示，它们在复制时不进行调整。采用形象法输入坐标时．可用〈F4〉改变坐标的性质。

（3）操作过程

① 定义表格格式，使 A、C、E、G、I、K 和 M 列用来画竖线。

```
／WEY                  {清除内存中的表格}
／WCS1〈CR〉            {定义第 A 列列宽为 1}
→→／WCS1〈CR〉          {定义第 C 列列宽为 1}
→→／WCS1〈CR〉          {E 列宽＝1}
→→／WCS1〈CR〉          {G 列宽＝1}
```

→→／WCS1〈CR〉

→→／WCS1〈CR〉

→→／WCS1〈CR〉

② 输入表格标题

〈F5〉E1〈CR〉REVENUES－EXPENSES TABLE〈CR〉 {使用参数定位到E1填入标题}

③ 画表格的格子

〈F5〉A2〈CR〉＼－↓｜↓＼－〈CR〉／CA3·A8〈CR〉A5·A10〈CR〉 {输入第A列中的边线}

／CA2·A10〈CR〉B2·M2〈CR〉 {把第A列复制到整个表格区域中}

这时的屏幕如图7—7所示。其中有些单元中复制了不必要的竖线，这些竖线会在填入内容后自动消失。

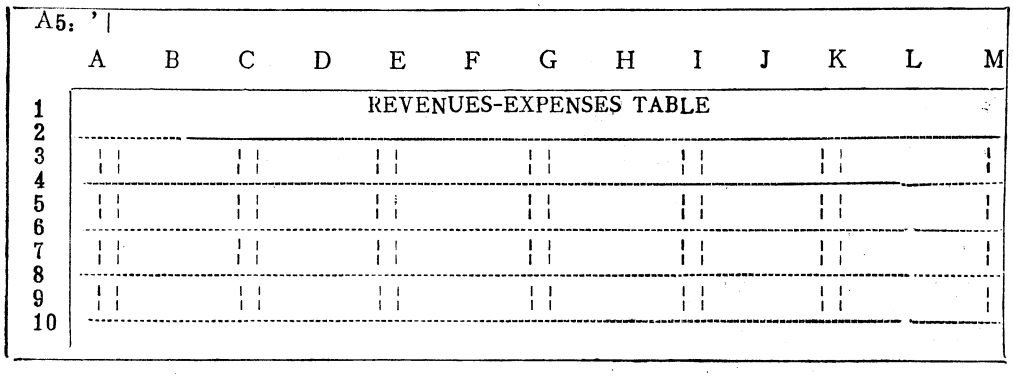


图7—7 表格中的格子

④ 填入表格内容

先在第3行中填入小标题，再在D5··J7区域中分别填入四个季度的收支原始数据，最后，在L5··L9中填入下列计算公式：

L5 全年收入：＋D5＋F5＋H5＋J5

L7 全年支出：＋D7＋F7＋H7＋J7

L9 结余：＋L5－L7

这时屏幕上显示出的结果如图7—8所示。

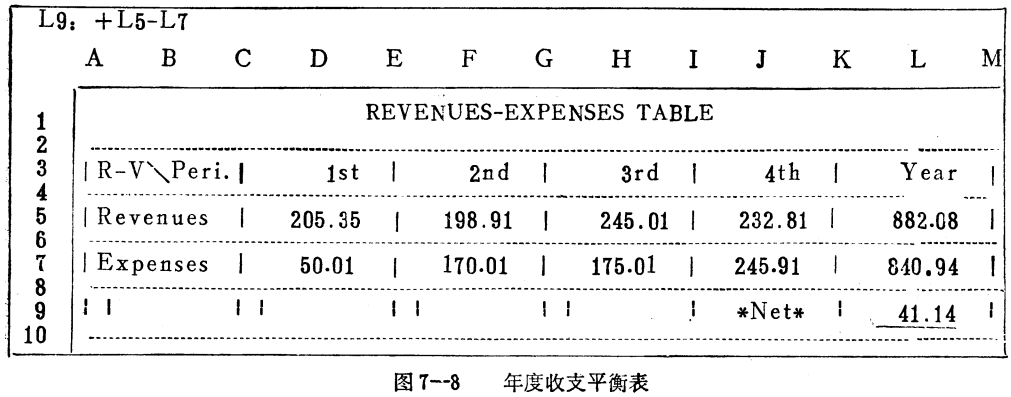


图7—8 年度收支平衡表

2· 表格的维护与管理

例7—4　表格的装入、修改与保存

（1）功能

假设B盘上有一个1 2-3表格文件LESNA-1·WKS，把它装入内存进行修改，修改后再存放到盘上（读者可按书中要求予先建立一个这样的表格文件）。

（2）/F命令介绍

/F命令主要用来完成表格与外存交换数据。它有下列子命令：

* Retrieve　把表格文件装入内存。
* Save　把内存中的表格保存到盘上。
* Combine　把指定的表格文件与当前表格合并。
* Xtract　把指定的表格区域保存到盘上。
* Erase　删除盘上指定的表格文件（·WKS文件）、打印文件（·PRN文件）或统计图文件（·PIC文件）。
* List　显示当前工作盘上的目录。
* Import　从标准系统文本文件中读入若干字符串，填入当前表格中。
* Directory(或Disk)　显示或设置当前工作盘。

（3）操作过程

① 装入LESNA-1·WKS并查看它的内容

/FRLESNA-1〈CR〉　{装入LESNA-1·WKS}

这时的屏幕如图7—9所示。其中单元行的内容为“A1：'Part List”，表示当前单元为A1，其内容为“Part List”，格式为向左对齐（'）。

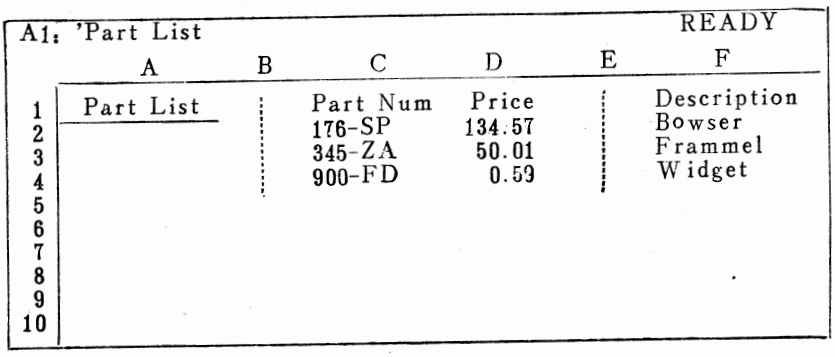


图7—9　装入LESNA-1.WKS后的画面

〈F5〉I88〈CR〉　{把单元光标定位到I88}

这时，屏幕上出现的画面如图7—10所示。

接着把光标分别移到K88和M88，可以发现：单元行显示的字符串都相同，只是它们的前导符不同（分别为'、"和^）。在1-2-3中，这三个前导符分别表示字符串的显示格

```
I88: 'Labels can be                                              READY
            I           J           K           L           M
 88  Labels can be   |       Labels can be|  Labels can be
     -------------   |                    |
 89  left-aligned in |     right-aligned in|   centered in
 90  their cells.    |          their cells.|  their cells.
 91                  |                    |
 92  Labels can be very,very,very,very long--and still be seen.
 93
 94
 95
 96
 97
 98  "Repeating" labels
 99  are good for drawing
100  horizontal lines:
101
102  -------------
103
104  =============
105
106  HiMomHiMomHiMomHiMomHi
107
```

图 7—10 定位到 I88 后的画面

式为对左、对右和对中。而列宽小于字符串长度时，则一律向左对齐。

② 修改 I92 中的字符串

把光标移到 I92 上，发现 I92 中是一个长字符串，它横跨 I92、J92、K92、L92 和 M92 五个单元。1-2-3对于超过列宽的字符串，只要其下一个单元是空的，就借这个单元进行延伸显示。修改这个字符串可在 I92 处进行，也可在需要修改之处所在的单元进行。

例如，把光标移至 K92 并打入“'〔abcdef〕”后，屏幕上的画面如图 7—11 所示。

```
K92: '[abcdef]
            I           J           K           L           M
 88  Labels can be   |       Labels can be |  Labels can be
 89  left-aligned in |     right-aligned in |   centered in
 90  their cells.    |          their cells. |  their cells.
 91                  |                     |
 92  Labels can be very,ver[abcdef]very long--and still be seen.
 93
 94
 95
 96
 97
 98  "Repeating" labels
 99  are good for drawing
100  horizontal lines:
101
102  -------------
103
104  =============
105
106  HiMomHiMomHiMomHiMomHi
107
```

图 7—11 修改后的结果

在指定的单元打入新数据，会替换原单元中的数据。若不想替换原来的数据，而只希望对原数据进行修改，则输入〈F2〉键，然后用编辑键进行修改。

③ 把修改后的结果存盘

假如修改后的表仍然要作为文件 LESNA-1·WKS 保存到盘上，则可进行下列操作：

/FSLESNA-1〈CR〉

这时因为盘上已有 LESNA-1·WRS 存在，1-2-3 显示出：

Cancel Replace

打入“R”，就更新盘上的文件；打入“C”则不更新。

3. 表格区域的管理

例7—5 文字处理

（1）功能 调整一段文字的版面，并对这段文字所在的区域命名。

（2）/R 命令介绍

/R 命令可用来定义区域格式，调整区域中的内容和对区域命名。/R 命令的子命令有以下几个：

- Format 定义区域中的数值格式。
- Labl-prefix 定义区域中的标号格式。
- Erase 删除区域中的所有内容。
- Name 命名表格区域。
- Justify 调整区域中文字的版面，调整以单词为单位。
- Protect 设置表格的指定区域为保护区域，仅当整个表格处于可保护状态时（由/WGPE 命令设置），保护性质才有效。
- Unprotect 解除保护。
- Input 定义指定的区域为输入区域，这时输入和编辑只能在输入区域的非保护单元中进行。在输入区域中，〈Home〉键移到区首，〈End〉键移到区末。连续输入两个〈CR〉

```
11: 'Four score and seven                                    READY
        H          I          J          K
 1          Four score and seven
 2          years ago,our
 3          fathers brought forth
 4          on this continent a
 5          new nation,conceived
 6          in liberty and
 7          dedicated to the
 8          proposition that all
 9          men are created equal.
10
```

图7—12 调整前的版面

键退出输入状态。

（3）操作过程

设表格中的I1～I9单元中已输入了一段文字，如图7—12所示。

现在要把这段文字调整为占第I、J和K列的形式，使这段文字的版面加宽。这可通过打入命令

/RJI1·K1〈CR〉

来完成。命令中的“I1·K1”表示版面要调整到的宽度，但其开始的坐标“I1”必须是这段文字起始位置。

调整后，这段文字由9行变为5行，列宽从1列调至3列。调整后的结果如图7—13所示。

```
I1: 'Four score and seven years ago,our fathers      READY
        H          I                J          K
 1
 2      Four score and seven years ago, our fathers
 3      brought forth on this continent a new
 4      nation,conceived in liberty and dedicated
 5      to the proposition that all men are created
 6      equal.
 7
 8
 9
10
```

图7—13　调整后的版面

另外，还可对这段文字所在的区域命名，供以后按名调用。命名的过程如下：

/RNCWORDAREA〈CR〉I1·K5〈CR〉

这段文字可用区域名“WORDAREA”来引用。

4．表格的打印

例7—6　打印表格单元中的内容和打印报表

（1）功能　打印表格的报表，并按行打印出表格的内容，以便对表格进行调试。

（2）/P命令介绍

/P命令可用来把报表从打印机输出（/PP）或建立报表文件（/PF）以供打印。/PP和/PF命令都可以有下列子命令：

* Range　指定报表范围。
* Line　换行。
* Page　换页。

* Clear　　清除打印参数。
* Align　　设置打印机每一页开始打印的位置。
* Go　　打印报表。
* Quit　　退出。
* Options Header　　设页首标题（/PPOH）。
* OF　　设页末标题。
* OP　　设置页长。
* OOC　　按行打印单元的内容。

（3）操作过程

设内存中已存在一张表格，如图7—14所示。

S41: 'Column S is very wide.

	S	T	U	V	W	X
41	Column S is very wide.				4,908 -	
42					7,999	
43	Column T is very narrow.				13,704	
44					4,957 -	
45					15,175 -	
46					2,702	
47					19,430	
48					2,346 -	
49					15,898 -	
50					13,627	
51					3,969	
52					15,036	

图7—14　　已存在的表格

打印报表的过程如下：

/RNCAREA〈CR〉S41.X52〈CR〉　　{定义表格区域为AREA}

/PP　　{使用打印机输出报表}

RAREA〈CR〉　　{定义打印区域为AREA}

OH　　* * Table of Area * *〈CR〉　　{定义页首标题}

QAG　　{打印报表}

打印出来的报表如下：

```
      * * Table of Area * *
Column S is very wide.                    4,908 —
                                          7,999
Column T is very narrow.                 13,704
                                          4,957 —
                                         15,175 —
                                          2,702
                                         19,430
                                          2,346 —
                                         15,898 —
                                         13,627
                                          3,969
                                         15,036
```

接着，按行打印表格中每个已使用了的单元的内容。其过程如下：

```
OF ** End of Page ** 〈CR〉    {定义页末标题}
P20〈CR〉    {设置页长为20}
OC    {使用/PPOOC命令设置逐行格式}
QAG    {打印单元的内容}
```

打印出的结果如下：

```
    ** Table of Area **
S41: 'Column S is very wide.
W41: (,0) 4908.3232787
X41: ' —
W42: (,0) 7998.9139084
S43: ' Column T is very narrow.
W43: (,0) 13704.4937583
W44: (,0) 4956.7814125
X44: ' —
W45: (,0) 15175.4528098
X45: ' —
    ** End of page **
    ** Table of Area **
W46: (,0) 2701.7339505
W47: (,0) 19429.854336
W48: (,0) 2346.1467819
X48: ' —
W49: (,0) 15898.2927818
X49: ' —
W50: (,0) 13627.2188555
W51: (,0) 3969.0009737
W52: (,0) 15036.2638757
    ** End of Page **
```

5．商用函数的使用

例7—7　贷款分析

（1）功能

某工厂向银行贷款20000元扩大再生产，年利率为12%，且按月利滚利计算，五年内应还清本利，问工厂每月应还款多少？本例除计算出月还款数，还应列出五年中每年年初欠款余额和年终结算以及每年归还的总数和利息。

（2）/WG命令介绍

/WG命令可用来定义表格的全局格式，它主要有下列子命令：

* Format　　定义全局数据格式。
* Label—Prefix　　定义全局标号格式。
* Column—Width　定义全局列宽。

（3）操作过程

① 设计表格的格式

首先，定义表格的格式：

/WGC15〈CR〉　　{定义全局列宽为15}

〈Home〉/WCS10〈CR〉　　{定义A列的局部列宽为10}

/WGF，2〈CR〉　　{全局数值格式为三位一撇，小数点后取二位}

/RFP1〈CR〉C2〈CR〉{C2为百分数格式，保留一位小数}

/RFF0〈CR〉A5·A30〈CR〉　　{A5··A30定义为整数格式}

/RFF0〈CR〉C3〈CR〉　　{C3取为整数格式}

其次，填入各栏标题。在B1··B4中填入："Principal（本金）、"Rate（年利率）、"Years（年限）和"Payment（月付款数）。再在A6··D6中填入："Year（年序号）、"Begin Bal.（上年余额）、"End Bal.（年终结算）、"Total Paid（归还的总数）和"Interest（付出的利息）。

接着，在C1填入本金20000，在C2中填年利率0.12，再在C3中填入还款年限5。

最后，在A7··A11中填入年序号1～5。这时的屏幕如图7—15所示。

A1:

	A	B	C	D	E
1		Principal	200,000.00		
2		Rate	12.0%		
3		Years	5		
4		Payment			
5					
6	Year	Begin Bal.	End Bal.	Total Paid	Interest
7	1				
8	2				
9	3				
10	4				
11	5				
12					

图7—15　贷款分析表的结构

② 输入公式并计算结果

已知贷款总额、月利率和还款期限（月），要求出每月应还的款数，可直接使用函数@PMT来计算。故在C4中填入下列公式：

@PMT（C1，C2/12，C3*12）

接着在B7中填入公式"+C1"，表示第1年年初的欠款为20000元。在C7中填入计算年终结算的公式：

@PV（C4，C2／12，12＊（C3－A7））

为复制这个公式，公式里使用了一些绝对坐标。先在D7中填入“＋C4＊12”，表示一年中的总还款数。再在E7中填入“＋D7－（B7－C7）”，用来计算还款中归还的利率数。

下面是第2至第5年的公式的复制过程：

〈F5〉B8〈CR〉＋C7〈CR〉　　{填入第2年的上年余额}

／CB8·B8〈CR〉B9·B11〈CR〉　　{复制到B9··B11中}

／CC7·E7〈CR〉C8·C11〈CR〉　　{把C7·E7复制到矩形区域C8·E11中}

公式复制好了之后，1—2—3会立即计算出结果，如图7—16所示。

C7: @PV(C4,C2/12,12*(C3-A7))　　READY

	A	B	C	D	E
1		Principal	20,000.00		
2		Rate	12.0%		
3		Years	5		
4		Payment	444.89		
5					
6	Year	Begin Bal.	End Bal.	Total Paid	Interest
7	1	20,000.00	16,894.20	5,338.67	2,232.86
8	2	16,894.20	13,394.50	5,338.67	1,838.97
9	3	13,394.50	9,450.95	5,338.67	1,395.12
10	4	9,450.95	5,007.26	5,338.67	894.98
11	5	5,007.26	0.00	5,338.67	331.41
12					

图7—16　贷款分析表

③ 打印报表

最后，用／P命令打印出报表，其过程如下：

/PPA1·E11〈CR〉　　{设置打印区域}

OH　＊＊＊ PAYMENT TABLE ＊＊＊ 1984.6.29〈CR〉　　{设置报表标题}

QAG　　{打印报表}

打印出的报表如下：

＊＊＊ PAYMENT TABLE ＊＊＊ 1984.6.29

	Principal	20,000.00		
	Rate	12.0%		
	Years	5		
	Payment	444.89		
Year	Begin Bal.	End Bal.	Total Paid	Interest
1	20,000.00	16,894.20	5,338.67	2,232.86
2	16,894.20	13,394.50	5,338.67	1,838.97
3	13,394.50	9,450.95	5,338.67	1,395.12
4	9,450.95	5,007.26	5,338.67	894.98
5	5,007.26	0.00	5,338.67	331.41

6．／M和／DF命令的使用

例7—8　20年贷款分析

（1）功能　本例对例7—7进行改进，把还款期限由5年改为20年。

（2）命令介绍

① 移动命令／M

／M命令用于移动指定区域中的内容，移动后所有坐标（包括绝对坐标）都要作适当的调整，这与／C命令不同。

② 步长填充命令／DF

／DF命令是数据处理命令／Data下的一个子命令，可对指定的区域填上等步长的一系列数值。在使用这条命令时，用户须向1-2-3提供四个参数：填数的区域、数的初值、数的步长和数的终值。

（3）操作过程

① 修改表格的格式

首先，把区域B3··C4中的内容移到区域D1··E2中去：

／MB3·C4〈CR〉D1·E2〈CR〉

这时，移到E2中的月付款额的计算公式被更改成：

@PMT（C1，C2／12，E1＊12）

接着，删去第4和第5行，并在第5行处插入一行并填入一条线：

／WDRC4·C5〈CR〉　｛删去第4和第5行｝
／WIRC5·C5〈CR〉　｛在第5行插入一行空行｝
〈F5〉A5〈CR〉＼－〈CR〉　｛在A5中填满减号｝
／CA5·A5〈CR〉B5·E5〈CR〉　｛复制A5，生成一条横线｝

随后，在A11··A25中填入年序号6～20，本例采用步长填充命令／DF来完成这一功能：

／DFA11·A25〈CR〉　｛指定填充区域｝
6〈CR〉1〈CR〉20〈CR〉　｛输入初值6、步长1和终值20｝

② 复制公式与显示结果

格式修改好之后，就复制公式：

／CB10·E10〈CR〉B11·B25〈CR〉

当公式复制好之后，在E1中填入20，就会立即计算出结果，如图7—17所示。

图7—17仅显示出了前15年的数据，输入〈PgDn〉键会显示出表格的其余部分。接着再对表格作进一步的修改：

E1: (F0) 20　　READY

	A	B	C	D	E
1		Principal	20000.00	Years	20
2		Rate	12.0%	Payment	220.22
3					
4	Year	Bgin Bal.	End Bal.	Total Paid	Interest
5					
6	1	20,000.00	19,743.59	2,642.61	2,386.20
7	2	19,743.59	19,454.67	2,642.61	2,353.68
8	3	19,454.67	19,129.10	2,642.61	2,317.04
9	4	19,129.10	18,762.25	2,642.61	2,275.75
10	5	18,762.25	18,348.87	2,642.61	2,229.22
11	6	18,348.87	17,883.06	2,642.61	2,176.80
12	7	17,883.06	17,358.17	2,642.61	2,117.72
13	8	17,358.17	16,766.71	2,642.61	2,051.15
14	9	16,766.71	16,100.25	2,642.61	1,976.14
15	10	16,100.25	15,349.26	2,642.61	1,891.62
16	11	15,349.26	14,503.02	2,642.61	1,796.37
17	12	14,503.02	13,549.46	2,642.61	1,689.05
18	13	13,549.46	12,474.97	2,642.61	1,568.11
19	14	12,474.97	11,264.20	2,642.61	1,431.84
20	15	11,264.20	9,899.87	2,642.61	1,278.28

图 7—17　20 年贷款分析表之一

〈F5〉A26〈CR〉＼—〈CR〉

／CA26·A26〈CR〉B26·E26〈CR〉　　{ 在第 26 行中画线 }
〈F5〉D27〈CR〉@SUM (D6·D25) 〈CR〉→　　{ 计算还款总计 }
@SUM (E6·E25) 〈CR〉　　{ 计算利息总计 }
〈Home〉〈F5〉E6〈CR〉／WTB〈CR〉　　{ 固定标题区 }

E21: +D21—(B21—C21)　　READY

	A	B	C	D	E
1		Principal	200000.00	Years	20
2		Rate	12.0%	payment	220.22
3					
4	Year	Begin Bal.	End Bal.	Total Paid	Interest
5					
21	16	9,899.87	8,362.52	2,642.61	1,105.25
22	17	8,362.52	6,630.19	2,642.61	910.28
23	18	6,630.19	4,678.16	2,642.61	690.58
24	19	4,678.16	2,478.56	2,642.61	443.01
25	20	2,478.56	0.00	2,642.61	164.04
26					
27				52,852.13	32,852.13
28					
29					
30					
31					
32					
33					
34					
35					

图 7—18　20 年贷款分析表之二

这时打入〈PgDn〉键，屏幕上的画面如图 7—18 所示。

③ 打印报表

```
/PPRA1·E27〈CR〉
OH＝＝＝Payment Table＝＝＝〈CR〉

QAG      { 打印报表 }
```

打印出的报表如下：

===Payment Table===

	Principal	20000.00	Years	20
	Rate	12.0%	Payment	220.22
Year	Begin Bal.	End Bal.	Total Paid	Interest
1	20,000.00	19,743.59	2,642.61	2,386.20
2	19,743.59	19,454.67	2,642.61	2,353.68
3	19,454.67	19,129.10	2,642.61	2,317.04
4	19,129.10	18,762.25	2,642.61	2,275.75
5	18,762.25	18,348.87	2,642.61	2,229.22
6	18,348.87	17,883.06	2,642.61	2,176.80
7	17,883.06	17,358.17	2,642.61	2,117.72
8	17,358.17	16,766.71	2,642.61	2,051.15
9	16,766.71	16,100.25	2,642.61	1,976.14
10	16,100.25	15,349.26	2,642.61	1,891.62
11	15,349.26	14,503.02	2,642.61	1,796.37
12	14,503.02	13,549.46	2,642.61	1,689.05
13	13,549.46	12,474.97	2,642.61	1,568.11
14	12,474.97	11,264.20	2,642.61	1,431.84
15	11,264.20	9,899.87	2,642.61	1,278.28
16	9,899.87	8,362.52	2,642.61	1,105.25
17	8,362.52	6,630.19	2,642.61	910.28
18	6,630.19	4,678.16	2,642.61	690.58
19	4,678.16	2,478.56	2,642.61	443.01
20	2,478.56	0.00	2,642.61	164.04
			52,852.13	32,852.13

7. /D 命令的使用

例7—9　利率变化分析

（1）功能

设计一张年利率从 6.0% 开始，以步长 0.5% 增长到 14.5% 时，每月付款额、20 年付款总额、总利息和利息与本金比的变化表（以例 7—8 为基础）。

（2）命令介绍

① 数据处理命令 /D

* Fill　　步长填充数据。
* Table　　制作 1-2-3 预测表。
* Sort　　对记录进行排序。
* Query　　对 1-2-3 的数据库文件进行检索。
* Distribution　　对指定的字段进行计数统计，作频度分布表。

② 预测表命令 /DT

1-2-3 的预测表常用于分析表格随输入变量变化而变化的情况（也称为灵敏性分析）。在使用预测表时，不需要在每个单元中填入公式，而仅在预测表的第 1 行和第 1 列填入公式和输入变量的数据。1-2-3 会在预测表的其它单元计算出对应的结果。/DT 命令又有以下三条子命令：

* 1　第 1 类预测表，它有一个输入变元 I（可以取值为 E1，E2，E3…）和多个计算公式（F1，F2，F3…），如图 7—19（a）所示。
* 2　第 2 类预测表，它有两个输入变元（I1，I2）和一个计算公式（F），如图 7—19（b）所示。
* Reset　清除预测表参数。

I	输入变量			
	F1	F2	F3	…
E1	V11		V13	
E2				
E3	V31			
⋮				结果区

(a) 第 1 类预测表

I1	输入变量			I2
F	Y1	Y2	Y3	…
X1	V11		V13	
X2				
X3	V31			
⋮				结果区

(b) 第 2 类预测表

图 7—19　1—2—3 的预测表

（3）操作过程

设当前表格为例 7—8 的结果（如图 7—17 和图 7—18 所示）。把表格窗口下移 40 行并在 A41··E41 中送入预测表的标题：

```
/WTC      { 清除标题 }
〈F5〉A41〈CR〉"Rate→"Payment→"Total Paid→"Interest→"Int./Pri.〈CR〉
```

接着填入预测表第 1 行中的公式和第 1 列中的输入变量数据：

```
/RFTA42.E42〈CR〉      { 把 A42··E42置为公式显示格式 }
〈F5〉A42〈CR〉→+E2→+D27→+E27→+E27/C1〈CR〉      { 填入公式 }
/RFP1〈CR〉A43.A60〈CR〉      { 置 A43··A60 格式为xx.x% }
/DFA43.A60〈CR〉6.0%〈CR〉0.5%〈CR〉15%〈CR〉      { 填入数据 }
```

再使用 /DT 命令建立预测表：

```
/DT1      { 选择第 1 预测表 }
A42.E60〈CR〉      { 置预测表的范围 }
C2〈CR〉      { 选 C2 为输入变量 }
```

这时屏幕上显示出预测表的结果，如图 7—20 所示。

B42: (T)+E2 READY

	A	B	C	D	E
41	Rate	Payment	Total Paid	Interest	Int./Pri.
42	+E2	+D27	+E27	+E27/C1	
43	6.0%	143.29	34,388.69	14,388.69	0.72
44	6.5%	149.11	35,787.51	15,787.51	0.79
45	7.0%	155.06	37,214.35	17,214.35	0.86
46	7.5%	161.12	38,668.47	18,668.47	0.93
47	8.0%	167.29	40,149.12	20,149.12	1.01
48	8.5%	173.56	41,655.52	21,655.52	1.08
49	9.0%	179.95	43,186.85	23,186.85	1.16
50	9.5%	186.43	44,742.30	24,742.30	1.24
51	10.0%	193.00	46,321.04	26,321.04	1.32
52	10.5%	199.68	47,922.23	27,922.23	1.40
53	11.0%	206.44	49,545.04	29,545.04	1.48
54	11.5%	213.29	51,188.62	31,188.62	1.56
55	12.0%	220.22	52,852.13	32,852.13	1.64
56	12.5%	227.23	54,534.75	34,534.75	1.73
57	13.0%	234.32	56,235.63	36,235.63	1.81
58	13.5%	241.47	57,953.98	37,953.98	1.90
59	14.0%	248.70	59,689.00	39,689.00	1.98
60	14.5%	256.00	61,439.89	41,439.89	2.07

图7—20 利率变化分析表

8. 趋势图的生成与输出

例7—10 利率变化分析图

（1）功能 把例7—9中利率变化预测表表示成XY图（趋势图）。

（2）命令介绍

① 选图命令 /GT

这条命令用于选择欲作图的类型。可供选择的类型有：折线图（L）、条形图（B）、叠置条形图（S）、扇形图（P）和XY图（X）。

② /GX和/GA命令

/GX命令用来指明X轴的小标题所在的区域。/GA…/GF命令用来指明图的六个数据区，用户可任意指定1～6个数据区。

③ 图标题命令 /GOT

/GOT命令可用于设置图的标题，它又有下列子命令

* First 图的主标题。
* Second 图的副标题。
* X—axis X轴标题。
* Y—axis Y轴标题。

④ 图标注命令 /GOD

/GOD命令可用来对数据区A～F加标注，例如/GODA表示对A数据区加标注。使用这组命令后，首先要向1-2-3提供标注的信息区，再指明标注的位置。标注的位置有五种：Centered（中心）、Left（左）、Above（上）、Right（右）和Below（下）。

（3）操作过程

① 生成趋势图

```
/GTX                    {选择 XY 图}
XA43.A60〈CR〉           {把利率作为 X 坐标}
AE43.E60〈CR〉           {把利息和本金比作为图的数据区}
OTF **20 Years Interest/Principal ** 〈CR〉    {设置图的主标题}
TX==Years Rates ==〈CR〉     {设置 X 轴标题}
DAE43.E60〈CR〉R         {使用数据本身作标注，位于右边}
QSXY〈CR〉Q              {把生成好的图以文件名 XY·PIC 存盘}
```

② 打印趋势图

```
/QY                     {退出 1-2-3，返回 LOTUS 系统并插入图形打印盘片}
G                       {调用图形打印程序}
SXY〈CR〉                {选择要打印的文件为 XY·PIC}
OF1〈CR〉AG              {选择字体并打印}
```

打印出的趋势图如图 7—21 所示。

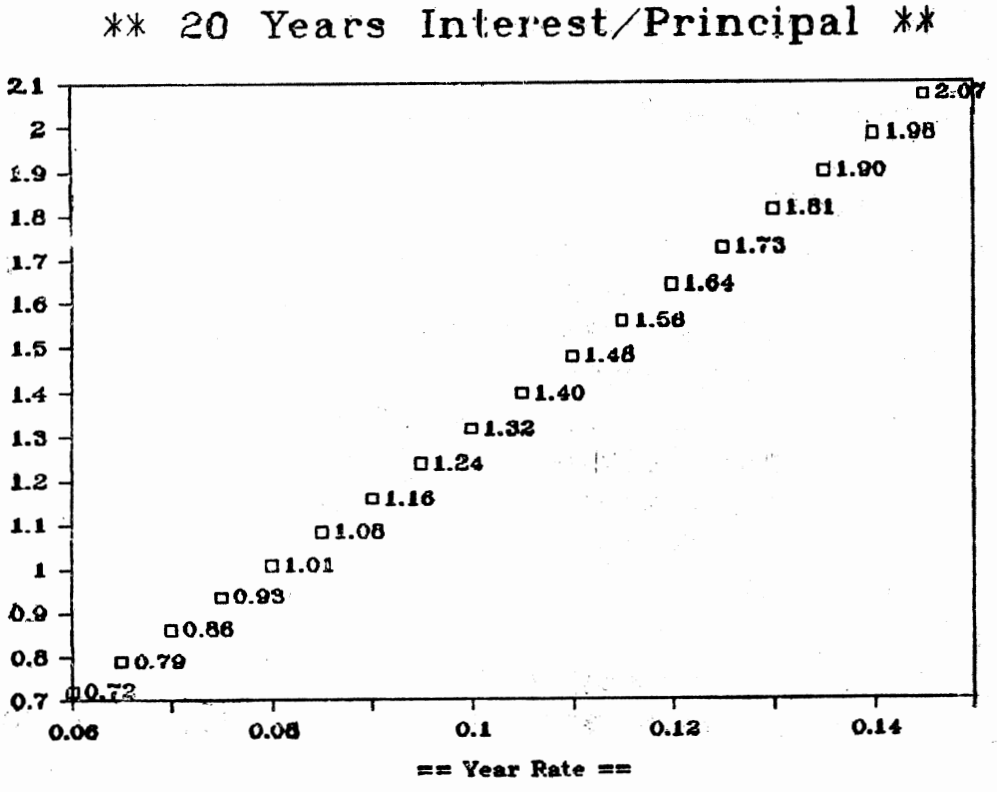


图 7—21　利率变化趋势图

9．统计图的生成与输出

例 7—11　　销售额统计图

（1）功能

某百货商店有三个销售部 A、B 和 C，其 1983 年四个季度的销售额报表如图 7—22 所示。

A3: ^(1983)　　MENU

Worksheet Range Copy Move File Print Graph Data Quit

Create a graph

	A	B	C	D	E	F
1	Sales Summary		Jan-Mar	Apr-Jun	Jul-Sep	Oct-Dec
2	Branch Offices	Adept.	40,100	55,600	66,000	65,500
3	(1983)	Bdept.	83,900	77,050	94,300	87,900
4		Cdept.	22,590	44,100	47,900	12,660
5						
6		*Total*	146,590	176,750	208,200	166,060
7						
8						
9						
10						

图 7—22　1983 年销售额报表

本例分别用条形图、叠置条形图、折线图和百分比图等不同形式来反映该百货商店1983年度的销售情况。

（2）／G 命令介绍

／G 命令主要用于作商用统计图，它有下列子命令：

- Type　　确定统计图的类型。
- X A…F　　定义 X 轴小标题和六组数据区在表格中的区域。
- Reset　　清除设置的作图参数。
- View　　在图形显示器上显示生成的统计图。
- Save　　把生成的统计图以图形文件的形式保存。
- Options　　设置统计图的标题、比例和标注等参数。
- Name　　对生成的统计图命名。
- Quit　　退出。

（3）设计步骤

设计一张统计图，一般可按下列步骤进行：

① 选择统计图的类型：条形图、叠置条形图、扇形图（百分比图）、折线图和 XY 坐标图。

② 指明 X 轴的取值区域和 A～F 数据区的取值区域。

③ 定义统计图的各种参数，例如标题、比例、标注和颜色等。

④ 显示、命名和保存统计图。显示统计图可用 View 子命令或〈F10〉功能键来完成。

⑤ 使用 LOTUS 管理系统下的图形文件打印程序打印生成的图形文件。

图形打印程序的初态画面如图 7—23 所示。其第 3 行为命令“菜单”，从第 6 行开始为缺省的参数，它们可用 Options 命令更改。

```
GRAPH Copyright (C) 1982,1983 Lotus Development Corp. All Rights Reserved MENU

Select Options Go Configure Align Page Quit
Select pictures

SELECTED GRAPHS  COLORS          SIZE    HALF      DRIVES

                 Grid:    Black  Left Margin:   .785 Pictures: B
                 A Range: Black  Top Margin:    .500 Fonts: A
                 B Range: Black  Width:        6.922
                 C Range: Black  Height:       5.000 GRAPHICS DEVICE
                 D Range: Black  Rotation:      .000 Epson MX double
                 E Range: Black
                 F Range: Black  MODE                 INTERFACE
                                 Eject: No            Parallel
                 FONTS           Pause: No

                 1:                                   PAGE SIZE
                 2:                                   Length    11.000
                                                      Width      8.000
```

图 7—23　图形打印程序的初态画面

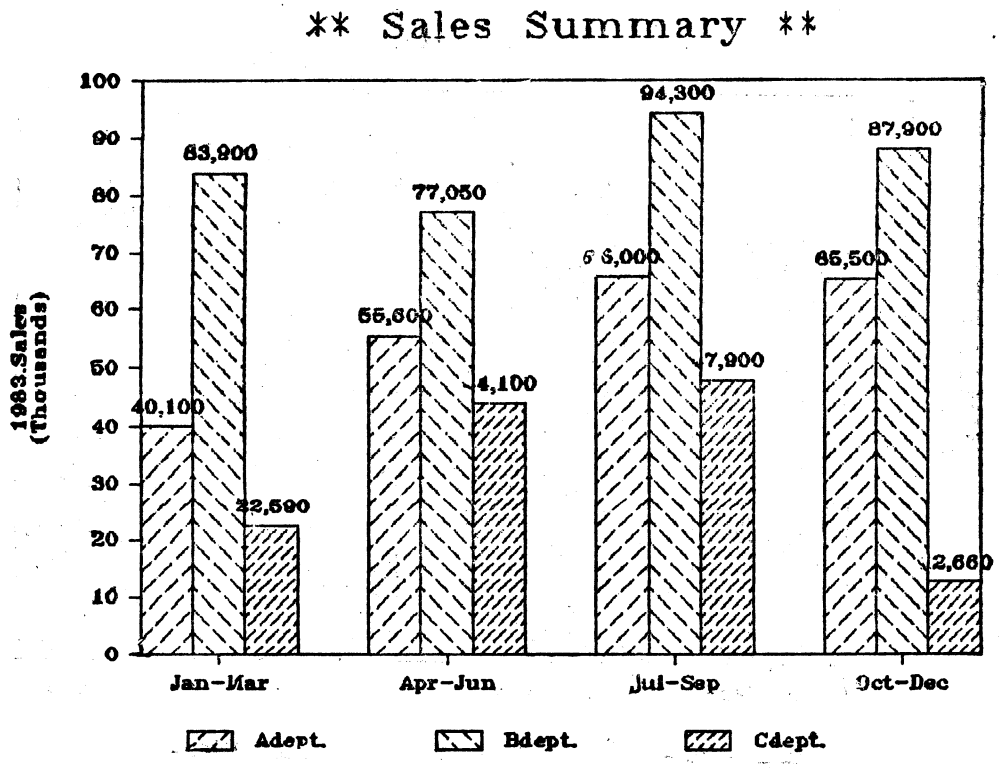


图 7—24　销售额条形图

使用 Options 可设置的统计图参数有以下几个：

* Color　为彩色打印机（或彩色绘图仪）设置统计图的颜色。
* Font　选择字体，可选的字体有：BLOCK（方块体）、SCRIPT（手写体）、ROMAN（印刷体）和ITALIC(斜体)。它们又分 1 号字体(主标题) 和 2 号字体。
* Size　图幅的大小，有全页（Full）、半页（Half）和自定义（Manual）三种。
* Pause　打印两幅图之间暂停一次，以便用户调整打印纸，按〈SP〉键就继续下去。
* Eject　打印一幅图要自动走到下一页的页首。

（4）操作过程

下面先作条形图，作图的过程如下：

```
/GTB      {选条形图}
XC1·F1〈CR〉      {定义 C1··F1 中的季名为 X 轴小标题}
AC2·F2〈CR〉BC3·F3〈CR〉CC4·F4〈CR〉      {定义 A—C 数据区}
ODAC2·F2〈CR〉ABC3·F3〈CR〉ACC4·F4〈CR〉A      {定义标注}
QTF**Sales Summary**〈CR〉      {定义大标题}
TY1983·Sales〈CR〉      {定义 Y 轴标题}
LA\B2〈CR〉LB\B3〈CR〉LC\B4〈CR〉      {给 A—C 数据区加图例：Adept, Bdept, Cdept}
```

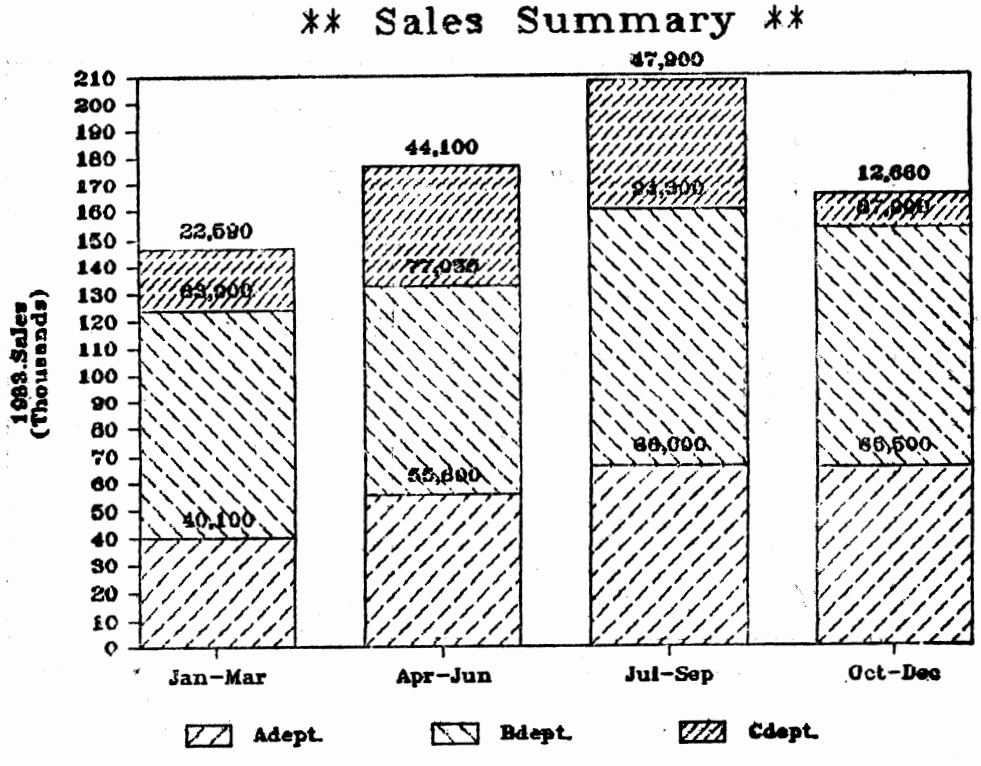


图 7—25　销售额叠置条形图

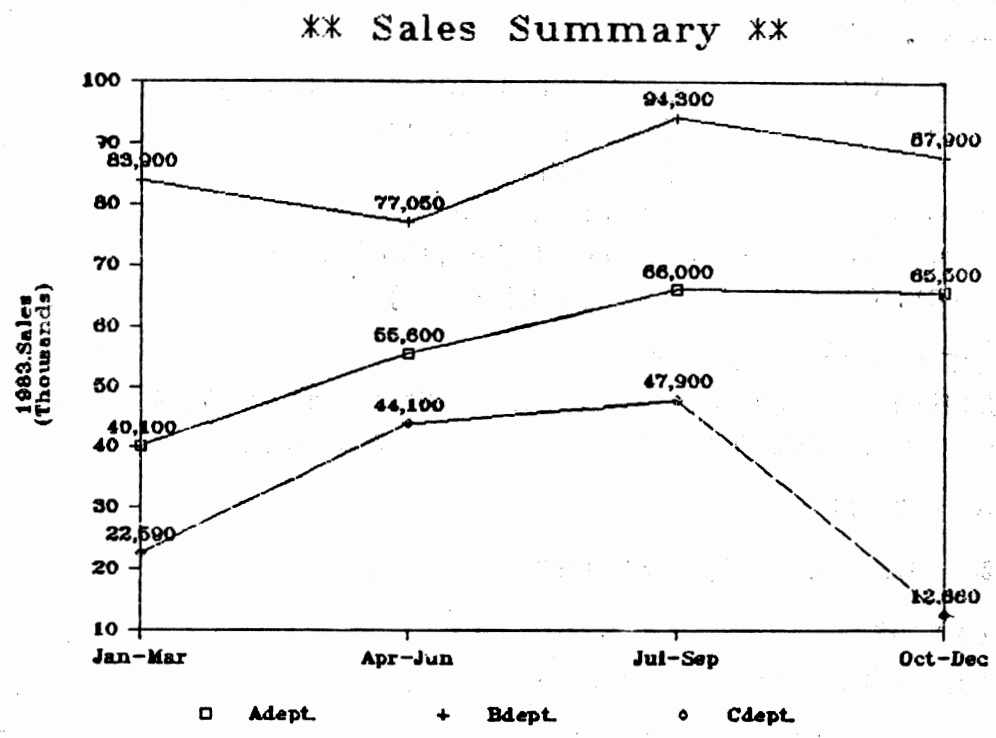


图 7—26　销售额折线图

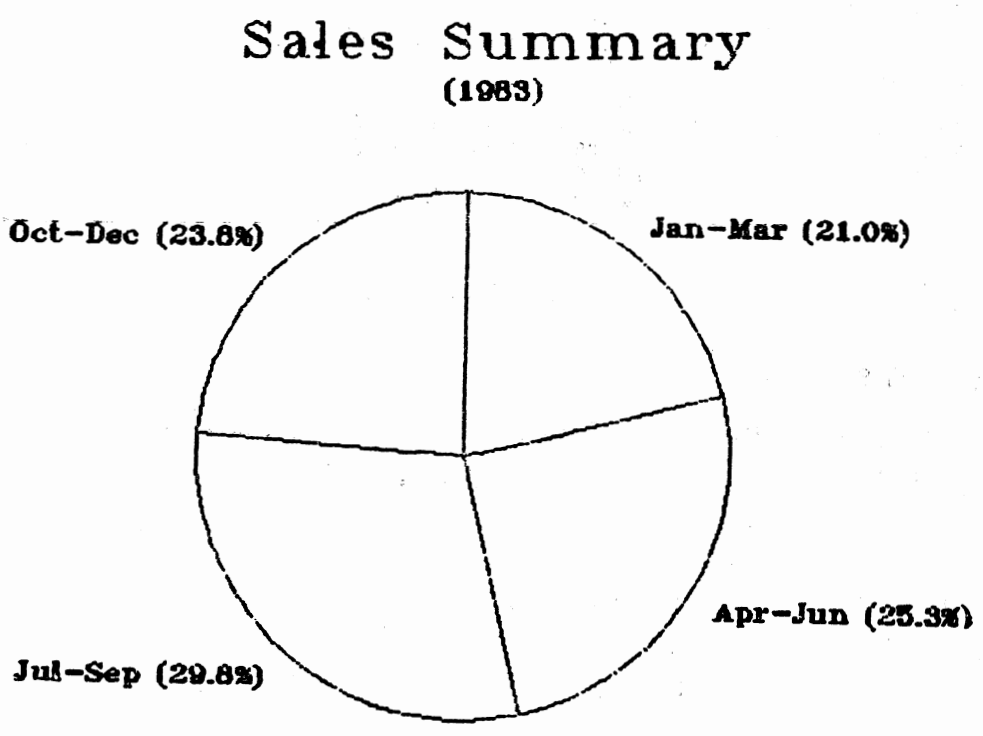


图 7—27　总销售额扇形图

上面命令中的＼B2、＼B3 和＼B4 分别表示 B2..B4 单元中的内容“Adept”、“Bdept”和“Cdept”。同样，也可用＼A1 表示“Sales Summary”。

```
QSBARGP〈CR〉     {把条形图以 BARGP·PIC 为名存盘}
NCBARGP〈CR〉     {把这幅图命名为 BARGP}
V                {显示条形图}
Q／QY            {退出 1-2-3，装入图形打印程序盘片}
GSBARGP〈CR〉     {调 GRAPH 程序并选 BARGP·PIC}
OF1〈CR〉AG       {选择字体并打印图形}
```

打印出的条形图如图 7—24 所示。

用同样方法可作出叠置条形图（图 7—25）和折线图（图 7—26）。同样也可作出以三个部门的季度合计（C6.F6）为数据区的扇形图，如图 7—27 所示。

10．数据排序

例 7—12　对职员数据库文件进行排序

（1）功能

对已存在的职员数据库文件（图 7—28）中的记录按关键字进行排序。

B4: 29　　READY

	A	B	C	D	E	F	G	H
1	Employee Database:Name,c10;Age,n2;Sala,n8,2;Dept,c6;Manag,c10							
2								
3	Name	Age	Sala	Dept	Manag			
4	Cau-a	29	103.80	Adept	Zhou-e			
5	Qen-b	42	59.40	Bdept	Li-d			
6	Shen-c	22	53.40					
7	Li-d	30	100.76	Bdept	Wang-f			
8	Zhou-e	35	115.68	Adept	Wang-f			
9	Wang-f	50	116.30	Cdept				
10	Fong	40	47.30	Cdept	Li-d			

图 7—28　职员数据库文件

（2）／DS 命令介绍

／DS 可用来对数据库按关键字排序。它的参数由下列子命令来设置：

* Data—Range　指定要排序的记录范围。
* Primary—key　指定主关键字。
* Secondary—key　指定次关键字。
* Reset　取消排序参数。
* Go　执行排序。
* Quit　退出。

（3）操作过程

```
/DSDA4·E10〈CR〉      {指定排序范围为 A4··E10}
PD4〈CR〉A〈CR〉       {以部门(Dept)为主关键字，增序}
SB4〈CR〉A〈CR〉       {以姓名为次关键字，按增序进行排序}
G                    {排序}
```

排序后的结果如图 7—29 所示。

B4: 22

	A	B	C	D	E	F	G	H
1	Employee Database: Name,c10;Age,n2;Sala,n8,2;Dept,c6;Manag,c10							
2								
3	Name	Age	Sala	Dept	Manag			
4	Shen-c	22	53.40					
5	Cau-a	29	103.80	Adept	Zhou-e			
6	Zhou-e	35	115.68	Adept	Wang-f			
7	Li-d	30	100.76	Bdept	Wang-f			
8	Qen-b	42	59.40	Bdept	Li-d			
9	Fong	40	47.30	Cdept	Li-d			
10	Wang-f	50	116.30	Cdept				

图 7—29　排序后的职员数据库文件

11. 数据检索

例 7—13　对职员数据库文件进行检索

(1) 功能　对例 7—12 排序后的职员数据库文件进行多种查询操作。

(2) /DQ 命令介绍

在使用 /DQ 命令检索时。一般都涉及到三个区域：存放数据的输入区域、存放检索条件的条件区域和存放检索结果的输出区域。/DQ 命令下的子命令有：

* Input　定义输入区域。
* Criterion　定义条件区域。
* Output　定义输出区域。
* Find　根据给定的条件在输入区域中查找，若找到就用光标指出查找到的记录。
* Extract　根据条件检索，检索的结果送输出区域。
* Unique　功能类似 Extract 子命令，但不复制重复的记录到输出区域。
* Delete　删除所有满足检索条件的记录。
* Reset　清除检索参数。
* Quit　退出 /DQ 命令。

(3) 操作过程

① 查找 A 部门名为 Cau—a 的职员的情况

```
/CA3·E3〈CR〉A12·E12〈CR〉      {设置条件区域的字段名}
```

```
/CA3·E3〈CR〉A16·E16〈CR〉          {设置输出区域的字段名}
/MD16·D20〈CR〉F16·F20〈CR〉        {把部门区段调到F列}
〈F5〉A13〈CR〉Cau—a→→→Adept〈CR〉   {输入查询条件}
/DQIA3·E10〈CR〉    {指定输入区域}
CA12·E13〈CR〉      {指定检索条件区}
OA16·F20〈CR〉      {指定输出区域}
E                  {执行检索}
```

检索后的结果如图 7—30 所示。

```
A13: 'Cau-a                                                    MENU
Input  Criterion  Output  Find  Extract  Unique  Delete  Reset  Quit
Copy all records that match criteria to Output range
        A        B        C        D        E        F        G        H
 1  Employee Database: Name, c10; Age, n2; Sala, n8, 2; Dept, c6; Manag, c10
 2
 3      Name      Age      Sala     Dept     Manag
 4  Shen-c         22      55.40
 5  Cau-a          29     103.80    Adept    Zhou-e
 6  Zhou-e         35     115.68    Adept    Wang-f
 7  Li-d           30     100.76    Bdept    Wang-f
 8  Qen-b          42      59.40    Bdept    Li-d
 9  Fong           40      47.30    Cdept    Li-d
10  Wang-f         50      116.3    Cdept
11
12      Name      Age      Sala     Dept     Manag
13  Cau-a                           Adept
14
15
16      Name      Age      Sala              Manag     Dept
17  Cau-a          29     103.80             Zhou-e    Adept
18
19
20
```

图 7—30　人员检索结果

② 查找 A 部门和 B 部门中的所有职员

在 1-2-3 中，条件区域的同一行的条件其关系为“AND”，同一列的条件其关系为“OR”。

```
Q/REA13·A13〈CR〉          {删去A13中的内容}
〈F5〉D14〈CR〉Bdept〈CR〉    {在D14中输入Bdept}
/DQCA12·E14〈CR〉          {指定条件区域}
E                         {执行检索}
```

检索后的结果如图 7—31 所示。

③ 查找小于 30 岁的职员

```
Q/RED13·D14〈CR〉          {清除原来的条件}
〈F5〉B13〈CR〉/RFTB13·B13〈CR〉    {显示B13中的公式}
+B4<30〈CR〉               {输入检索条件}
```

/DQCA12·E13〈CR〉E　　{指明条件区域并检索}

检索的结果如图7—32所示。

A13:　　READY

	A	B	C	D	E	F	G	H
1	Employee Database:Name, c10;Age,n2;Sala,n8,2;Dept,c6;Manag,c10							
2								
3	Name	Age	Sala	Dept	Manag			
4	Shen-c	22	53.40					
5	Cau-a	29	103.80	Adept	Zhou-e			
6	Zhou-e	35	115.68	Adept	Wang-f			
7	Li-d	30	100.76	Bdept	Wang-f			
8	Qen-b	42	59.40	Bdept	Li-d			
9	Fong	40	47.30	Cdept	Li-d			
10	Wang-f	50	116.3	Cdept				
11								
12	Name	Age	Sala	Dept	Manag			
13				Adept				
14				Bdept				
15								
16	Name	Age	Sala	Manag	Dept			
17	Cau-a	29	103.80	Zhou-e	Adept			
18	Zhou-e	35	115.68	Wang-f	Adept			
19	Li-d	30	100.76	Wang-f	Bdept			
20	Qen-b	42	59.40	Li-d	Bdept			

图7—31　A、B部门中的职员

B13: (T) +B4<30　　MENU

Input Criterion Output Find Extract Unique Delete Reset Quit

Copy all records that match criteria to Output range

	A	B	C	D	E	F	G	H
1	Employee Database:Name,c10;Age,n2;Sala,n8,2;Dept,c6;Manag,c10							
2								
3	Name	Age	Sala	Dept	Manag			
4	Shen-c	22	53.40					
5	Cau-a	29	103.80	Adept	Zhou-e			
6	Zhou-a	35	115.68	Adept	Wang-f			
7	Li-d	30	100.76	Bdept	Wang-f			
8	Qen-b	42	59.40	Bdept	Li-d			
9	Fong	40	47.30	Cdept	Li-d			
10	Wang-f	50	116.3	Cdept				
11								
12	Name	Age	Sala	Dept	Manag			
13		+B4						
14								
15								
16	Name	Age	Sala		Manag	Dept		
17	Shen-c	22	53.40					
18	Cau-a	29	103.80		Zhou-e	Adept		
19								
20								

图7—32　小于30岁的职员

12. 频度分布

例7—14　职员年龄分布

（1）功能

制作职员年龄频度分布表，并作出相应的分布条形图和分布扇形图。已存在的职员数据库文件如图 7—33 所示。

D4: 'Office

	A	B	C	D	E	F	G	H
1	Employee Database:Name,c10;Age,n2;Sala,n8,2;Dept,c8;Manag,c10							
2								
3	Name	Age	Sala	Dept	Manag			
4	Shen-c	22	53.40	Office				
5	Cau-a	47	103.80	Adept	Zhou-e			
6	Zhou-e	35	115.68	Adept	Wang-f			
7	Li-d	31	100.76	Bdept	Wang-f			
8	Qen-b	42	59.40	Bdept	Li-d			
9	Fong	40	47.30	Cdept	Li-d			
10	Wang-f	51	116.30	Cdept				
11	Yie-g	63	270.40	Office				
12	Zhou-h	48	100.69	Cdept	Wang-f			
13	Qu-i	46	73.40	Bdept				
14	Ge-j	44	68.90	Ddept				
15	Zhang-k	28	59.40	Adept				
16	Gao-l	80	330.68					
17	Hua-m	56	80.40	Office				
18	Lin-n	48	90.43	Cdept				
19	Se-o	45	68.40	Ddept				
20	Zen-p	49	102.80	Office				

图 7—33　职员数据库文件

（2）操作过程

① 制作年龄频度分布表

```
Q〈F5〉A21〈CR〉Distribution of Age〈CR〉
〈F5〉A22〈CR〉"Age<=30↓"31——40↓
"41——50↓"51——60↓"Age>60〈CR〉      {输入标题}
```

C27: (F2)　　READY

	A	B	C	D	E
21	Distribution of Age				
22	Age<=30	30	2		
23	31——40	40	3		
24	41——50	50	8		
25	51——60	60	2		
26	Age>60		2		
27					
28					
29					
30					

图 7—34　年龄频度分布表

〈F5〉B22〈CR〉30↓40↓50↓60〈CR〉　　{输入年龄组的下界}

／DDB4·B20〈CR〉B22·B25〈CR〉　　{作统计频度分布}

统计后的结果如图 7—34 所示。

② 作分布图

／GTB　　{选条形图}

XB22·B26〈CR〉　　{B22··B26为 X 坐标}

AC22·C26〈CR〉　　{数据区为 C22··C26}

OTF＊＊Distribution of Age＊＊〈CR〉　　{标题}

QSDOAB〈CR〉　　{以 DOAB·PIC 为名记盘}

TP　　{选扇形图}

QSDOAP〈CR〉　　{以 DOAP·PIC 为名保存}

随后，可退出 1-2-3 返回 LOTUS 管理系统，并调用图形打印程序打印出这两张图来，如图 7—35和图 7—36 所示。

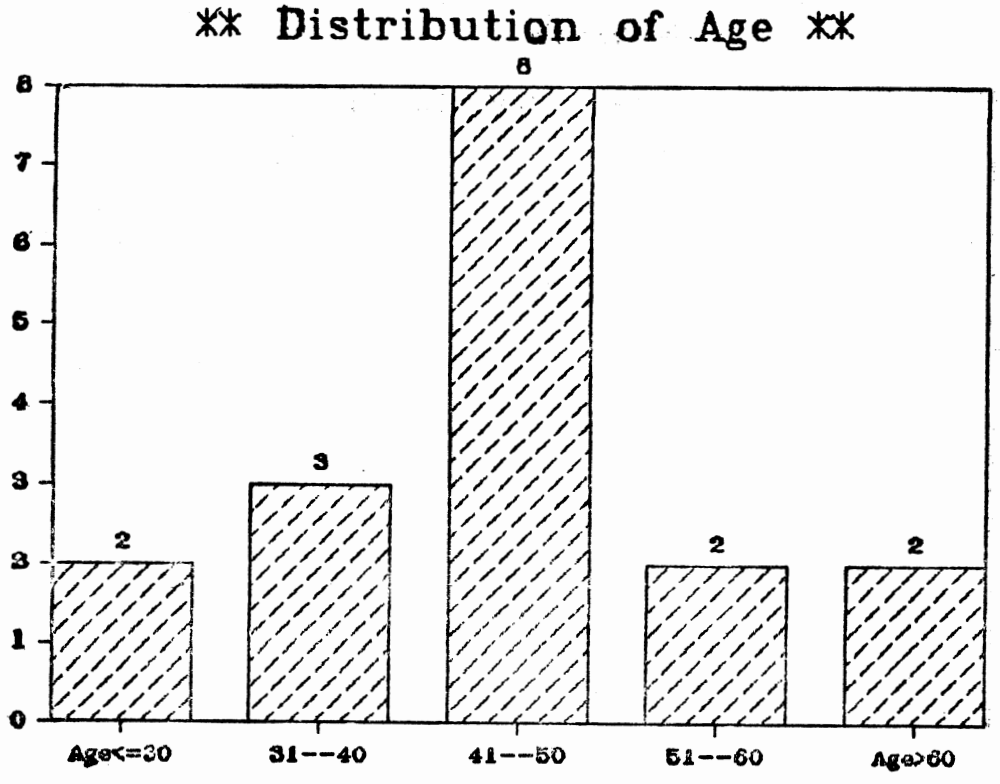


图 7—35　　年龄频度分布条形图

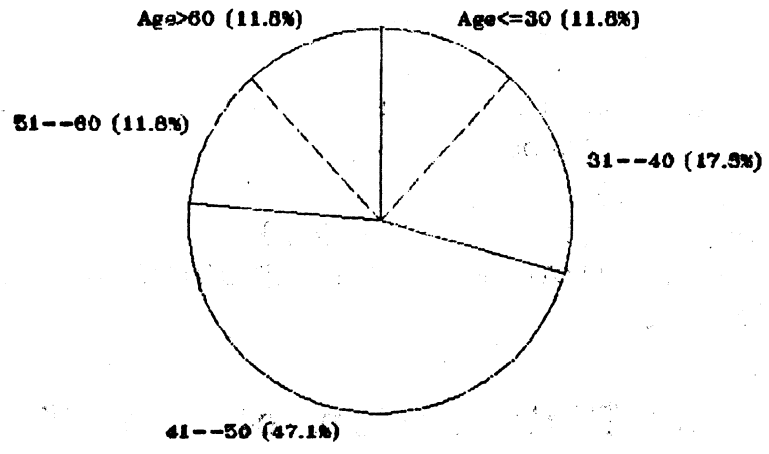


图7—36　年龄频度分布扇形图

第八章　IBM PC实用图形学

由于大规模集成电路技术的飞跃发展，半导体存贮器的性能越来越高，成本也越来越低，因此，采用光栅扫描技术的图形显示器在微型计算机中日益普及。图形显示的理论和技术已经引起了系统开发人员和计算机用户们的普遍兴趣。

图形显示是计算机的一种输出形式，它比文字信息的输出具有更大的优越性。俗话说得好，“一幅画抵得上千言万语”。这是因为一幅好的图画能容纳大量的信息，并且更容易为人们所理解和记忆。

图形显示技术有着广泛的应用领域。例如，计算机辅助设计和辅助制造（CAD／CAM），计算机模拟，图象处理，地学信息处理及地图绘制，管理信息系统中的数据分析与辅助决策，游戏，计算机辅助教学（CAI）以及办公室自动化等等。

IBM PC的字符显示器在图形显示方面只具有相当有限的能力。例如，可以利用一些专用的图形字符和某些特殊符号显示出诸如条形图、游戏图案之类的图形。只有在配置了彩色图形显示控制器及彩色（或黑白）监视器之后，IBM PC才真正具备图形显示的能力。

本章将通过若干实例较为详细地叙述二维图形的显示，交互式作图，图形操作与变换，动画等计算机图形学中的一些基本原理和常用技术。限于篇幅，对于三维图形只作入门性的介绍。全部例子都是用IBM PC的高级BASIC语言写成的，但是也能很方便地把它们移植到用其它语言所开发的软件中去。为了便于读者参考使用，本章的例题较为完整，并根据机器输出的程序清单和运行结果照相排印。

第一节　图形显示原语

在第一章里已经介绍了IBM PC的彩色图形显示器的逻辑结构、工作原理和程序设计的基本方法。彩色图形显示器有多种显示模式，在图形显示模式下，屏幕上可以由程序直接控制其亮度或颜色的最小单位叫做“象元”（Pixel），通常也简称为一个“点”。一幅图画实际上是由千万个这样的点所组成。

为方便用户在开发应用软件时能较好地使用硬件所提供的各种图形显示功能，避免直接与低级的硬件逻辑进行接口，系统软件往往通过下列各种不同的形式，向用户提供一组用于图形显示的例行程序。例如：

* 在操作系统中增加有关图形显示的各种功能调用。
* 在程序库中添加一组用于图形显示的各种函数和子程序。
* 在高级语言中扩充若干图形显示语句。

无论采用哪一种形式，这种提供给用户直接使用的、具有最基本的图形显示操作功能的例行程序就称为“图形显示原语”。

IBM PC高级BASIC语言所提供的图形显示原语（即图形显示语句）主要有两类：第1类是画图原语，例如画点，画线，画矩形，画圆弧等；第2类是交互原语，它们与光笔、操纵杆等有关，用于交互式图形显示。本节先介绍第1类原语及其使用。

1．显示模式的选择

在 BASIC 语言的控制下，IBM PC 的彩色图形显示器可以有下列一些显示模式：

- 字符显示模式
  - 40 列模式（显示缓冲器分成 8 页）
  - 80 列模式（显示缓冲器分成 4 页）
- 图形显示模式
  - 中分辨率模式（每屏 200 线，每线 320 点）
  - 高分辨率模式（每屏 200 线，每线 640 点）

为了选择彩色图形显示器的显示模式，BASIC 语言提供三条语句，现分别予以介绍：

（1）SCREEN 语句（屏幕语句）

屏幕语句的语法如下：

SCREEN M，BST，AP，VP

其中参数 M 为显示模式，它是一个整数，其定义为：

0　选择字符显示模式（40 列还是 80 列由 WIDTH 语句决定）

1　选择中分辨率图形显示模式

2　选择高分辨率图形显示模式

参数 BST 用来选择是黑白显示还是彩色显示，其定义为：

- 0
  - 中分辨率图形显示时　彩色显示
  - 字符显示时　黑白显示
- 非0
  - 中分辨率图形显示时　黑白显示
  - 字符显示时　彩色显示

因为在高分辨率图形显示模式下，只能使用黑白显示，所以当 M=2 时参数 BST不起作用。最后两个参数 AP 和 VP 只在字符显示模式下才有意义，AP 指出哪一个页面是工作页面（即输出显示的信息被写入的页面），而 VP 指出哪一个页面是可见页面（即屏幕上看到的那个页面）。

SCREEN 语句所选择的显示模式如果与程序中以前的模式一致，则不影响屏幕上的显示内容。若使用 SCREEN 语句选择了新的显示模式，则屏幕被清除，底色和边框色被置为黑色，显示色被置为白色。

SCREEN 语句中的参数可以被省略，被省略的参数仍采用原来的旧值。

下面是使用 SCREEN 语句的例子。

```
10 SCREEN 2      {选择高分辨率图形显示模式}
20 SCREEN 1，0   {选择中分辨率彩色图形显示模式}
30 SCREEN ，1    {仍为中分辨率图形显示模式，但无彩色}
```

（2）WIDTH 语句（宽度语句）

WIDTH 语句在图形显示中应用时，其语法为：

WIDTH S

其中参数S只允许是40或80。在字符显示模式时，WIDTH语句用来选择屏幕上每行显示40个还是80个字符；在图形显示模式时，WIDTH 40的作用相当于SCREEN 1语句，它强行使显示器进入中分辨率模式，而WIDTH 80则相当于SCREEN 2语句，它使显示器进入高分辨率模式。

（3）COLOR语句（彩色语句）

在中分辨率图形显示模式下，屏幕上的每一点都可以有多种颜色。程序中可以用COLOR语句来选择屏幕的底色，并决定象元的颜色使用哪一种配色器来定义。COLOR语句在图形显示模式时的语法为：

COLOR B，P

其中，参数B是整型常数或表达式，取值范围应在0到15之间，它用来选择屏幕的底色，其定义为：

0	黑	8	深灰
1	蓝	9	浅蓝
2	绿	10	浅绿
3	青	11	浅青
4	红	12	淡红
5	洋红	13	淡洋红
6	棕	14	黄
7	淡灰	15	白

参数P用于选择配色器0还是1，如果它的值为偶数，则选择配色器0，若为奇数，则选择配色器1。如同第一章所述，中分辨率图形显示模式时，每个象元的取值（称为“彩色码”）可以有四种，即0、1、2或者3，它们所对应的颜色是由COLOR语言所选择的配色器决定的，具体规定为：

彩色码	配色器 0	配色器 1
0	与底色B相同	与底色B相同
1	绿	青
2	红	洋红
3	黄	白

COLOR语句中的参数B和P可以缺省，缺省某参数时就表示保持原来的底色或配色器不变。当使用SCREEN 1语句进入中分辨率图形显示模式时，其初态为：底色是黑色，配色器为1号。此后，程序中可以任意地使用COLOR语句。每次执行COLOR语句后，屏幕的底色及已有的图形画面，均按照COLOR语句的要求而立即改变其颜色。

由于高分辨率图形显示模式只能是黑白工作方式（底色为黑，显示色为白），因此不能使用COLOR语句，否则引起语法错误。

COLOR语句也可以在字符显示模式下使用，其作用为选择字符的显示色、底色及屏幕

上边框的颜色，这里就不细述了。

（4）CLS 语句（屏幕清除）

CLS 语句没有任何参数。在字符显示模式下，它把工作页面的内容全部清除，并使光标返回到屏幕左上角。在图形显示模式下，它把整个屏幕都置为所选择的底色，也就是说把屏幕上的全部画面都擦除干净。

2．屏幕坐标系统

为了给出欲画出的点或线等在屏幕上的位置，必须使用一个坐标系统。在 PC BASIC 语言中，该坐标系统以屏幕左上角为原点（0，0），横坐标的刻度为 0，1，2，…，319（或 639），纵坐标的刻度为0，1，2，…，199。在这种坐标系统中，画面上的任意一点的位置，均可由一对坐标（x，y）表示。其中 x，y 均为整数，且 0≤x≤319（或 639），0≤y≤199。这种坐标系统称为“屏幕坐标系统”，如图 8—1 所示。

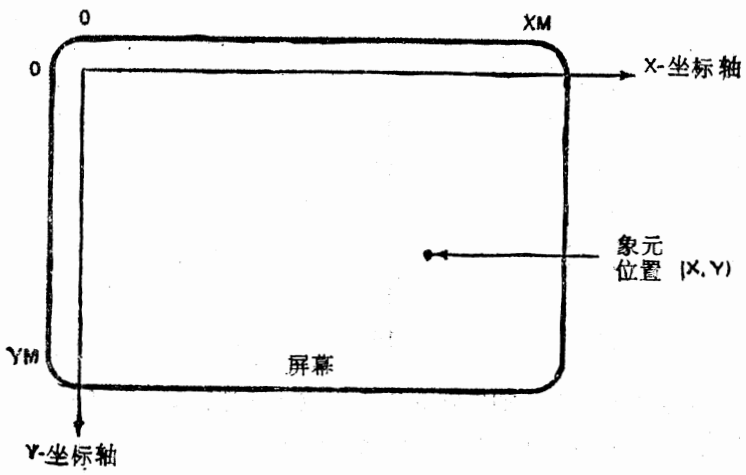


图 8—1　屏幕坐标系统

如果把 CRT 图形显示器看作为一台平板式绘图机的话，那么，屏幕就相当于一张画纸，另外它还有一支“虚拟”的画笔，画笔将按照程序给出的坐标数据在纸上移动并画出点、线、圆等等。每次操作之后，画笔就移动到一个新的位置，在执行下一条绘图命令之前，它将停留在那里不动。这个位置就叫做画笔的现行位置，或简称“现行位置”。

屏幕清除语句（CLS）使用之后，或者用 SCREEN 语句选择新的图形显示模式而引起屏幕清除之后，画笔的初始位置是（160，100）或（320，100），前者对应于中分辨率模式，后者对应于高分辨率模式。也就是说，屏幕被清除之后（可以比方为在绘图机上换了一张空白的画纸），画笔总是停在屏幕的中心位置处，等待着执行画图命令。

画图命令中的坐标数据通常可以有两种形式给出。一种是“绝对坐标形式”，它在 BASIC 语句中用（x，y）来表示，另一种是“相对坐标形式”，它用 STEP（Dx，Dy）来表示。PC BASIC 语言的画图语句中，凡需要给出坐标的地方，这两种形式都可以使用，它们的具体意义将在下面的各个有关画图语句中再作进一步的解释。

3．画点原语

前面已经讲过，IBM PC 的图形显示器是采用普通电视技术中的光栅扫描原理进行工作的。因此，CRT显示屏幕上出现的每一幅画面均由千千万万个点（象元）所组成，换句话说，计算机为了在屏幕上产生一幅图画，它必须具备的最基本的能力，是在屏幕的任意位置上用需要的颜色来画“点”。

IBM PC 的所有 BASIC 语言都具有如下两条用于画点的语句：

（1）PSET 语句（画点语句）

PSET 语句的形式是：

PSET（x，y），C

其中，（x，y）是欲画点的屏幕坐标，x、y 应为整型常数或表达式。如果它们的值超出了屏幕坐标系统所允许的范围，则本语句不起作用。参数 C 是该点的彩色码，其值应在 0 到 3（中分辨率）或 0 到 1（高分辨率）的范围之内，若缺省，则该点的彩色码就选用 3 或者 1。

点的位置也可以用相对坐标形式 STEP（Dx，Dy）来给出。假设画点之前画笔的现行位置为（Cx，Cy），则该点的实际位置就是（Cx+Dx，Cy+Dy）。

每次使用 PSET 画出一点后，画笔的现行位置就被修改，新的现行位置就是所画那一点的屏幕位置。

（2）PRESET 语句（擦点语句）

PRESET 语句的形式是：

PRESET（x，y），C

其中有关参数的含义及规定与 PSET 语句完全一样，实际上它的作用也是在指定的屏幕位置处按彩色码 C 画出一点。与 PSET 不同之处在于，当 C 缺省时，它的缺省值选用的是 0，亦即，使用底色画出一个点，这就起到了擦去指定位置上已画出的一个点的作用。

不难发现，PSET（x，y）和 PRESET（x，y）的作用正好相反，而 PSET（x，y），0 与 PRESET（x，y）以及PSET（x，y）与PRESET（x，y），3 的作用却完全相同。

（3）PSET 和 PRESET 的应用实例

下面是使用 PSET 和 PRESET 语句的 BASIC 程序的一个例子。它的功能是在图形显示器屏幕上随机地显示出一些点并进行闪烁，从而说明了前面已介绍过的SCREEN、CLS、PSET和 PRESET 等语句在程序中的使用方法。

例 8—1　闪烁的象元

```
10 'PROGRAM        BLINKING PIXELS
20  SCREEN 1: CLS
30  X = INT(320 * RND)
40  Y = INT(199 * RND)
50  PSET (X,Y)
60  FOR DELAY = 1 TO 500: NEXT
70  PRESET (X,Y)
80  GOTO 30
90  END
```

4. 画线原语

为了在屏幕上画出一条直线，通常可以有两种做法。一是使用 PSET（或 PRESET）语句来画线，这需要预先编制一个子程序；另一种是直接使用 BASIC 语句所提供的画线语句，这样更方便，速度也会快得多。

（1）画线子程序

画线的算法有多种，这里先介绍一种最简单的算法。从解析几何已知，一条直线的方程可以用它的斜率和截矩来表示：

$$Y=M*X+B$$

其中，M 为该直线的斜率，B 为它在 Y 轴上的截矩。如果已知该直线的起点和终点坐标分别为（X1，Y1）和（X2，Y2），则

$$M=(Y2-Y1)/(X2-X1)$$
$$B=Y1-M*X1$$

下面是使用 PSET 语句画直线的一个子程序。

```
10 'SUBROUTINE GENERAL LINE DRAWING USING PSET
20 M = (Y2 - Y1) / (X2 - X1)
30 B = Y1 - M * X1
40 FOR X = X1 TO X2
50   Y = M * X + B
60   PSET (X, Y)
70 NEXT
80 RETURN
```

注意，PSET 和 PRESET 以及下面将要介绍的一些画图语句，如果坐标值不是整数，则将被自动舍入成整数。

显然这个子程序有许多局限性。例如，当所画线段与 X 轴的夹角小于 45度 时，画出的直线质量较好，而当夹角大于45度时，所画直线的质量就较差（组成线条的点太稀），特别是在画垂直线时，该子程序根本不能工作。

下面是该子程序的改进，它采用直线的参数方程

$$X=X1+(X2-X1)*U$$
$$Y=Y1+(Y2-Y1)*U$$

作为算法的基础，子程序给出如下：

```
10 'SUBROUTINE IMPROVED LINE DRAWING ALGORITHM
20 DX = X2 - X1 : DY = Y2 - Y1
30 STEPS = INT(ABS(DX)) + 1
40 IF ABS(DX) < ABS(DY) THEN STEPS = INT(ABS(DY)) + 1
50 DXSTEP = DX / STEPS : DYSTEP = DY / STEPS
60 X = X1 : Y = Y1
70 FOR U = 1 TO STEPS
80   PSET (X, Y)
90   X = X + DXSTEP : Y = Y + DYSTEP
100 NEXT
110 PSET (X, Y)
120 RETURN
```

（2）LINE 语句（画线语句）

IBM PC 的 BASIC 语言可以直接使用 LINE 语句来画出一条直线，或者画出一个矩形。LINE 语句的形式有三种，第 1 种形式为：

LINE（X1，Y1）－（X2，Y2），C

其中（X1，Y1）和（X2，Y2）分别是直线的起点和终点，它们也可以使用相对坐标形式来给出。（X1，Y1）如果缺省的话，则表示起点位置就是画笔现行位置。参数 C 表示直线的彩色码，其缺省值为 3。下面是 LINE 语句的例子（假设为中分辨率模式）：

```
LINE（0，0）－（319，199）      {在屏幕上画出一条对角线}
LINE（160，100）－STEP（159，0）    {从中心点向右画一条水平线}
```

LINE 语句的第 2 种形式为：

LINE（X1，Y1）－（X2，Y2），C，B

它的作用是用来画出以（X1，Y1）和（X2，Y2）为对顶角的一个矩形，C 用来指出线条的彩色码，取值范围是 0 到 3。B 不是参数，它是一个标志字符，在语句中应照写，表示 LINE 语句的作用是画矩形而不是画线条。

LINE 语句的第 3 种形式与第 2 种完全相似，但标志字符为 BF：

LINE（X1，Y1）－（X2，Y2），C，BF

它表示不但要画出一个顶点为（X1，Y1）和（X2，Y2）的矩形，而且要用 C 选择的彩色去把这个矩形填满，所以，实际上画出的是长方块。

下面是使用 LINE 语句来画长方形和长方块的例子：

```
LINE（0，0）－（319，199），，B      {画出最大的一个矩形方框}
LINE（0，0）－STEP（160，100），1，BF      {画一个填满彩色 1 的长方块}
```

（3）DRAW 语句（连续画线语句）

一幅画往往是由许多线条组成的。因此，程序中需要大量地使用 LINE 语句。为了更方便地产生图形的显示输出，IBM PC的高级 BASIC语言提供了一条 DRAW 语句，它的功能较强，可以方便地用来连续画出各种直线。DRAW 语句的形式为：

DRAW　字符串

其中字符串由一系列的画图命令所组成，每个画图命令有两部分，一个命令字母以及一个数值 n。画图命令相互之间用分号隔开。可以使用的主要画图命令有：

* Cn　选择颜色 n，中分辨率时的 n 范围为0～3，高分辨率时为 0～1。
* Sn　设置比例因子。n 的范围是1～255，n 被 4 除是比例因子，其范围为1/4～64。
* An　设置画图的角度，n 可以是 0、1、2和 3，分别对应 0 度、90 度、180 度和 270 度。
* Un　向上画线。
* Dn　向下画线。
* Ln　向左画线。

* Rn　向右画线。
* En　向右上对角线画线。
* Fn　向右下对角线画线。
* Gn　向左下对角线画线。
* Hn　向左上对角线画线。
* Mx，y　从现行位置向（x，y）画线，若x前面冠以正负号，则表示以现行位置为起点，按相对坐标画线。

上面的U、D、L、R、E、F、G、H及M命令，它们在画线时所移动的点数，是n倍的比例因子。这些命令字母的前面还可以放一个前缀字母B，它表示只移动不画线。如果冠以前缀字母N，则表示在画线之后画笔又返回原来的起始位置。

所有命令中的常数n也可以改用变量，这时命令中的n将由“=变量名”来代替。

命令字符串中又可以包括一个或几个子命令字符串，这只要用X命令就行了，格式为：

X 字符串

这条命令非常有用，因为它在DRAW语句所画出的一个“对象”中，独立地定义了其中的某些部分，从而可以单独对它们进行某些处理。

下面是使用DRAW语句的一些简单例子（假设选择了中分辨率彩色显示模式）：

```
DRAW "R100; U100; L100; D100"        {画一个方框}
DRAW "R100; H50; G50"                {画出一个三角形}
DRAW "M260, 100; M210, 50; M160, 100"        {同上}
DRAW "M+100, 0; M-50, -50; M-50, 50"         {同上}
DRAW "R25; BR10; R25; BR10; R25"             {同一水平位置的三条直线}
DRAW "NE100; F100"                   {从同一点出发的两条斜线}
DRAW "BM110, 125; C2; R100; H50; G50"        {起点在(110, 125)用洋红色画出的一个三角形}
DRAW "BM125, 125; S2; L100"          {起点在(125, 125)向左画出一条长度50的水平线}
```

画线语句LINE和连续画线语句DRAW的完整的使用例子，将在下面给出。

（4）LINE和DRAW语句的应用实例

例8—2　使用LINE语句画五角星

程序：

```
10 'PROGRAM        CONNECTING RELATIVE STAR POINTS
20  SCREEN 2: CLS
30  READ X,Y
40  PSET (X,Y)
50  FOR P = 1 TO 5
60      READ X,Y
70      LINE - (X,Y)
80  NEXT
90  DATA 274,84,90,141,159,50,231,141,46,84,274,84
100 END
```

例 8—3　　使用 DRAW 语句画帆船

程序：

```
10 'PROGRAM        SAILBOAT USING DRAW STATEMENT
20  SCREEN 1: COLOR 0,0: CLS
30  DRAW "BM60,140R200G30L140H30"          'MAKE BOAT BOTTOM
40  DRAW "BM150,130R110H110D120"           'MAKE LARGE SAIL
50  DRAW "BM150,130L80E80"                 'MAKE SMALL SAIL
60  IF INKEY$ = "" THEN 60
70  END
```

运行结果：

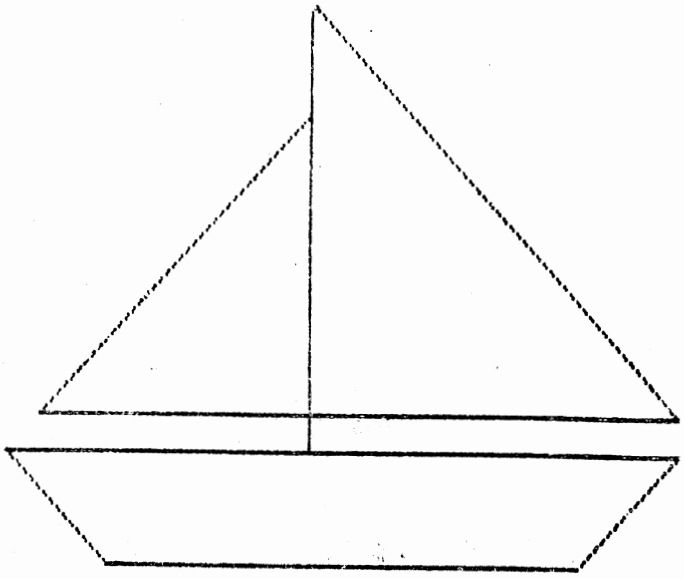


图 8—2　例 8—3 输出显示的帆船

例 8—4　　使用 DRAW 语句画古城堡

程序：

```
10 'PROGRAM          DRAW STATEMENTS USED TO DRAW CASTLE
20  SCREEN 1: COLOR 0,0: CLS
30  TURRET$ = "U10 R10 D10 R10"
40  TURRET$ = TURRET$ + TURRET$ + "U10 R10 D10"
50  MAINTURRET$ = TURRET$ + "R10" + TURRET$ + "R10" + TURRET$
60  LEFTSMALLTURRET$ = "S2" + TURRET$ + "S4 R5"
70  RIGHTSMALLTURRET$ = "R5 S2" + TURRET$ + "S4"
80  FLAG$ = "U15; M 180,12; M 160,15"
90  DRAW "BM75,190 L50 U90; X TURRET$; D15 R20 U30; X LEFTSMALLTURRET$;"
100 DRAW "U30 M160,20; X FLAG$; D5 M195,55 D30; X RIGHTSMALLTURRET$;"
110 DRAW "D30 R20 U15 X TURRET$; D90 L220 U55; X MAINTURRET$; D55"
120 DRAW "BM140,190; U20; M150,162; M160,160; M170,162; M180,170; D20"
130 WINDOW$ = "R5 D10 L5 U10"
140 DRAW "BM100,90; X WINDOW$;"
150 DRAW "BM110,90; X WINDOW$;"
160 DRAW "BM205,90; X WINDOW$;"
```

```
170 DRAW "BM256,105; S5; X WINDOW$;"
180 DRAW "BM276,105; X WINDOW$;"
190 DRAW "BM130,170; S3; X WINDOW$;"
200 DRAW "BM120,170; X WINDOW$;"
230 IF INKEY$ = "" THEN 230
240 END
```

运行结果：

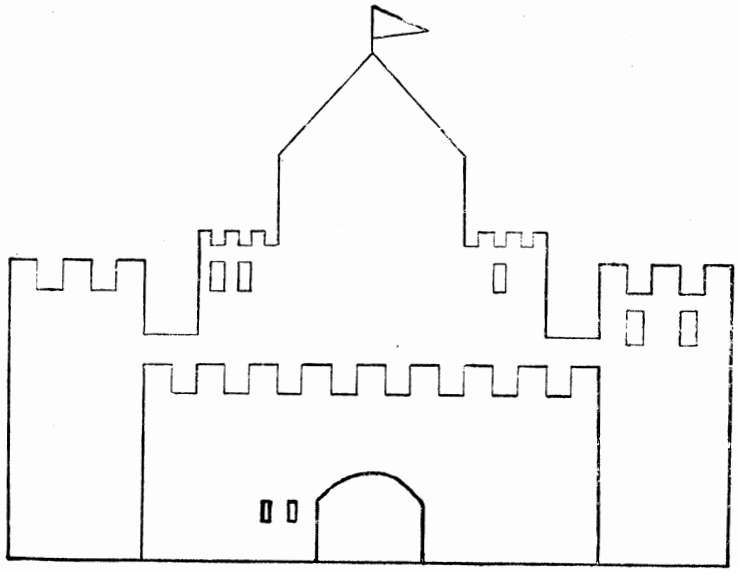


图8—3 古城堡

说明：

①本例中DRAW语句使用了许多X命令，它可以用来画出图画中一些重复出现的或者比较独立的子图画，例如，TURRET$（塔楼），WINDOW$（窗），FLAG$（旗）等等。

② S命令用来改变画图的尺寸比例，从而可以节省许多图形数据。

5．圆、圆弧及曲线的显示

（1）画圆的算法

在图形显示器上显示圆或圆弧的算法有多种。假设已知圆心的坐标是（XC，YC），圆的半径是R，则圆的参数方程可以给出如下：

$$X=XC+R*COS(A)$$
$$Y=YC+R*SIN(A)$$

其中参数A表示角度，它的单位是弧度，在画圆时它的变化范围为$0\sim2\pi$。利用这个参数方程，当A从0离散地变化到2π时，即可算出圆上每一点的坐标，并利用PSET语句把它们逐点显示出来，就能得到一个由一系列点所组成的一个圆。或者，利用参数方程算出圆上各点的坐标后，利用LINE语句把相邻两点之间连成一线，这样也就产生了一个由多边形所逼

近的圆。

显然，无论采用哪种方法，参数A的变化（即角增量）越大，则画得就越快，但画出的圆越不精确；变化越小，画出的圆就越精确，但画得也就越慢。如何选择一个合适的角增量呢？参见图8—4。为了使圆相邻两点的距离为1，则有：

$$SIN(DA)=1/R$$

因为在角度很小的情况下，SIN（DA）≈DA，所以角增量：

$$DA\approx 1/R$$

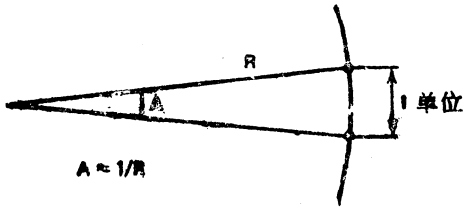


图8—4　角增量与半径之间的关系

在画圆的过程中，由于需要完成许多计算，因而影响画圆的速度。为了减少计算量，从图8—5中可以看出，由于圆的对称性，圆上任何一点的坐标（X，Y）被计算出来之后，

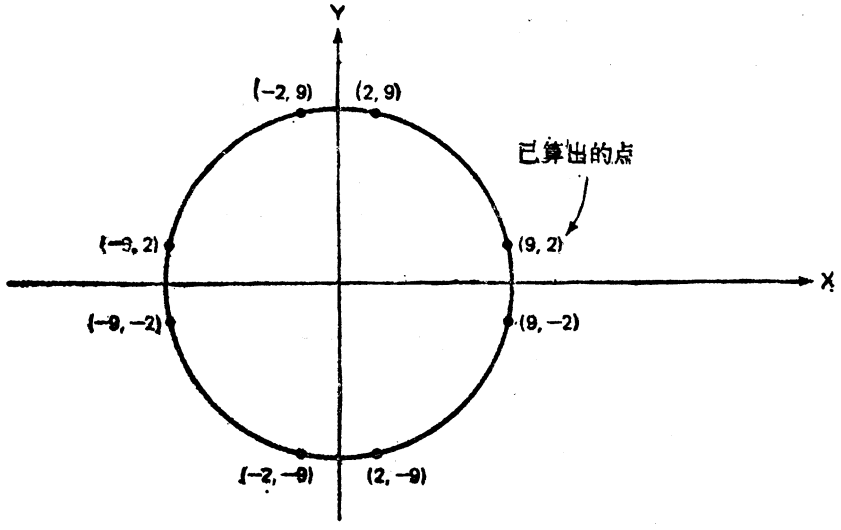


图8—5　八路对称算法原理

与它对称的圆上另外七点的坐标也极易推算得到，它们分别是（X，−Y）、（−X，Y）、（−X，−Y）、（Y，X）、（Y，−X）、（−Y，X）和（−Y，−X）。因此，在画圆的过程中，只要计算出角度A从0到π/4范围中各点的坐标，就可以得到圆上所有各点的坐标，从而大大加快了画圆的过程，这种方法叫做“八路对称法”。下面是利用这种算法画圆

的一个子程序：

```
10 'SUBROUTINE CIRCLE GENERATOR
20 DA = 1 / R
30 RADIN45 = 45*3.141593 / 180
40 FOR ANGLE = 0 TO RADIAN45 STEP DA
50     DX = R * CON(ANGLE)
60     DY = R * SIN(ANGLE)
70  GOSUB 100
80 NEXT
90 RETURN
100    'PLOT ALL SYMMETRIC POINTS
110 PSET ( XC + DX, YC + DY * .919999)
120 PSET ( XC - DX, YC + DY * .919999)
130 PSET ( XC + DX, YC - DY * .919999)
140 PSET ( XC - DX, YC - DY * .919999)
150 PSET ( XC + DY, YC + DX * .919999)
160 PSET ( XC - DY, YC + DX * .919999)
170 PSET ( XC + DY, YC - DX * .919999)
180 PSET ( XC - DY, YC - DX * .919999)
190 RETURN
```

其中，语句110～180中常数 .919999 的作用是使得画出的圆不至于变成椭圆，其原因以后再作解释。

（2）CIRCLE 语句（画圆语句）

在高级 BASIC 语言中，可以使用 CIRCLE 语句直接画出圆或者圆弧，它的语法为：

CIRCLE (XC, YC), R, C, AS, AE, ASP

其中，(XC,YC) 为圆心的坐标，R 为半径，C 为彩色码，AS 和 AE 分别为圆弧的起始角和结束角（单位为弧度）。取值范围为 $-2\pi \sim 2\pi$，以逆时钟方向进行度量，如图 8—6 所示。当角度为负值时，表示圆弧的端点应与圆心相连。如果参数 AS 或 AE 缺省，则其缺省值为 0。

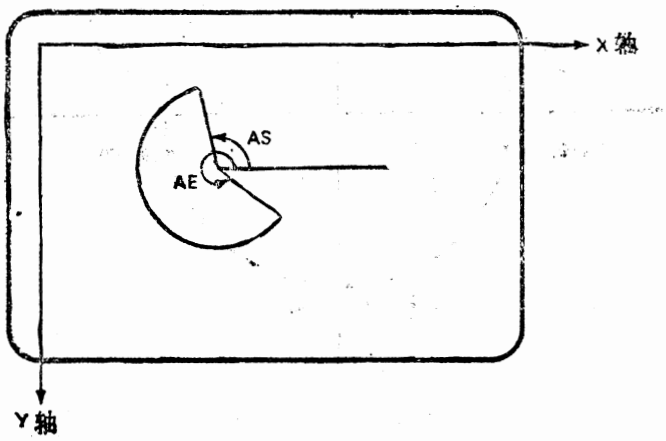


图 8—6　圆弧的起始角与结束角

CIRCLE 语句中最后一个参数 ASP 表示圆的“纵横比”，这是为了避免屏幕上纵向两点距离与横向两点距离的不等（在 IBM PC 上，前者大于后者）造成圆变成椭圆的现象而设置的。如果给出参数 ASP 的话，则 Y 方向的半径应为 R * ASP。当 ASP 参数缺省时，

若显示模式为中分辨率，则缺省值为5/6，若显示模式为高分辨率，则缺省值为5/12。

下面是使用CIRCLE语句的几个简单例子：

CIRCLE（30，10），10　　{以（30，10）为圆心，半径为10的一个圆}

CIRCLE（160，100），75,,1.57080，4.71239　　{以（160，100）为圆心，半径为75的左半圆}

CIRCLE（160，100），75,,,,1　　{以（160，100）为圆心，半径（实际上为短轴）为75的椭圆}

（3）画圆语句的应用实例

例8—5　使用CIRCLE语句画“月中人”

程序：

```
10 'PROGRAM      MAN IN THE MOON FROM DIFFERENT-SHAPED ARCS
20  SCREEN 2: CLS
30  CIRCLE (250,100),200,,4.9,1.4,.37              'OUTER CURVE
40  CIRCLE (250,100),90,,5.1,5.73,.8499999         'CHIN
50  CIRCLE (250,100),90,,.2,1.2,.8499999           'FOREHEAD
60  CIRCLE (330,110),16,,2.2,5.2,.4                'NOSE
70  CIRCLE (320,90),20,,4.8,.5,.7                  'BRIDGENOSE
80  CIRCLE (250,100),90,,5.85,6.05,.8499999        'MOUTH-TO-NOSE
90  CIRCLE (400,110),35,,1.9,4.7,.4                'CHEEK
100 CIRCLE (345,115),45,,4.4,5.4,.35               'MOUTH TOP
110 CIRCLE (335,126),35,,4.5,6.05,.35              'MOUTH BOTTOM
120 CIRCLE (360,90),9                              'EYE
140 CIRCLE (400,110),35, ,1.9,4.7,.4               'CHEEK
150 CIRCLE (360,90),18,, ,.3,2.5,.9                'EYEBROW
160 IF INKEY$ = "" THEN 160
170 END
```

运行结果：

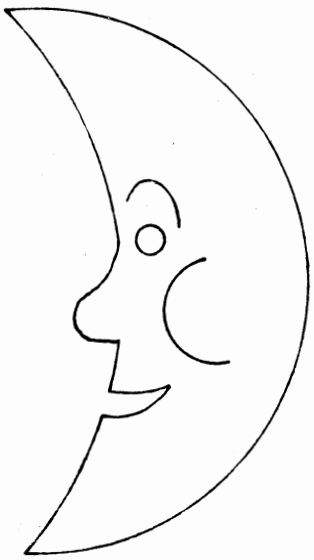


图8—7　使用CIRCLE语句画出的“月中人”

例 8—6 装饰图案

程序：

```
10 'PROGRAM       CURVE PATTERNS
20  SCREEN 1: COLOR 0,0: CLS
30      '********************** CLOVER ***********************************
40  COLCODE = 2
50  XCENTER = 160: YCENTER = 100
60  FOR RADIUS = 20 TO 50 STEP 15
70      PSET (XCENTER,YCENTER),COLCODE
80      FOR ANGLE = 0 TO 6.28318 STEP 1/RADIUS
90          R1 = RADIUS * SIN(2 * ANGLE)
100         X = XCENTER + R1 * COS(ANGLE)
110         Y = YCENTER + R1 * SIN(ANGLE)
120         LINE - (X,Y),COLCODE
130     NEXT
140 NEXT
150 END
```

运行结果：

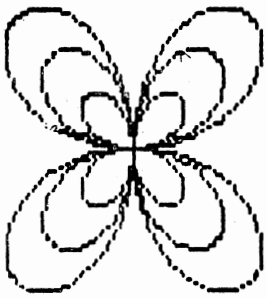


图 8—8 装饰图案

例8—7 利用 CIRCLE 语句画扇形统计图

程序：

```
10  PROGRAM       PIECHART.
20  DIM SECTION$(8), VALUE(8)
30  SCREEN 0: WIDTH 80: CLS
40  INPUT "CENTER COORDINATES FOR PIECHART"; XCENTER,YCENTER
50  INPUT "RADIUS"; RADIUS
60  IF XCENTER+RADIUS > 319 OR XCENTER-RADIUS < 0 OR YCENTER+RADIUS > 199
      OR YCENTER-RADIUS < 0 THEN 580
70  INPUT "TITLE OF CHART"; TITLE$
80  INPUT "NUMBER OF DIVISIONS (UP TO 8)"; N
90  PRINT "ENTER NAME AND VALUE FOR EACH DIVISION"
100 TOTAL = 0
110     'INPUT DATA. FIND TOTAL OF ALL VALUES
120 FOR K = 1 TO N
130     INPUT SECTION$(K), VALUE(K)
140     TOTAL = TOTAL + VALUE(K)
150 NEXT
160 SCREEN 1: CLS
170 COLUMN = 20 - INT(LEN(TITLE$) / 2 + .5)      'CENTER TITLE
180 LOCATE 1,COLUMN: PRINT TITLE$
190 CIRCLE (XCENTER,YCENTER),RADIUS,,,,.9199999
200 BEFORE = 0              'BEFORE IS ANGLE THAT DETERMINED PRECEDING LINE
210 FOR K = 1 TO N
220         'LINE TO PLOT IS BASED ON THE PERCENTAGE OF THE
230         'CIRCLE EQUAL TO VALUE(K) / TOTAL PLUS THE
240         'PRECEDING DIVISIONS
250     ANGLE = BEFORE + 6.28318 * VALUE(K) / TOTAL
```

```
260     XP = XCENTER + RADIUS * COS(ANGLE)
270     YP = YCENTER + RADIUS * SIN(ANGLE) * .9199999
280     LINE (XCENTER,YCENTER) - (XP,YP)
290         'PUT LABEL ON DIVISION
300         'FIND A POINT 4 UNITS OUTWARD FROM THE CENTER
310         'POINT OF THIS DIVISION'S ARC
320     BISECT = BEFORE + (ANGLE - BEFORE) / 2          'BISECT IS THE ANGLE
330     XLABEL = XCENTER + (RADIUS + 4) * COS(BISECT)   'WHOSE LINE WOULD
340     YLABEL = YCENTER + (RADIUS + 4) * SIN(BISECT)   'HALVE THIS DIVISION
350         '(XLABEL,YLABEL) IS THE POINT USED TO ANCHOR LABEL
360         'USE THE POINT AS START OF LABEL IF IT'S ON RIGHT
370         'SIDE OF CIRCLE, AS END OF LABEL IF IT'S ON LEFT,
380         'AS MIDPOINT IF IT'S ON TOP OR BOTTOM OF CIRCLE
390         'POINT IS START OF LABEL
400     IF XLABEL > XCENTER + 10 THEN 500
410         'POINT IS END OF LABEL
420     IF XLABEL < XCENTER - 10 THEN 470
430         'OTHERWISE POINT IS MIDPOINT OF LABEL. ADJUST XLABEL BY
440         'ONE-HALF THE NUMBER OF PIXELS NEEDED FOR LABEL
450     XLABEL = XLABEL - LEN(SECTION$(K)) / 16
460     GOTO 500
470         'POINT IS END OF LABEL. MOVE BACK BY THE NUMBER
480         'OF PIXELS REQUIRED FOR LABEL
490     XLABEL = XLABEL - LEN(SECTION$(K)) * 8
500         'CONVERT THE PIXEL LOCATION (XLABEL,YLABEL) TO THE CLOSEST
510         'CORRESPONDING PRINT POSITION
520     ROW = INT(YLABEL / 8) + 1
530     COLUMN = INT(XLABEL / 8) + 1
540     LOCATE ROW,COLUMN: PRINT SECTION$(K);
550     BEFORE = ANGLE              UPDATE BEFORE TO REFLECT DOING THIS DIVISION
560 NEXT
570 GOTO 590
580 PRINT "COORDINATE OUT OF RANGE"
590 IF INKEY$="" THEN 590
600 END
```

说明：

这是利用 CIRCLE 语句编制的画扇形统计图的一个通用程序。图的绘制虽比较简单，但图中各部分标号的处理却较为复杂。每个标号的起始打印位置由它的长度以及它所对应的

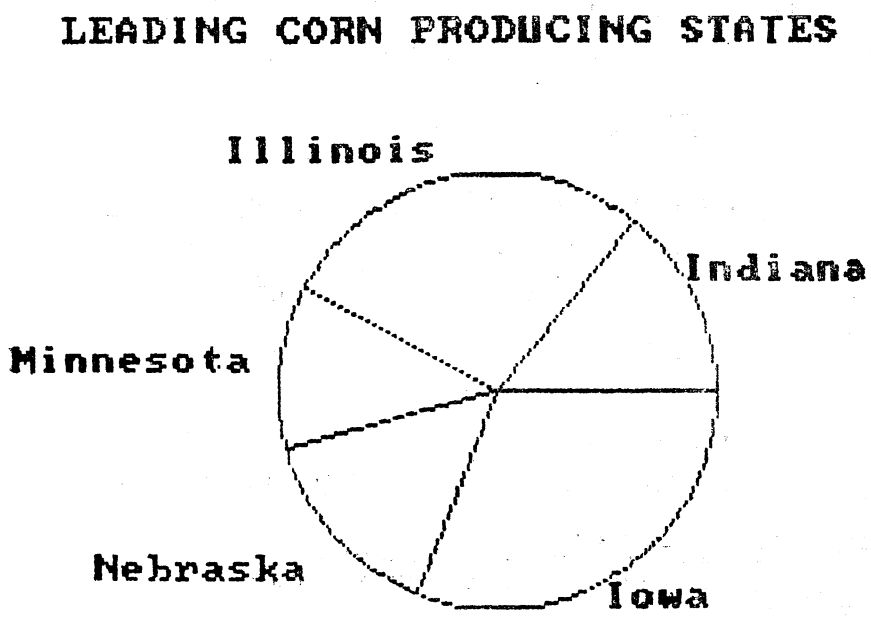


图8—9　扇形统计图的绘制

扇区的位置所决定，标号本身由 PRINT 语句输出。

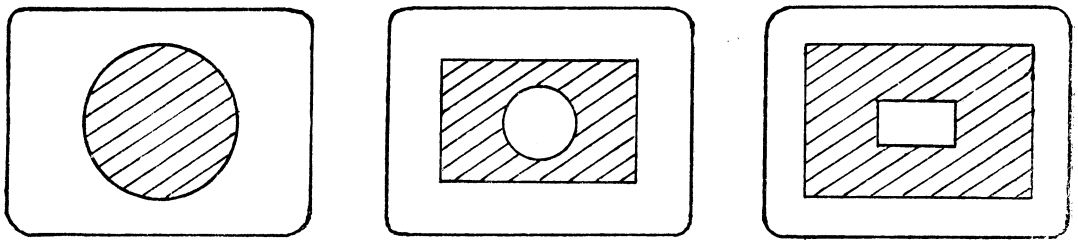
运行结果：见图 8—9。

6．着色、涂阴影及其它

前面所介绍的画点、画线、画圆或画圆弧等语句，已经可以用来构造出许多复杂的图画了。但是，这些图画大多数都仅仅是由各种线条所组成（称为“线条图”）。为了增加画面的效果，常常需要采用着色、涂阴影等技术，下面对此作简单介绍。

（1）着色

所谓“着色”，是指把画面上某一封闭区域的内部或外部涂满某一种指定的颜色。图 8—10 给出了三种情况：（a）表示在某一边界线内着色。（b）表示在两个边界线之间着色，这时，两个边界线本身的颜色应该相同。（c）表示在边界线之外进行着色，直到屏幕边框线为止。



(a)　　(b)　　(c)

图 8—10　　着色的三种情况

矩形的着色很简单。假设（X1，Y1）和（X2，Y2）分别是某个矩形的左上角和右下角的坐标，则使用下面的循环语句即可实现矩形的着色：

```
FOR X = X1 TO X2
    LINE (X, Y1) - (X, Y2), BOXCOLOR
NEXT
```

实际上前面已经介绍过，IBM PC 的 BASIC 语言直接提供了矩形着色的语句。因此，上面的语句也就可以用下面的语句来代替：

LINE（X1，Y1）－（X2，Y2），BOXCOLOR，BF

圆的着色也是不难解决的，因为圆周上的任意点都应满足：

（X－XC）^2＋（Y－YC）^2＝R^2

因此，每一个在 XC－R 到 XC＋R 范围中的 X，都有两个纵坐标与之对应，从而可以画出一条垂直线（弦）。画出所有垂直线，便实现了圆的着色。下面是程序的片断：

```
FOR X = XC - R TO XC + R
    MIDY = SQR(R * R - (X - XC) * (X - XC))
    LINE (X, YC - MIDY) - (X, YC   MIDY), CCOLOR
NEXT
```

为了解决更一般的着色问题，PC 的高级 BASIC 提供了一条 PAINT 语句：

PAINT（X，Y），CP，CB

其中，（X，Y）为边界线内的任意一点（称为内点）的坐标，它可以是绝对坐标形式，也可以是相对坐标形式。CP表示需要在此区域内着的颜色，CB表示边界线的颜色。如果参数CP缺省，则其缺省值为3（中分辨率）或1（高分辨率）；如果参数CB缺省或CB不是任何边界线的颜色，则着色将对整个屏幕进行。如果边界线上有“孔”，则着的彩色会从孔中漏出，并沿着边界线进行着色。

如果在例8—4（画古城堡）的程序中再增加两条语句：

```
210 PAINT (160, 140), 3, 3
220 PAINT (160, 100), 2, 3
```

则着了色的古城堡更具有立体感，其输出图形如图8—11所示。

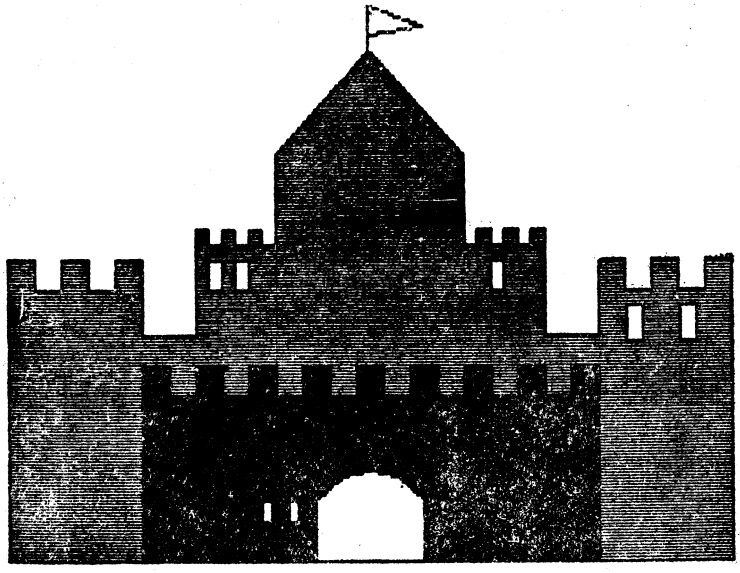


图8—11　着色后输出显示的古城堡

图8—12　修改后的月中人

同样，例 8—5 程序中也可增加一条 PAINT 语句，并修改 140、150 两条语句，使得“月中人”的画面更加形象（图 8—12 ）。语句的修改如下：

```
130 PAINT (355, 110), 1, 1
140 CIRCLE (400, 110), 35, 0, 1.9, 4.7, 0.4
150 CIRCLE (360, 90), 18, 0, 0.3, 2.5, 0.9
```

（2）涂阴影

为了取得更好的画面效果，除了着色以外，有时还可以在某些区域内涂上阴影，甚至画上一定的纹理（图 8—13 ）。涂阴影和画纹理的方法有多种，可以视实际需要而定，但大多数都只是着色技术的一些变化。例如，在着色的算法中，每隔一条或两条线着色一次，或者使间隔逐渐增加或减小，就能得到明暗有变化的阴影。在边线所规定的区域内，也可以用画点的方法来产生阴影，它们的间距可以是有规则地变化，也可以是随机地变化，均能达到预定的效果。如果使用不同颜色的点，则效果更佳。例如，轮流画出红色和棕色的点子，便可产生一个桔黄色的阴影区。

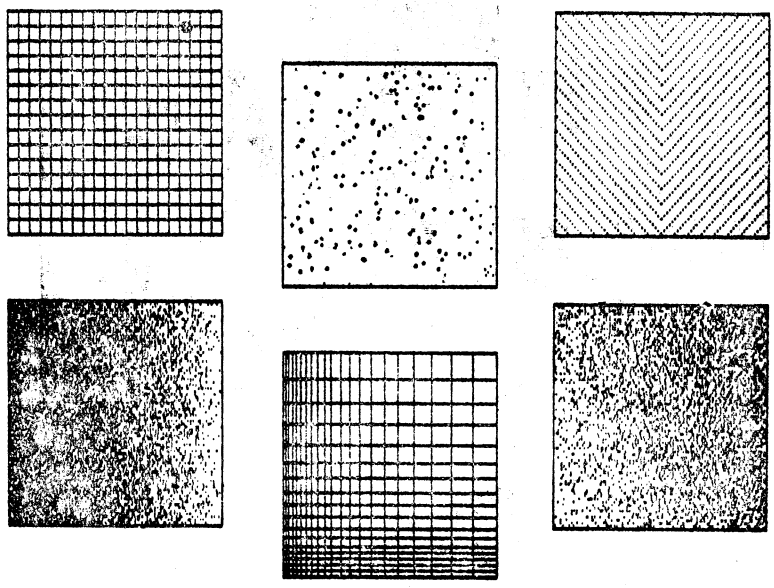


图 8—13　阴影和纹理

例 8—8 是一个使用阴影技术的条形图显示程序。该条形图用来表示三个不同地区每季度的商品营业额。不同地区在图上用不同阴影的矩形来表示，矩形的大小则代表着营业额的多少。这种条形图具有清晰、直观、对比性较强的优点。

例 8—8　涂阴影的条形图

程序：

```
10 'PROGRAM      SHADED COLUMN CHART USING PIXEL GRAPHICS
50  SCREEN 2: CLS
60  PRINT TAB(28); "QUARTERLY SALES BY REGION"
```

```
70  LINE (128,24) - (608,162),,B          'MAKE BOX FOR CHART
80  FOR Y = 36 TO 156 STEP 8              'MAKE NOTCHES FOR LABELING
90      LINE (125,Y) - (131,Y)
100 NEXT
110 ROW = 20
120 FOR SALESAMOUNT = 0 TO 30 STEP 10              'LABEL THE NOTCHES
130     LOCATE ROW,13: PRINT USING "##"; SALESAMOUNT
140     ROW = ROW - 5
150 NEXT
160     'LABEL QUARTERS
170 LOCATE 23,22: PRINT " FIRST       SECOND        THIRD       FOURTH"
180 PRINT TAB(22); "QUARTER     QUARTER     QUARTER     QUARTER";
190 LOCATE 12,3: PRINT "SALES"
200 LOCATE 13,1: PRINT "(millions)"
210 LOCATE 8,67: PRINT "WEST"                      'LABEL REGIONS
220 LOCATE 13,67: PRINT "SOUTH"
230 LOCATE 18,67: PRINT "MIDWEST"
240     'CONSTRUCT BARS, ONE FOR EACH QUARTER
250 RANGERATIO = (156 - 36) / (30 - 0)
260 XLEFT = 160
270 XRIGHT = XLEFT + 64
280 FOR QUARTER = 1 TO 4
290     YBOTTOM = 156
340     ADJUSTMENT = 0
350     FOR DISTRICT = 1 TO 3
360         READ SALES
370         'CONVERT TO MILLIONS
380         SALES = SALES / 1000000!
390         YTOP = INT((30 - SALES) * RANGERATIO + 36 + .5)
400         'ADJUST YTOP BY ADJUSTMENT - THE AREA THAT HAS ALREADY
410         'BEEN TAKEN UP BY PREVIOUS REGIONS SALES MAGNITUDES.
420         YTOP = YTOP - ADJUSTMENT
430         LINE (XLEFT,YTOP) - (XRIGHT,YBOTTOM),,B
440         'FILL IN UPPER LEFT  TRIANGLE
450         FOR X1 = XRIGHT TO XLEFT STEP -DISTRICT * 3
460             X = X1
470             Y = YTOP
480             PSET (X,Y)
490             Y = Y + 1
500             X = X - 1
510             IF Y <= YBOTTOM AND X > XLEFT THEN 480
520         NEXT
530         'FILL IN LOWER RIGHT TRIANGLE
540         FOR Y1 = YTOP TO YBOTTOM STEP DISTRICT * 3
550             Y = Y1
560             X = XRIGHT
570             PSET (X,Y)
580             Y = Y + 1
590             X = X - 1
600             IF Y < = YBOTTOM AND X > XLEFT THEN 570
610         NEXT
620         'FIND ADJUSTMENT THAT WILL BE NEEDED FOR NEXT YTOP
630         ADJUSTMENT = 156 - YTOP
640         YBOTTOM = YTOP
650     NEXT
660     'ADVANCE ACROSS TO NEXT QUARTER'S COLUMN
670     XLEFT = XLEFT + 96
680     XRIGHT = XLEFT + 64
690 NEXT
700 DATA 7000000,11000000,4000000
710 DATA 8000000,10500000,7000000
720 DATA 4000000,12000000,8500000
730 DATA 7000000,11333000,10500000
740 IF INKEY$ = "" THEN 740
750 END
```

运行结果：见图 8—14。

说明：

本例选择的是高分辨率图形显示模式，每季度总营业额的范围为 0～30 兆元，对应的纵坐标变化范围是 156～36。阴影由点组成，涂阴影是分两步进行的，先涂矩形中的左上方三角

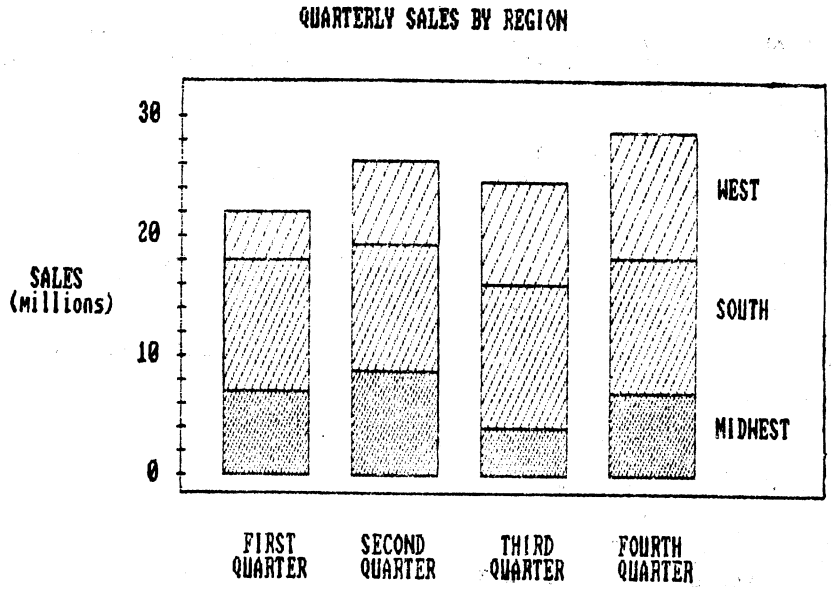


图 8—14　有阴影的条形图

区，再涂右下三角区，它们分别与程序中的 440—520 语句及 530—650 语句相对应。

第二节　交互式图形显示技术

在上一节所介绍的图形显示技术中，用于生成图形的数据或者直接来自程序中的DATA语句，或者间接地由程序计算所产生，这种构造图形的方式，称为“被动式图形显示”。但是在实际应用中，用户往往需要在图形显示期间，能够对画面上的某些部分进行修改（例如改变颜色，改变形状，移动位置，旋转角度等），删除或者增添某些部分。还有些用户甚至希望整个画面全部由自己通过一定的操作或命令来动态地生成，就好象画家在画布上作画一样。这种构造图形的技术，称为“交互式图形显示技术”（简称“交互式作图”）。本节先介绍几个有关交互式作图的基本概念，然后通过一些实例，说明在 IBM PC 机上如何使用键盘、光笔和操纵杆来进行交互式作图。

1．基本概念

在文本编辑程序或者文字处理程序等软件的作用下，用户对一段程序、一篇公文、一张表格或一封信件的处理（建立、修改、格式调整、删除等等）一般总是通过控制台或终端的键盘来进行的。键盘是文字处理的最主要的设备，它可以方便地完成对文本的各种处理和操作。但是在交互式作图的操作过程中，单有键盘还不够，因为在一幅图画的组成中，不但有字符（在图中用作标号或注解），而且主要包含有大量的点、线、圆弧等，对这些图形信息的处理和操作要比对字符复杂得多，因此使用的设备也就相应要复杂一些。

用于交互式作图的设备叫做“图形输入设备”或“图形交互设备”。正如下面几节将要说明的那样，实际上它们多数并不具备把图形信息输入到计算机去的功能，只是在软件的作

用下，用户在宏观上感觉到它们具有画图的能力而已。

图形输入设备有多种多样，如光笔，操纵杆，操纵球，电位器，按钮，图形数字化桌，鼠形光标定位器（俗称“老鼠”），摸感屏，键盘等等。按照它们的逻辑功能来划分，通常可把图形输入设备分成下列五类：

① 定位设备　用来指出画面上的某一特定位置，以便在该位置上添加图形、符号、或进行其它有关操作。

② 选择设备　用来指出画面上的某一特定的成分，如一条线段、一个圆弧、一个图形区、一个机械零件等，以便对该画面成分进行放大、缩小、旋转、移动、删除等操作。

③ 定标设备　用于向系统输入一个实数值，以便指出放大或缩小的倍数，旋转的角度，移动的距离等。

④ 字符设备　输入字符串，用于在画面上添加标号或注解。

⑤ 命令设备　用于向系统发出各种操作命令，实现对图形的各种操作。

不同的物理设备，由于其构造和特性不同，它们往往只具有上述的某一种或两种逻辑功能。例如，光笔主要用作选择设备，操纵杆或操纵球、数字化桌、光标定位器等主要用作定位设备。由于它们常常附带有按钮，因此也可兼做命令设备，电位器可用作定标设备，键盘是字符设备，按钮或键盘上的功能键是命令设备。但是，在交互式作图的过程中，五种类型的逻辑功能常常都必须具备。不过限于成本方面的原因，微型机一般并不配置所有各种类型的物理设备，只能选择某一种或两种进行配置。因此，必须利用软件使已有的某种物理设备来仿真其它设备的功能，有关的技术将在下面几节中详细叙述。

2．使用键盘的交互式技术

键盘是一种最便宜的图形输入设备，它的功能键可以作为命令设备，数字键可以作为定标设备。在一定的软件配合下，利用键盘上的光标键（←，→，↑，↓），可以使键盘起着一个定位设备的作用。例 8—9 就是利用键盘作为定位设备来交互式地进行画图的。

例 8—9　利用键盘进行素描

程序：

```
10 'PROGRAM      INTERACTIVELY SKETCHES PICTURES.
70  SCREEN 0: WIDTH 80: CLS
180 PRINT "0 - BLACK     4 - RED          8 - GRAY          12 - LIGHT RED      "
190 PRINT "1 - BLUE      5 - MAGENTA      9 - LIGHT BLUE    13 - LIGHT MAGENTA "
200 PRINT "2 - GREEN     6 - BROWN       10 - LIGHT GREEN   14 - YELLOW         "
210 PRINT "3 - CYAN      7 - WHITE       11 - LIGHT CYAN    15 - INTENSE WHITE "
220 PRINT
230 INPUT "COLOR CHOICE FOR BACKGROUND"; BACKGROUND
240 IF BACKGROUND >= 0 AND BACKGROUND <= 15 THEN 260
250 PRINT "INVALID COLOR CHOICE. TRY AGAIN": GOTO 230
260 PRINT
270 PRINT "       1 - GREEN       4 - CYAN     "
280 PRINT "       2 - RED         5 - MAGENTA  "
290 PRINT "       3 - BROWN       6 - WHITE    "
300 PRINT
310 INPUT "COLOR CODE FOR FOREGROUND"; FORE
320 IF FORE >= 1 AND FORE <= 6 THEN 340
330 PRINT "INVALID COLOR CHOICE. TRY AGAIN": GOTO 310
340 IF FORE = 1 OR FORE = 2 OR FORE = 3 THEN PALETTE = 0 ELSE PALETTE = 1
350 IF FORE = 1 OR FORE = 4 THEN DRAWCOLOR = 1
360 IF FORE = 2 OR FORE = 5 THEN DRAWCOLOR = 2
370 IF FORE = 3 OR FORE = 6 THEN DRAWCOLOR = 3
380 PRINT
```

```
390 PRINT "HIT THE DOWN  ARROW TO EXTEND YOUR LINE DOWNWARD"
400 PRINT "HIT THE  UP   ARROW TO EXTEND YOUR LINE UPWARD"
410 PRINT "HIT THE RIGHT ARROW TO EXTEND YOUR LINE TO THE RIGHT"
420 PRINT "HIT THE LEFT  ARROW TO EXTEND YOUR LINE TO THE LEFT"
430 PRINT
440 INPUT "INPUT STARTING COORDINATES"; X,Y
450 SCREEN 1: COLOR BACKGROUND, PALETTE: CLS
460 LINE (0,0) - (319,199),DRAWCOLOR,B          'DRAW BOX AROUND SCREEN EDGES
470 IF X < 0 OR X > 319 OR Y < 0 OR Y > 199 THEN 580
480 PSET (X,Y),DRAWCOLOR
490 IF DRAWING$ = "N" THEN FOR J=1 TO 50: NEXT: PRESET (X,Y)     'MOVING TO
500 A$ = INKEY$: IF A$ = "" THEN 500                             'NEW POSITION
510 IF RIGHT$(A$,1) = CHR$(80) THEN Y = Y + 1: GOTO 480          'DOWN
520 IF RIGHT$(A$,1) = CHR$(72) THEN Y = Y - 1: GOTO 480          'UP
530 IF RIGHT$(A$,1) = CHR$(77) THEN X = X + 1: GOTO 480          'RIGHT
540 IF RIGHT$(A$,1) = CHR$(75) THEN X = X - 1: GOTO 480          'LEFT
550 IF A$ = "D" OR A$ = "d" THEN DRAWING$ = "Y": GOTO  490
560 IF A$ = "B" OR A$ = "d" THEN DRAWING$ = "N": GOTO 490
570 GOTO 470
580 IF INKEY$ = "" THEN 580
590 END
```

说明：

①本程序的功能是让用户利用键盘在屏幕上画图，选择的显示模式为中分辨率，屏幕的底色及画图的颜色可由用户指定，但一经指定便不能改动。

②操作开始时，屏幕上为空白，画笔的起始位置由用户输入确定。程序提供了两种操作模式："D"或"d"命令（画图模式）和"B"或"b"命令（擦图模式）。通过键盘输入可以任意选择某一模式。在画图模式下，可以移动画笔并画出图来。在擦图模式下，移动画笔时并不画图，若原来位置上有图时还会把它擦去。

③画笔的移动（定位功能）是由键盘上的四个光标键来控制的，每按一键，画笔即往指定方向移动一个单位。在擦图模式下，为了让用户知道画笔在画面上的位置，画笔移动后，总是在停留位置处先画出一个点，稍过片刻再把它擦去。

④草图画毕后，可以打入"E"或"e"命令结束本程序的工作。

例 8—9 是利用键盘进行交互式作图的一个简单例子，显然这是很不实用的。因为一幅图画如果完全这样一"点"一"点"地来画出，则太浪费时间了。为此可利用键盘先画线，再由线条组成一幅图画，参见例 8—10 。

例 8—10　　利用线条交互式作图

程序：

```
10 'PROGRAM        INTERACTIVE PICTURE CONSTRUCTION WITH LINES
70  DIM X1(20), Y1(20), X2(20), Y2(20), LINECOLOR(20)
80  SCREEN 1: CLS
90  COUNT = 1     'COUNT IS NUMBER OF LINES
100       'INPUT COORDINATES OF EACH LINE
110 PRINT "ENTER -1,-1 TO QUIT"
120 INPUT "FIRST POINT OF LINE"; X, Y
130 IF X = -1 AND Y = -1 THEN 420
140 IF X >= 0 AND X <= 319 AND Y >= 0 AND Y <= 199 THEN 160
150 PRINT "FIRST POINT OUT OF RANGE. TRY AGAIN": GOTO 120
160 X1(COUNT) = X: Y1(COUNT) = Y
170 INPUT "SECOND POINT OF LINE"; X, Y
180 IF X = -1 AND Y = -1 THEN 420
190 IF X >= 0 AND X <= 319 AND Y >= 0 AND Y <= 199 THEN 210
200 PRINT "SECOND POINT OUT OF RANGE. TRY AGAIN": GOTO 170
210 X2(COUNT) = X: Y2(COUNT) = Y
220       'CHOOSE COLOR OF LINE
230 INPUT "COLOR CODE FOR LINE (1, 2, OR 3)"; COLCODE
240 IF COLCODE = 1 OR COLCODE = 2 OR COLCODE = 3 THEN 260
250 PRINT "COLOR CHOICE INVALID. TRY AGAIN": GOTO 230
260 LINECOLOR(COUNT) = COLCODE
```

```
270 GOSUB 360              'DRAW PICTURE
280     'SHOULD LINE BE KEPT OR DISCARDED?
290 INPUT "TYPE K OR E - KEEP OR ERASE LAST LINE"; A$
300 IF A$ = "K" OR A$ = "k" OR A$ = "E" OR A$ = "e" THEN 320
310 PRINT "ENTER K OR E ONLY. TRY AGAIN": GOTO 290
320 IF A$ = "E" OR A$ = "e" THEN COUNT = COUNT - 1       'DISCARD LINE
330 GOSUB 360
340 COUNT = COUNT + 1                    'GET READY FOR NEXT LINE
350 GOTO 100
360     'DRAWS PICTURE
370 CLS
380 FOR EACH = 1 TO COUNT
390     LINE (X1(EACH),Y1(EACH)) - (X2(EACH),Y2(EACH)),LINECOLOR(EACH)
400 NEXT
410 RETURN
420 GOSUB 360
430 IF INKEY$ = "" THEN 430
440 END
```

说明：

①本程序允许一幅图画由 20 条线段组成，如果需要更多的线条，可修改第 70 号语句。

②用户在键盘上依次输入每条线段的起点和终点的坐标位置以及该线段的彩色码，输入完一条线段的数据后，程序就把产生的画面显示一次，然后询问用户：新增加的一条线段是保留还是擦掉？若用户回答“E”或“e”，则擦掉；若回答“K”或“k”，则保留。然后用户可再次输入组成画面所需要的下一线段的有关数据。

③若输入的坐标数据为（－1，－1），则作图过程结束，程序把最终画面显示出来，等待用户任意输入一键后，程序即告结束。

3．光笔及其使用

（1）光笔的结构与工作原理

光笔是很早就开发出来的一种用于交互式作图的设备，它的结构原理如图 8—15 所示。

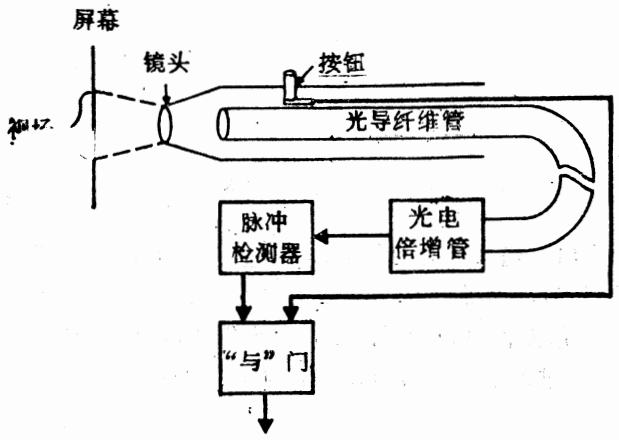


图 8—15　光笔的结构原理

光笔的外形很象一支笔，它的头部是一个镜头，调节光圈的大小即可控制其视场，视场内的光通量（即亮度的强弱）经过光导纤维管送到光电倍增管进行放大，然后送到脉冲检测电路去进行检测。由于图形显示器采用的是光栅扫描技术，荧光屏上的每一亮点实际上都是动态变化的，这样脉冲检测电路便产生一连串的脉冲。当用户按下光笔上的按钮时，便会使

控制门产生一个输出信号。当然，如果光笔的视场内什么亮点都没有，则按下按钮时也就不会有什么信号输出。

IBM PC 的彩色图形显示控制器与光笔的接口信号有两个，一个是“－光笔开关”，它反映了光笔上的按钮的状态是接通还是断开。另一个信号是“光笔脉冲”，即按钮按下时光笔中脉冲检测电路所产生的输出信号。光笔脉冲被送到一个D触发器，如果该触发器原来处于复位状态，则它使触发器置位，触发器的输出信号(“光笔触发”)即由低变高，从而使得MC6845 CRTC控制器把该瞬间的刷新地址锁存到光笔寄存器R16和R17中去。“－光笔开关”和“光笔触发”这两个信号还被保存在图形显示器的状态寄存器中，CPU可以通过输入指令从3DA端口读入其内容，以便了解光笔的操作状态。

光笔控制电路的原理可参见图8—16。其中，D触发器的复位是用输出指令(端口地址为3DB，数据可任意)来进行的，置位也用端口地址为3DC的输出指令来进行，目的是为了软件模拟光笔操作，以便进行故障诊断。

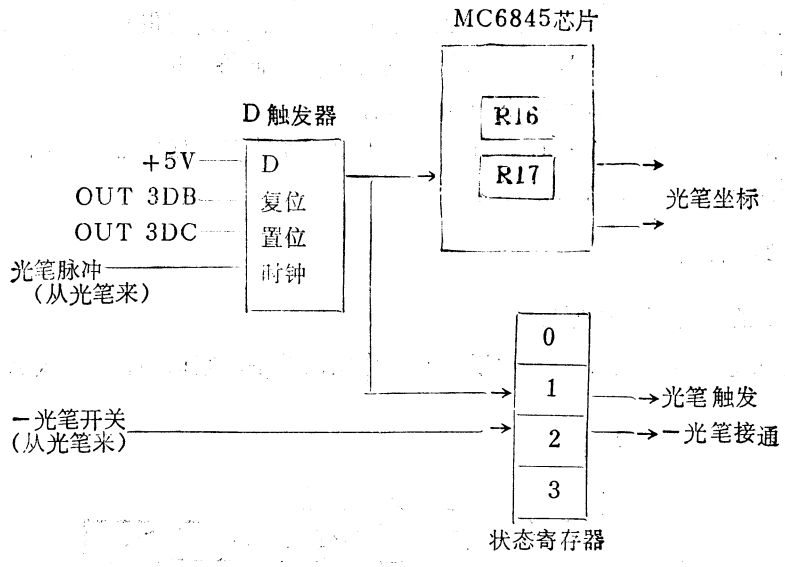


图8—16　光笔控制线路的原理

从图8—16可以看出，当光笔对准了屏幕上的亮点，并按下按钮时，便会产生光笔脉冲送入控制器，使触发器置位，这时称光笔已被“激活”，或已被“触发”。光笔被激活一次后，如果软件没有把D触发器复位的话，用户对光笔的再次操作是无效的。

BIOS中光笔的处理过程，可以用图8—17的流程图来表示。

(2)光笔原语

IBM PC的BASIC语言中，提供了几条为光笔所专用的语句和一个光笔函数，从而为使用光笔进行交互式作图带来了很大的方便。下面对这些语句和函数作简单介绍。

① PEN ON(允许光笔操作)

为了允许用户使用光笔进行操作，程序中必须使用“PEN ON”语句。当BASIC解释程序执行了“PEN ON”语句之后，它就会定期地自动查询光笔的状态。如果光笔被激活，

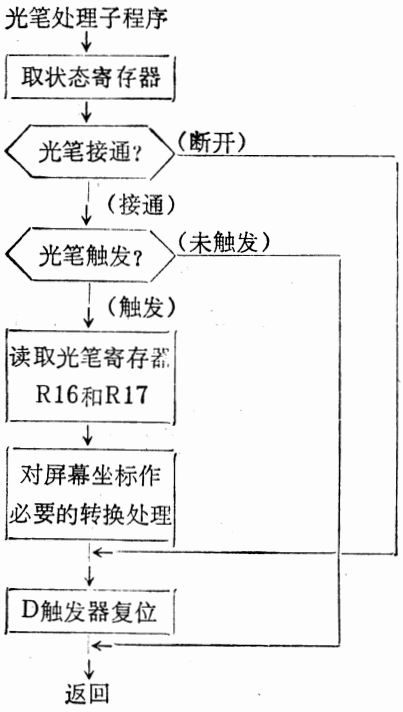


图 8—17　光笔处理子程序流程

则立即保存该瞬间的屏幕坐标。显然，这样做会增加 BASIC 解释程序的开销，降低了程序的执行速度。因此，只有在快要使用光笔进行操作之前，BASIC 程序中才使用 PEN ON 语句来“打开”光笔的操作，一旦光笔操作结束，则立即使用下面的 PEN OFF 语句来“关闭”光笔的查询操作。

② PEN OFF（关闭光笔操作）

关闭光笔操作，此后 BASIC 解释程序即正常执行，不再查询光笔的状态。

③ 函数 PEN（N）（光笔函数）

光笔函数只能在使用了 PEN ON 语句把对光笔的查询操作打开后才能使用，否则会引起语法错误。函数 PEN（N）用来取回光笔的状态，或者取回最近一次光笔激活时的屏幕坐标。自变量 N 不同，它的返回值也不一样。N 是整数，其取值范围为 0～9，它们的含义分别为：

PEN（0）——检查光笔状态，若上次查询以来，光笔尚未再次被激活，则函数值为0，若又已被激活，则函数值为－1。

PEN（1）——光笔上一次激活时所在位置的横坐标，中分辨率时为 0～319，高分辨率时为 0～639。

PEN（2）——光笔上一次激活时所在位置的纵坐标，范围为 0～199。

PEN（3）——光笔开关的现行状态，若此时光笔开关正处于接通状态，则函数值为－1，否则为 0。

PEN（4）——光笔开关持续处于接通状态时，本函数用来取回最近一次光笔激活时所

在位置的横坐标。

PEN(5)——同PEN(4)，但取回的是纵坐标。

PEN(6)——同PEN(2)，但光笔所在的纵向位置应折合成字符行的行号，其范围为1～24。

PEN(7)——同PEN(1)，但光笔所在的横向位置应折合成字符列的列号，其范围为1～40（中分辨率）或1～80（高分辨率）。

PEN(8)——同PEN(5)，但光笔所在的纵坐标值应折合成行序号，范围为1～24。

PEN(9)——同PEN(4)，但光笔所在的横坐标值应折合成列序号，范围为1～40或1～80。

④ ON PEN GOSUB〈行号〉（光笔陷井）

光笔陷井只在打开了光笔操作后才有效。若〈行号〉为0，则禁止光笔陷井，否则，此后每一个BASIC语句开始执行之前，都将查看光笔是否被激活，假如已被激活，则转向行号所指出的子程序去进行处理。处理完毕后，仍将返回原来的程序继续执行下去。

本语句仅限用于高级BASIC语言。

⑤ PEN STOP （停止光笔陷井）

使用本语句后，光笔陷井即告失效。但是，如果光笔被激活的话，则会被记住，当再次执行PEN ON语句后，就会被立即捕获而得到处理。

PEN STOP与PEN OFF不同之处在于：PEN OFF也使光笔陷井失效，但此后光笔即使被激活，也不会记住。

PEN STOP语句只限用于高级BASIC语言。

（3）使用光笔进行交互式作图的实例

下面给出四个实例，它们分别描述了如何使用光笔作为选择设备和定位设备，来解决菜单”选择、画面着色、构图及素描等问题的。

例8—11 利用光笔进行“菜单”选择

程序：

```
10 'PROGRAM      LIGHT PEN SELECTION FROM A MENU
50  SCREEN 1: COLOR 1,1: CLS
60      'DISPLAY MENU
70  LOCATE 1,1: PRINT "SELECT AN ACTIVITY --"
80  LINE (80,32) - (232,64),2,B
90  LINE (80,88) - (232,120),2,B
100 LINE (80,144) - (232,174),2,B
110 LINE (10,24) - (309,183),2,B
120 PAINT (20,40),1,2            'PAINT AROUND THE MENU BOXES IN CYAN
130 LOCATE 6,12: PRINT "ADD STUDENT NAMES"
140 LOCATE 7,14: PRINT "TO GRADE FILE"
150 LOCATE 13,15: PRINT "ADD SET OF"
160 LOCATE 14,15: PRINT "TEST SCORES"
170 LOCATE 20,13: PRINT "CALCULATE FINAL"
180 LOCATE 21,14: PRINT "COURSE GRADES"
190     'READ PEN CHOICE
300 PEN ON
210 IF PEN(0) <> -1 THEN 210     'CHECK TO SEE IF PEN IS ACTIVATED
220 X = PEN(1): Y = PEN(2)       'WHEN IT IS, SAVE COORDINATES
230     'WAS PEN IN CORRECT AREA OF SCREEN?
240 IF X < 80 OR X > 232 OR Y < 32 OR Y > 174 THEN 220
250 PEN OFF
260 IF Y > 32 AND Y < 64 THEN GOSUB 320: GOTO 380
```

```
270 IF Y > 88 AND Y < 120 THEN GOSUB 340: GOTO 380
280 IF Y > 144 AND Y < 174 THEN GOSUB 360: GOTO 380
290     'CHOICE WAS IN-BETWEEN BOXES. ASK FOR RE-ENTRY
300 LOCATE 1,1: PRINT "TRY AGAIN. POINT PEN INSIDE A BOX"
310 GOTO 200
320     'CODE TO ADD STUDENT NAMES TO GRADE FILE
330 RETURN
340     'CODE TO ADD A SET OF TEST SCORES
350 RETURN
360     'CODE TO CALCULATE FINAL COURSE GRADES
370 RETURN
380 END
```

说明：

① 语句50—180用来显示一张命令“菜单”（图8—18）。上面有三个命令供用户选

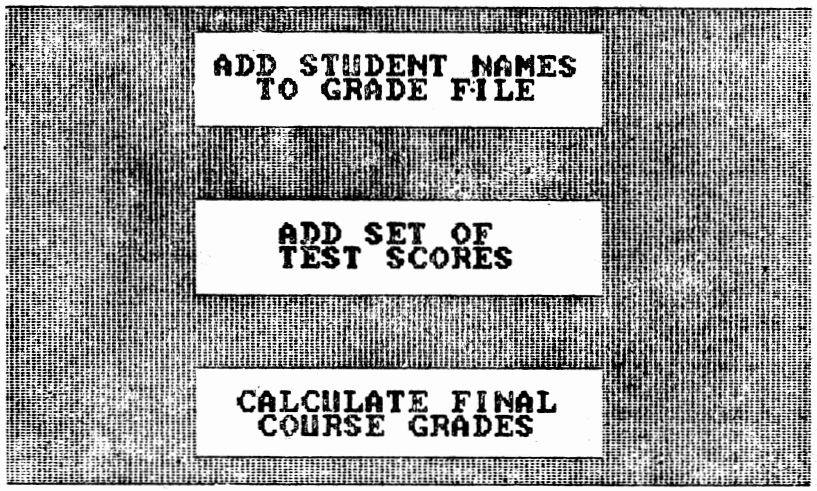


图8—18　例8—11所产生的命令“菜单”

择：在成绩册上添加学生姓名；加入一组测验分数；计算期末成绩。屏幕的底色为蓝色，配色器选用1号，菜单的底色为青色，边框为洋红，菜单本身为蓝底白字。

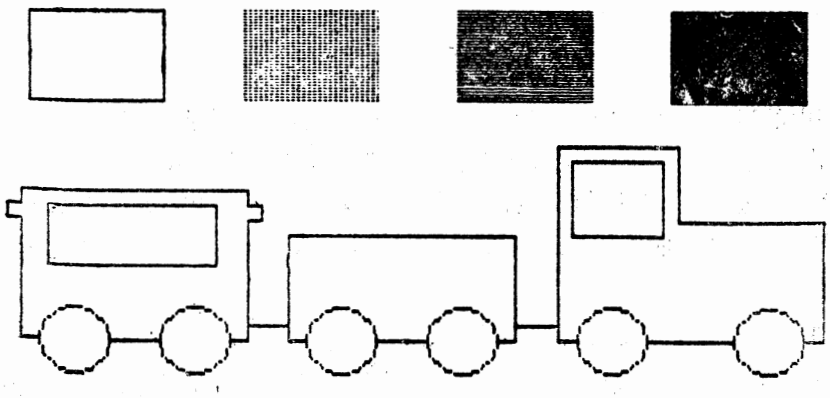


图8—19　用光笔进行着色

② 光笔允许操作后，由于底色不是黑色，因此光笔指向命令框内任何位置上都会被激活，然后根据其坐标而判定选择的是哪一个命令，再转去执行。如果光笔指向的位置不属于任何一个命令，则程序输出提示信息，请用户重新选择。

例 8—12　利用光笔进行着色

下面的程序分两部分，第一部分是在屏幕上显示出一列尚未着色的火车及四种彩色方块（图 8—19），第二部分的功能是使用光笔选择某种颜色，然后再用光笔指向火车上这种颜色应涂的部位，程序即可自动着色。

程序：

```
10 'PROGRAM      CHOOSING COLORS WITH LIGHT PEN
730 SCREEN 1: COLOR BACKGROUND,PALETTE          GO TO CHOSEN COLORS
740 ENGINE$ = "BM315,135;D40L7;BL26;L33;BL26;L7U65R45D25R55"
750 BOXCAR$ = "BM200,140;D35L7;BL26;L19;BL26;L7U35R85"
760 CABOOSE$ = "BM100,125;D5R5D5L5D40L7;BL26;L19;BL26;L7U40L5U5R5U5R85"
770 TRAIN$ = ENGINE$ + BOXCAR$ + CABOOSE$
780 DRAW TRAIN$
790 CIRCLE (35,175),13: CIRCLE (80,175),13
800 CIRCLE (135,175),13: CIRCLE (180,175),13
810 CIRCLE (236,175),13: CIRCLE (295,175),13
820 LINE (221,115) - (255,140),,B
830 LINE (25,130) - (88,150),,B
840 LINE (100,170) - (114,170)
850 LINE (200,170) - (215,170)
860     'DRAW COLOR CHOICES
870 LINE (15,20) - (65,50),,B
880 FOR EACHCOLOR = 1 TO 3
890     LINE (15+EACHCOLOR*80,20) - (15+EACHCOLOR*80+50,50),EACHCOLOR,BF
900 NEXT
910     'COLOR TRAIN
920 PEN ON
930 IF PEN(0) <> -1 THEN 930
940 X = PEN(1): Y = PEN(2)
950 IF X < 15 OR X > 305 OR Y < 20 OR Y > 50 THEN 930
960 IF X > 15 AND X < 65 THEN DRAWCOLOR = 0: GOTO 1020
970 IF X > 95 AND X < 145 THEN DRAWCOLOR = 1: GOTO 1020
980 IF X > 175 AND X < 225 THEN DRAWCOLOR = 2: GOTO 1020
990 IF X > 255 AND X < 305 THEN DRAWCOLOR = 3: GOTO 1020
1000 LOCATE 1,1: PRINT "TRY AGAIN": FOR K=1 TO 30: NEXT
1010 LOCATE 1,1: PRINT "          ": GOTO 920
1020 SOUND 600,10
1030 PEN ON
1040 IF PEN(0) <> -1 THEN 1040
1050 X = PEN(1): Y = PEN(2)
1060 SOUND 800,10
1070 PAINT (X,Y),DRAWCOLOR,3
1080 GOTO 910
1090 END
```

说明：

① 程序中的语句 1020 和 1060 均为音响语句。它们的作用是：每当光笔激活一次，且指向有效屏幕位置时，机器将发出音响来告诉用户，否则，就表示光笔操作无效。

② 着色是通过 PAINT 语句实现的，其参数 X，Y 的值由 1050 语句中的光笔函数 PEN(1)和 PEN(2)取得，程序中假设它们均处于火车图案的内部，因此未对其进行范围检查。

例 8—13　利用光笔交互式地构图

本例利用光笔进行交互式构图。假设欲构造的画面是由若干基本图形符号所组成，例如，电阻、电容、晶体管等可以构成电路图；桌、椅、书架、文件柜等可以构成办公室平面

布置图等等。用户只要先使用光笔选择某一种图形符号，然后再用光笔指出该符号应该放置的位置，反复进行，即可得到所需要的画面。

程序：

```
10 'PROGRAM      INTERACTIVE PICTURE CONSTRUCTION WITH LIGHT PEN
50  SCREEN 2: CLS
60  LINE (0,0) - (639,199),1,BF
70  LINE (80,0) - (80,199),0
80  LINE (0,50) - (80,50),0
90  LINE (0,100) - (80,100),0
100 LINE (0,150) - (80,150),0
110 BOX$ = "R50D25L50U25"
120 TRIANGLE$ = "R50H25G25"
130 DRAW "C0BM15,10;XBOX$;BM15,140;XTRIANGLE$;"
140 CIRCLE (40,75),25,0
150 LINE (16,168) - (64,190),0,BF
160 LOCATE 23,4: PRINT "STOP";
170 LINE (400,189) - (639,199),0,BF
180 LOCATE 25,53
190 PRINT "USE PEN TO CHOOSE A SHAPE ";
200 PEN ON
210 IF PEN(0) <> -1 THEN 210
220 X = PEN(1): Y = PEN(2)
230 IF X > 80 THEN 210
240 PEN OFF
250 IF Y < 50 THEN SHAPE = 1: SOUND 400,10: GOTO 290
260 IF Y > 50 AND Y < 100 THEN SHAPE = 2: SOUND 500,10: GOTO 290
270 IF Y > 100 AND Y <150 THEN SHAPE = 3: SOUND 600,10: GOTO 290
280 IF Y > 150 THEN SOUND 300,10: GOTO 540
290     'FIND WHERE TO PLACE THE SHAPE
300 LOCATE 25,53
310 PRINT "USE PEN TO PLACE THE SHAPE";
320 PEN ON
330 IF PEN(0) <> -1 THEN 330
340 X = PEN(1): Y = PEN(2)
350 IF X < 80 THEN 330
360 PEN OFF
370 ON SHAPE GOSUB 390,450,480
380 GOTO 180
390     'DRAW BOX
400 XSTART = X - 25     'LET PEN COORDINATES BE CENTER OF SHAPE
410 YSTART = Y - 13
420 PSET (XSTART,YSTART)
430 DRAW "C0;XBOX$;"
440 RETURN
450     'DRAW CIRCLE
460 CIRCLE (X,Y),25,0
470 RETURN
480     'DRAW TRIANGLE
490 XSTART = X - 25
500 YSTART = Y + 13
510 PSET (XSTART,YSTART)
520 DRAW "C0;XTRIANGLE$;"
530 RETURN
540 LINE (400,189) - (639,199),1,BF
550 IF INKEY$ = "" THEN 550
560 END
```

说明：

① 本程序工作在高分辨率显示模式下，因此只有黑白两种颜色。程序中选择屏幕的底色为白，显示色为黑，否则无法使光笔激活。

② 程序首先在屏幕左部边缘处画出三个基本图形符号（方块，圆，三角）及一个停止标志（STOP），然后在屏幕右下角给出提示："使用光笔选择一个图形符号"（图 8—20）。于是，用户就可以开始用光笔构图。如果用户选用了停止标志，即表示构图完毕，程序把右下角的提示擦掉并结束工作。

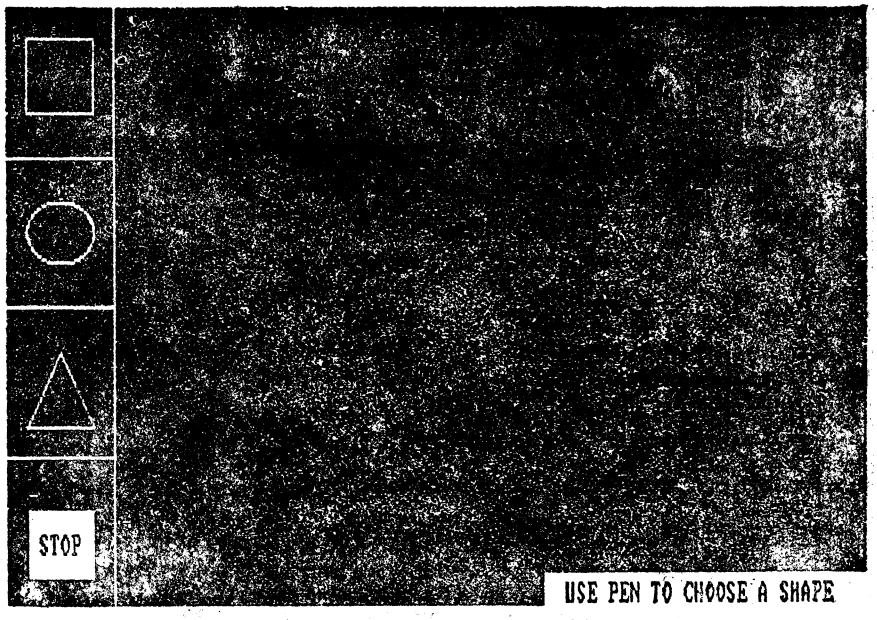


图 8—20　使用光笔进行构图

③ 使用光笔选择基本图形符号时，如果光笔指在某个符号所在的方框内，则机器发声以示选择操作有效。同样，当放置该符号时，也只有光笔指向屏幕上的有效画面区时，才能画出对应的符号来（光笔指向位置是符号的中心位置）。

例 8—14　　利用光笔进行素描

前面的几个例子都是应用光笔来选择某些项目而进行交互式作图的。本例则把光笔模拟成普通的画笔，使用户可以在屏幕上任意地作画写字。

程序：

```
10 'PROGRAM      INTERACTIVE SKETCHING WITH THE LIGHT PEN
20  SCREEN 1: COLOR 1,1: CLS
30       'MAKE STOP BOX
40  LINE (0,176) - (48,199),1,BF          'MAKE BLUE BOX
50  LINE (4,180) - (44,195),0,BF          'MAKE BLACK BOX INSIDE BLUE
60  LOCATE 24,2: PRINT "STOP";
70       'READ FIRST POINT FROM PEN
80  PEN ON
90  CLEER = PEN(0)                        'CLEAR OUT PRIOR PEN ACTIVITY
100 IF PEN(0) <> -1 THEN 100               HAS PEN BEEN ACTIVATED?
110 X = PEN(1): Y = PEN(2)
120 PSET (X,Y),2                          'SET THE FIRST POINT
130      'READ ADDITIONAL POINTS
140 IF PEN(3) <> -1 THEN 140
150 IF PEN(4) < 48 AND PEN(5) > 176 THEN 190     'WE'RE IN THE STOP BOX
160 X = PEN(4): Y = PEN(5)
170 LINE - (X,Y),2
180 GOTO 140
190 LINE (0,176) - (48,199),0,BF          'COLOR IN STOP BOX WITH BLUE
200 PEN OFF
210 IF INKEY$ = "" THEN  210
220 END
```

说明：

① 本例工作在中分辨显示模式，屏幕底色为蓝色，配色器为1号。

② 屏幕左下角显示一个黑色方框，内有“STOP”标记，表示光笔若移动到此处，则绘画过程即告结束。

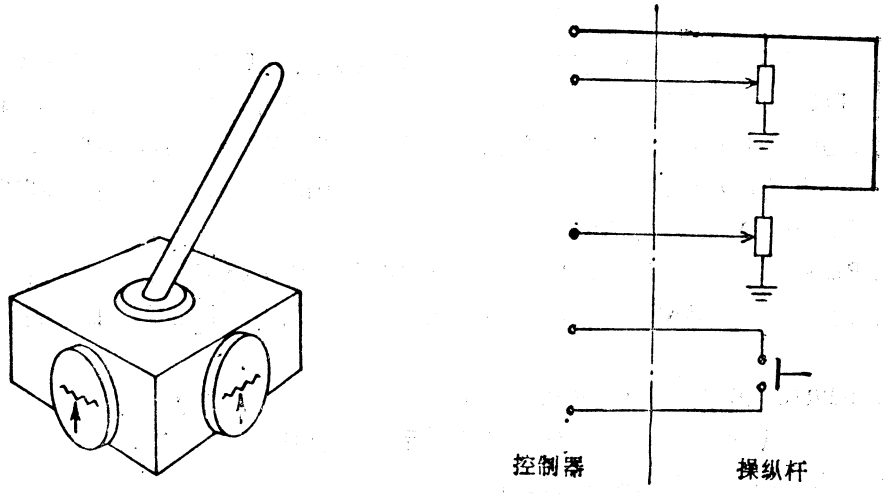
③ 90号语句用来清除光笔坐标的存贮区，否则，本程序的前一次运行所留下来的坐标值，会在下一次运行时起作用。

④ 用户操作时，光笔上的按钮始终被按下，由于屏幕底色为蓝色，因此光笔也连续地处于激活状态。程序中使用函数PEN（3）来判断光笔的状态，用PEN（4）和PEN（5）来取得光笔现行位置的坐标，照后用LINE语句跟踪光笔的移动而画出线来。

4．操纵杆及其使用

（1）操纵杆的结构与原理

IBM PC通过游戏控制器选件板，可以连接两个操纵杆（Joystick）。图8—21（a）是操纵杆的结构示意图。它由两个0～100KΩ的电位器及一个（或两个）按钮所组成，操纵杆上、下、左、右四面八方活动时，两个电位器的输出阻值也就跟着改变，它们与控制器的接口见图8—21（b）。



(a)　　(b)

图8—21　操纵杆原理及其接口

操纵杆两个电位器的输出送到游戏控制器之后，用来控制单稳电路的输出信号宽度，软件通过对该信号宽度的测量，便可算出其阻值的大小，因此也就能判断出操纵杆的位置了。

（2）操纵杆原语

IBM PC的BASIC语言是通过下列函数和语句来使用操纵杆的：

① STICK（N）（操纵杆函数）

STICK函数用来取得操纵杆的现行位置。参数N为整型，取值为0～3，它们各自的含义为：

N＝0　A 操纵杆的水平位置

N＝1　A 操纵杆的垂直位置

N＝2　B 操纵杆的水平位置

N＝3　B 操纵杆的垂直位置

由于不同型号的操纵杆其阻值范围不同，因此 STICK 函数的取值范围也不一样。另外，不同应用中操纵杆所控制的屏幕区域也有大有小。因此，程序中常常需要进行一定的换算，才能求出操纵杆所在位置与屏幕坐标之间的对应关系。

设操纵杆横向最小值（操纵杆处于最左位置）为 XJMIN，横向最大值（操纵杆处于最右位置）为XJMAX，需要控制的屏幕区域的横坐标在范围 XMIN～XMAX 中，则从 STICK 函数所取得的操纵杆横向位置 XJ 所对应的屏幕坐标 X，可按下式算出：

$$X=XMIN+(XJ-XJMIN)*(XMAX-XMIN)/(XJMAX-XJMIN)$$

同样，屏幕的纵坐标 Y 与操纵杆纵向位置的对应关系是：

$$Y=YMIN+(YJ-YJMIN)*(YMAX-YMIN)/(YJMAX-YJMIN)$$

例如，假设操纵杆 A 的横向位置取值范围是 3～160，在中分辨率时需要对满屏幕进行控制（0～319），如果操纵杆位置 XJ＝STICK（0）＝100，则它所对应的屏幕坐标 X＝197。

注意，STICK（0）和其它三个函数 STICK（1）、STICK（2）、STICK（3）是有区别的。STICK（0）用来检索两个操纵杆的所有四个位置值，然后把其中操纵杆 A 的横向位置作为函数值返回。其它三个函数却不再检索操纵杆的状态，仅把 STICK（0）已检索到的对应位置作为函数值返回。所以，为了使用后三个函数，必须首先使用 STICK（0）。

② STRIG ON（操纵杆按钮打开）

与 PEN ON 语句类似。STRIG ON 语句用来通知 BASIC 解释程序：下面即将使用按钮进行操纵。于是，BASIC 解释程序就会在每次执行新的语句之前，自动查询按钮的状态。

③ STRIG OFF（操纵杆按钮关闭）

停止对操纵杆按钮的查询操作，表示此后不再使用按钮了。

④ STRIG（N）（操纵杆按钮函数）

这是一个函数，自变量 N 的范围只能是 0、1、2 或者3，它们的函数值分别定义如下：

N＝0　自从上一次调用了函数 STRIG（0）以后，操纵杆 A 上的按钮又再次被按下，则函数值为－1，否则为 0。

N＝1　操纵杆 A 上的按钮正处于按下状态时，函数值为 －1，否则为 0。

N＝2　同 N＝0，但指的是操纵杆 B 上的按钮。

N＝3　同 N＝1，但指的是操纵杆 B 上的按钮。

如果每个操纵杆上有两个按钮，则在高级 BASIC 语言中，N 的取值允许扩充到 4、5、6和 7，它们分别用来处理操纵杆 A 和 B 上的第 2 只按钮的状态。定义与上述N＝0～3的情况完全类似。

使用函数 STRIG（N）之前，必须先用 STRIG ON把对操纵杆按钮的查询操作打开。

不再使用 STRIG (N) 时，应该用 STRIG OFF 将查询操作关闭，以提高程序的执行速度。

⑤ STRIG (N) ON （打开按钮陷井）

N 可以是 0、2、4 或 6，它们分别表示允许对操纵杆 A 和 B 上的四个按钮 (A1, A2, B1, B2) 进行陷井操作。

⑥ STRIG (N) OFF (关闭按钮陷井)

同上，但表示陷井操作关闭。

⑦ STRIG (N) STOP (暂停陷井操作)

陷井操作暂停，但按钮的动作会被记下，一旦陷井又再次打开时，该事件便会立即受到处理。

⑧ ON STRIG (N) GOSUB 〈行号〉

若行号为 0，则不进行陷井操作。否则，如果对应的 STRIG (N) ON 语句被执行，则 BASIC 程序每次执行新的语句之前，都去查询指定的按钮是否被按下。若按下则转向〈行号〉所指出的子程序去处理，子程序处理完毕后再返回原来的断点处继续执行。

下面是程序中使用按钮陷井的一个示例：

```
 100  ON STRIG ( 0 ) GOSUB 2000
 110  STRIG ( 0 ) ON
  ⋮
2000  REM Subroutine for button A1
  ⋮
2100  RETURN
```

这里介绍的有关按钮陷井的四个语句都只能在高级 BASIC中使用。

(3) 使用操纵杆进行交互式作图的实例

前面已经介绍过利用键盘和光笔在屏幕上进行素描的例子，下面介绍如何使用操纵杆在屏幕上画图。操纵杆使用于素描相当粗糙，但它在模拟和游戏等应用中还是很有效的。

例 8—15　利用操纵杆进行素描

程序：

```
10 'PROGRAM       INTERACTIVE SKETCHING WITH JOYSTICKS
20       'SKETCHES FROM CENTER JOYSTICK POSITION TO OTHER POINTS.
30       'SKETCHING IS TERMINATED BY HITTING ANY KEY ON THE KEYBOARD.
40  SCREEN 1: COLOR 1,0: CLS
50  FIRSTPOINT = 1                          'FLAG TO INDICATE FIRST POINT
60  READ XMIN, XMAX, YMIN, YMAX             'SET SCREEN AREA IN WHICH TO DRAW
70  READ XJMIN, XJMAX, YJMIN, YJMAX         'READ MIN AND MAX FOR THIS JOYSTICK
80  XCONST1 = (XMAX - XMIN) / (XJMAX - XJMIN)
90  XCONST2 = XMIN - (XJMIN * XCONST1)
100 YCONST1 = (YMAX - YMIN) / (YJMAX - YJMIN)
110 YCONST2 = YMIN - (YJMIN * YCONST1)
120      'READ POINTS OF JOYSTICK
130 XJ = STICK(0): YJ = STICK(1)
140 X = XCONST2 + (XJ * XCONST1)
150 Y = YCONST2 + (YJ * YCONST1)
160 IF FIRSTPOINT = 0 THEN LINE - (X,Y) ELSE PSET (X,Y): FIRSTPOINT = 0
170 IF INKEY$ = "" THEN 130                 'IF NO KEY PRESSED, CONTINUE
```

```
180 IF INKEY$ = "" THEN 180              'HOLD PICTURE WITHOUT "OK".
190 DATA 0,319,0,199
200 DATA 3,166,3,174:    'JOYSTICK RANGE
210 END
```

说明：

① 本例选择的是中分辨率显示模式，底色为蓝色，线条为白色。允许全屏幕范围内用操纵杆画图。

② 操纵杆的起始位置是中心位置，它与屏幕中心点相对应，此后移动操纵杆时，即在屏幕上不停地画出线条来。如果任意敲入一键，画图即告结束。

例 8—16　利用操纵杆及按钮进行素描

程序：

```
10 'PROGRAM      INTERACTIVE SKETCHING USING JOYSTICKS AND BUTTONS
20  SCREEN 2: CLS : FIRSTPOINT = 1
30  READ XMIN, XMAX, YMIN, YMAX          SET AREA OF SCREEN IN WHICH TO DRAW
40  READ XJMIN, XJMAX, YJMIN, YJMAX
50  XCONST1 = (XMAX - XMIN) / (XJMAX - XJMIN)
60  YCONST1 = (YMAX - YMIN) / (YJMAX - YJMIN)
70  XCONST2 = XMIN - (XJMIN * XCONST1)
80  YCONST2 = YMIN - (YJMIN * YCONST1)
90  STRIG ON
100 IF STRIG(0) <> -1 THEN 100           'WAIT FOR BUTTON
110 GOSUB 150                            'CONVERT TO SCREEN POINTS
120 IF FIRSTPOINT = 0 THEN LINE - (X,Y) ELSE PSET (X,Y): FIRSTPOINT = 0
130 IF INKEY$ = "" THEN 100 ELSE 210     'HITTING KEYBOARD KEY ENDS PROGRAM
140     'CONVERT JOYSTICK COOORDINATES TO SCREEN COORDINATES
150 XJ = STICK(0): YJ = STICK(1)
160 X = XCONST2 + (XJ * XCONST1)
170 Y = YCONST2 + (YJ * YCONST1)
180 RETURN
190 DATA 0,639,0,199
200 DATA 3,166,3,175
210 STRIG OFF: END
```

说明：

① 本例选择的是高分辨率显示模式，利用操纵杆可以全屏幕进行画图。

② 用户移动操纵杆时，屏幕上并不立即画出线来，只有在按下按钮时，操纵杆的位置才能被程序取得、并画出线条来。

③ 与上例一样，任意在键盘上输入一键，即引起程序结束。

除了上面介绍过的键盘、光笔、操纵杆等可以用来作为交互式作图的工具之外，数字化桌、鼠形光标定位器等设备也均可以与 IBM PC 联用（通过通讯控制器的 RS—232C 接口），由于它们的基本原理相同，又限于篇幅，故本书就不再介绍了。

第三节　图形变换与窗口操作

在计算机辅助设计（CAD）和数据处理的许多场合，用户经常需要对计算机显示输出的图形进行各种操作。例如，把屏幕上的某些图形移动其位置，修改它们的形状或尺寸大小，改变它们的方向角度；有时为了突出图形中的某些部分，还可以对它们进行加亮、裁剪、放大等处理。所有这些都是通过一些基本的图形变换和窗口操作来实现的。

1. 平移变换

把图形从屏幕上的一处移到另外一处，称为图形的“平移”。假设图上某点的老坐标位置为（X，Y），平移的水平距离为H，垂直距离为V，则经过平移之后，该点的新坐标为：

$$\begin{cases} XT = X + H \\ YT = Y + V \end{cases}$$

不难看出，其中H为正表示向右移动，H为负表示向左移动；V为正表示向下移动，V为负表示向上移动。

为了对一幅图形或图形中的某个部分进行平移操作，程序中必须完成：

① 计算出被平移的图形中每一点的新坐标，这个过程叫做“平移变换”；

② 擦去原来位置上的图形；

③ 按新坐标位置画出该图形。

由于图形常常是由一些点、线条、矩形及圆或圆弧等组成的，因此平移变换并不需要对图中每一点逐点进行，只要对线条的端点、矩形的顶点、圆或圆弧的中心以及一些孤立点进行新坐标的计算就行了。

下面是利用平移变换在屏幕上移动图形位置的一个例子。

例 8—17　图形的平移

程序：

```
10 'PROGRAM       TRANSLATION OF PICTURE PARTS.
60  DIM X(5,50), Y(5,50), POINTCOUNT(5)
70  SCREEN 1: COLOR 4,0: CLS
80      '*************** READ PICTURE PARTS AND DRAW ***************
90  PICTUREPART = 0      'PICTUREPART IS PART NUMBER
100 READ XD, YD
110 WHILE XD <> -100                          '-100 IS END OF DATA SIGNAL
120     PICTUREPART = PICTUREPART + 1
130     EACHPOINT = 0
140     WHILE XD => 0                         '-1 IS END OF PICTURE PART SIGNAL
150           EACHPOINT = EACHPOINT + 1
160           X(PICTUREPART,EACHPOINT) = XD
170           Y(PICTUREPART,EACHPOINT) = YD
180           READ XD,YD
190     WEND
200     'STORE # OF ELEMENTS IN PICTUREPART IN POINTCOUNT(PICTUREPART)
210     POINTCOUNT(PICTUREPART) = EACHPOINT
220     READ XD,YD
230 WEND
240     'DRAW PICTURE
250 DRAWCOLOR = 3
260 FOR K = 1 TO PICTUREPART
270     GOSUB 570                 'DRAW EACH PART
280 NEXT
290     '******************* CONTROL PROGRAM FLOW ********************
300 GOSUB 430                     'GET CHOICE OF WHICH PART TO MOVE
310 WHILE CHOICE <> 0             'USER ENTERS 0 TO END PROGRAM
320     GOSUB 480                 'HOW MUCH TO MOVE?
330     K = CHOICE
340     DRAWCOLOR = 0             'ERASE CURRENT DISPLAY OF CHOSEN PART
350     GOSUB 570
360     GOSUB 510                 'RECALCULATE POINTS
370     DRAWCOLOR = 3
380     GOSUB 570                 'DISPLAY IN NEW POSITION
390     GOSUB 430                 'GET CHOICE OF PART TO MOVE OR QUIT
```

```
400 WEND
410 GOTO 760
420     '********** PRINT INSTRUCTIONS FOR TRANSLATING **********
430 LOCATE 22,1: PRINT "1-BOY  2-DOG  3-HYDRANT"
440 INPUT "PICTURE PART TO MOVE (0 TO END)"; CHOICE
450 LOCATE 22,1: PRINT STRING$(78," ");
460 RETURN
470     '****************** HOW MU  H TO MOVE? ********************
480 LOCATE 23,1: INPUT "H AND V C  UNT TO MOVE"; H, V
490 RETURN
500     '****************** RECALCULATE POINTS ********************
510 FOR J = 1 TO POINTCOUNT(CHOICE)
520     X(CHOICE,J) = X(CHOICE,J) + H
530     Y(CHOICE,J) = Y(CHOICE,J) + V
540 NEXT
550 RETURN
560     '********************* DRAW ROUTINE ***********************
570 FOR J = 1 TO POINTCOUNT(K)-1
580     LINE (X(K,J), Y(K,J)) - (X(K,J+1), Y(K,J+1)),DRAWCOLOR
590 NEXT
600 RETURN
610 '*************************************************************
620 DATA 85,70,90,75,105,60,105,80,85,110,95,110,110,85
630 DATA 125,110,135,110,115,80,115,60,130,75,135,70,115,45
640 DATA 115,40,125,30,125,15,110,10,95,15,95,30,105,40,105,45,85,70
650 DATA -1,-1
660 DATA 50,90,62,110,58,110,50,90,42,110,38,110,50,90
670 DATA 45,100,20,100,10,90,22,110,18,110,10,90,3,110
680 DATA 0,110,10,90,0,80,10,90,50,90,40,80,50,70,55,75,60,78,60,82,50,90
690 DATA -1,-1
700 DATA 290,110,290,98,288,98,288,92,286,92,286,88
710 DATA 286,88,288,88,288,82,290,82,290,73,287,73,287,70,290,70,290,65
720 DATA 295,60,305,60,310,65,310,70,313,70,313,73,310,73,310,110,290,110
730 DATA - ,-1
740 DATA -100,-100
750 '*************************************************************
760 IF INKEY$ = "" THEN 760
770 END
```

说明：

① 本例的图形由三部分组成：男孩、小狗和消防水龙头，它们均由线条所组成。生成这些图形的数据由DATA 语句（620 语句～740语句）给出，其中 －1表示每个部分的数据结束，－100 表示全部图形数据结束。

② 为便于进行平移变换，图形数据首先读入数组X（PICTUREPART, EACHPOINT）和Y（PICTUREPART, EACHPOINT）中，PICTUREPART=0,1,2分别表示它们是三个不同部分的图形数据。POINTCOUNT（PICTUREPART）表示每一部分的端点总数。

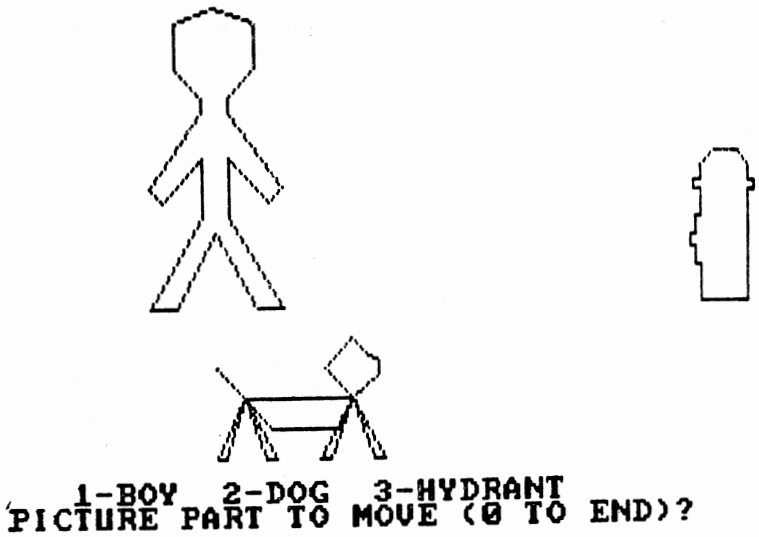
③ 语句250～280用来画出图形，如图8—22（a）所示。画面的底部是提示符。用户输入1表示需要平移男孩，输入2表示平移小狗，输入3表示平移水龙头，输入0表示结束本程序的工作。

④用户选择了平移的对象之后，程序再提示用户输入平移的水平距离H和垂直距离V（语句480—490）。

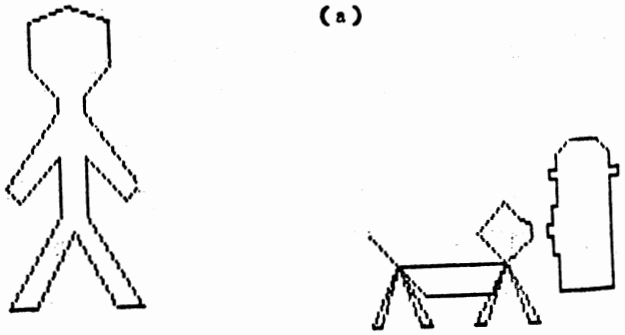
⑤ 程序擦除原来的图形后，再进行平移变换，算出被平移的图形的新的坐标值，按新的坐标数据重新画出该图形（图8—22（b））。

2．比例变换

屏幕上显示出来的图形，有时为要看得更清楚些需要把它放大，有时为要在屏幕上显示



(a)

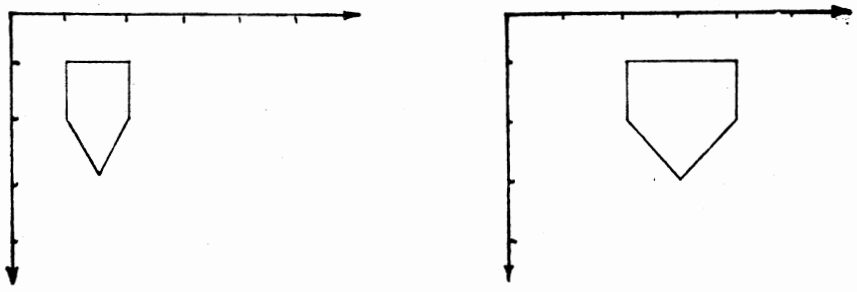


(b)

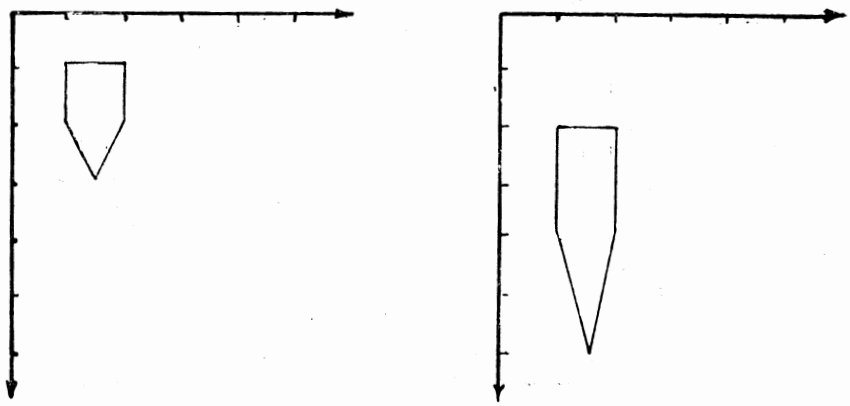
图8—22 图形的平移

更多的内容需要把它缩小，也有的时候需要调整图形的纵横比以改变其形状。以上这些都可以通过比例变换（Scaling）来实现。

所谓“比例变换”，是指把图形沿X方向扩大（或缩小）HS倍，沿Y方向扩大（或缩小）VS倍（图8—23）。HS和VS就分别称为水平比例因子和垂直比例因子。如果HS＝VS，则被变换的图形只改变大小而不改变其形状。



（a）HS=2，VS=1



(b) HS=1, VS=2

图 8—23 比例变换的定义

进行比例变换的方法，是把图形上各点的横坐标乘以 HS，纵坐标乘以 VS，即

$$\begin{cases} XS = X * HS \\ YS = Y * VS \end{cases}$$

这种比例变换是相对于坐标原点来进行的。但在实际问题中，常常需要相对于屏幕上的某一点来进行比例变换。假设该参考点的坐标为（XF，YF），则相对于该点进行比例变换的公式如下：

$$\begin{cases} XS = XF + (X - XF) * HS \\ YS = YF + (Y - YF) * VS \end{cases}$$

即

$$\begin{cases} XS = X * HS + XF * (1 - HS) \\ YS = Y * VS + YF * (1 - VS) \end{cases}$$

当然，与平移变换一样，一个图形作比例变换时，只要对线段的端点、圆的中心及半径等进行变换就行了，并不需要对图中各点都进行计算。

例 8—18 是利用光笔交互式地对屏幕上显示的一部汽车外形图进行比例变换的例子。

例 8—18 利用光笔进行比例变换

程序：

```
10 'PROGRAM       SCALING A PICTURE WITH THE LIGHT PEN
20     'DRAWS CAR AND ALLOWS USER TO SCALE PART OF ALL OF
30     'CAR. SCALED POINTS REPLACE ORIGINAL VALUES IN X AND Y
40     'X AND Y.
50  DIM X(5,100), Y(5,100), POINTCOUNT(5)
60  SCREEN 1: COLOR 1,0: CLS
70       '*************** CONTROL PROGRAM FLOW **********************
80  GOSUB 210                      'READ PICTURE PARTS
90  GOSUB 440                      'MAKE MENUS
100 GOSUB 560                      'GET CHOICE OF WHICH PART TO SCALE
110 WHILE CHOICE <> 0
120     GOSUB 700                  'HOW MUCH TO SCALE?
```

```
130     DRAWCOLOR = 0
140     IF CHOICE <> 3 THEN K = CHOICE: GOSUB 1020
            ELSE FOR K = 1 TO PICTUREPART: GOSUB 1020: NEXT
150     GOSUB 910                'RECALCULATE SCALED POINTS
160     DRAWCOLOR = 2
170     IF CHOICE <> 3 THEN K = CHOICE: GOSUB 1020
            ELSE FOR K = 1 TO PICTUREPART: GOSUB 1020: NEXT
180     GOSUB 560                'GET CHOICE OF PART TO SCALE (OR QUIT)
190 WEND
200 GOTO 1280
210     '**************** READ PICTURE PARTS AND DRAW ****************
220 PICTUREPART = 0
230 READ XD,YD
240 WHILE XD <> -100
250     PICTUREPART = PICTUREPART + 1
260     EACHPOINT = 0
270     WHILE XD => 0            '-1 IS END OF POINTS FOR THIS PAR:
280             EACHPOINT = EACHPOINT + 1
290             X(PICTUREPART,EACHPOINT) = XD
300             Y(PICTUREPART,EACHPOINT) = YD
310             READ XD,YD
320     WEND
330     'STORE # OF POINTS IS PICTUREPART IN POINTCOUNT(PICTUREPART)
340     POINTCOUNT(PICTUREPART) = EACHPOINT
350     READ XD,YD
360 WEND
370     'DRAW PICTURE
380 DRAWCOLOR = 2
390 FOR K = 1 TO PICTUREPART
400     GOSUB 1020
410 NEXT
420 RETURN
430     '***************** MAKE MENUS FOR LIGHT PEN SELECTION **********
440 LOCATE 18,1: PRINT "PART -";
450 LINE (3,149) - (53,161),1,B: LOCATE 20,2: PRINT "FRONT";
460 LINE (3,165) - (53,177),1,B: LOCATE 22,2: PRINT "REAR";
470 LINE (3,181) - (53,193),1,B: LOCATE 24,3: PRINT "ALL";
480 LOCATE 18,10: PRINT "SCALING FACTOR - "
490 LINE (75,193) - (315,199),1,BF
500 LOCATE 20,10: PRINT ".1              .5              .9";
510 LINE (75,161) - (315,167),1,BF
520 LOCATE 24,11: PRINT "1               2               3";
530 LINE (272,2) - (319,20),1,BF
540 LINE (278,6) - (313,16),0,BF: LOCATE 2,36: PRINT "STOP";
550 RETURN
560     '***************** READ PART TO SCALE FROM PEN ********************
570 PEN ON
580 WHILE PEN(0) <> -1           'WAIT FOR PEN TO BE ACTIVATED
590     LOCATE 18,1: PRINT "      ";: FOR DELAY = 1 TO 80: NEXT
600     LOCATE 18,1: PRINT "PART -";: FOR DELAY = 1 TO 80: NEXT
610 WEND
620 X = PEN(1): Y = PEN(2)
630 PEN OFF
640     'HAVE WE HIT THE STOP BOX; OR, IS THE PEN READING INVALID?
650 IF X > 270 AND Y < 25 THEN CHOICE = 0
    ELSE IF X > 53 OR Y < 149 THEN 570
660 IF Y > 179 THEN CHOICE = 3: GOTO 690
670 IF Y > 163 THEN CHOICE = 2: GOTO 690
680 IF Y > 149 THEN CHOICE = 1
690 RETURN
700     '********************* HOW MUCH TO SCALE? ************************
710 PEN ON
720 WHILE PEN(0) <> -1           'WAIT UNTIL PEN IS ACTIVATED
730     LOCATE 18,10:PRINT "                    ";:FOR D=1 TO 100:NEXT
740     LOCATE 18,10:PRINT "SCALING - HORIZONTAL";:FOR D=1 TO 100:NEXT
750 WEND
760 X = PEN(1): Y = PEN(2)
770 PEN OFF
780 IF X < 75 OR Y < 160 THEN 710        'INVALID READ - TRY AGAIN
790 IF Y < 188 THEN HS = (X - 75) * (.8 / 240) + .1
            ELSE HS = (X - 75) * (2 / 240) + 1
800     'READ VERTICAL SCALING FACTOR
```

```
810 PEN ON
820 WHILE PEN(0) <> -1          'WAIT UNTIL PEN IS ACTIVATED
830     LOCATE 18,10:PRINT "                          ";:FOR D=1 TO 100:NEXT
840     LOCATE 18,10:PRINT "SCALING - VERTICAL  ";:FOR D=1 TO 100:NEXT
850 WEND
860 X = PEN(1): Y = PEN(2)
870 PEN OFF
880 IF X < 75 OR Y < 160 THEN 810       'INVALID READ - TRY AGAIN
890 IF Y < 188 THEN VS = (X - 75) * (.8 / 240) + .1
               ELSE VS = (X - 75) * (2 / 240) + 1
900 RETURN
910     '******** RECALCULATE POINTS FOR APPROPRIATE PICTURE PART ********
920 ON CHOICE GOTO 930, 940, 950
930 XFIXED = X(1,2): YFIXED = Y(1,2): GOTO 960
940 XPIXED = X(2,2): YFIXED = Y(2,2): GOTO 960
950 XFIXED = X(1,7): YFIXED = Y(1,7): GOTO 980
960 SCALEPART = CHOICE: GOSUB 1080      'DO SCALING
970 GOTO 1010
980 FOR SCALEPART = 1 TO PICTUREPART            'SCALE EACH PART
990     GOSUB 1080
1000 NEXT
1010 RETURN
1020 '############################## DRAW ROUTINE ##############################
1030 FOR J = 1 TO POINTCOUNT(K) - 1
1040         IF X(K,J) < 0 OR X(K,J) > 319 OR Y(K,J) < 0 OR Y(K,J) > 199
                THEN 1300
1050         LINE -(X(K,J),Y(K,J)) - (X(K,J+1),Y(K,J+1)),DRAWCOLOR
1060 NEXT
1070 RETURN
1080 '############################ SCALING ROUTINE ############################
1090 FOR J = 1 TO POINTCOUNT(SCALEPART)
1100     X(SCALEPART,J) = X(SCALEPART,J) * HS + XFIXED * (1 - HS)
1110     Y(SCALEPART,J) = Y(SCALEPART,J) * VS + YFIXED * (1 - VS)
1120 NEXT
1130 RETURN
1140 '########################################################################
1150     'FRONT OF CAR
1160 DATA 220,90,160,90,160,60,205,60,187,35,160,35
1170 DATA 160,30,190,30,210,60,250,65,260,90,250,90
1180 DATA 250,100,240,110,230,110,220,100,220,90
1190 DATA 230,80,240,80,250,90
1200 DATA -1,-1
1210     'BACK OF CAR
1220 DATA 120,90,160,90,160,60,120,60,135,35,160,35
1230 DATA 160,30,130,30,115,60,80,60,75,90,90,90
1240 DATA 90,100,100,110,110,110,120,100,120,90,110,80,100,80,90,90,90,100
1250 DATA -1,-1
1260 DATA -100,-100
1270 '########################################################################
1280 IF INKEY$ = "" THEN 1280
1290 GOTO 1310
1300 PRINT "COORDINATE OUT OF RANGE"
1310 END
```

说明：

① 本例首先在屏幕上显示出一部汽车，车身的前半部、后半部或整个汽车都可以进行放大或缩小等操作（图8—24）。操作由光笔进行。屏幕的下半部是“菜单”，用户可以用光笔来选择需要进行变换的是汽车的哪一部分。旁边的两条直线作为比例尺来使用，光笔可以在0.1～3的范围内任意选择水平和垂直比例因子。所以，也可以形象地把 FRONT、REAR 和 ALL 叫做“光按钮”，把两条直条叫做“光尺”。

② 语句50～200为主程序部分，它完成下列功能：

* 把图形数据读入数组，画出汽车的原始图形。
* 生成光按钮和光尺，并在屏幕右上角画出一个停止标志 STOP，供结束操作使用。
* 用光笔选择需要变换的是哪一部分，前车身、后车身或全车。

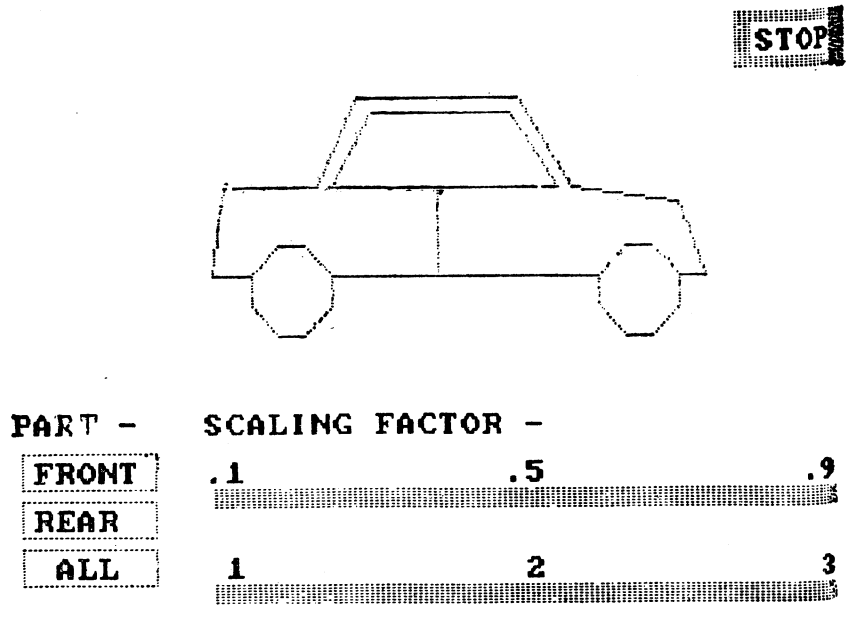


图 8—24　利用光笔进行比例变换

* 两次用光笔在光尺上指出水平比例因子 HS 和垂直比例因子 VS 的大小。
* 擦去需要进行比例变换的老的图形部分。
* 对图形进行比例变换，算出一组新的坐标值。
* 画出变换后的新的图形部分。
* 重复进行上述操作，直到选择了 STOP 为止。

③ 为了引起用户的注意，提示信息“PART—”和“SCALING—HORIZONTAL”或“SCALING—VERTICAL”在进行到相应的操作步骤时会不断地闪烁，直到用户成功地用光笔进行了选择为止。

④ 对前车身或后车身的比例变换，其参考点为（160，90），即车身中间下部位置处。对全车进行的比例变换，其参考点为（160，30），即车身中间上部位置处。两者是不一样的。

⑤ 每一次比例变换，都是相对于上一次变换结果进行的，而不是相对于最初的图形进行的。因为，每次比例变换后得到的坐标数据都取代了原来的坐标数据。

3．旋转变换

除了平移变换和比例变换之外，图形操作中经常需要用到的第 3 种基本变换是旋转变换，即把屏幕上显示的图形绕某一点旋转一定角度。

参见图 8—25。设屏幕上某一点的坐标为（X，Y），围绕旋转的基准点的坐标为（XO，YO），旋转的角度为 A（逆时针方向为正，顺时针方向为负，以弧度为单位），则旋转后该点的坐标为：

$$\begin{cases} XR=XO+(X-XO)*COS(A)+(Y-YO)*SIN(A) \\ YR=YO+(Y-YO)*COS(A)-(X-XO)*SIN(A) \end{cases}$$

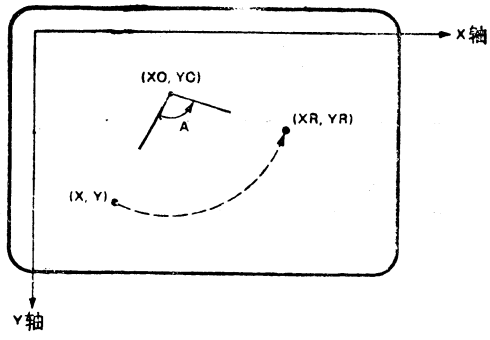


图 8—25　点的旋转变换

注意，基准点的选择可以是任意的，它既可以在屏幕内，也可以在屏幕外，视实际需要而定。

下面是一个综合使用了三种基本变换交互式地进行构图的通用程序。程序中使用的交互设备是键盘，基本的图形符号有直线、矩形、圆、三角形等四种。可以用于构图的基本操作有平移、比例变换、旋转、擦除等。图 8—26 是本程序运行过程中的一个画面。

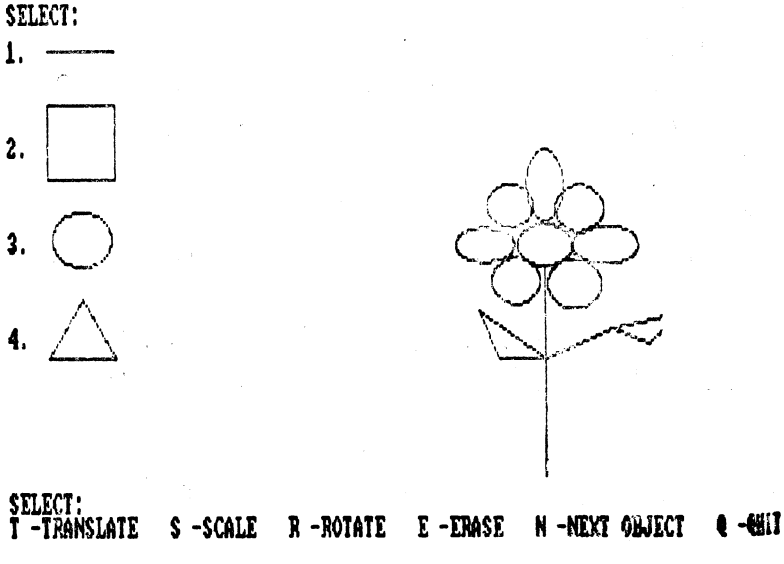


图 8—26　　利用三种基本变换进行构图

例 8—19　　利用三种基本变换进行交互式构图

程序：

```
10 'PROGRAM      BUILDING PICTURES FROM COMPONENT PARTS.
70 SCREEN 2: CLS
80 YA = 5/12              'YA AND XA ARE RESOLUTION ADJUSTMENTS
90 XA = 12/5
100 GOSUB 460             'MAKE MENUS
110 AA = 0        'AA IS ACCUMULATED ANGLE THAT A CIRCLE HAS BEEN ROTATED
120      '************************** INPUT CHOICES **********************
130 LOCATE 1,1: PRINT "          ";: FOR DELAY=1 TO 300: NEXT
140 LOCATE 1,1: PRINT "SELECT:";: FOR DELAY=1 TO 300: NEXT
150 A$ = INKEY$           'A$ IS CHOICE OF SHAPE
160 IF A$ = "" OR A$ < "1" OR A$ > "4" THEN 130          'INVALID CHOICE
170 RO = VAL(A$) * 4
180 N = 0                 'N INDICATES WHETHER CURRENT DISPLAY OF SHAPE
190 LOCATE 23,1: PRINT "SELECT:";          'SHOULD BE ERASED OR NOT
200 LOCATE RO,1: PRINT " ";: FOR DELAY=1 TO 300: NEXT
210 LOCATE 23,1: PRINT "          ";
220 LOCATE RO,1: PRINT A$: FOR DELAY=1 TO 300: NEXT
230 B$ = INKEY$           'B$ IS CHOICE OF TRANSFORMATION
240 IF B$ = "" OR B$<>"T" AND B$<>"S" AND B$<>"R" AND B$<>"E" AND
          B$<>"N" AND B$<>"Q" THEN 190
250 IF B$ = "Q" THEN 2310
260 IF B$ <> "N" THEN 310
270 RESTORE                        'GO BACK TO BEGINNING OF DATA
280 GOSUB 600                      'READ DATA POINTS
290 GOTO 110
300      '******** ERASING? ********
310 IF B$ = "E" THEN C = 0: ON VAL(A$) GOSUB 680, 710, 770, 990: RESTORE:
          GOSUB 600 : GOTO 110
320      '**************************
330 LOCATE 23,1: PRINT "          ";        'ERASE MENU TO MAKE ROOM FOR OTHER
340 LOCATE 24,1: PRINT STRING$(79," ");          'TRANSFORMATIONS INSTRUCTIONS
350      '****** TRANSLATING? ******
360 IF B$ = "T" THEN GOSUB 1310: C = 0: ON VAL(A$) GOSUB 1050, 1110, 1190, 1240
370      '******** SCALING? ********
380 IF B$ = "S" THEN GOSUB 1670: C = 0: ON VAL(A$) GOSUB 1350, 1420, 1520, 1580
390      '******** ROTATING? ********
400 IF B$ = "R" THEN GOSUB 2220: C = 0: ON VAL(A$) GOSUB 1720, 1840, 2020, 2080
410      '************************************************************
420 N = 1                 'FROM NOW ON, ERASE OLD PICTURE OF SHAPE
430 GOSUB 560             'REDISPLAY TRANSFORMATION MENU
440 GOTO 190
450 '
460 '######################## DISPLAY MENUS ##########################
470 LOCATE 1,1: PRINT "SELECT:";
480 FOR K=1 TO 4:LOCATE K*4,1:PRINT RIGHT$(STR$(K),1);".";:NEXT
490 C = 3
500 GOSUB 600             'READ DATA POINTS
510 GOSUB 690             'DRAW LINE
520 GOSUB 720             'DRAW BOX
530 AA = 0
540 GOSUB 780             'DRAW CIRCLE
550 GOSUB 1000            'DRAW TRIANGLE
560 LOCATE 23,1: PRINT "SELECT:"; STRING$(50," ");
570 LOCATE 24,1: PRINT "T -TRANSLATE   S -SCALE   R -ROTATE";
580 LOCATE 24,39: PRINT "E -ERASE   N -NEXT OBJECT   Q -QUIT";
590 RETURN
600      '************* READ DATA POINTS *********************
610 READ XP,YP,XQ,YQ
620 READ XU,YU,XV,YV,XW,YW,XX,YX
630 READ XC,YC,RX,RY
640 READ XR,YR,XS,YS,XT,YT
650 RETURN
660 '
670 '######################## DRAW ROUTINES ##########################
680      '***************** DRAW LINE ************
690 LINE (XP,YP) - (XQ,YQ),C
700 RETURN
710      '***************** DRAW BOX ************
720 LINE (XU,YU) - (XV,YV),C
730 LINE (XV,YV) - (XW,YW),C
740 LINE (XW,YW) - (XX,YX),C
750 LINE (XX,YX) - (XU,YU),C
```

```
760 RETURN
770     '*************** DRAW CIRCLE ***********
780 IF RX = RY THEN CIRCLE (XC,YC),RX,C: GOTO 980
790 IF RX < RY THEN R = RX ELSE R = RY
800 IF AA <> 0 THEN 890
810 FOR A = 0 TO 1.5708 STEP 1/R
820     DX = RX * COS(A): DY = RY * SIN(A)
830     PSET (XC+DX,YC+DY*YA),C
840     PSET (XC+DX,YC-DY*YA),C
850     PSET (XC-DX,YC+DY*YA),C
860     PSET (XC-DX,YC-DY*YA),C
870 NEXT
880 GOTO 980
890 CX = SIN(AA) * XA           'CALCULATE CONSTANT PART OF EQUATION
900 CY = SIN(AA) * YA
910 FOR A = 0 TO 3.14159 STEP 1/R
920    X = RX * COS(A): Y = RY * SIN(A) * YA
930    XH = X
940    X = X * COS(AA) + Y * CX: Y = Y * COS(AA) - XH * CY
950    PSET (XC+X,YC+Y),C
960    PSET (XC-X,YC-Y),C
970 NEXT A
980 RETURN
990     '************ DRAW TRIANGLE ***********
1000 LINE (XR,YR) - (XS,YS),C: LINE - (XT,YT),C: LINE - (XR,YR),C
1010 RETURN
1020
1030 '############################## TRANSLATE ##################################
1040     '*********** TRANSLATE LINE **********
1050 IF N <> 0 THEN GOSUB 680   'IF N IS 0 SHAPE IS STILL IN MEMO-DON'T ERASE
1060 XP = XP + HT: YP = YP + VT
1070 XQ = XQ + HT: YQ = YQ + VT
1080 C = 3: GOSUB 680                   'DRAW
1090 RETURN
1100     '*********** TRANSLATE BOX ***********
1110 IF N <> 0 THEN GOSUB 710
1120 XU = XU + HT: YU = YU + VT
1130 XV = XV + HT: YV = YV + VT
1140 XW = XW + HT: YW = YW + VT
1150 XX = XX + HT: YX = YX + VT
1160 C = 3: GOSUB 710                   'DRAW
1170 RETURN
1180    '********** TRANSLATE CIRCLE **********
1190 IF N <> 0 THEN GOSUB 770
1200 XC = XC + HT: YC = YC + VT
1210 C = 3: GOSUB 770                   'DRAW
1220 RETURN
1230    '********** TRANSLATE TRIANGLE *******
1240 IF N <> 0 THEN GOSUB 990
1250 XR = XR + HT: YR = YR + VT
1260 XS = XS + HT: YS = YS + VT
1270 XT = XT + HT: YT = YT + VT
1280 C = 3: GOSUB 990                   'DRAW
1290 RETURN
1300    '************ INSTRUCTIONS ************
1310 LOCATE 23,1: INPUT "HORIZONTAL AND VERTICAL DISTANCE TO TRANSLATE"; HT,VT
1320 RETURN
1330 '
1340 '############################## SCALING ####################################
1350    '************* SCALE LINE *************
1360 IF N <> 0 THEN GOSUB 680
1370 XF = (XP + XQ) / 2: YF = (YP + YQ) / 2
1380 XP = XP * HS + XF * (1 - HS): YP = YP * VS + YF * (1 - VS)
1390 XQ = XQ * HS + XF * (1 - HS): YQ = YQ * VS + YF * (1 - VS)
1400 C = 3: GOSUB 680           'DRAW
1410 RETURN
1420    '*************** SCALE BOX ************
1430 IF N <> 0 THEN GOSUB 710
1440 XF = (XU + XV + XW + XX) / 4
1450 YF = (YU + YV + YW + YX) / 4
1460 XU = XU * HS + XF * (1 - HS): YU = YU * VS + YF * (1 - VS)
1470 XV = XV * HS + XF * (1 - HS): YV = YV * VS + YF * (1 - VS)
```

```
1480 XW = XW * HS + XF * (1 - HS): YW = YW * VS + YF * (1 - VS)
1490 XX = XX * HS + XF * (1 - HS): YX = YX * VS + YF * (1 - VS)
1500 C = 3: GOSUB 710          'DRAW
1510 RETURN
1520    '************ SCALE CIRCLE **********
1530 IF N <> 0 THEN GOSUB 770
1540 RX = RX * HS
1550 RY = RY * VS
1560 C = 3: GOSUB 770          'DRAW
1570 RETURN
1580    '************ SCALE TRIANGLE *********
1590 IF N <> 0 THEN GOSUB 990
1600 XF = (XR + XS + XT) / 3
 610 YF = (YR + YS + YT) / 3
1620 XR = XR * HS + XF * (1 - HS): YR = YR * VS + YF * (1 - VS)
1630 XS = XS * HS + XF * (1 - HS): YS = YS * VS + YF * (1 - VS)
1640 XT = XT * HS + XF * (1 - HS): YT = YT * VS + YF * (1 - VS)
1650 C = 3: GOSUB 990          'DRAW
1660 RETURN
1670    '************ INSTRUCTIONS ***********
1680 LOCATE 23,1: INPUT "ENTER X AND Y SCALING FACTORS"; HS,VS
1690 RETURN
1700 '
1710 '################################ ROTATION ##############################
1720    '************** ROTATE LINE ************
1730 IF N <> 0 THEN GOSUB 680
1740 XO = (XP + XQ) / 2
1750 YO = (YP + YQ) / 2
1760 XH = XP                   'HOLD VALUE OF XP FOR USE IN Y CALCULATION
1770 XP = XO + (XP - XO) * COS(AR) + (YP - YO) * SIN(AR) * XA
1780 YP = YO + (YP - YO) * COS(AR) - (XH - XO) * SIN(AR) * YA
1790 XH = XQ
1800 XQ = XO + (XQ - XO) * COS(AR) + (YQ - YO) * SIN(AR) * XA
1810 YQ = YO + (YQ - YO) * COS(AR) - (XH - XO) * SIN(AR) * YA
1820 C = 3: GOSUB 680          'DRAW
1830 RETURN
1840    '*********** ROTATE BOX **************
1850 IF N <> 0 THEN GOSUB 710
1860 XO = (XU + XV + XW + XX) / 4
1870 YO = (YU + YV + YW + YX) / 4
1880 XH = XU                   'HOLD CURRENT VALUE OF XU
1890 XU = XO + (XU - XO) * COS(AR) + (YU - YO) * SIN(AR) * XA
1900 YU = YO + (YU - YO) * COS(AR) - (XH - XO) * SIN(AR) * YA
1910 XH = XV
1920 XV = XO + (XV - XO) * COS(AR) + (YV - YO) * SIN(AR) * XA
1930 YV = YO + (YV - YO) * COS(AR) - (XH - XO) * SIN(AR) * YA
1940 XH = XW
1950 XW = XO + (XW - XO) * COS(AR) + (YW - YO) * SIN(AR) * XA
1960 YW = YO + (YW - YO) * COS(AR) - (XH - XO) * SIN(AR) * YA
1970 XH = XX
1980 XX = XO + (XX - XO) * COS(AR) + (YX - YO) * SIN(AR) * XA
1990 YX = YO + (YX - YO) * COS(AR) - (XH - XO) * SIN(AR) * YA
2000 C = 3: GOSUB 710          'DRAW
2010 RETURN
2020    '*********** ROTATE CIRCLE ********************
2030 IF N <> 0 THEN AA = AS: GOSUB 770
2040 AA = AR + AS
2050 C = 3: GOSUB 770          'DRAW
2060 AS = AA
2070 RETURN
2080    '*********** ROTATE TRIANGLE **********
2090 IF N <> 0 THEN GOSUB 990
2100 XO = (XR + XS + XT) / 3: YO = (YR + YS + YT) / 3
2110 XH = XR
2120 XR = XO + (XR - XO) * COS(AR) + (YR - YO) * SIN(AR) * XA
2130 YR = YO + (YR - YO) * COS(AR) - (XH - XO) * SIN(AR) * YA
2140 XH = XS
2150 XS = XO + (XS - XO) * COS(AR) + (YS - YO) * SIN(AR) * XA
2160 YS = YO + (YS - YO) * COS(AR) - (XH - XO) * SIN(AR) * YA
2170 XH = XT
2180 XT = XO + (XT - XO) * COS(AR) + (YT - YO) * SIN(AR) * XA
2190 YT = YO + (YT - YO) * COS(AR) - (XH - XO) * SIN(AR) * YA
```

```
2200 C = 3: GOSUB 990
2210 RETURN
2220     '********** INSTRUCTIONS ****************
2230 LOCATE 23,1: INPUT "NUMBER OF DEGREES TO ROTATE"; AR
2240 AR = AR * 3.14159 /180          'CONVERT AR TO RADIANS
2250 RETURN
2260 '[illegible]
2270 DATA 30,27,80,27
2280 DATA 80,45,80,70,30,70,30,45
2290 DATA 55,90,22,22
2300 DATA 30,130,80,130,55,110
2310 END
```

说明：

① 本程序工作在高分辨显示模式下，初始的画面为：左面是四个基本图形符号组成的“菜单”，供用户选择构图。下面是六个命令组成的“菜单”，供用户选择对基本符号进行各种变换。其中E命令（ERASE）表示擦掉正在进行处理的图形符号并重新另选一个；N命令（NEXT OBJECT）表示这一个符号的变换处理已经结束，开始选择下一个新的图形符号；Q命令表示构图完毕，结束程序的工作。

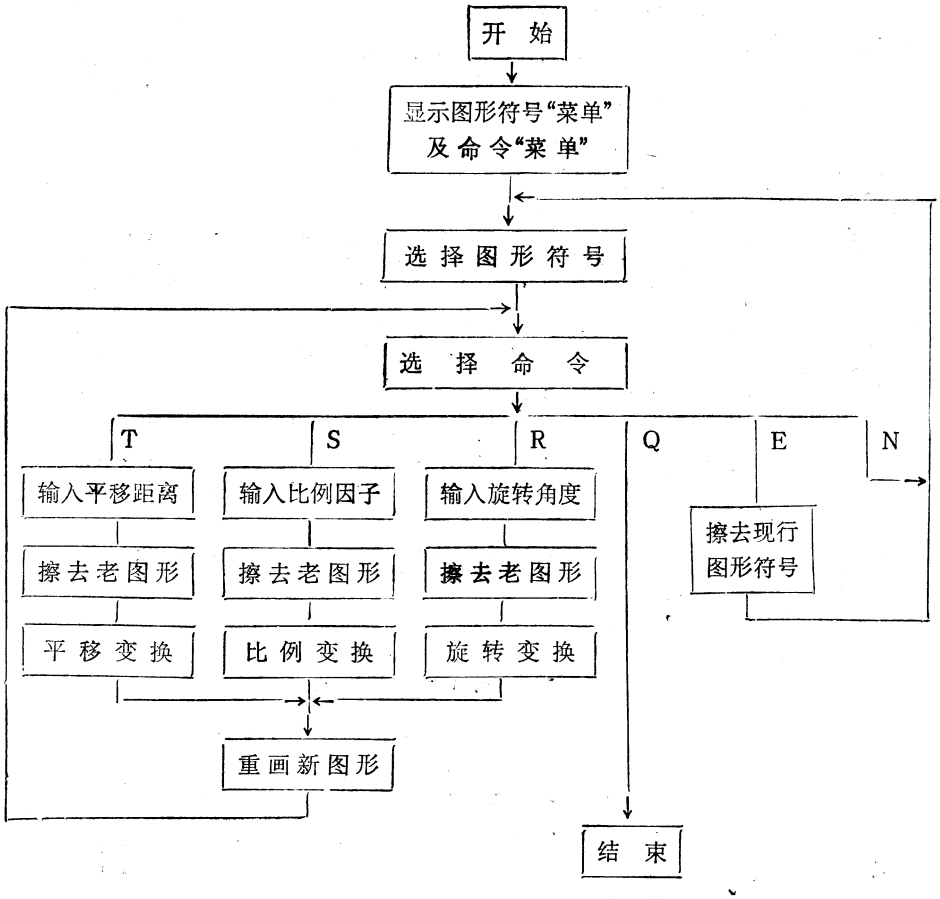


图8—27　例8—19的控制流程

② 与例8—18类似，操作过程中程序会通过不断闪烁着的提示符来提醒用户应该进行何种操作，直到用户做了正确的选择为止。

③ 整个程序的控制流程如图8—27所示。从流程中可以看出，本程序的缺点是只能向画面添加新的图形符号，而不能对已有的符号再次进行修改或擦去。

④ 三种基本变换中，比例变换及旋转变换都是相对于每个图形符号的中心位置来进行的，即直线的中点、矩形的中心、圆的圆心、三角形的中心等，这样，便于用户选择合适的变换因子或变换角度来达到预想的要求。

前面介绍的三种基本变换，在使用PSET、LINE及CIRCLE语句作图时都经常用到。但是，如果使用IBM PC的高级BASIC语言中所提供的DRAW语句来构造图形，由于DRAW语句本身具有处理一些简单的图形变换的能力，因此使用起来比较方便。例如，下面语句中的S12命令就把紧接着的一个三角形放大三倍。

```
DRAW "BM10, 150; S12; R40; H20; G20; BM50, 150; S4; R40; H20;
      G20"
```

又如，下面的DRAW语句中的A2命令把紧接着的一个三角形旋转180度，旋转的基准点坐标是(100, 100)。

```
DRAW "BM100, 100; A2; R50; H25; G25"
```

4．窗口与裁剪操作

往往有这样的情况，用户只对某一幅图画的其中一个特定区域内的画面部分发生兴趣，例如，根据需要可以把它放大，以便仔细地研究其细节，或者为突出它所表达的内容，而把

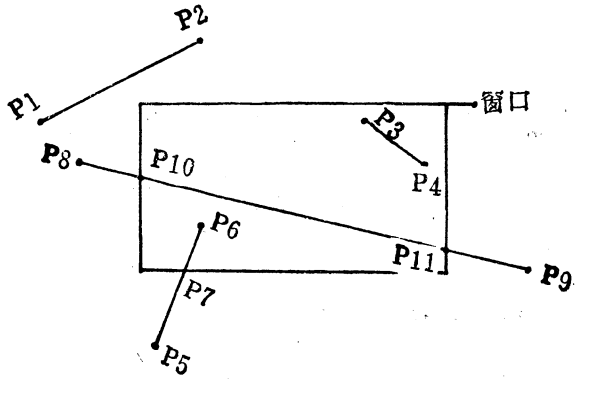


图8—28　对线段进行裁剪的各种情况

其周围的画面部分全部擦去。图画上的这种特定区域往往是用矩形来指定的，该矩形区域就叫做“窗口”，保留窗口内的画面而把窗口外的画面部分全部擦去，则叫做“裁剪”操作。

一条线段的裁剪操作有各种情况。例如，图8—28中的线段P1 P2应被完全擦去，而P3 P4需全部保留，P8 P9应剪去P8 P10和P11 P9两段，P5 P6则应剪去窗口外的P5 P7

部分。

下面通过一个实例来介绍直线的裁剪算法。

例 8—20　图形的裁剪

本例首先在屏幕上显示出一架飞机的图形，然后由用户任意定义一个窗口，如图 8—29（a）所示，于是程序把窗口外的图形全部擦去，只保留窗口及其中所含的图形部分（图8—29（b））。

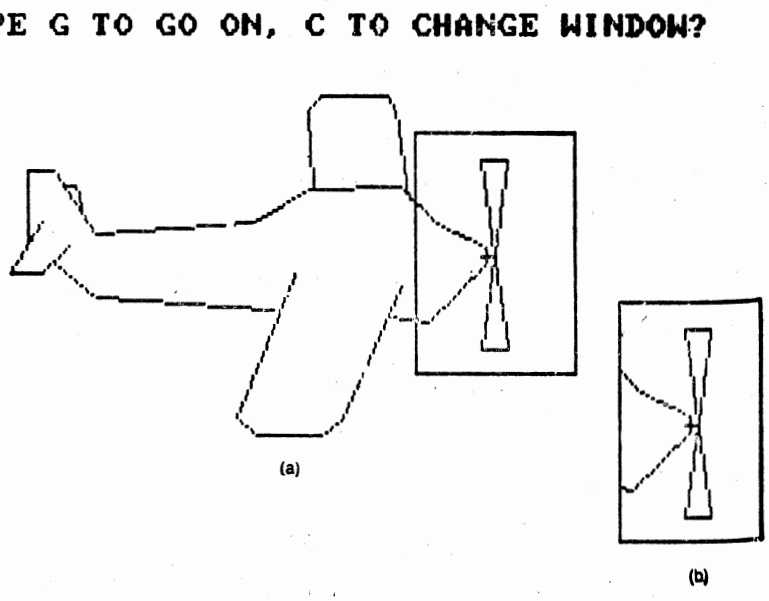


图 8—29　图形的裁剪

程序：

```
10 'PROGRAM      CLIPPING OUTISDE A WINDOW.
160 DIM X(8,20),Y(8,20),X1(8,20),Y1(8,20),X2(8,20),Y2(8,20)
170 SCREEN 1: COLOR 1,1: CLS
180 GOSUB 240            'READ PICTURE POINTS
190 GOSUB 330            'DRAW PICTURE
200 GOSUB 420            'ESTABLISH WINDOW
210 GOSUB 630            'CLIP
220 GOSUB 2010           'DRAW CLIPPED POINTS
230 GOTO 2240
240     '*************** READ PICTURE PARTS ********************
250 READ TOTAL                'TOTAL IS NUMBER OF PICTURE PARTS
260 FOR PART = 1 TO TOTAL
270     READ POINTCOUNT(PART)        'NUMBER OF ELEMENTS IN PART P
280     FOR VERTEX = 1 TO POINTCOUNT(PART)
290         READ X(PART,VERTEX), Y(PART,VERTEX)
300     NEXT
310 NEXT
320 RETURN
330     '*****************  DRAW ROUTINE **********************
340 FOR PART = 1 TO TOTAL
350     IF POINTCOUNT(PART) = 1 THEN PSET (X(PART,1),Y(PART,1)): GOTO 400
360     PSET (X(PART,1),Y(PART,1))
370     FOR VERTEX = 2 TO POINTCOUNT(PART)
380         LINE - (X(PART,VERTEX),Y(PART,VERTEX))
390     NEXT
```

```
400 NEXT
410 RETURN
420     '**************** ESTABLISH WINDOW ********************
430 LOCATE 1,1: INPUT "TOP LEFT CORNER OF WINDOW"; XWINDOW,YWINDOW
440 LOCATE 1,1: PRINT STRING$(80," ");
450 LOCATE 1,1: INPUT "WIDTH AND HEIGHT OF WINDOW";WINDOWWIDTH,WINDOWHEIGHT
460 LOCATE 1,1: PRINT STRING$(80,32);
470 LEFT = XWINDOW
480 RIGHT = XWINDOW + WINDOWWIDTH
490 TOP = YWINDOW
500 BOTTOM = YWINDOW + WINDOWHEIGHT
510 IF LEFT < RIGHT AND LEFT >= 0 AND RIGHT <= 319 AND TOP < BOTTOM AND
      TOP >= 0 AND BOTTOM <= 199 THEN 550
520 LOCATE 1,1: PRINT "WINDOW OFF SCREEN. TRY AGAIN"
530 LOCATE 1,1: PRINT STRING$(80,32);
540 GOTO 430
550 LINE (LEFT,TOP) - (RIGHT,BOTTOM),,B
560 LOCATE 1,1: INPUT "TYPE G TO GO ON, C TO CHANGE WINDOW"; M$
570 IF M$ = "G" THEN 620
580    'ERASE CURRENT POSITION
590 LOCATE 1,1: PRINT STRING$(80,32);
600 LINE (LEFT,TOP) - (RIGHT,BOTTOM),0,B
610 GOTO 420
620 RETURN
630 '########################## CLIPPING ROUTINE ##############################
640    'CLIP POINTS IN X AND Y AGAINST LEFT EDGE. STORE IN X1 AND Y1.
650 PARTSIN = 1
660 FOR PART = 1 TO TOTAL
670     VERTIN = 0
680     FOR VERT = 1 TO POINTCOUNT(PART) - 1
690         IF X(PART,VERT) < LEFT THEN 770
700         VERTIN = VERTIN + 1                    'FIRST PART OF LINE IS IN
710         X1(PARTSIN,VERTIN) = X(PART,VERT)
720         Y1(PARTSIN,VERTIN) = Y(PART,VERT)
730             'WHAT ABOUT SECOND POINT?
740         IF X(PART,VERT+1) < LEFT THEN GOSUB 900    'FIND INTERSECTION
750         GOTO 790             'ELSE IT'S IN, SO JUST CONTINUE
760             '-------------------------------------------------
770            'FIRST POINT IS OUT. WHAT ABOUT SECOND POINT?
780     .      IF X(PART,VERT+1) >= LEFT THEN GOSUB 900   'FIND INTERSECTION
790     NEXT
800     IF X(PART,POINTCOUNT(PART)) < LEFT THEN 840          'DO FINAL POINT
810     VERTIN = VERTIN + 1
820     X1(PARTSIN,VERTIN) = X(PART,POINTCOUNT(PART))
830     Y1(PARTSIN,VERTIN) = Y(PART,POINTCOUNT(PART))
840        VERTIN = 0 THEN 870                  'NO ELEMENTS IN WINDOW
850     POINTCOUNTIN(PARTSIN) = VERTIN          'ELSE, SAVE COUNT OF IN POINTS
860     PARTSIN = PARTSIN + 1                   'GO ON TO NEXT PART
870 NEXT
880 TOTALIN = PARTSIN - 1       'TOTALIN IS NUMBER OF PARTS WITH POINTS INSIDE
890 GOTO 970                   'GO ON TO NEXT EDGE
900         '^^^^^^^^ FIND INTERSECTION ROUTINE ^^^^^^^^^^
910 VERTIN = VERTIN + 1
920 SLOPE = (Y(PART,VERT+1) - Y(PART,VERT)) / (X(PART,VERT+1) - X(PART,VERT))
930 Y1(PARTSIN,VERTIN) = SLOPE * (LEFT - X(PART,VERT)) + Y(PART,VERT)
940 X1(PARTSIN,VERTIN) = LEFT
950 RETURN
960                 '****************************
970    'CLIP POINTS IN X1, Y1 AGAINST TOP. STORE INSIDE POINTS IN X2, Y2
980 PARTSIN = 1
990 FOR PART = 1 TO TOTALIN
1000     VERTIN = 0
1010     FOR VERT = 1 TO POINTCOUNTIN(PART) - 1
1020         IF Y1(PART,VERT) <  TOP THEN 1100
1030         VERTIN = VERTIN + 1                'FIRST PART OF LINE IS IN
1040         Y2(PARTSIN,VERTIN) = Y1(PART,VERT)
1050         X2(PARTSIN,VERTIN) = X1(PART,VERT)
1060            'WHAT ABOUT SECOND POINT?
1070         IF Y1(PART,VERT+1) < TOP THEN GOSUB 1230   'FIND INTERSECTION
1080         GOTO 1120          'ELSE IT'S IN SO JUST CONTINUE
1090            '-------------------------------------------------
1100            'FIRST POINT IS OUT. WHAT ABOUT SECOND POINT?
```

```
1110          IF Y1(PART,VERT+1) >= TOP THEN GOSUB 1230  'FIND INTERSECTION
1120      NEXT
1130      IF Y1(PART,POINTCOUNTIN(PART)) < TOP THEN 1170        'DO FINAL POINT
1140      VERTIN = VERTIN + 1
1150      Y2(PARTSIN,VERTIN) = Y1(PART,POINTCOUNTIN(PART))
1160      X2(PARTSIN,VERTIN) = X1(PART,POINTCOUNTIN(PART))
1170      IF VERTIN = 0 THEN 1200                   'NO ELEMENTS IN WINDOW
1180      POINTCOUNTIN(PARTSIN) = VERTIN            'ELSE SAVE COUNT OF IN POINTS
1190      PARTSIN = PARTSIN + 1                     'GO ON TO THE NEXT PART
1200  NEXT
1210  TOTALIN = PARTSIN - 1       'TOTALIN IS # OF PARTS WITH POINTS INSIDE
1220  GOTO 1330                   'GO ON TO NEXT EDGE
1230            '^^^^^^^^^^^^^ FIND INTERSECTION ROUTINE ^^^^^^^^^^^^^^^
1240  VERTIN = VERTIN + 1
1250  IF X1(PART,VERT+1) <> X1(PART,VERT) THEN 1280
1260  X2(PARTSIN,VERTIN) = X1(PART,VERT)                  'VERTICAL LINE
1270  GOTO 1300
1280  SLOPE = (Y1(PART,VERT+1)-Y1(PART,VERT)) / (X1(PART,VERT+1)-X1(PART,VERT)
1290  X2(PARTSIN,VERTIN) = (TOP - Y1(PART,VERT)) / SLOPE + X1(PART,VERT)
1300  Y2(PARTSIN,VERTIN) = TOP
1310  RETURN
1320                    '**************************
1330     'CLIP POINTS IN X2, Y2 AGAINST RIGHT. STORE INSIDE POINTS IN X1, Y1.
1340  PARTSIN = 1
1350  FOR PART = 1 TO TOTALIN
1360      VERTIN = 0
1370      FOR VERT = 1 TO POINTCOUNTIN(PART) - 1
1380          IF X2(PART,VERT) >  RIGHT THEN 1460
1390          VERTIN = VERTIN + 1                   'FIRST POINT IS IN
1400          X1(PARTSIN,VERTIN) = X2(PART,VERT)
1410          Y1(PARTSIN,VERTIN) = Y2(PART,VERT)
1420             'WHAT ABOUT SECOND POINT
1430          IF X2(PART,VERT+1) > RIGHT THEN GOSUB 1590   'FIND INTERSECTION
1440          GOTO 1480           'ELSE IT'S IN SO JUST CONTINUE
1450             '---------------------------------------
1460             'FIRST POINT IS OUT. WHAT ABOUT SECOND POINT?
1470          IF X2(PART,VERT+1)< RIGHT THEN GOSUB 1590  'FIND INTERSECTION
1480      NEXT
1490      IF X2(PART,POINTCOUNTIN(PART)) > RIGHT THEN 1530      'DO FINAL POINT
1500      VERTIN = VERTIN + 1
1510      X1(PARTSIN,VERTIN) = X2(PART,POINTCOUNTIN(PART))
1520      Y1(PARTSIN,VERTIN) = Y2(PART,POINTCOUNTIN(PART))
1530      IF VERTIN = 0 THEN 1560                   'NO ELEMENTS IN WINDOW
1540      POINTCOUNTIN(PARTSIN) = VERTIN            'ELSE SAVE COUNT OF IN POINTS
1550      PARTSIN = PARTSIN + 1                     'GO ON TO NEXT PART
1560  NEXT
1570  TOTALIN = PARTSIN - 1       'TOTALIN IS NUMBER OF PARTS WITH POINTS INSIDE
1580  GOTO 1660                   'GO ON TO NEXT EDGE
1590            '^^^^^^^^^^^^^ FIND INTERSECTION ROUTINE ^^^^^^^^^^
1600  VERTIN = VERTIN + 1
1610  SLOPE = (Y2(PART,VERT+1)-Y2(PART,VERT)) / (X2(PART,VERT+1)-X2(PART,VERT))
1620  Y1(PARTSIN,VERTIN) = SLOPE * (RIGHT - X2(PART,VERT)) + Y2(PART,VERT)
1630  X1(PARTSIN,VERTIN) = RIGHT
1640  RETURN
1650                    '**************************
1660     'CLIP POINTS IN X1, Y1 AGAINST BOTTOM. STORE INSIDE POINTS IN X2, Y2
1670  PARTSIN = 1
1680  FOR PART = 1 TO TOTALIN
1690      VERTIN = 0
1700      FOR VERT = 1 TO POINTCOUNTIN(PART) - 1
1710          IF Y1(PART,VERT) >  BOTTOM THEN 1790
1720          VERTIN = VERTIN + 1                   'FIRST POINT IS IN
1730          Y2(PARTSIN,VERTIN) = Y1(PART,VERT)
1740          X2(PARTSIN,VERTIN) = X1(PART,VERT)
1750             'WHAT ABOUT SECOND POINT?
1760          IF Y1(PART,VERT+1)>BOTTOM THEN GOSUB 1920  'FIND INTERSECTION
1770          GOTO 1810           'ELSE IT'S IN SO JUST CONTINUE
1780             '---------------------------------------
1790             'FIRST POINT IS OUT. WHAT ABOUT SEGOND POINT?
1800          IF Y1(PART,VERT+1)<=BOTTOM THEN GOSUB 1920 'FIND INTERSECTION
1810      NEXT
1820      IF Y1(PART,POINTCOUNTIN(PART)) > BOTTOM THEN 1860      'DO FINAL POINT
```

```
1830      VERTIN = VERTIN + 1
1840      Y2(PARTSIN,VERTIN) = Y1(PART,POINTCOUNTIN(PART))
1850      X2(PARTSIN,VERTIN) = X1(PART,POINTCOUNTIN(PART))
1860      IF VERTIN = 0 THEN 1890                    'NO ELEMENTS IN WINDOW
1870      POINTCOUNTIN(PARTSIN) = VERTIN             'ELSE SAVE COUNT OF IN POINTS
1880      PARTSIN = PARTSIN + 1                      'GO ON TO NEXT PART
1890 NEXT
1900 TOTALIN = PARTSIN - 1        'TOTALIN IS # OF PARTS STILL INSIDE
1910 RETURN                       'END OF CLIPPING ROUTINE
1920            '^^^^^^^^^^^^ FIND INTERSECTION ROUTINE ^^^^^^^^^^
1930 VERTIN = VERTIN + 1
1940 IF X1(PART,VERT+1) <> X1(PART,VERT) THEN 1970
1950 X2(PARTSIN,VERTIN) = X1(PART,VERT)           'VERTICAL LINE
1960 GOTO 1990
1970 SLOPE = (Y1(PART,VERT+1)-Y1(PART,VERT)) / (X1(PART,VERT+1)-X1(PART,VERT))
1980 X2(PARTSIN,VERTIN) = (BOTTOM - Y1(PART,VERT)) / SLOPE + X1(PART,VERT)
1990 Y2(PARTSIN,VERTIN) = BOTTOM
2000 RETURN
2010 '###################### DRAW CLIPPED POINTS ####################
2020 CLS
2030 LINE (LEFT,TOP) - (RIGHT,BOTTOM),,B
2040 FOR PART = 1 TO TOTALIN
2050      FOR VERT = 1 TO POINTCOUNTIN(PART) - 1
2060           LINE (X2(PART,VERT),Y2(PART,VERT)) -
                    (X2(PART,VERT+1),Y2(PART,VERT+1))
2070      NEXT
2080 NEXT
2090 RETURN
2100 '##########################################################################
2110 DATA 7
2120      'OUTLINE
2130 DATA 12,195,127,210,130,232,105,232,100,213,90,200,75,165,77
2140 DATA 145,90,85,95,70,70,60,70,60,97
2150 DATA 3,69,106,85,120,152,125
2160      'WINGS
2170 DATA 6,160,110,138,168,145,175,170,175,178,168,200,115
2180 DATA 6,203,80,198,45,195,40,170,40,165,45,167,76
2190      'TAIL
2200 DATA 3,73,75,78,75,80,85
2210 DATA 4,65,90,53,110,65,110,75,100
2220      'PROPELLOR
2230 DATA 7,230,103,235,102,250,65,240,65,230,140,240,140,235,103
2240 IF INKEY$ = "" THEN 2240
2250 END
```

说明：

① 飞机的图形由七段折线所组成：机身两段，机翼两段，机尾两段。螺旋桨一段。这些折线的顶点的坐标数据开始时被读入数组 X 和 Y 中，并在画面上用 LINE 语句把飞机显示出来。

② 用户通过输入矩形的左上角坐标（XWINDOW，YWINDOW）及其宽度（WINDOWWIDTH）和高度（WINDOWHEIGHT）来定义一个窗口，因此，该窗口的左、右边框的 X 坐标及上、下边框的 Y 坐标可分别计算如下：

LEFT＝XWINDOW
RIGHT＝XWINDOW＋WINDOWWIDTH
TOP＝YWINDOW
BOTTOM＝YWINDOW＋WINDOWHEIGHT

此后，裁剪操作就是针对这四条边框线进行的。

③ 程序中采用 Sutherlan-Hodgman 算法对每条折线逐条进行裁剪。裁剪是按照下列次序进行的：先用左边框线进行裁剪，裁剪后所得折线的各顶点坐标放入数组 X1 和 Y1

中，再用顶边进行裁剪，所得折线的数据放入 X2 和 Y2 中，然后用右边框进行裁剪，生成的数据放在 X1 和 Y1 中，最后用底边进行裁剪，这样就得到裁剪后的最后结果。整个操作过程可以用图 8—30 来表示。

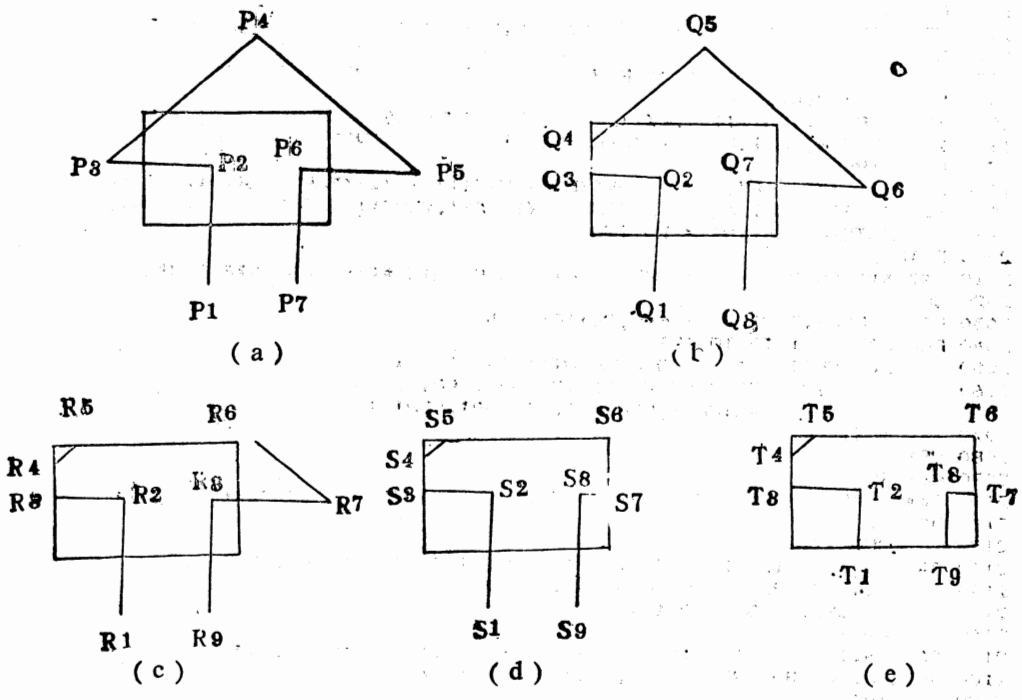


图 8—30　折线的裁剪过程

从图 8—30 不难看出，某顶点 V_i（例如 Pi，Q_i，…，T_i）用某条边框线裁剪时，需进行下列的判断和操作（图 8—31）：

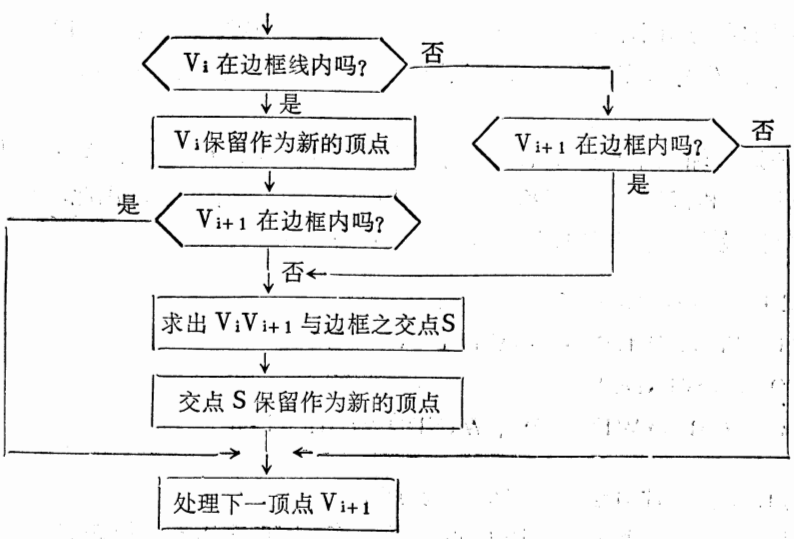


图 8—31　折线的裁剪算法

不过，折线最后一点的处理是特殊的，读者可从程序中仔细加以体会，此处不再赘述。

5．视见变换

当画面上定义了一个窗口并进行裁剪操作之后，用户经常需要把窗口内的画面部分在屏幕上指定位置处的指定区域内显示出来，以便达到放大或缩小窗内图形的目的。屏幕上用于显示窗内画面的那个区域，叫做“视见区”（Viewing area）或简称“视区”（Viewport）。

窗口与视区的差别在于：窗口用来指出在屏幕上要看到的是些什么，而视区则用来指出在屏幕的何处显示其内容。视区通常也是一个矩形区域，它可以比窗口大，也可以比窗口小或者两者完全相同。窗口和视区可以在不同位置上，也可以相互重迭，甚至完全覆盖。事实上，窗口和视区往往是在两个不同的坐标系统下定义的，这种情况本书不予讨论。

当窗口与视区的大小或位置不同时，为了把窗内画面在指定视区中显示出来，程序须进行必要的变换处理，这种变换叫做“视见变换”（Viewing transformation）。

视见变换的原理为：假设窗口左上角坐标为（XW，YW），窗口高度为WH，宽度为WW；而视区左上角坐标为（XV，YV），高度为VH，宽度为VW（图8—32）。设窗口内任一点的坐标为(X，Y)，则它在视区中的对应位置（XN，YN）可用以下的公式来计算：

$$\begin{cases} XN=(X-XW)*(VW/WW)+XV \\ YN=(Y-YW)*(VH/WH)+YV \end{cases}$$

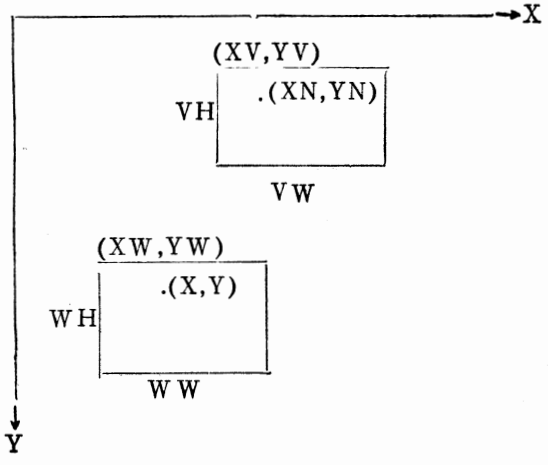


图8—32　视见变换

其中，（VW/WW）和（VH/WH）实际上是比例变换中的比例因子。若它们大于1，则表示图形被放大，小于1表示图形被缩小，如果两者不相等，则视见变换后它们的形状（纵横比）会发生改变。式中的XV和YV是平移变换中的平移距离，若它们与XW，YW不相等，则表示图形位置有所移动。

下面是例8—20中程序的一个扩充，它把例8—20窗口中的图形显示在用户任意指定的一个视区中。两部分程序合并在一起后（删去语句2250），就是一个完整的窗口操作（裁剪与视见变换）的例子。程序运行的结果见图8—33，其中（a）表示定义一个窗口，（b）表

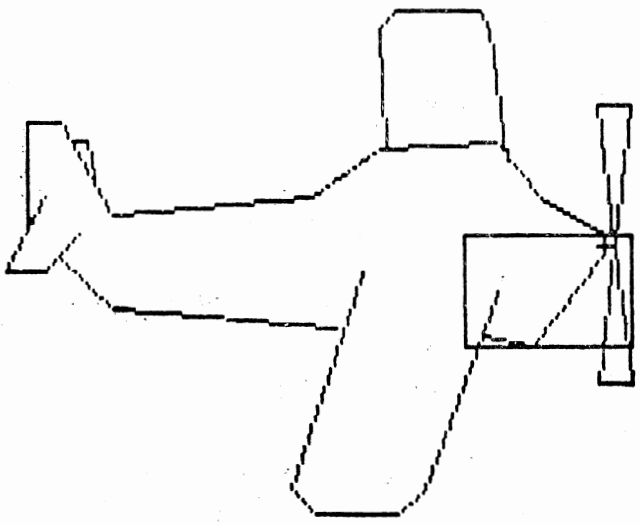
示窗口内的画面被放大显示在一个全屏幕大小的视区中。图中五角星的数据在程序中没有提供，这是为了说明视见变换的作用而故意画出的。

例 8—21　　视见变换

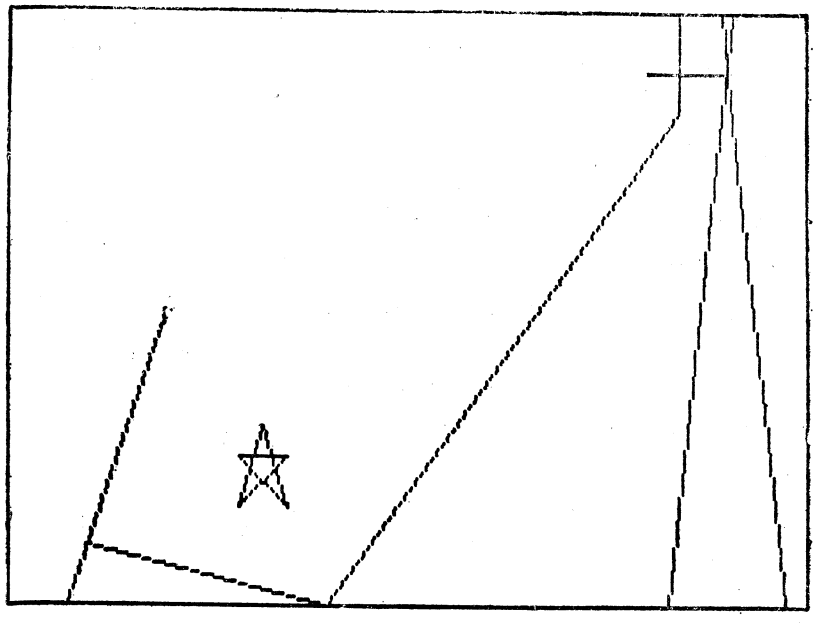
程序：

```
2240 '***** PROGRAM:      DISPLAY WINDOW AREA IN VIEWPORT
2280      '*************************************************************
2290 GOSUB 2330          'ESTABLISH VIEWPORT
2300 GOSUB 2460          'CONVERT WINDOW AREA TO VIEWPORT AREA
2310 GOSUB 2010          'DRAW
2320 GOTO 2540
2330 '********************* ESTABLISH VIEWPORT **********************
2340 LOCATE 1,1: INPUT "TOP LEFT CORNER OF VIEWPORT"; XVIEW,YVIEW
2350 LOCATE 1,1: PRINT STRING$(80,32);
2360 LOCATE 1,1: INPUT "WIDTH AND HEIGHT OF VIEWPORT"; VIEWWIDTH,VIEWHEIGHT
2370 LOCATE 1,1: PRINT STRING$(80,32);
2380 LEFT   = XVIEW
2390 RIGHT  = XVIEW + VIEWWIDTH
2400 TOP    = YVIEW
2410 BOTTOM = YVIEW + VIEWHEIGHT
2420 IF LEFT < RIGHT AND LEFT >= 0 AND RIGHT <= 319 AND TOP < BOTTOM AN
     TOP >= 0 AND BOTTOM <= 199 THEN 2450
2430 LOCATE 1,1: PRINT "VIEWPORT OFF SCREEN. TRY AGAIN"
2440 LOCATE 1,1: PRINT STRING$(80,32);: GOTO 2340
2450 RETURN
2460 '************ CONVERT WINDOW AREA TO VIEWPORT AREA *********
2470 FOR PART = 1 TO TOTALIN
2480     FOR VERT = 1 TO POINTCOUNTIN(PART)
2490         X2(PART,VERT) = (X2(PART,VERT) - XWINDOW) *
                             (VIEWWIDTH / WINDOWWIDTH) + XVIEW
2500         Y2(PART,VERT) = (Y2(PART,VERT) - YWINDOW) *
                             (VIEWHEIGHT / WINDOWHEIGHT) + YVIEW
2510     NEXT
2520 NEXT
2530 RETURN
2540 END
```

```
TYPE G TO GO ON, C TO CHANGE WINDOW?
```



(a)　原始画面及定义的窗口



（b）执行视见变换后的画面

图8—33　视见变换

第四节　动画技术

所谓动画技术，是指屏幕上显示出来的画面或画面中的一部分，能够按一定规则或要求在屏幕上进行活动，使计算机显示输出的图形具有更强的真实感。动画在模拟、教学、游戏等应用场合尤为有用。

下面把画面上的活动部分称为“动画对象”，有时简称“对象”。实际上，动画是对动画对象重复地进行图形变换操作的结果。即程序中对动画对象反复地进行如下操作：先显示出该对象的图形，再进行变换，然后擦去原来的图形，最后再显示出变换后所得到的图形。就好象动画影片的制作过程一样。

本节将通过一组实例介绍在IBM PC上常用的动画技术。

1、字符动画

由特殊字符或图形字符构成的动画对象，可以在屏幕上按一定规则移动，例如从左到右或自上而下等。每次移动的距离，是以字符的行和列来计算的。为了减少画面的闪烁，应当在每次计算出动画对象的新的行、列位置之后，才把原来的图形擦去，然后立即显示新的图形，并保持一定的时间。

当画面上既有动画对象（如一个球），又有静止对象（如一堵墙）时，或者有两个以上动画对象(如一个球和一个球拍)时，程序中往往需要判断两个对象是否即将发生“碰撞”。比较直接的做法是在程序中记下每个对象的现行坐标位置，然后进行测试。这种方法在动

画对象较多时显得很不方便。为此，IBM PC 的 BASIC 语言提供了一个SCREEN函数（屏幕函数）。借助于这个函数，任何一个动画对象都能很方便地探测出在它活动的路径上是否即将碰上其它的对象。SCREEN 函数的格式如下：

SCREEN (R, C, CA)

其中，R 表示行号，C 表示列号。当参数 CA 缺省或为 0 时，函数的值等于屏幕上位于第 R 行第 C 列上的字符的 ASCII 代码的值。例如，如果函数值为 32（空格），就表示该位置上没有任何东西。当参数 CA 为非 0 时，SCREEN 函数的值就是第 R 行第 C 列上的字符属性的值（字符的属性请参见第一章）。字符的显示色及底色可决定如下：

显示色=SCREEN (R, C, CA) MOD 16

底　色=SCREEN (R, C, CA) MOD 128－显示色

若函数值大于 127，则表示该字符正在闪烁。

下面是字符动画及利用 SCREEN 函数来判断动画对象是否碰撞的一个例子。

例 8—22　　导弹打飞机

程序：

```
10 'PROGRAM      SHOOT THE AIRPLANE!
60  SCREEN 0: COLOR 7,0,0: WIDTH 80: LOCATE ,,0: CLS
70  PLANEROW = 2: PLANECOLUMN = 1'STARTING POSITIONS
80  MISSILEROW = 21: MISSILECOLUMN = 39
90  WHILE A$ <> "Q" AND A$ <> "q"
100     COLOR 0: CLS    'SET ATTRIBUTE OF UNUSED SCREEN AREAS TO 0
110     COLOR 7: GOSUB 260       'DRAW PLANE
120     IF PLANECOLUMN < 70 THEN PLANECOLUMN = PLANECOLUMN + 2
           ELSE PLANECOLUMN = 1
130     A$ = INKEY$
140     IF A$ = " " THEN FIRE$ = "YES"
150     IF FIRE$ <> "YES" THEN 240
160     MISSILEROW = MISSILEROW - 2: MISSILECOLUMN = MISSILECOLUMN + 2
170     IF MISSILEROW < 1 OR MISSILECOLUMN > 80 THEN FIRE$ = "NO":
           MISSILEROW = 21: MISSILECOLUMN = 39: GOTO 240
180           'IF THE FOREGROUND OF THE MISSILE'S NEXT POSITION
190           'IS 7 (WHITE) WE MUST BE AT THE PLANE!!
200           'FOREGROUND OF NON-PLANE AREAS WOULD BE 0 (BLACK)
210           'SINCE IN LINE 60 WE SET FOREGROUND TO BLACK AND THEN
220           'CLEARED THE SCREEN
230     IF SCREEN(MISSILEROW,MISSILECOLUMN,1) MOD 16 = 7 THEN GOSUB 310
           ELSE LOCATE MISSILEROW,MISSILECOLUMN: PRINT CHR$(254);
240 WEND
250 GOTO 340
260    'MAKE AIRPLANE
270 LOCATE PLANEROW  ,PLANECOLUMN: PRINT "  " + CHR$(220) + "   " + CHR$(219);
280 LOCATE PLANEROW+1,PLANECOLUMN: PRINT "  " + STRING$(6,219) + CHR$(254);
290 LOCATE PLANEROW+2,PLANECOLUMN: PRINT "  " + CHR$(223) + "   " + CHR$(219)
300 RETURN
310 COLOR 23: PRINT "BULLSEYE!!!";: FOR HOLD = 1 TO 3000: NEXT
320 PLANECOLUMN = 1: FIRE$ = "NO": MISSILEROW = 21: MISSILECOLUMN = 39
330 RETURN
340 END
```

说明：

① 程序中的语句 60，用来选择图形显示器的工作模式为字符模式：COLOR 7, 0, 0 使屏幕的边界色及底色为黑色，字符的显示色为白色，LOCATE ,, 0 用来关闭光标。

② 语句 100 用 COLOR 0 来擦除动画对象，使整个屏幕成为黑色，不显示任何图形，然后立即再通过 110～120 语句画出飞机的图形（白色），并计算出它的下一屏幕位置。程

序工作时显示器上的画面如图 8—34 所示。

③ 飞机是由 12 个图形字符（三行四列）组合而成的，语句 270～300 的子程序用来画出飞机的图形。由于它的显示位置不断改变，看起来好象它在空中自左向右飞行一样（110～120）。

④ 导弹仅用一个图形字符来表示，其编码为 254。当用户按下空格键后，它就从屏幕底部向屏幕右上角射出。在它的运动过程中，如它的下一屏幕位置上字符属性中显示色为1（白色）则表示它即将碰上飞机，如果为 0（黑色），则表示未碰上飞机。

⑤ 如果导弹击中了飞机，则屏幕上将闪烁地出现“BULLSEYE”（命中），然后再继续开始。

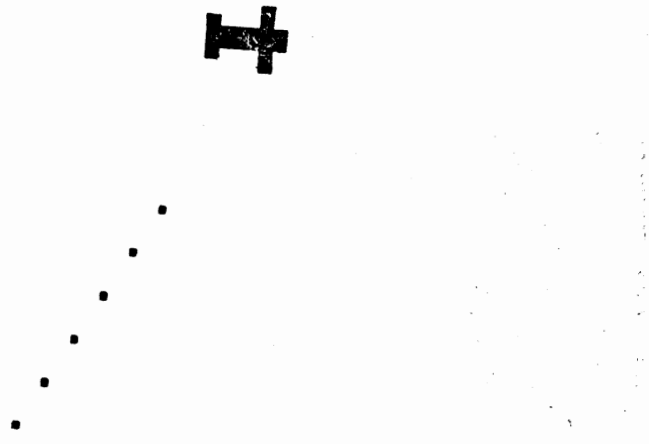


图 8—34 导弹打飞机

从例 8—22 可以看出，动画对象的图形构成越复杂，程序处理的速度就越慢，动画的效果就会受到影响。在字符显示模式下，为了加快动画的速度，可以使用多个显示页面同时进行操作。例 8—23 是同时使用四个页面实现快速动画的一个例子。

例 8—23 使用多个页面实现快速动画

程序：

```
10  PROGRAM      CREEPING WORM ON MULTIPLE SCREENS
20  SCREEN 0: COLOR 2,0,0: WIDTH 80: LOCATE ,,0: CLS
30  ROW = 20: COLUMN = 1
40  A$ = INKEY$
50  SCREEN ,,1,0: CLS
60  WHILE A$ = ""
70      GOSUB 280          'DRAW POSITION 1
80      SCREEN ,,2,1: CLS
90      FOR DELAY = 1 TO 400: NEXT
100     IF COLUMN < 65 THEN COLUMN = COLUMN + 8 ELSE COLUMN = 1
110     GOSUB 310          'DRAW POSITION 2
120     SCREEN ,,3,2: CLS
130     FOR DELAY = 1 TO 300: NEXT
140     IF COLUMN < 68 THEN COLUMN = COLUMN + 7 ELSE COLUMN = 1
150     GOSUB 380          'DRAW POSITION 3
160     SCREEN ,,0,3: CLS
170     FOR DELAY = 1 TO 300: NEXT
180     IF COLUMN < 68 THEN COLUMN = COLUMN + 5 ELSE COLUMN = 1
190     GOSUB 310          'DRAW POSITION 2
```

```
200     SCREEN ,,1,0: CLS
210     FOR DELAY = 1 TO 300: NEXT
220     IF COLUMN < 65 THEN COLUMN = COLUMN + 7 ELSE COLUMN = 1
230     A$ = INKEY$
240 WEND
250 SCREEN ,,0,0          'GO BACK TO FIRST SCREEN
260 GOTO 460
270     'DRAWS POSITION #1
280 LOCATE ROW,COLUMN: PRINT STRING$(8,219);: COLOR 14,0: PRINT CHR$(223);
290 LOCATE ROW-1,COLUMN:PRINT STRING$(8,32)+CHR$(220);:COLOR 2,0
300 RETURN
305     'DRAWS POSITION #2
310 LOCATE ROW,COLUMN
320 PRINT STRING$(2,CHR$(219)) + "   " + STRING$(2,CHR$(219));
330 COLOR 14,0: PRINT CHR$(223);: COLOR 2,0
340 LOCATE ROW-1,COLUMN: PRINT " " + CHR$(219) + "   " + CHR$(219) + " ";
350 COLOR 14,0: PRINT CHR$(220);:COLOR 2,0
360 LOCATE ROW-2,COLUMN: PRINT "  " + STRING$(3,CHR$(219)) + " ";
370 RETURN
375     'DRAWS POSITION #3
380 LOCATE ROW,COLUMN: PRINT CHR$(219) + "   " + CHR$(219);
390 COLOR 14,0: PRINT CHR$(223);: COLOR 2,0
400 LOCATE ROW-1,COLUMN: PRINT " " + CHR$(219) + " " + CHR$(219) + " ";
410 COLOR 14,0: PRINT CHR$(220);: COLOR 2,0
420 LOCATE ROW-2,COLUMN: PRINT " " + CHR$(219) + " " + CHR$(219);
430 LOCATE ROW-3,COLUMN: PRINT " " + CHR$(219) + " " + CHR$(219);
440 LOCATE ROW-4,COLUMN: PRINT "  "  + CHR$(219);
450 RETURN
460 END
```

说明：

① 本例使屏幕上显示出一条尺蠖在地面上匍匐前进（图8—35）。

（a）　（b）　（c）　（d）　（a）

图8—35　尺蠖在匍匐前进

其中（a）、（b）、（c）、（d）四个图形是分别画在第1,2,3,0四个页面中的，通过轮流地显示不同的页面，即可得到快速动画的效果。

② 程序中使用SCREEN, , AP, VP选择工作页面（AP）和显示页面（VP），例如SCREEN, , 2,1（语句80）表示在显示1号页面的同时，在2号页面中画出新的图形。

2．直线运动

在图形显示模式下，动画对象是由象元所构成的，它在屏幕上的运动，其原理与字符动画一样，但由于对运动的控制比较精密，能够按照所要求的任意轨道进行运动，因而会产生更好的动画效果。

下面是一个很简单的例子。动画对象是一个点，它在一个方框中进行运动，运动的路线是45°的直线，当它碰到边框线后即改变其运动方向，如此循环往复不已（图8—36）。

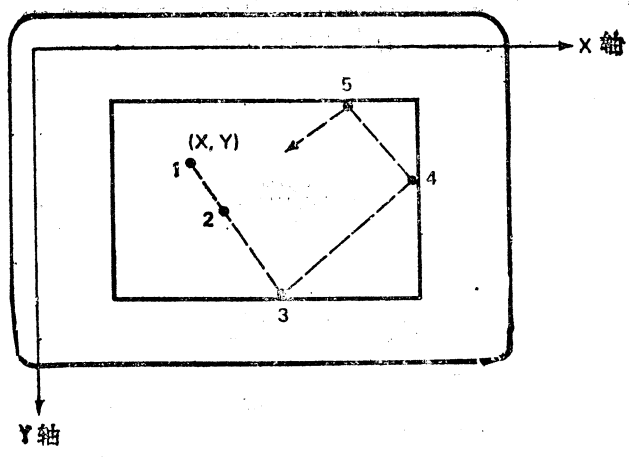


图 8—36　直线运动

例 8—24　质点在方框中弹跳

程序：

```
10 'PROGRAM      BOUNCING A POINT WITHIN A BOX.
20       'DX = DY = 1.
30  SCREEN 1: COLOR 3,0: CLS
40  INPUT "ENTER X VALUES FOR LEFT AND RIGHT WALL"; XLEFT,XRIGHT
50  IF XLEFT >= XRIGHT OR XLEFT < 0 OR XRIGHT > 319 THEN 220
60  INPUT "ENTER Y VALUES FOR TOP AND BOTTOM OF BOX"; YTOP,YBOTTOM
70  IF YTOP >= YBOTTOM OR YTOP < 0 OR YBOTTOM > 199 THEN 220
80  DX = 1: DY = 1
90  X = XLEFT + INT((XRIGHT - XLEFT) / 2)
100 Y = YTOP + INT((YBOTTOM - YTOP) / 2)      'START THE POINT IN MIDDLE OF BOX
110 CLS
120      '************************* DRAW BOX ******************************
130 LINE (XLEFT,YTOP) - (XRIGHT,YBOTTOM),2,B
140      '************************ BOUNCE POINT ***************************
150 PSET (X,Y),2
160 IF X-1=XLEFT OR X+1=XRIGHT THEN DX = -DX     'IF WE'RE ONE UNIT AWAY FROM
170 IF Y-1=YTOP OR Y+1=YBOTTOM THEN DY = -DY     'BOX SIDE, REVERSE DIRECTION
180 PRESET (X,Y)          'ERASE CURRENT POINT
190 X = X + DX: Y = Y + DY                'CALCULATE NEW POINT
200 GOTO 150
210      '*****************************************************************
220 PRINT "ERROR IN CHOICE OF BOX WALLS"
230 END
```

从程序中可以看出，如果改变 DX 和DY 的初值，则质点的运动速度将会发生变化。其值越大，则运动速度越快，其值越小，运动速度就越慢。如果在运动过程中，DX 或 DY 的绝对值也发生变化，则质点的运动就不再是匀速运动了。

当动画的对象每次移动的距离不止一个单位时（即|DX|，|DY|大于 1），在它弹跳即将碰到边框之前，程序中最好能计算出它的运动路径与边框的交点，并在紧靠边框线处画出该动画对象，然后再进行回弹，这样就会产生更好的真实感。

画面上有两个以上动画对象时，可以用下面的 POINT 函数来判断是否会发生碰撞：

POINT (X, Y)

其中，X，Y为屏幕坐标，函数的值等于位于（X，Y）处的象元的彩色码，如果 X，Y 不在

允许范围内，则函数值为 −1，否则取值为 0～3（中分辨率）或 0，1（高分辨率）。下面是使用 POINT 函数来测试动画对象是否发生碰撞的一个例子。

例 8—25　迷宫

程序：

```
10 'PROGRAM       GETTING THROUGH MAZE WITH THE SCREEN FUNCTION
20  SCREEN 1: COLOR 1,1: CLS
30      'MAKE MAZE
40  LINE (150,95) - (200,100),1,BF        'MAKE INTERIOR BOX
50  R = 1
60  UA = 5                'UA,RA,LA,DA ARE AMOUNTS TO DRAW UP,RIGHT,DOWN,LEFT
70  PSET(150,100)         'START BY THE CENTER
80  DRAW "C2"
90  FOR TIME = 1 TO 8
100     RA = UA + 45: DA = UA + 10: LA = UA + 55
110     DRAW "U=UA; R=RA; D=DA; L=LA;"
120     UA = UA + 20
130 NEXT
140     'DRAW BALL
150 X = 70: Y = 175
160 CIRCLE (70,175),R,1
170 A$ = INKEY$: IF A$ = "" THEN 170
180 B$ = A$
190 WHILE A$ <> "Q" AND A$ <> "q" AND ITHACA = 0        'QUIT OR INTERIOR?
200     CIRCLE (X,Y),R,0                  'ERASE CURRENT BALL
210     'IF THE COLOR OF THE NEXT POINT IS BACKGROUND OR MAZE INTERIOR BOX
         THEN ADVANCE TO POINT AND DRAW CIRCLE. OTHERWISE WE'RE ON WALL
220     IF RIGHT$(B$,1) = CHR$(80) THEN IF POINT(X,Y+1+R) < 2 THEN
            Y=Y+1:GOTO 260 ELSE BEEP:GOTO 270
230     IF RIGHT$(B$,1) = CHR$(72) THEN IF POINT(X,Y-1-R) < 2 THEN
            Y=Y-1:GOTO 260 ELSE BEEP:GOTO 270
240     IF RIGHT$(B$,1) = CHR$(77) THEN IF POINT(X+1+R,Y) < 2 THEN
            X=X+1:GOTO 260 ELSE BEEP:GOTO 270
250     IF RIGHT$(B$,1) = CHR$(75) THEN IF POINT(X-1-R,Y) < 2 THEN
            X=X-1:GOTO 260 ELSE BEEP:GOTO 270
260     IF POINT(X+R,Y) = 1 THEN ITHACA = 1
270     CIRCLE (X,Y),R,1                  'DRAW NEW BALL
280     IF POINT(X,Y) = 1 THEN ITHACA = 1
290     A$ = INKEY$
300     IF A$ <> "" THEN H = ASC(RIGHT$(A$,1))          'NEW, VALID KEY?
310     IF H = 80 OR H = 72 OR H = 77 OR H = 75 THEN B$ = A$:H = 0
320 WEND
330 IF ITHACA THEN PRINT "HOORAY!!"
340 END
```

说明：

① 本例是一个游戏程序。它首先在屏幕上显示一座迷宫和一个圆球（图 8—37），接着，操作者就可以用键盘来控制球的移动，直到球移到迷宫中心为止就算取得成功。

② 球的移动用四个光标键来进行，分别对应着向上、向下、向左或向右。如果球碰到墙则引起喇叭发出警告声。

③ 按下“Q”或“q”键，使游戏结束。

3．曲线运动

在大量的应用场合，常常需要动画对象进行某种曲线运动，例如炮弹的发射，物体的振动等等。下面给出两个例子以示一般。

例 8—26　球的弹跳运动

本例用来模拟一个圆球从一定高度掉到地面后不断弹跳的运动轨迹（图 8—38），这是

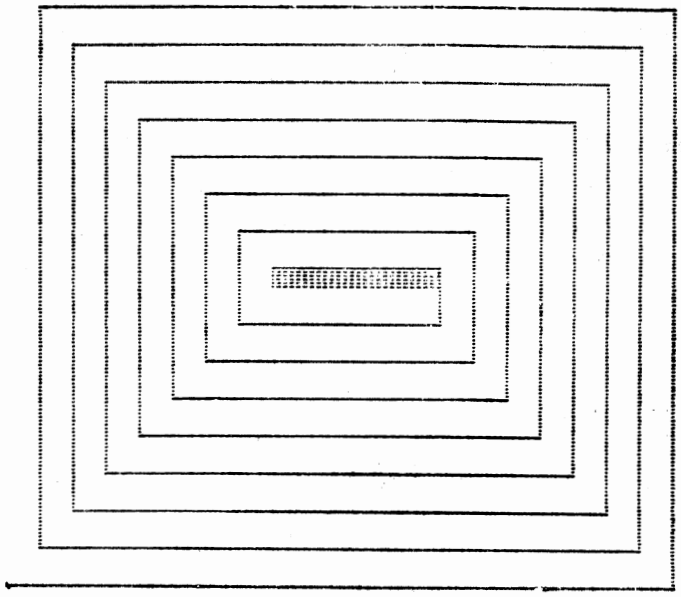


图8—37　迷宫运球

一种阻尼运动。其轨迹可以用公式

$$Y=H*SIN(X+D)*EXP(-K*X)$$

来表示，程序中选择衰减系数K=0.01。

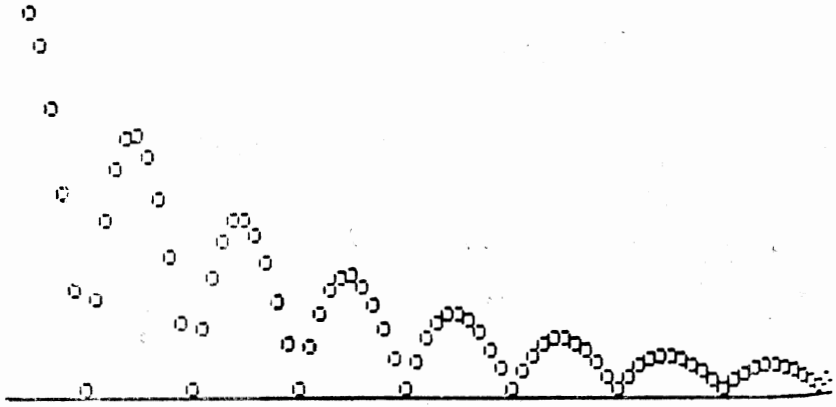


图8—38　球的落体运动

程序：

```
10 'PROGRAM       BOUNCING BALL DROPPED FROM SOME HEIGHT.
20       'PROGRAM SIMULATES MOVEMENT OF A BALL DROPPED FROM
30       'SOME HEIGHT
40  SCREEN 1: CLS
```

```
50  INPUT "BALL IS DROPPED FROM WHAT HEIGHT"; HEIGHT
60  W = 3.14159 / 40              'DISTANCE FROM BOUNCE TO BOUNCE IS 40
70  D = 90 * 3.14159 / 180        'DISPLACE BY 90 DEGREES (EXPRESSED AS RADIANS)
80  K = .01                       'K IS DAMPING FACTOR
90  CLS
100 LINE (0,199) - (319,199)              'DRAW GROUND
110     '****************** DROP BALL AND BOUNCE ******************
120 FOR XNEW = 0 TO 319-10 STEP 4          '4 EVENLY DIVIDES INTO 40
130     YNEW = HEIGHT * SIN(W * XNEW + D) * EXP(-K * XNEW)
140     YNEW = 199 - ABS(YNEW) - 3
150     CIRCLE (X,Y),2,0              'ERASE CURRENT POSITION
160     CIRCLE (XNEW + 10,YNEW),2     'DRAW NEW POSITION
170     X = XNEW + 10       'STORE CURRENT POSITION IN X AND Y
180     Y = YNEW
190 NEXT
200 END
```

例 8－27　抛物运动

物体作抛物运动时的轨迹，与它的初始速度S和发射角度A有关（图8—39）。在已

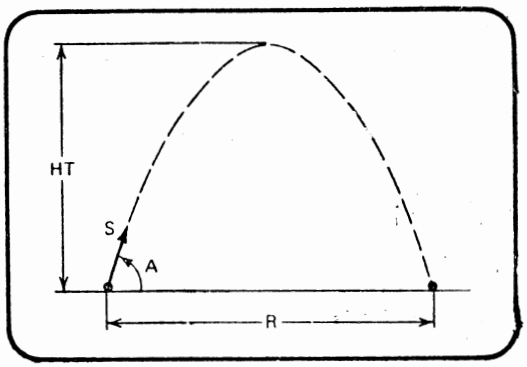


图8—39　抛物运动

知S和A的情况下，发射距离R和最大高度HT可按下式计算：

$$R=S*S*SIN(2*A)/G$$

$$HT=((S*SIN(A))\hat{}2)/(2*G)$$

其中G为重力加速度常数，它等于980。

在直角坐标系中，当X从0变化到R时，其对应的纵坐标值为：

$$Y=C1*X\hat{}2+C2*X+Y0$$

其中，Y0为起始纵坐标值，C1和C2是两个常数，它们由S和A所决定：

$$C1=G/(2*(S*COS(A))\hat{}2)$$

$$C2=-TAN(A)$$

程序：

```
10 'PROGRAM        MOVING ALONG A PARABOLIC CURVE
20  SCREEN 0: WIDTH 80: CLS
```

```
30  INPUT "ENTER COORDINATES OF START POSITION"; XSTART,YSTART
40  IF XSTART >= 0 AND XSTART <= 319 AND YSTART >= 0 AND YSTART <= 199 THEN 60
50  PRINT "RE-ENTER START POSITION": GOTO 30
60  INPUT "ENTER ANGLE (0 - 90)"; ANGLE
70  ANGLE = ANGLE * 3.14159 / 180        'CONVERT A TO RADIANS
80  INPUT "ENTER SPEED (100 - 600)"; SPEED
90  GRAVITY = 980
100 RANGE = SPEED * SPEED * SIN(2 * ANGLE) / GRAVITY
110                       'WILL WHOLE CURVE FIT ON SCREEN?
120 IF XSTART + RANGE <= 319 THEN 160
130                       'IF NOT, ENTER NEW VALUES
140 PRINT "RE-ENTER ANGLE AND SPEED":GOTO 60
150                       'HEIGHT IS HEIGHT OF CURVE
160 HEIGHT = ((SPEED * SIN(ANGLE)) ^ 2) / (2 * GRAVITY)
170                       'WILL WHOLE CURVE FIT ON SCREEN?
180 IF HEIGHT > 0 AND HEIGHT <= 199 THEN 220
190                       'IF NOT, ENTER NEW VALUES
200 PRINT "RE-ENTER ANGLE AND SPEED": GOTO 60
210                       'CALCULATE COEFFICIENTS OF EQUATION
220 C1 = GRAVITY / (2 * (SPEED * COS(ANGLE)) ^ 2)
230 C2 = - TAN(ANGLE)
240 SCREEN 1: CLS
250     '********************* MOVE BALL ALONG CURVE ************************
260 FOR X = 0 TO RANGE STEP 2
270     Y = C1 * X ^ 2 + C2 * X + YSTART
280     CIRCLE (X + XSTART,Y),3,3,,,.9199999
290     CIRCLE (X + XSTART,Y),3,0,,,.9199999
300 NEXT
310 END
```

4. 快速动画

在前面所介绍的一些例子中，如果动画对象不是简单的一个点或一个圆，则每次化费在擦去图形和重画图形上的时间就要增加。动画对象的图形越复杂，增加的时间就越多。如果遇到运动轨迹的计算又比较复杂时，则动画的速度就会慢到不能允许的程度，无法满足使用的要求。

为了解决图形显示模式下提高动画速度的问题，IBM PC 的高级 BASIC 语言提供了两条专用的图形语句——GET 和 PUT 语句。使用这两条语句，可以避免一条一条线段地擦去和重画每一个图形。这两条语句的格式及功能解释如下：

GET (X1,Y1)－(X2,Y2), ARR

其中，(X1,Y1) 和 (X2,Y2) 是一个矩形的两个对顶角的坐标，ARR 是一个数组。GET 语句的功能是把 (X1,Y1)－(X2,Y2) 所指出的矩形区域内的每一象元的彩色码存放到数组 ARR 中去。值得注意的是 ARR 必须是一个数字数组，其大小应能容纳下矩形中所有象元的彩色码。它的最小容量（以字节为单位）可以按下式进行计算：

MINSIZE=4+H * INT ((W * BITS+7) /8)

其中，H 为矩形的高度 (H=Y2+1−Y1)，W 为矩形的宽度 (W=X2+1−X1)，BITS 是每个象元所对应的二进位的数目（中分辨率时 BITS=2，高分辨率时 BITS=1），4 个附加的字节是用来存放 W 和 H 的数值的。作为例子，语句

GET (50, 50)－(69, 79), OBJECT%

中矩形的宽为 20 点，高为 30 点，中分辨率时矩形中所有象元的彩色码需两个二进位。所以

MINSIZE=4+30 * INT ((20 * 2) +7) /8) =154 字节

因为整型数组的每个元素由两个字节构成，因此OBJECT%数组至少必须包含77个元素。

GET 语句的作用是把矩形方框内的一个动画对象贮存起来，而与此对应的PUT语句则用来把贮存的动画对象在屏幕上指定位置处再显示出来。PUT 语句的格式及含义如下：

```
PUT (X, Y)  ARR, HOW
```

其中，ARR 是一个数组，它存放了一个矩形的动画对象的象元信息。PUT 语句把该动画对象在屏幕上（X，Y）处重新显示出来，（X，Y）表示矩形左上角的坐标。参数 HOW用来指出显示的方式，它共有五种选择（缺省时表示 XOR）：

- PSET　按动画对象的象元信息原样地在屏幕上显示出来。
- PRESET　把动画对象的象元信息取反后在屏幕上显示出来。
- XOR　把动画对象的象元信息与屏幕上的象元信息进行“按位加”后再显示出来。
- OR　把动画对象的象元信息与屏幕上的象元信息进行“或”运算后再显示出来。
- AND　把动画对象的象元信息与屏幕上的象元信息进行“与”运算后再显示出来。

	数组	屏幕原始彩色码			
	彩色码	0	1	2	3
PSET	0	0	0	0	0
	1	1	1	1	1
	2	2	2	2	2
	3	3	3	3	3
PRESET	0	3	3	3	3
	1	2	2	2	2
	2	1	1	1	1
	3	0	0	0	0
XOR	0	0	1	2	3
	1	1	0	3	2
	2	2	3	0	1
	3	3	2	1	0
OR	0	0	1	2	3
	1	1	1	3	3
	2	2	3	2	3
	3	3	3	3	3
AND	0	0	0	0	0
	1	0	1	0	1
	2	0	0	2	2
	3	0	1	2	3

上面的表是选择不同的显示方式时，数组中的彩色码与屏幕对应位置上的原始彩色码在各种不同组合的情况下，PUT 语句所产生的结果彩色码的一览表（中分辨率模式）。

例 8—28 和 8—29 是使用 GET 和 PUT 语句来实现快速动画的两个例子。

例 8—28　卡车运动

程序：

```
10  PROGRAM       MOVING TRUCK (in back of poles) WITH GET AND PUT STATEMENTS
20  DIM TRUCK%(47)
30  SCREEN 2: CLS
40  LINE (96,100) - (126,110),,BF          'DRAW TRUCK BODY
50  LINE (126,105) - (136,110),,BF
60  CIRCLE (101,110),4                     'DRAW WHEELS
70  CIRCLE (131,110),4
80  GET (96,100) - (136,114),TRUCK%        'STORE OBJECT SHAPE
90  CLS
100 LINE (0,65) - (639,65)                 'DRAW ROAD LINED WITH POLES
110 FOR X = 70 TO 580 STEP 80
120     LINE (X,45) - (X,65)
130 NEXT
140 FOR X = 0 TO 576 STEP 48               'ANIMATE TRUCK
150     PUT (X,50),TRUCK%,XOR              'XOR TRUCK IMAGE ONTO SCREEN
160     FOR DELAY = 1 TO 100: NEXT
170     PUT (X,50),TRUCK%,XOR              'XOR TRUCK IMAGE ONTO SCREEN
180 NEXT
190 GOTO 140
200 END
```

说明：

①本例是使用 GET 和 PUT 语句来实现动画的例子，动画对象是一部卡车，它在公路上飞驶（参见图 8—40）。显然，GET 和 PUT 语句对于有阴影的或着色的对象实现动画尤为合适。

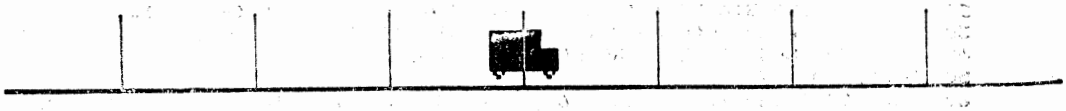


图 8—40　卡车在电线杆后面运动

② 语句 140～180 用来实现卡车的移动，其中 PUT 语句选用的显示方式为 XOR，因此当卡车与电线杆重迭时，电线杆会把卡车“档住”，产生了卡车在电线杆后面运动的效果。如果希望卡车能在电线杆前面运动，则可把程序中 150～170 三条语句用下面的四条语句来代替：

```
150 GET(X, 50) - (X + 40, 64), SAVEBACK%
160 PUT(X, 50), TRUCK%, OR
170 FOR DELAY = 1 TO 80 : NEXT
175 PUT(X, 50)  SAVEBACK%, PSET
```

当然，语句 20 中必须添加SAVEBACK%（47）的说明。

例 8—29　帆船入海

本例与前面一些例子的不同之处在于：动画对象在运动过程中本身的大小也跟着改变，因而能产生物体逐渐远去或逐渐靠近的效果。例中使用了 12 个数组来存放帆船的 12 帧大

小不同的图形，然后使用 PUT 语句连续地把它们在屏幕上不同位置处画出来，从而产生了一幅帆船入海的动画（图 8--41 ）。

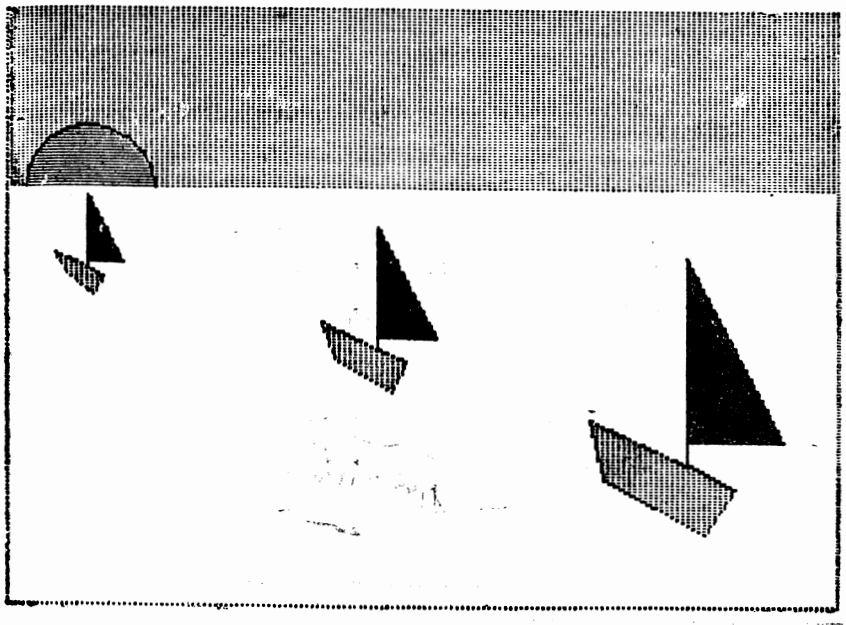


图 8—41　帆船入海

程序：

```
10 'PROGRAM        SAILING INTO THE SUNSET.
20       'SCALES SIALBOAT IN RELATION TO A FIXED POINT (-30,70).
30       'REPEATED SCALING MOVES THE BOAT FROM RIGHT TO LEFT AND
40       'SHRINKS IT AS IT RECEDES INTO THE DISTANCE.
50  DIM X(10), Y(10)
60  DIM BOAT1%(886), BOAT2%(716), BOAT3%(572), BOAT4%(485), BOAT5%(405),
        BOAT6%(310), BOAT7%(257), BOAT8%(209), BOAT9%(191), BOAT10%(154),
        BOAT11%(121), BOAT12%(95), BLANK%(886)
70  SCREEN 1: COLOR 1,1: CLS
80  FOR K = 1 TO 9: READ X(K),Y(K): NEXT          'READ DATA POINTS
90  GOSUB 410
100 CLS
110 LINE (0,0) - (319,199),1,B                    'DRAW BORDER
120 LINE (0,60) - (319,60),1                      'DRAW HORIZON
130 PAINT (160,10),1,1                            'PAINT IN SKY
140 CIRCLE (40,60),25,0,0,3.14159                 'DRAW SUN OUTLINE
150 PAINT (40,50),2,0                             'PAINT IN SUN
160 CIRCLE (40,60),25,3,0,3.14159                 'DRAW SUN OUTLINE IN WHITE
170 TIMER = 75
180       'SAIL BOAT INTO SUNSET
190 X = 247
200 GOSUB 350: PUT (X,Y),BOAT1%:  GOSUB 370
210 GOSUB 350: PUT (X,Y),BOAT2%:  GOSUB 370
220 GOSUB 350: PUT (X,Y),BOAT3%:  GOSUB 370
230 GOSUB 350: PUT (X,Y),BOAT4%:  GOSUB 370
240 GOSUB 350: PUT (X,Y),BOAT5%:  GOSUB 370
250 GOSUB 350: PUT (X,Y),BOAT6%:  GOSUB 370
260 GOSUB 350: PUT (X,Y),BOAT7%:  GOSUB 370
270 GOSUB 350: PUT (X,Y),BOAT8%:  GOSUB 370
280 GOSUB 350: PUT (X,Y),BOAT9%:  GOSUB 370
290 GOSUB 350: PUT (X,Y),BOAT10%: GOSUB 370
```

```
300 GOSUB 350: PUT (X,Y),BOAT11%: GOSUB 370
310 GOSUB 350: PUT (X,Y),BOAT12%
320 GOTO 820
330     '*************************************************************
340     'CALCULATE EACH X AND Y DISPLAY POSITION
350 X = X - 20: Y = .1 * X + 60: RETURN
360     'SLOW BOAT DOWN WITH TIMER AND DELAY, THEN ERASE
370 TIMER = TIMER + 50: FOR DELAY = 1 TO TIMER: NEXT
380 PUT (X,Y),BLANK%,PSET: RETURN
390 RETURN
400     '******************** SAIL INTO SUNSET ********************
410 GET (210,97) - (282,189),BLANK%     'GET AREA TO USE FOR ERASING
420 XFIXED = 210: YFIXED = 97
430 HSCALING = .9: VSCALING = .9
440     'CALCULATE CONSTANT PARTS OF SCALING EQUATIONS
450 XCONST = XFIXED * (1 - HSCALING): YCONST = YFIXED * (1 - VSCALING)
460 FOR TIME = 1 TO 12
470     FOR K = 1 TO 9          'SCALE EACH OF 9 POINTS
480         X(K) = INT(X(K) * HSCALING + XCONST + .5)
490         Y(K) = INT(Y(K) * VSCALING + YCONST + .5)
500     NEXT
510     CLS: GOSUB 680          'DRAW NEW POSITION
520             'GET INTO APPROPRIATE ARRAY
530     ON TIME GOTO 540,550,560,570,580,590,600,610,620,630,640,650
540     GET (210,97) - (282,189),BOAT1%: GOTO 660
550     GET (210,97) - (275,180),BOAT2%: GOTO 660
560     GET (210,97) - (269,172),BOAT3%: GOTO 660
570     GET (210,97) - (263,165),BOAT4%: GOTO 660
580     GET (210,97) - (258,158),BOAT5%: GOTO 660
590     GET (210,97) - (253,152),BOAT6%: GOTO 660
600     GET (210,97) - (249,147),BOAT7%: GOTO 660
610     GET (210,97) - (245,142),BOAT8%: GOTO 660
620     GET (210,97) - (242,138),BOAT9%: GOTO 660
630     GET (210,97) - (239,134),BOAT10%: GOTO 660
640     GET (210,97) - (236,130),BOAT11%: GOTO 660
650     GET (210,97) - (233,127),BOAT12%: GOTO 660
660 NEXT
670 RETURN
680 '************************ DRAW SAILBOAT ***************************
690 FOR K = 1 TO 8
700     LINE (X(K),Y(K)) - (X(K+1),Y(K+1)),3
710 NEXT
720 XINSAIL = (X(1) + X(2) + X(3)) / 3
730 YINSAIL = (Y(1) + Y(2) + Y(3)) / 3
740 PAINT (XINSAIL,YINSAIL),3,3
750 XINBOAT = (X(4) + X(5) + X(6) + X(7) + X(8) + X(9))
760 YINBOAT = (Y(4) + Y(5) + Y(6) + Y(7) + Y(8) + Y(9))
770 PAINT (XINBOAT,YINBOAT),1,3
780 RETURN
790 '*************************************************************
800 DATA 250,165,290,165,250,97,250,173,270,183
810 DATA 258,199,215,179,210,157,250,174
820 IF INKEY$ = "" THEN 820
830 END
```

说明：

① 12帧大小不同的帆船图形是使用比例变换来产生的。比例变换的参考点是（－30，70），比例因子是0.9。子程序410—670用来完成此项任务。

② 子程序350用来计算帆船运动的轨迹，它使帆船逐渐地驶向远处日出的地方。子程序370用作时间控制，它使得画面上帆船的位置离人越来越远时其速度也逐渐减慢，因而有很好的真实感。

5、复合运动与背景运动

有许多应用场合，动画对象在运动过程中，本身的一部分或全部也都要同时进行活动。例如，火车行进时车轮不断旋转，人跑动时手脚的摆动等，这就是所谓“复合运动”。

复合运动需要使用多个数组，每个数组用来存放一帧图形，它们虽然代表着同一个动画对象，但形状各异，以达到复合运动的目的。下面通过一个复合运动的例子，说明处理此类问题的一些基本技巧。

例 8—30　跑步运动员

程序：

```
10 'PROGRAM       RUNNER
70  DIM X1(15), Y1(15), X2(15), Y2(15)
80  SCREEN 1: CLS
90  FOR K = 1 TO 13                          'READ POSITION #1
100     READ X1(K),Y1(K)
110 NEXT
120 FOR K = 1 TO 12                          'READ POSITION #2
130     READ X2(K),Y2(K)
140 NEXT
150 XDISP = 0                                'XDISP IS DISPLACEMENT ADROSS SCREEN
160 IF XDISP+X1(12) > 319 THEN 520           'WOULD POSITION #1 STILL BE ON SCREEN?
170 DRAWCOLOR = 1: GOSUB 250                      'DRAW POSITION #1
180 DRAWCOLOR = 0: GOSUB 250                      'ERASE POSITION #1
190 XDISP = XDISP + 15                            'MOVE OVER 15 UNITS
200 IF XDISP+X2(11) > 319 THEN 520           'WOULD POSITION #2 STILL BE ON SCREEN?
210 DRAWCOLOR = 1: GOSUB 360                      'DRAW POSITION #2
220 DRAWCOLOR = 0: GOSUB 360                      'ERASE POSITION #2
230 XDISP = XDISP + 20                            'MOVE OVER 20 UNITS
240 GOTO 170
250 '########################## POSITION #1 ##########################
260 FOR K = 1 TO 3
270     LINE (XDISP+X1(K),Y1(K)) - (XDISP+X1(K+1),Y1(K+1)),DRAWCOLOR
280 NEXT
290 LINE (XDISP+X1(5),Y1(5)) - (XDISP+X1(6),Y1(6)),DRAWCOLOR
300 LINE (XDISP+X1(6),Y1(6)) - (XDISP+X1(7),Y1(7)),DRAWCOLOR
310 FOR K = 8 TO 11
320     LINE (XDISP+X1(K),Y1(K)) - (XDISP+X1(K+1),Y1(K+1)),DRAWCOLOR
330 NEXT
340 CIRCLE (XDISP+X1(13),Y1(13)),10,DRAWCOLOR
350 RETURN
360 '########################## POSITION #2 ##########################
370 FOR K = 1 TO 2
380     LINE (XDISP+X2(K),Y2(K)) - (XDISP+X2(K+1),Y2(K+1)),DRAWCOLOR
390 NEXT
400 LINE (XDISP+X2(4),Y2(4)) - (XDISP+X2(5),Y2(5)),DRAWCOLOR
410 LINE (XDISP+X2(5),Y2(5)) - (XDISP+X2(6),Y2(6)),DRAWCOLOR
420 FOR K = 7 TO 10
430     LINE (XDISP+X2(K),Y2(K)) - (XDISP+X2(K+1),Y2(K+1)),DRAWCOLOR
440 NEXT
450 CIRCLE (XDISP+X2(12),Y2(12)),10,DRAWCOLOR
460 RETURN
470 '##########################################################################
480 DATA 14,150,20,133,15,120,20,93,5,145,25,133,15,120
490 DATA 20,115,10,110,19,92,20,108,30,113,20,83
500 DATA 2,132,25,136,40,93,43,150,50,130,30,120
510 DATA 30,111,22,103,38,95,43,110,58,104,40,83
520 END
```

说明：

①本例在屏幕上显示出一个跑步运动员的动画（图 8—42），实际上，这只不过是两帧不同图形在屏幕上交替地显示出来而已。这两帧图形的坐标数据分别在数组X1，Y1 和X2，Y2中。

②由于动画对象的图形比较简单，它完全由线条构成，既无着色也无阴影等，因此本例没有使用 GET 和 PUT 语句，这样速度显得反而更快一些。

如果动画对象相当复杂而它们的背景图形比较简单，或者动画对象的图形比较大不便于

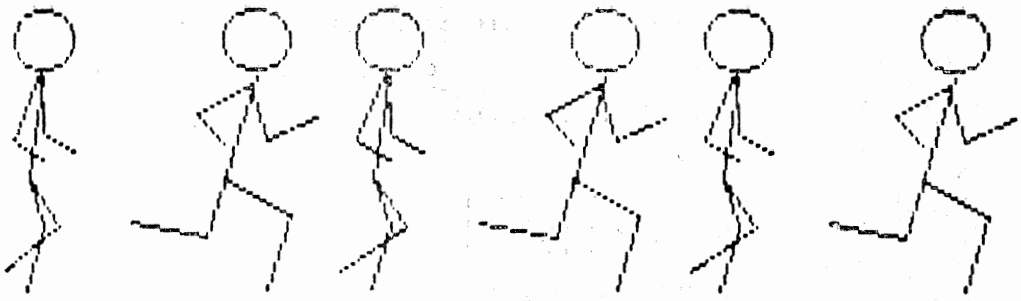


图 8—42　跑步运动的动画

在屏幕上大幅度运动时，可以采用使背景作反方向运动而对象保持不动、或者两者同时相对运动的办法来达到动画的目的，这种方法叫做“背景运动”。下例是通过背景运动模拟火车行进的动画。

例 3—31　用背景运动来模拟火车在铁轨上行进

本例在屏幕上产生一幅火车正在铁轨上行进的动画（图 8—43）。动画的效果可这样来达到：程序一方面使火车的车轮不断地旋转，另一方面又使铁轨从左向右移动，这样看起来火车就在飞速地从右向左行驶了。由于程序本身并不复杂，故不再作进一步的解释。

程序：

```
10 'PROGRAM        TRAIN WITH MOVING WHEEL BAR AND TRACKS.
60  SCREEN 1: COLOR 1,1: CLS
70  YADJUST = 5/6                          'YA IS RESOLUTION ADJUSTMENT
80      'DRAW TRAIN
90  READ X1,Y1: PSET (X1,Y1)
100 FOR K = 1 TO 25
110     READ X2,Y2: LINE - (X2,Y2)
120 NEXT
130 READ XL, XR, YT, YB: LINE (XL,YT) - (XR,YB),,BF      'WINDOW
140 FOR K = 1 TO 6                                       'ADD DETAIL LINES
150     READ X1, Y1, X2, Y2: LINE (X1,Y1) - (X2,Y2)
160 NEXT
170 FOR K = 1 TO 4                                       'ADD WHEELS
180     READ XC, YC, R
190     CIRCLE (XC,YC),R        'FINAL XC AND YC VALUES ARE FOR REAR WHEEL
200 NEXT
210     '******************** ROTATE WHEEL BAR, MOVE TRACKS ********************
220 RADIUS = 15                            'RADIUS OF WHEEL BAR'S ROTATION
230 DA = 50  * 3.14159 / 180
240 A$ = ""
250 WHILE A$ = ""
260     FOR ANGLE = DA TO 6.28318 STEP DA
270         XBAR = XC + RADIUS * SIN(ANGLE)
280         YBAR = YC + RADIUS * COS(ANGLE) * YADJUST
290         LINE (XBAR,YBAR) - (XBAR-90,YBAR)    'DRAW NEW BAR POSITION
300         FOR X = XSTART TO 319 STEP 35        'PLOT TRACKS 35 UNITS APART
310             PSET (X,152)
320         NEXT
330         FOR X = XSTART TO 319 STEP 35        'ERASE TRACKS
340             PRESET (X,152)
350         NEXT
360         XSTART = XSTART + 7          'NEXT POINT SET WILL BE 7 PIXELS OVER
370         IF XSTART >= 30 THEN XSTART = 0      'POINTS KEEP COMING FROM LEFT
380         LINE (XBAR,YBAR) - (XBAR-90,YBAR),0          'ERASE CURRENT POSITION
390     NEXT
400     A$ = INKEY$
```

```
410 WEND
420 '*************************************************************************
430     'OUTLINE
440 DATA 270,130,290,130,290,50,300,50,300,30,220,30,220,70,200,70
450 DATA 200,50,170,50,170,70,90,70,90,50,100,40,60,40,70,50,70,70
460 DATA 60,70,50,80,50,110,60,120,40,120,20,150,50,150,50,130,90,130
470     'WINDOW
480 DATA 230,280,40,70
490     'DETAIL LINES
500 DATA 220,70,220,100,220,100,65,100
510 DATA 65,100,45,125,130,130,140,130
520 DATA 0,154,319,154,180,130,230,130
530     'WHEELS
540 DATA 65,140,10,110,130,20,160,130,20,250,130,20
550 END
```

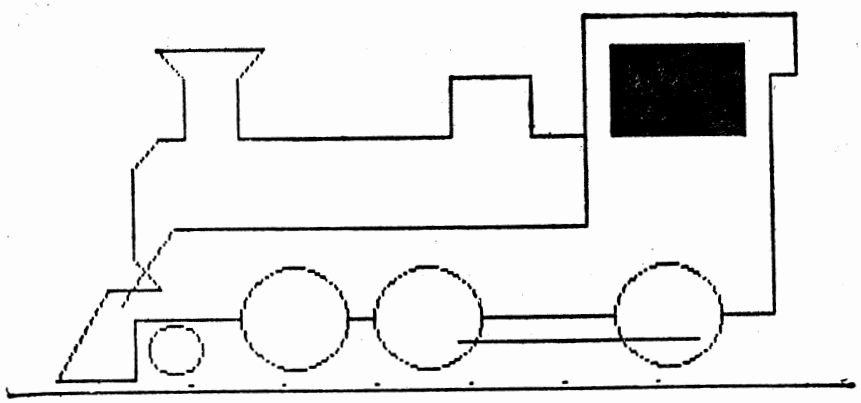


图8—43　火车在铁轨上行进

第五节　三维图形简介

前面所介绍的关于图形显示及其操作的一些原理和技术，都是就二维的情况而言的。也就是说，屏幕上所显示的图形，只能反映出物体的宽度和高度，而没有反映出它们的厚度（深度）。尽管二维图形在许多应用场合已能满足用户的要求，但具有立体感的三维图形由于真实性好，因而更受用户的欢迎。

三维图形的显示技术比二维图形要复杂得多，它们涉及到一些比较复杂的算法，对处理机的运算速度也有较高的要求。本节将通过几个实例，对三维图形显示的基本技术和问题作一简单介绍。

1．空间坐标系统和透视变换

为了描述具有立体形状的物体的位置和尺寸大小，必须使用空间坐标系统。图8—44是本节所使用的一种空间直角坐标系统。Z轴指向屏幕背后，物体上任何一点的空间位置均用三个分量（X,Y,Z）来表示，其中Z表示物体的厚度（深度），当Z大于零时，表示该点在屏幕的背后。

由于荧光屏本身只是一个二维的平面，而三维图形的显示，就是要在屏幕上表示出物体空

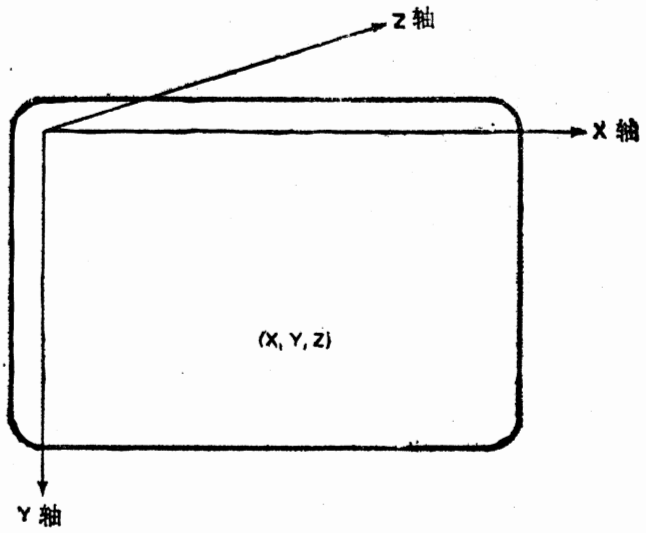


图 8—44　三维直角坐标系统

间位置的全部信息，即不仅是物体的长和宽，还应该表示出它们的厚度，为此，必须采用投影技术。

常用的投影技术有两种：透视投影和正交投影。透视的基本原理如图 8—45 所示。图中的一排房子，离观察者越近，它显得越大，离观察者越远，它显得越小，直到在远处的一个所谓“会聚点”处完全消失。用透视原理画出的图形具有很好的立体感。

使用透视原理在屏幕上显示物体的图形时，需要把物体每一点的空间位置投影到屏幕上，投影的方法如图 8—46 所示。其中观察者的位置（叫做“视点”）在空间坐标系统中为

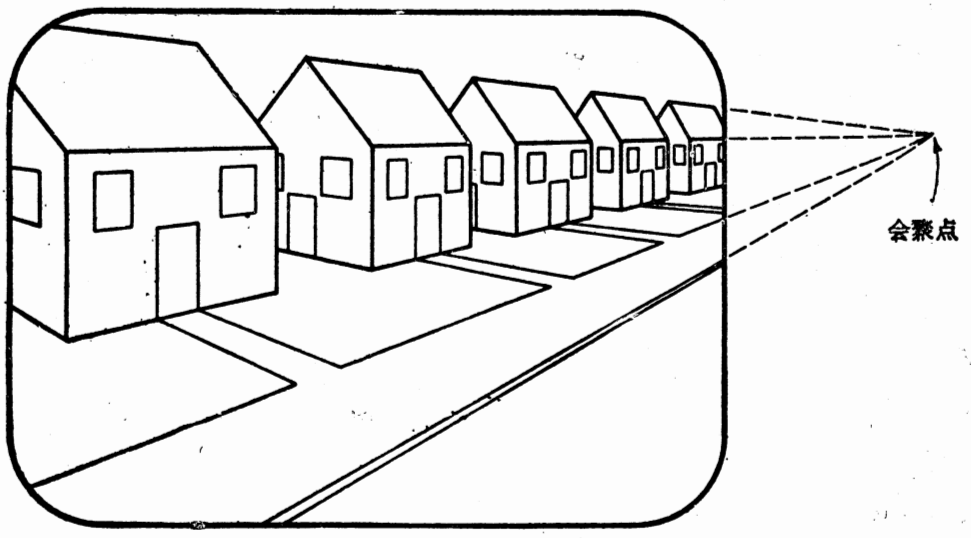


图 8—45　透视原理

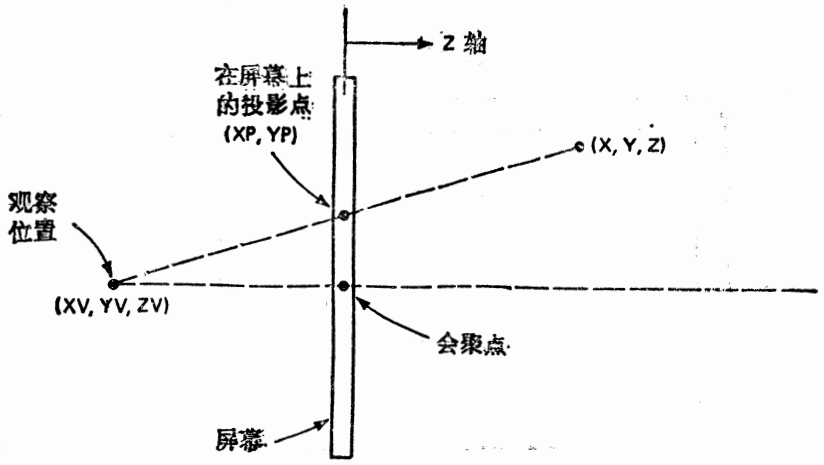


图 8—46　透视投影

(XV, YV, ZV)，物体某一点在屏幕上的投影就是该点与视点的连线与屏幕的交点(XP, YP)。不难推出，空间任意一点(X, Y, Z)在屏幕上的投影的坐标为：

$$\begin{cases} XP = XV + (XV - X) * ZV/(Z - ZV) \\ YP = YV + (YV - Y) * ZV/(Z - ZV) \end{cases}$$

利用上述公式，就可以把物体的空间位置按透视原理计算出它们在屏幕上的坐标位置，这就叫做“透视变换”。显然，透视变换与视点的选择有关，视点离屏幕越近，投影得到的图形就越小，当 ZV=0 时，图形就退化成为一个点，这就是会聚点；当视点位置离屏幕越远时，投影所得到的图形就越接近于原来物体的尺寸，当距离无穷远时，XP=X，YP=Y，这就是“正交投影”。

下面是使用透视投影产生三维图形的两个例子。

例 8—32　公路的透视投影图

程序：

```
10 'PROGRAM        DRAWING TELEPHONE POLES IN PERSPECTIVE.
60  DIM X1(8), Y1(8), X2(8), Y2(8)
70  SCREEN 0: WIDTH 80: CLS
80      'ESTABLISH VIEWING POINT
90  PRINT "X AND Y MUST BE ON SCREEN, Z MUST BE NEGATIVE"
100 INPUT "X,Y,Z OF VIEWING POINT"; XV,YV,ZV
110 IF XV<0 OR XV>319 OR YV<0 OR YV>199 OR ZV>=0 THEN  90
120 SCREEN 1: CLS
130 FOR K = 1 TO 6
140     READ X1, Y1, X2, Y2
150     FOR Z = 0 TO 5000 STEP 500
160         'CALCULATE CONSTANT PART OF EQUATION
170         P = -ZV / (Z - ZV)
180         'CALCULATE POINTS FOR LINE AT THIS Z VALUE
190         XA = XV + (X1 - XV) * P
200         YA = YV + (Y1 - YV) * P
210         XB = XV + (X2 - XV) * P
220         YB = YV + (Y2 - YV) * P
230         LINE (XA,YA) - (XB,YB)
240     NEXT
250 NEXT
260     'DRAW IN ROAD EDGES AND CENTER
```

```
270 FOR K = 1 TO 3
280     READ XA,YA
290     XB = XV + (XA - XV) * P
300     YB = YV + (YA - YV) * P
310     LINE (XA,YA) - (XB,YB)
320 NEXT
330     '**************************************
340 DATA 50,45,50,155
350 DATA 40,60,60,60
360 DATA 40,50,60,50
370 DATA 260,45,260,155
380 DATA 250,60,270,60
390 DATA 250,50,270,50
400 DATA 70,155,160,155,240,155
410 IF INKEY$ = "" THEN 410
420 END
```

说明：

① 本例在屏幕上显示出两旁各有一排电线杆的一条公路的透视投影图（图8—47），

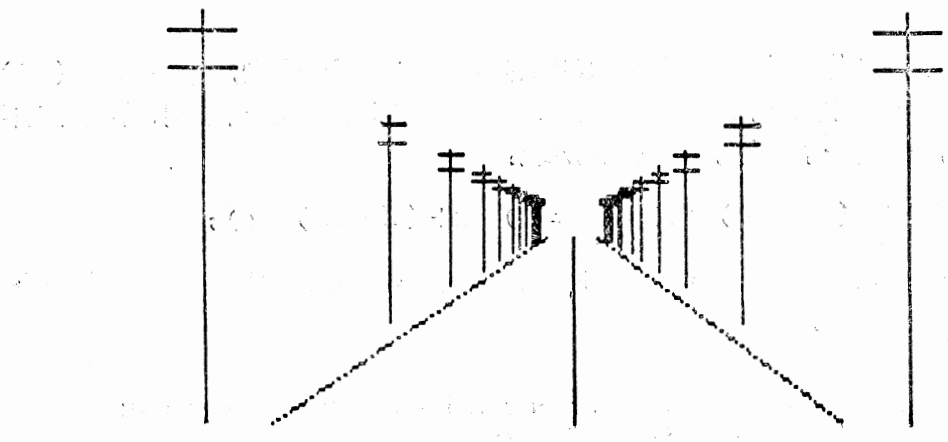


图8—47　公路的透视图

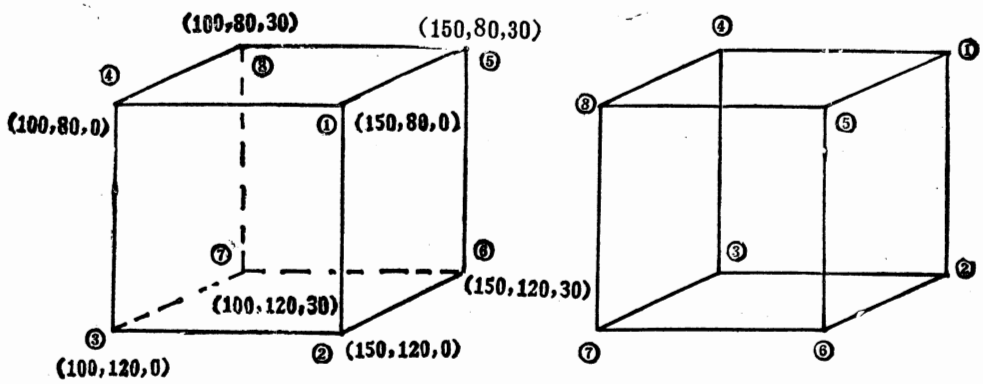
其中11对电线杆的横坐标、纵坐标均相同，只是离视点的距离远近不同，它们的Z坐标值分别为0，500，1000，1500，…，5000。

② 视点的位置是在程序运行时输入的，XV和YV必须在屏幕范围之内，ZV必须为负值。读者不妨选择几个不同的视点，比较一下输出的图形有什么变化。

例8—33　立方体的显示

本例用来在屏幕上画出一个正立方体的透视图，该立方体的八个顶点的编号及其坐标如图8—48（a）所示。当选择了视点的位置以后，程序将逐一地计算出这八个顶点在屏幕上的投影位置，然后把它们互连起来就得到该立方体的透视图（图8—48（b））。

但是，这时所产生的的图形由于难以区分出哪一些面在前，哪一些面在后，因而缺乏逼真感。为此，程序中应当判别出哪些线段（或面）被前面位置上的面所遮住，它们不必再在图中画出来（或者用虚线画出），这样一些被遮住的线条或面叫做“隐藏线”或“隐藏面”。例如图8—48（b）中的线段⑤—①、⑤—⑥和⑤—⑧都是隐藏线，⑤—⑥—②—①、⑤—⑧—④—①和⑤—⑥—⑦—⑧都是隐藏面。



（a） 立方体的空间坐标　　（b） 未消除隐藏线的透视图

图8—48　立方体的显示

由于一个三维物体投影到两维平面后线段之间的相交情况十分复杂，所以隐藏线（面）消除在三维图形显示中是一个比较复杂的问题。本例的立方体图形是一个特例，程序中先按下面的公式计算出每一个顶点与视点之间的距离：

$$D=SQR((X-XV)^2+(Y-YV)^2+(Z-ZV)^2);$$

然后把它们加以比较，利用立方体的对称性，就能判断出哪些面是可见的，应该画出；哪些面是隐藏面，应该消除。

程序：

```
10  PROGRAM        DEFINING AN OBJECT IN 3 DIMENSIONS USING PERSPECTIVE
70  DIM X(8), Y(8), Z(8), XP(8), YP(8), D(8)
80  SCREEN 1: CLS
90      '********************** READ DATA POINTS **********************
100 READ N
110 FOR K = 1 TO N
120       READ X(K), Y(K), Z(K)
130     IF X(K)<0 OR X(K)>319 OR Y(K)<0 OR Y(K)>199 THEN 760
140 NEXT
150 FOR K = 1 TO N/2 - 1
160     LINE (X(K),Y(K)) - (X(K+1),Y(K+1))
170 NEXT
180 LINE (X(N/2),Y(N/2)) - (X(1),Y(1))
190     '****************** ESTABLISH VIEWPOINT & DRAW ***************
200 LOCATE 1,1: PRINT "ENTER 0,0,0 TO QUIT"
210 PRINT "Z COORDINATE MUST BE NEGATIVE"
220 INPUT "COORDINATES OF VIEW POSITION"; XV,YV,ZV
230 IF XV = 0 AND YV = 0 AND ZV = 0 THEN 760
240 IF ZV >= 0 THEN 200
250     'PUT POINTS IN PERSPECTIVE
260 FOR K = 1 TO N
270     XP(K) = XV + (X(K) - XV) * -ZV / (Z(K) - ZV)
280     YP(K) = YV + (Y(K) - YV) * -ZV / (Z(K) - ZV)
290 NEXT
300     'FIND DISTANCES OF POINTS FROM VIEWPOINT
310 FOR K = 1 TO N
320     D(K) = SQR((X(K)-XV)^2 + (Y(K)-YV)^2 + (Z(K)-ZV)^2)
330 NEXT
340 GOSUB 360           'DRAW
350 GOTO 190
360 '************************************ DRAW ROUTINE ********************************
370 CLS
```

```
380 IF D(1) = D(5) THEN 490
390 IF D(1) > D(5) THEN 450
400 FOR K = 1 TO 3                                    'DRAW FACE CONTAINING POINT 6
410     LINE (XP(K),YP(K)) - (XP(K+1),YP(K+1))
420 NEXT
430 LINE (XP(4),YP(4)) - (XP(1),YP(1))
440 GOTO 490
450 FOR K = 5 TO 7                                    'DRAW FACE CONTAINING POINT 5
460     LINE (XP(K),YP(K)) - (XP(K+1),YP(K+1))
470 NEXT
480 LINE (XP(8),YP(8)) - (XP(5),YP(5))
490 IF D(1) = D(4) THEN 600
500 IF D(1) > D(4) THEN 560
510 LINE (XP(1),YP(1)) - (XP(2),YP(2))                'DRAW FACE CONTAINING POINT 1
520 LINE (XP(2),YP(2)) - (XP(6),YP(6))
530 LINE (XP(6),YP(6)) - (XP(5),YP(5))
540 LINE (XP(5),YP(5)) - (XP(1),YP(1))
550 GOTO 600
560 LINE (XP(4),YP(4)) - (XP(3),YP(3))                'DRAW FACE CONTAINING POINT 4
570 LINE (XP(3),YP(3)) - (XP(7),YP(7))
580 LINE (XP(7),YP(7)) - (XP(8),YP(8))
590 LINE (XP(8),YP(8)) - (XP(4),YP(4))
600 IF D(1) = D(2) THEN 710
610 IF D(1) > D(2) THEN 670
620 LINE (XP(1),YP(1)) - (XP(4),YP(4))                'DRAW FACE CONTAINING POINT 1
630 LINE (XP(4),YP(4)) - (XP(8),YP(8))
640 LINE (XP(8),YP(8)) - (XP(5),YP(5))
650 LINE (XP(5),YP(5)) - (XP(1),YP(1))
660 GOTO 710
670 LINE (XP(2),YP(2)) - (XP(3),YP(3))                'DRAW FACE CONTAINING POINT 2
680 LINE (XP(3),YP(3)) - (XP(7),YP(7))
690 LINE (XP(7),YP(7)) - (XP(6),YP(6))
700 LINE (XP(6),YP(6)) - (XP(2),YP(2))
710 RETURN
720     '*****************************************************************
730 DATA 8
740 DATA 150,80,0,150,120,0,100,120,0,100,80,0
750 DATA 150,80,30,150,120,30,100,120,30,100,80,30
760 END
```

说明：

① 程序首先在屏幕上显示出该立方体的正交投影图，然后由用户输入视点的位置坐标。

② 由于被显示物体是正立方体，因此隐藏面的判断比较简单。例如，若视点与点①、②、③的距离都相等，且小于到其它顶点的距离，则仅有①—②—③—④是可见面，其余五面均不可见；若视点与点①和②的距离相等，且小于与点④之间的距离，则①—②—③—④和①—②—⑥—⑤是两个可见面，其余的均为隐藏面；若视点与点③的距离最近，则有三个可见面和三个隐藏面，它们的透视图如图 8－49 所示。

```
ENTER COORDINATES OF VIEWING POSITION
Z COORDINATE MUST BE NEGATIVE
ENTER 0,0,0 TO QUIT
```

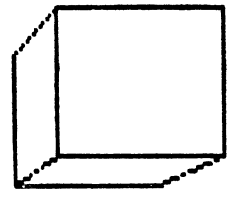


图 8—49　消除隐藏面后立方体的透视图

2．曲面的显示与隐藏线的消除

曲面是三维图形的一种，它在许多科学和工程计算中非常有用。下面是用于显示二元函数

$$Y=H*SIN(F*SQR(X*X+Z*Z))$$

图形的一个例子，其中H和F均为常数，自变量X的变化范围为 −135～325，步长为1，它用来选择该曲面沿着X方向的某一段。Z的变化范围为 −100～130，步长为5。图8—50是该二元函数在对应范围内的一段曲面图形。

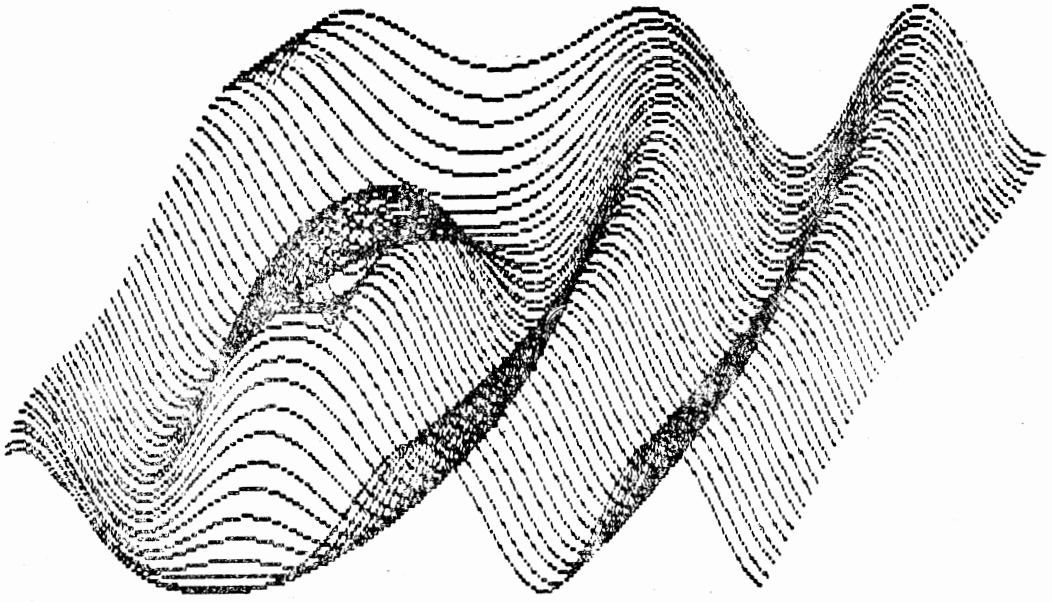


图8—50　未消除隐藏线的曲面

不难看出，一个曲面的图形实际上是由一组曲线所形成的，每条曲线都与一个固定的Z值相对应。

例8—34　曲面的图形

程序：

```
10  PROGRAM        THREE-DIMENSIONAL PLOT.
70  SCREEN 2: CLS
80  XCENTER = 214: YCENTER = 100
90  HEIGHT = 20
100 FREQUENCY = .05
110 FIRSTX = -135
120 FOR Z = -100 TO 130 STEP 5
130     ZSQUARED = Z * Z
140     XSCREENADJUST = XCENTER + .75 * Z
150     YSCREENADJUST = 199 - (YCENTER + .5 * Z)
160     FOR X = FIRSTX TO 325
170         Y = INT(HEIGHT * SIN(FREQUENCY * SQR(X * X + ZSQUARED)) + .5)
```

```
180          XSCREEN = XSCREENADJUST + X
190          YSCREEN = YSCREENADJUST - Y
200          IF X > FIRSTX THEN LINE - (XSCREEN,YSCREEN) ELSE
                PSET (XSCREEN,YSCREEN)
210       NEXT
220 NEXT
230 END
```

说明：

为了使曲面的立体感更好些，图形上的每一点并不直接按照方程所计算的（X，Y）值画出，而是加上一定的调整值后再画线（或画点）。由于调整量本身是Z的函数（语句140和150），因此每画出一条曲线之后，下一条曲线将稍微向屏幕右上角偏移一点再行画出。这就好比用户是站在屏幕左前方位置上来观察这个曲面一样，图形的变化和层次就更加清晰可见。

为了使曲面的真实感更好，曲面显示中同样也存在着隐藏线消除的问题。消除曲面中的隐藏线可以采用如下方法：首先，必须按照从前到后的顺序来画出组成曲面的每一条曲线。其次，在画曲线的过程中，对每一个X值都保存两个数据，即在已画出的所有曲线上与该X值对应的所有Y的最小值和最大值。然后，在画下一条曲线时，每计算出一个点（X,Y），若Y值落在最小值与最大值之间，则表示它已被前面的曲线所遮住，应该消去而不必画出。若Y值比最小值小或者比最大值大，则表示该点为可见点，应该画出。同时还要修改最小值或最大值。

下面是例8—34的改进，它采用上述的算法消除了图8—50中的隐藏线，因而得到更加形象、更能清晰地反映出函数性质的曲面图形（图8—51）。

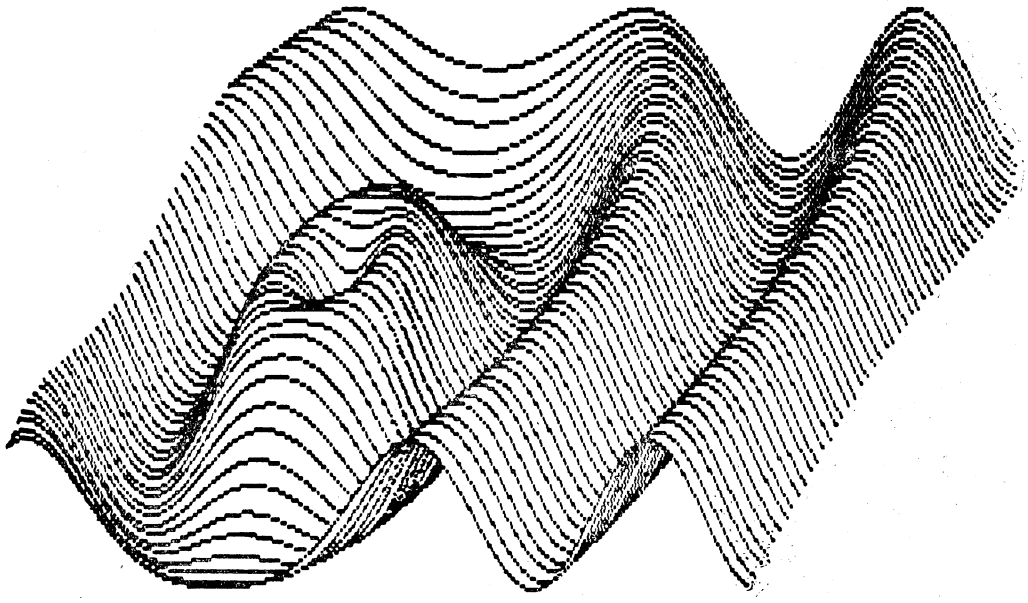


图8—51　消除隐藏线后曲面的图形

例8—35　消除曲面中的隐藏线

程序：

```
 10 'PROGRAM       VISIBLE LINE PLOT OF A THREE-DIMENSIONAL GRAPH.
190 SCREEN 2: CLS
200 DIM SMALLEST(639), BIGGEST(639)
210 XCENTER = 214: YCENTER = 100
220 HEIGHT = 20
230 FREQUENCY = .05
240 FIRSTX = -135
250 LASTPOINT = 1
260       'INITIALIZE ARRAYS
270 FOR ELEMENT = 1 TO 639
280     BIGGEST(ELEMENT) = 0: SMALLEST(ELEMENT) = 1000
290 NEXT
300 FOR Z = -100 TO 130 STEP 5
310     ZSQUARED = Z * Z
320     XSCREENADJUST = XCENTER + .75 * Z
330     YSCREENADJUST = 199 - (YCENTER + .5 * Z)
340     FOR X = FIRSTX TO 325 STEP 1
350         Y = INT(HEIGHT * SIN(FREQUENCY * SQR(X * X + ZSQUARED)) + .5)
360         XSCREEN = XSCREENADJUST + X
370         YSCREEN = YSCREENADJUST - Y
380         'IS THIS POINT WITHIN THE BOUNDS OF WHAT'S ALREADY DRAWN?
390         IF YSCREEN >= SMALLEST(XSCREEN) AND YSCREEN <= BIGGEST(XSCREEN)
                THEN LASTPOINT = 0: GOTO 450
400         IF LASTPOINT = 0 THEN CHANGE = 1
410         LASTPOINT = 1
420         IF YSCREEN < SMALLEST(XSCREEN) THEN SMALLEST(XSCREEN) = YSCREEN:
                IF BIGGEST(XSCREEN)=0 THEN BIGGEST(XSCREEN) = YSCREEN: GOTO 440
430         IF YSCREEN > BIGGEST(XSCREEN) THEN BIGGEST(XSCREEN) = YSCREEN
440         IF X = FIRSTX OR CHANGE = 1 THEN PSET(XSCREEN,YSCREEN): CHANGE = 0
                ELSE LINE - (XSCREEN,YSCREEN)
450     NEXT
460 NEXT
470 END
```

例 8—35 介绍的曲面显示程序具有一定的通用性。无论是根据二元函数计算所得的数据，还是有关人口密度、海拔高度等统计或测量得到的数据，都可以使用这里介绍的方法来画出它们所对应的三维图形。

3. 三维变换

作为本章的结束，这里再扼要介绍一下三维变换。与二维的情况一样，三种最基本的三维变换是平移变换、比例变换和旋转变换。掌握了这些基本变换的原理之后，实现三维图形的交互式作图和动画等就不很困难了。

三维空间的平移变换最简单。设物体上某点的坐标为（X，Y，Z），它在X方向、Y方向、Z方向移动的距离分别为H、V、D，则经过平移后该点的坐标为：

$$\begin{cases} XT = X + H \\ YT = Y + V \\ ZT = Z + D \end{cases}$$

比例变换与二维的情况类似，但对物体的放大或缩小除了可以在X轴方向和Y轴方向进行之外，还允许在Z轴方向上进行。假设比例变换的参考点的坐标是（XF，YF，ZF），则三维的比例变换公式为：

$$\begin{cases} XS = X * HS + XF * (1 - HS) \\ YS = Y * VS + YF * (1 - VS) \\ ZS = Z * DS + ZF * (1 - DS) \end{cases}$$

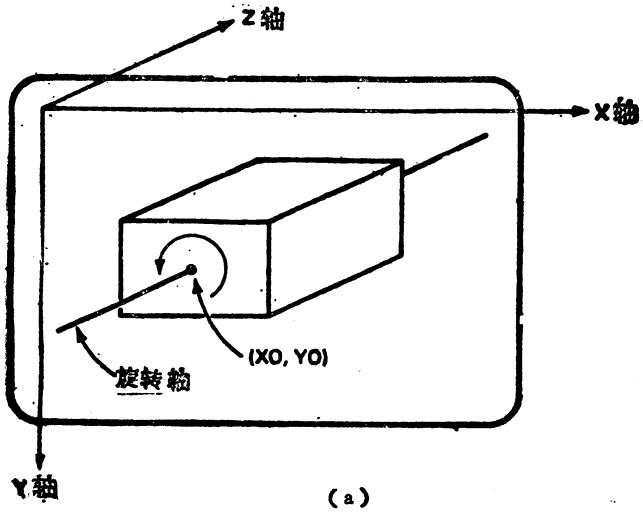
其中 HS、VS、和 DS 分别为在三个不同方向上的比例因子。

三维情况下的旋转变换要比二维复杂一些。首先考虑三种特殊情况：即物体绕与Z 方向平行的轴、绕与 Y 方向平行的轴、以及绕与 X 方向平行的轴旋转的情况。参看图 8 —52(a)，当旋转轴平行于 Z 轴时，物体上各点的 Z 坐标值均保均不变，而 X 坐标与 Y 坐标值的变化与二维情况完全一样。因此，物体上任何一点在旋转了角度 A 之后，新的坐标位置为：

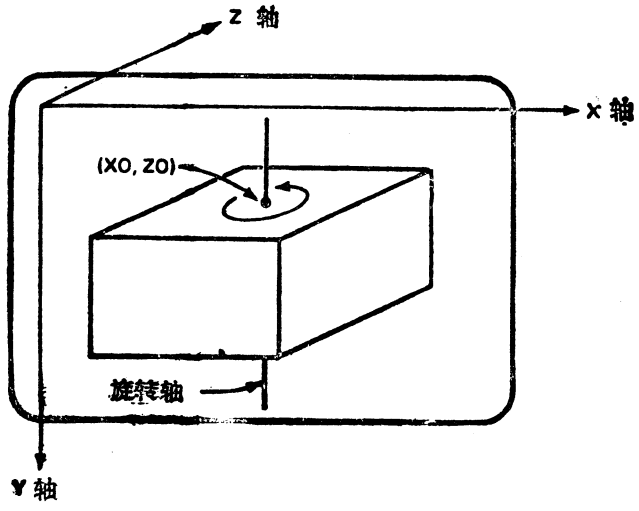
$$\begin{cases} XR = XO + (X - XO) * COS(A) + (Y - YO) * SIN(A) \\ YR = YO + (Y - YO) * COS(A) - (X - XO) * SIN(A) \\ ZR = Z \end{cases}$$

其中，（XO, YO）为旋转轴与屏幕交点的坐标，A 为旋转角，逆时针方向为正，单位为弧度。

同样，对于图 8—52（b）和（c）所示的两种特殊情况，旋转变换的公式分别为：



（a）



（b）

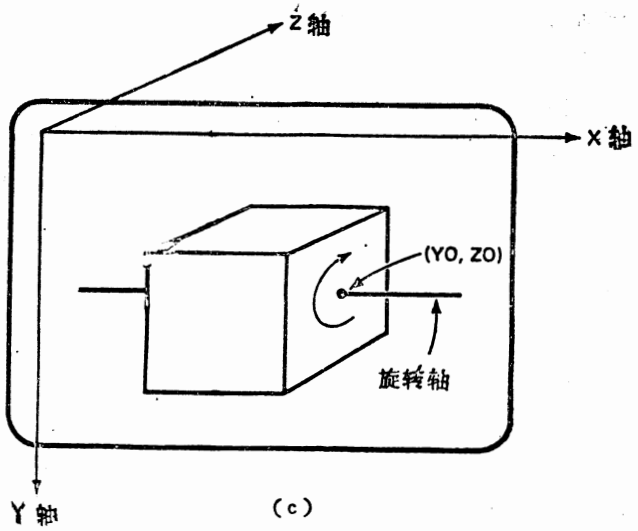


图8—52　三种特殊的旋转变换

$$\begin{cases} XR=XO+(X-XO)*COS(A)-(Z-ZO)*SIN(A) \\ YR=Y \\ ZR=ZO+(Z-ZO)*COS(A)+(X-XO)*SIN(A) \end{cases}$$

$$\begin{cases} XR=X \\ YR=YO+(Y-YO)*COS(A)+(Z-ZO)*SIN(A) \\ ZR=ZO+(Z-ZO)*COS(A)-(Y-YO)*SIN(A) \end{cases}$$

有了这三种基本的旋转变换之后，绕任意方向的轴进行旋转变换就不困难了。下面是一个三维旋转变换的实例，读者可以从中体会出任意角度的旋转是如何分解成三种基本旋转变换的。

例 8—36　三维旋转变换

程序：

```
10 'PROGRAM        ROTATION IN 3 DIMENSIONS
80  SCREEN 1: CLS
90  DIM XR(29), YR(29), ZR(29)
100 READ XO, YO, ZO
110 FOR K = 1 TO 29
120       READ XR(K), YR(K), ZR(K)
130       IF XR(K)<0 OR XR(K)>319 OR YR(K)<0 OR YR(K)>199 THEN 1020
140 NEXT
150 GOSUB 460
160 LOCATE 1,1: PRINT "ENTER Q TO QUIT"
170 INPUT "ROTATE AROUND WHAT AXIS"; R$
180 IF R$ = "Q" THEN 1020
190 INPUT "HOW MANY DEGREES"; A
200 A = A * 3.14159 / 180          'CONVERT A TO RADIANS
210 COSA = COS(A): SINA = SIN(A)
220     '***************** CALCULATE NEW POINTS **********************
230 IF R$ = "X" THEN 260
240 IF R$ = "Y" THEN 320
250 IF R$ = "Z" THEN 380
260 FOR K = 1 TO 29                'AROUND X AXIS
270       YS = YR(K)
280       YR(K) = INT(YO + (YR(K) - YO) * COSA + (ZR(K) - ZO) * SINA + .5)
290       ZR(K) = INT(ZO + (ZR(K) - ZO) * COSA - (YS - YO) * SINA + .5)
```

```
300 NEXT
310 GOTO 430
320 FOR K = 1 TO 29                    'AROUND Y AXIS
330     XS = XR(K)          'SAVE XR(K) FOR USE IN ZR CALCULATION
340     XR(K) = INT(XO + (XR(K) - XO) * COSA - (ZR(K) - ZO) * SINA + .5)
350     ZR(K) = INT(ZO + (ZR(K) - ZO) * COSA + (XS - XO) * SINA + .5)
360 NEXT
370 GOTO 430
380 FOR K = 1 TO 29                    'AROUND Z AXIS
390     XS = XR(K)          'SAVE XR(K) FOR USE IN YR CALCULATION
400     XR(K) = INT(XO + (XR(K) - XO) * COSA + (YR(K) - YO) * SINA + .5)
410     YR(K) = INT(YO + (YR(K) - YO) * COSA - (XS - XO) * SINA + .5)
420 NEXT
430 GOSUB 450
440 GOTO 160
450     '****************** DRAW ONLY VISIBLE FACES ******************
460 CLS
470 IF ZR(1) = ZR(5) THEN 630                  'NEITHER SIDE IS VISIBLE
480 IF ZR(1) > ZR(5) THEN 560
490 FOR K = 1 TO 3                             'DRAW FACE CONTAINING 1 SPOT
500     LINE (XR(K),YR(K)) - (XR(K+1),YR(K+1))
510 NEXT
520 LINE (XR(4),YR(4)) - (XR(1),YR(1))
530 CIRCLE (XR(9),YR(9)),1
540 GOTO 630
550     '
560 FOR K = 5 TO 7                             'DRAW FACE CONTAINING 6 SPOTS
570     LINE (XR(K),YR(K)) - (XR(K+1),YR(K+1))
580 NEXT
590 LINE (XR(8),YR(8)) - (XR(5),YR(5))
600 FOR K = 24 TO 29
610     CIRCLE (XR(K),YR(K)),1
620 NEXT
630 IF ZR(1) = ZR(4) THEN 770                  'NEITHER SIDE IS VISIBLE
640 IF ZR(1) > ZR(4) THEN 720
650 LINE (XR(1),YR(1)) - (XR(2),YR(2))         'DRAW FACE CONTAINING 4 SPOTS
660 LINE - (XR(6),YR(6)): LINE - (XR(5),YR(5)): LINE -(XR(1),YR(1))
670 FOR K = 15 TO 18
680     CIRCLE (XR(K), YR(K)),1
690 NEXT
700 GOTO 770
710     '
720 LINE (XR(4),YR(4)) - (XR(3),YR(3))         'DRAW FACE CONTAINING 3 SPOTS
730 LINE - (XR(7),YR(7)): LINE - (XR(8),YR(8)): LINE - (XR(4),YR(4))
740 FOR K = 12 TO 14
750     CIRCLE (XR(K),YR(K)),1
760 NEXT
770 IF ZR(1) = ZR(2) THEN 910                  'NEITHER SIDE IS VISIBLE
780 IF ZR(1) > ZR(2) THEN 860
790 LINE (XR(1),YR(1)) - (XR(4),YR(4))         'DRAW FACE CONTAINING 2 SPOTS
800 LINE - (XR(8),YR(8)): LINE - (XR(5),YR(5)): LINE - (XR(1),YR(1))
810 FOR K = 10 TO 11
820     CIRCLE (XR(K),YR(K)),1
830 NEXT
840 GOTO 910
850     '
860 LINE (XR(2),YR(2)) - (XR(3),YR(3))         'DRAW FACE CONTAINING 5 SPOTS
870 LINE - (XR(7),YR(7)): LINE - (XR(6),YR(6)): LINE - (XR(2),YR(2))
880 FOR K = 19 TO 23
890     CIRCLE (XR(K),YR(K)),1
900 NEXT
910 RETURN
920     '*****************************************************************
930 DATA 140,80,124
940 DATA 164,56,100,164,104,100,116,104,100,116,56,100
950 DATA 164,56,148,164,104,148,116,104,148,116,56,148
960 DATA 140,80,100
970 DATA 128,56,136,152,56,112
980 DATA 116,68,136,116,80,124,116,92,112
990 DATA 164,68,112,164,68,136,164,92,112,164,92,136
1000 DATA 128,104,112,128,104,136,140,104,124,152,104,112,152,104,136
1010 DATA 128,68,148,140,68,148,152,68,148,128,92,148,140,92,148,152,92,148
1020 END
```

说明：

本例为在屏幕上显示出一个骰子的正交投影图。骰子在空间的初始位置如图8—53（a）所示。用户可以输入骰子的旋转方向及旋转角度，经过旋转变换的处理之后，程序将显示出骰子的新的正交投影。图8—53（b）是骰子绕与X方向平行的轴（旋转轴通过骰子中心，下同）旋转155度后的图形，（c）是再绕与Z方向平行的轴旋转28度后的图形，（d）是绕与Y方向平行的轴旋转45度后所得到的图形。消除隐藏面的原理与例8—33类似，这里不再赘述。

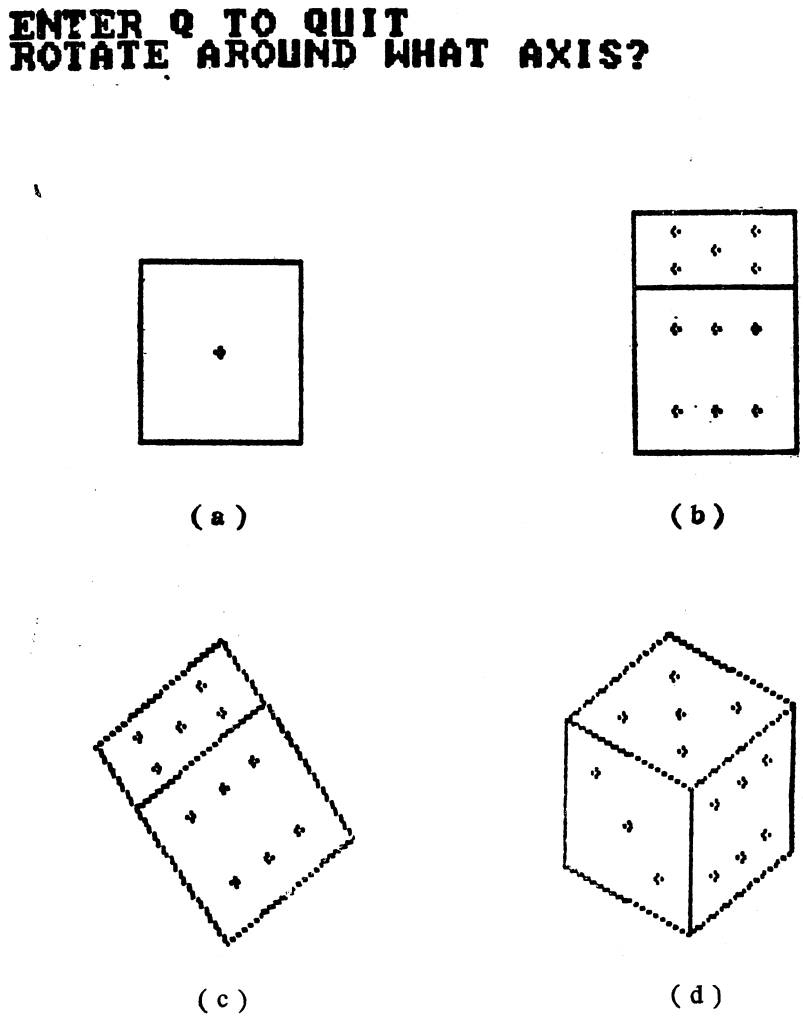


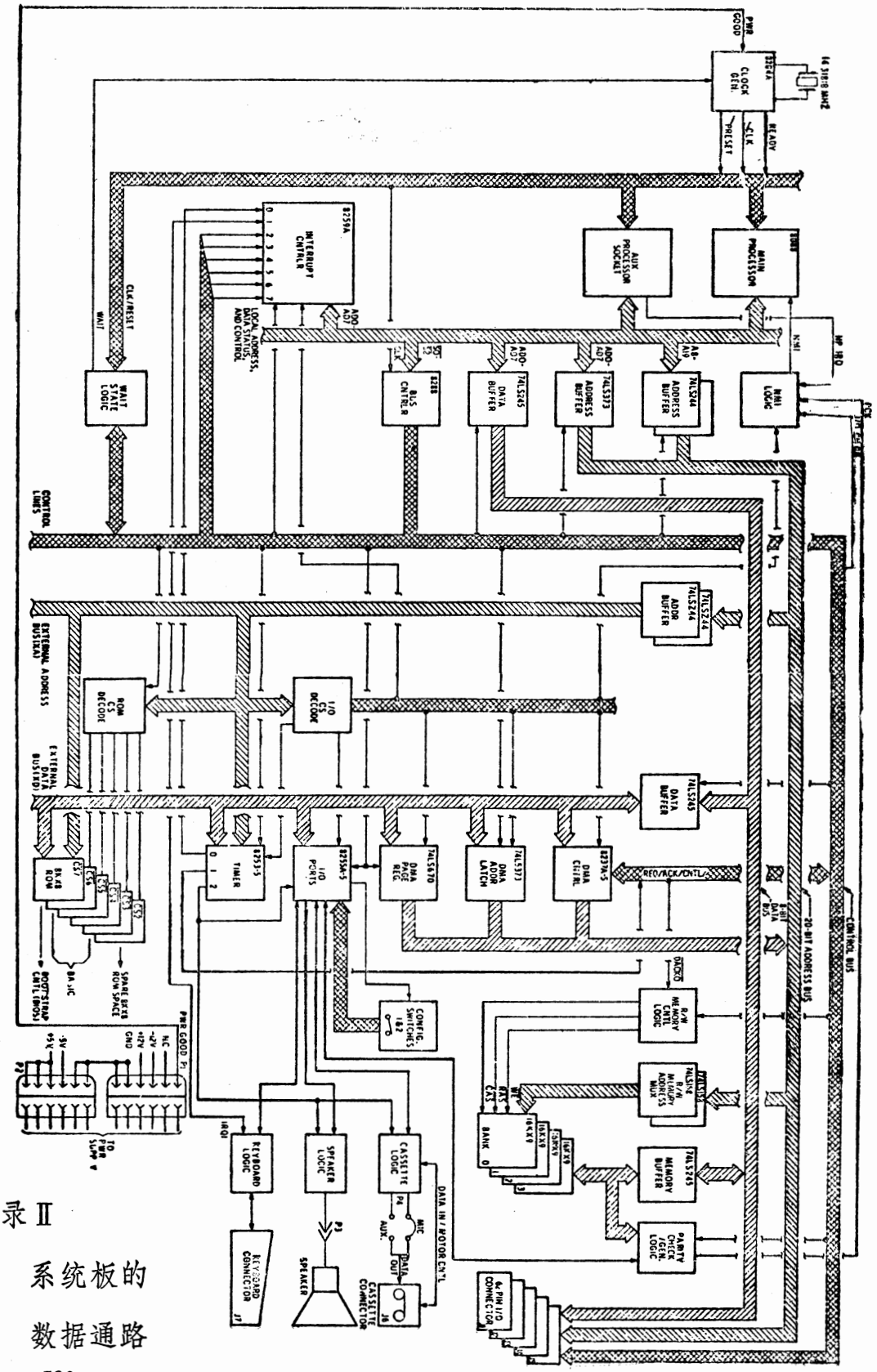
图8—53　三维旋转变换

附 录

附录 I　　CRT 显示输出码

DECIMAL VALUE	HEXA DECIMAL VALUE	0	16	32	48	64	80	96	112
		0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	▲	BLANK (SPACE)	0	@	P	‘	p
1	1	☺	▼	!	1	A	Q	a	q
2	2	☻	↕	"	2	B	R	b	r
3	3	♥	‼	#	3	C	S	c	s
4	4	♦	¶	$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↨	’	7	G	W	g	w
8	8	◘	↑	(	8	H	X	h	x
9	9	○	↓	)	9	I	Y	i	y
10	A	◙	→	*	:	J	Z	j	z
11	B	♂	←	+	;	K	[	k	{
12	C	♀	∟	,	<	L	\	l	|
13	D	♪	↔	-	=	M	]	m	}
14	E	♫	▲	.	>	N	^	n	~
15	F	☼	▼	/	?	O	_	o	△

DECIMAL VALUE	HEXA DECIMAL VALUE	128	144	160	176	192	208	224	240
		8	9	A	B	C	D	E	F
0	0	Ç	É	á	⋯	└	┴	∝	≡
1	1	ü	æ	í	⋯	┬	├	ß	±
2	2	é	Æ	ó	⋯	─	┼	Γ	≥
3	3	â	ô	ú	▯	├	╞	π	≤
4	4	ä	ö	ñ	▯	┌	╟	Σ	⌠
5	5	à	ò	Ñ	▯	┐	╚	σ	⌡
6	6	å	û	ª	▯	╔	╩	μ	÷
7	7	ç	ù	º	▯	╩	╦	τ	≈
8	8	ê	ÿ	¿	▯	╦	╠	Φ	°
9	9	ë	Ö	⌐	▯	╠	═	Θ	∙
10	A	è	Ü	¬	▯	═	╬	Ω	·
11	B	ï	¢	½	▯	╬	╧	δ	√
12	C	î	£	¼	▯	╧	╨	∞	n
13	D	ì	¥	¡	▯	╨	╤	ϕ	2
14	E	Ä	₧	«	▯	╤	╥	∈	■
15	F	Å	ƒ	»	▯	╥	╙	∩	BLANK FF



附录Ⅱ

系统板的

数据通路

附 录 Ⅲ　键盘输入码

1. ASCII码0～127的输入

键名 高位 / 低位		0	16	32	48	64	80	96	112
		0	1	2	3	4	5	6	7
0	0	Ctrl-2	Ctrl-P	空格	0	@	P	‘	p
1	1	Ctrl-A	Ctrl-Q	!	1	A	Q	a	q
2	2	Ctrl-B	Ctrl-R	"	2	B	R	b	r
3	3	Ctrl-C	Ctrl-S	#	3	C	S	c	s
4	4	Ctrl-D	Ctrl-T	$	4	D	T	d	t
5	5	Ctrl-E	Ctrl-U	%	5	E	U	e	u
6	6	Ctrl-F	Ctrl-V	&	6	F	V	f	v
7	7	Ctrl-G	Ctrl-W	'	7	G	W	g	w
8	8	BS	Ctrl-X	(	8	H	X	h	x
9	9	→|	Ctrl-Y	)	9	I	Y	i	y
10	A	Ctrl-J	Ctrl- Z	*	:	J	Z	j	z
11	B	Ctrl-K	Esc	+	;	K	[	k	{
12	C	Ctrl-L	Ctrl-\	,	<	L	\	l	|
13	D	←|	Ctrl-]	−	=	M	]	m	}
14	E	Ctrl-N	Ctrl-6	.	>	N	^	n	～
15	F	Ctrl-O	Ctrl-−	/	?	O	_	o	Ctrl-←

2. ASCII 码 128～255 的输入

ASCII码128～255的输入方法是按下〈Alt〉键的同时，再在右端小键盘上输入相应的十进制代码。例如要输入字符“β”，它的ASCII码是225，则输入Alt-225即可。

3. 扩充码的输入

对于不能用标准ASCII码表示的特殊键或组合键，IBM PC使用扩充码表示。IBM PC的扩充码由两个代码组成，第1个代码为00，第2个代码根据按键而定。常用的扩充码如下：

第2码	对应的按键
15	←
59～68	‹F1›～‹F10›
71	‹Home›
72	↑
73	‹PgUp›
75	←
77	→
79	‹End›
80	↓
81	‹PgDn›
82	‹Ins›
83	‹Del›
84～93	Shift-F1～Shift-F10
94～103	Ctrl-F1～Ctrl-F10
104～113	Alt-F1～Alt-F10
114	Ctrl-PrtSc
117	Ctrl-End
118	Ctrl-PgDn
119	Ctrl-Home
132	Ctrl-PgUp

附录Ⅳ　MS-DOS（2.00）常用操作命令

1. 磁盘操作

命令名称	命　令　格　式	类型	功　　能
CHKDSK	CHKDSK [‹盘符›] [‹文件名›][/F][/V]	外部	检查磁盘状态
DISKCOMP	DISKCOMP [‹盘符›] [‹盘符›][/1][/8]	外部	全盘比较
DISKCOPY	DISKCOPY [‹盘符›] [‹盘符›][/1]	外部	全盘复制
FDISK	FDISK	外部	硬盘分区
FORMAT	FORMAT [‹盘符›] [/S][/1][/8][/V][/B]	外部	初始化磁盘

2. 目录操作

命令名称	命　令　格　式	类型	功　　能
DIR	DIR [‹路径名›][/P] [/W]	内部	显示文件目录
RENAME	REN[AME] ‹路径名› ‹文件名›	内部	更换文件名

MKDIR	MD ‹目录路径名›	内部	建立一个子目录
CHDIR	CD [‹目录路径名›]	内部	改变或显示当前目录
RMDIR	RD ‹目录路径名›	内部	删除一个子目录
TREE	TREE [‹盘符›][/F]	外部	显示所有目录路径
PATH	PATH [‹目录路径名›] {;‹目录路径名›}	内部	改变外部命令搜索路径
VOL	VOL [‹盘符›]	内部	显示指定盘的卷名

3．文件操作

命令名称	命令格式	类型	功能
COPY	COPY [/A][/B] [‹路径名›] [/A][/B] [‹路径名›][/A][/B][/V]	内部	复制文件
SYS	SYS ‹盘符›	外部	复制系统
COMP	COMP [‹路径名›] [‹路径名›]	外部	比较文件
ERASE	DEL [‹路径名›]	内部	删除文件
TYPE	TYPE [‹路径名›]	内部	显示文件内容
PRINT	PRINT {‹文件引用名› [/T][/C][/P]}	外部	假脱机打印文件
VERIFY	VERIFY [ON|OFF]	内部	设置校验方式
BACKUP	BACKUP [‹路径名›] ‹盘符›[/S][/M][/A] [D:mm-dd-yy]	外部	复制硬盘文件的备份
RESTORE	RESTORE ‹盘符› [‹路径名›][/S][/P]	外部	把软盘上的文件复制到硬盘上
EXE2BIN	EXE2BIN [‹路径名›] [‹路径名›]	外部	把.EXE文件变换成.COM文件

4．批处理

命令名称	命令格式	类型	功能
(Batch)	‹文件名› [‹参数›]	内部	执行批处理文件
ECHO	ECHO [ON|OFF‹串›]	内部	设置屏幕显示状态

（续表 4）

FOR	FOR %%V IN(‹文件集›) DO ‹命令›	内部	循环子命令
GOTO	GOTO ‹标号›	内部	转移子命令
IF	IF [NOT] ‹条件›‹语句›	内部	条件子命令
SHIFT	SHIFT	内部	改变形参和实参对应关系
PAUSE	PAUSE [‹字符串›]	内部	暂停子命令
REM	REM [‹字符串›]	内部	注释子命令

5. I/O 命令

命令名称	命令格式	类型	功能
FIND	FIND[/V][/C][/N]‹串› {‹路径名›}	外部	检索过滤处理
MORE	MORE	外部	分页显示过滤处理
SORT	SORT [/R][/+n]	外部	排序过滤处理
ASSIGN	ASSIGN [x=y[...]]	外部	驱动器指派
CTTY	CTTY ‹设备名›	内部	控制台转让
GRAPHICS	GRAPHICS	外部	产生图形显示的硬拷贝
MODE	MODE LPT#:[n] [,[m][,P]] MODE [n],m[,T] MODE COMn:…	外部	设置打印机和显示器及通讯控制器的工作模式

6. 其它命令

命令名称	命令格式	类型	功能
BREAK	BREAK [ON|OFF]	内部	设置Ctrl-Break检查状态
CLS	CLS	内部	清除屏幕
DATE	DATE [mm-dd-yy]	内部	输入日期
RECOVER	RECOVER [‹路径名›]	外部	修复文件

（续表6）

TIME	TIME [hh:mm:ss.xx]	内部	输入时间
VER	VER	内部	显示MS-DOS的版本号
PROMPT	PROMPT [‹字符串›]	外部	设置新的MS-DOS的提示符
SET	SET [‹名›=[‹参数›]]	内部	设置命令处理程序环境

附录V　IBM PC BASIC的命令、语句和函数

1. 命令

命　令	作　用	命　令	作　用
AUTO	自动编行号	LOAD	装入程序文件
BLOAD	装入代码文件	MERGE	并入程序文件
BSAVE	保存代码文件	NAME..AS	文件更名
CLEAR	清除程序变量	NEW	清除当前程序
CONT	继续执行程序	RENUM	重编行号
DELETE	删除程序行	RESET	初始化盘片
EDIT	程序行编辑	RUN	执行程序
FILES	列出文件目录	SAVE	保存程序文件
KILL	删除文件	SYSTEM	返回MS-DOS
LIST	显示程序清单	TRON	跟踪接通
LLIST	打印程序清单	TROFF	跟踪断开

2. 语句

(1) 非I/O语句

语　句	作　用	语　句	作　用
CALL	调用代码程序	DIM	数组说明
CHAIN	链接程序	END	程序结束
COM(n) ON	允许捕俘通讯中断	ERASE	删除数组
COM(n) OFF	禁止捕俘通讯中断	ERROR n	模拟错误号n
COM(n) STOP	停止捕俘通讯中断	FOR..TO..STEP	步长循环语句
COMMON	公共变量语句	GOSUB	转子语句
DATE$=X$	设置日期	GOTO	转移语句
DEF FN	自定义函数	IF..THEN..ELSE	条件语句
DEF‹类型›	定义缺省变量类型	KEY ON	显示功能键清单
DEF SEG	定义段地址	KEY OFF	停止显示功能键清单
DEF USRn	定义代码程序n的始址	KEY LIST	列出功能键

语　句	作　用
KEY n,x$	定义功能键
KEY(n) ON	允许捕俘功能键中断
KEY(n) OFF	禁止捕俘功能键中断
KEY(n) STOP	停止捕俘功能键中断
LET	赋值语句
MID$(V$,n,m)=Y$	替代部分字符串
MOTOR	接通磁带机电机
NEXT	关闭步长循环语句
ON COM(n) GOSUB	设置通讯陷井
ON ERROR GOTO	设置出错陷井
ON n GOSUB	情况转子语句
ON n GOTO	情况分枝语句
ON KEY(n) GOSUB	设置功能键陷井
ON PEN COSUB	设置光笔陷井
ON STRIG(n) GOSUB	设置操纵杆陷井
OPTION BASE n	指定数组下标的最小初值
PEN ON	允许捕俘光笔中断
PEN OFF	禁止捕俘光笔中断
PEN STOP	停止捕俘光笔中断
POKE	写内存贮器
RANDOMIZE	初始化随机数序列
REM	注释语句
RESTORE	初始化DATA指示器
RESUME	从出错陷井续元中返回
RETURN	从子程序返回
STOP	停止执行程序
STRIG ON	允许使用操纵杆按钮
STRIG OFF	禁止使用操纵杆按钮
STRIG(n) ON	允许捕俘操纵杆中断
STRIG(n) OFF	禁止捕俘操纵杆中断
STRIG(n) STOP	停止捕俘操纵杆中断
SWAP	交换两个变量的值
TIME$=V$	设置时间
WAIT	等待端口信号
WEND	关闭当循环语句
WHILE	当循环语句

(2) I/O语句

语　句	作　用
BEEP	扬声器报警
CIRCLE	画圆语句
CLOSE #f	关闭文件
CLS	清除屏幕
COLOR	设置显示色、底色及边框色
DATA	数据语句
DRAW	画折线语句
FIELD #f	定义随机文件的字段
GET #f	读随机文件的记录
GET	读屏幕上的图形信息
INPUT	从键盘上读数据
INPUT #f	从文件读数据
LINE	画线
LINE INPUT	从键盘上读入一行
LINE INPUT #f	从文件中读入一行
LOCATE	设置光标位置
LPRINT	在打印机上打印数据
LPRINT USING	打印机按格式输出
LSET	靠左填字段缓冲
OPEN..FOR..AS	打开文件
OPEN	打开文件
OPEN"COMn:.."	打开异步通讯文件
OUT	输出字节到指定端口
PAINT	在指定区域涂颜色
PLAY	按字符串演奏音乐
PRINT	屏幕上显示数据
PRINT USING	按格式在屏幕上显示数据
PRINT #f	写数据到文件
PRINT #f, USING	按格式写数据到文件
PRESET	擦去一点
PSET	画点
PUT #f	写随机文件的一个记录
PUT	把图形信息输出到屏幕上
READ	从DATA语句中读数据
RSET	靠右填字段缓冲
SCREEN	设置显示器工作模式
SOUND	产生指定频率的声音
WIDTH	设置屏幕宽度
WRITE	显示数据
WRITE #f	向文件输出数据

3. 函数

(1) 算术函数

函　数	含　义	函　数	含　义
ABS(X)	取X的绝对值	INT(X)	取<=X的最大整数
ATN(X)	反正切函数	LOG(X)	对数函数
CDBL(X)	转换为长实型数	RND(X)	0～1的随机数
CINT(X)	舍入为整型数	SGN(X)	符号函数
COS(X)	余弦函数	SIN(X)	正弦函数
CSNG(X)	转换为实型数	SQR(X)	取X的平方根
EXP(X)	指数函数	TAN(X)	正切函数
FIX(X)	截断成整型数		

(2) 字符串函数

函　数	含　义	函　数	含　义
ASC(X$)	求出ASCII码	MKI$(X)	转换为整型内码串
CHR$(n)	求出ASCII码为n的字符	MKS$(X)	转换为实型内码串
CVI(X$)	把内码串转换成整数	MKD$(X)	转换为长实型内码串
CVS(X$)	把内码串转换成实数	RIGHT$(X$,n)	取最右的n个字符
CVD(X$)	把内码串转换成长实数	SPACE$(n)	取由n个空格组成的串
HEX$(n)	转换为十六进制字符串	STR$(X)	把X转换为字符串
INSTR(n,X$,Y$)	在X$中查找Y$	STRING$(n,m)	取n个CHR$(m)组成串
LEFT$(X$,n)	取最左的n个字符	STRING$(n,X$)	取n个X$首字符组成串
LEN(X$)	取X$的长度	VAL(X$)	求X$的数值表示
MID$(X$,n,m)	取中间的m个字符		

(3) I/O函数和辅助函数

函　数	含　义	函　数	含　义
CSRLIN	光标所在行的行号	POINT(x,y)	读指定点的颜色
DATE$	系统日期	POS(n)	光标所在列的列号
EOF(f)	文件结束	SCREEN(x,y,z)	读指定位置处的 ASCII 码或彩色码
ERL	出错所在行行号		
ERR	出错号	SPC(n)	设置n个空格
FRE(X$)	自由空间数	STICK(n)	读操纵杆坐标
INKEY$	从键盘上读入一个字符	STRIG(n)	读操纵杆按钮状态
INP(n)	从指定端口输入一个字节	TAB(n)	设置制表位置
INPUT$(n,#f)	从文件中读入n个字符	TIME$	系统时间
LOC(f)	文件读写指针的位置	USRn(X)	调用代码程序
LOF(f)	文件的长度	VARPTR(V)	变量的内存地址
LPOS(n)	打印头所在列的列号	VARPTR(#f)	文件控制块FCB的地址
PEEK(n)	读内存贮器的一个字节	VARPTR$(V)	变量的类型及地址串
PEN(n)	读光笔		

参 考 资 料

[1] The IBM Personal Computer Technical Reference Manual(Revised Edition), IBM Corp., 1983.

[2] iAPX86/88 User's Manual, Intel Corp., 1981.

[3] 周明德编著，微型计算机硬件软件及其应用，清华大学出版社,1983.

[4] 1982 Catalog, NEC Electronics U.S.A. Inc. Microcomputer Division,1982.

[5] Disk Operating System, Microsoft Inc., 1983.

[6] 清华大学等编，CP/M 微型计算机磁盘操作系统，科学技术文献出版社重庆分社，1982.

[7] WordStar General Information Manual, MicroPro International Corp., 1981.

[8] IBM BASIC MANUAL, IBM Corp., 1982.

[9] Microsoft BASIC Compiler User's Manual for MS-DOS, Microsoft Inc., 1981.

[10] 潭浩强、田淑清、谢锡迎编著，BASIC语言，科学普及出版社，1980.

[11] R. Clark, S. Koehler, The UCSD Pascal Handbook: A Reference and Guide Book for Programmers, Prentice-Hall Inc., 1982.

[12] N. 沃思, K. 詹森,(俞嘉惠译)，程序设计语言PASCAL，科学出版社，1983.

[13] K. L. Bowles, Problem Solving Using PASCAL, Springer Verlag, 1977.

[14] UCSD P-System,User's Manual(Version IV),SofTech Microsystems Inc.,1981.

[15] UCSD P-System, Internal Architecture Guide, SofTech Microsystems Inc., 1982.

[16] D. Beil,The Visicalc Book for the IBM Personal Computer,Prentice-Hall,1983.

[17] Supercalc User's Guide & Reference Manual, SORCIM, 1981.

[18] Multiplan Electronic Tutorial, Microsoft Corp., 1982.

[19] Junichiro Nishi,PIPS Revolution: Emergence of A Strategic Computer, SORD Computer System Inc., 1982.

[20] C.J. Date, An Introduction to Database System, IBM Corp., 1981.

[21] W. Ratliff, dBASE Ⅱ Assembly-Language Relational Database Management System, Ashton Tate, 1981.

[22] MELOM データベース入門 テキスト,三菱電機株式会社,昭和54年.

[23] Lotus 1-2-3 User's Manual(Release 1A), Lotus Development Corp., 1984.

[24] Lotus Electronic Tutorial(Version 1.00), Lotus Development Corp., 1983.

[25] MBA Tutorial and Reference Manual, Context Corp., 1983.

[26] D. Hearn, M.P. Baker, Computer Graphics for the IBM Personal Computer, Prentice-Hall Inc., 1983.

[27] R. J. Traister, Graphics Programs for the IBM PC,TAB Books Inc.,1983.

[28] J.D.Foley, A.Van Dam, Fundamentals of Interactive Computer Graphics, Addison-Wesley Publishing Company, 1982.

[29] S. Harrington, Computer Graphics: A Programming Approach, McGraw-Hill Book Company, 1983.

[30] R. E. Myers, Microcomputer Graphics, Addison-wesley Publishing Company, 1982.