Quick Basic
简明教程

计 算 机 教 程
青苹果电子图书系列

Quick BASIC 简明教程

杨小平　　编著
张　晋　　审校

前　言

对于计算机程序设计，有的人认为很难，也有些人说并不难。其实程序设计的学习不仅是简单地学习计算机语言本身，而是要学会如何用计算机编程来解决实际问题。

计算机技术的应用领域非常广泛，虽然不可能完全掌握计算机技术的各个方面，但完全可以根据自己的需要掌握或熟悉所需要的某一部分，有选择性地逐步学习。对于计算机语言的学习也是这样，要有所选择地从一门简单好学的语言开始，Quick BASIC 无疑是很好的选择。

我们推荐选择 Quick BASIC 进行学习，还因为它是目前非常流行、而且功能很强的一种结构化程序设计语言。它具有如下的特点：

1．Quick BASIC 采用编译方式，大大提高了运行速度，同时又保留了解释性会话式语言的易学易用的特点；

2．它是一种结构化的语言，易于阅读，便于设计和维护；

3．增强非数值计算（文字处理、图形、声音）的功能；

4．采用屏幕菜单技术，使人机会话更直观方便，操作简单。

在编写中，我们以实际应用为出发点，讲解编程的基础知识和基本技能，同时又照顾到初学者的接受能力，坚持少而精的原则，结合实例进行讲解。

需要说明的是，Quick BASIC 是实践性很强的编程语言。因此，在学习时，要特别注意实践能力的训练，既要培养分析具体问题的逻辑思维能力，又要养成结构化编程的习惯，形成良好的编程风格。

由于编著者水平有限，书中的缺点错误在所难免，恳请读者批评指正。

编　者

内容提要

Quick BASIC 是一门容易入门的编程语言，特别适合初学者学习。本书共组织了 13 章内容，介绍了 Quick BASIC 语言的基本概念、语法规则和利用 Quick BASIC 语言进行程序设计的方法，并提供了大量实例和习题，是广大用户学习和使用 Quick BASIC 语言的良师益友。

本书可作为大专院校师生学习 Quick BASIC 语言的教材，也可作为参加计算机等级考试的读者作为辅导教材。

目　　录

第一章　计算机的基本知识

什么是计算机，它有什么特点和用途，它的基本结构和组成是怎样的，它的工作原理如何？了解这些最基本的知识，对掌握计算机的应用是非常必要的。本章主要内容有：

- 信息化社会与计算机
- 计算机的特点
- 计算机的应用
- 计算机的发展与种类
- 计算机的硬件系统
- 计算机中数的表示与编码
- 计算机的工作原理
- 从机器语言到高级语言
- 计算机的系统软件

对于刚刚接触计算机的人来说，本章有的知识可能显得比较生涩，本章目的是让大家比较系统全面地了解计算机系统。等对计算机知识有一定深入了解以后，这些内容自然就会清楚明了。

1.1　信息化社会与计算机

商品上的条形码、收款台上的扫描器、电视屏上的气像预报、办公室的程控电话、银行里的信用卡、激光照排印刷的报刊书籍等等，在日常生活中，人们都已感触到信息化社会脉搏的跳动，人们已经不知不觉地在与计算机打交道。更不用说喷气飞机、人造卫星、洲际运载火箭、巡航导弹、遥感遥测、气像云图、工业机器人、蛋白质人工合成等等，这一系列的科学技术成就，无一不与计算机密切相关。

计算机自 20 世纪 40 年代诞生以来，对社会的发展产生了巨大的影响。18 世纪下半叶蒸汽机的发明标志着工业革命的开始，开创了人类利用机械代替体力劳动的时代，从而带来社会生产力的飞跃发展，创造了工业革命的物质文明；计算机的发明标志着一个新的信息革命时代的来临，开创了利用机械代替部分脑力劳动的时代，它又带来社会生产力的再一次飞跃。

科学技术成为第一生产力，信息科学技术又是其中的主导和核心。它包含了信息科学和信息技术两个方面，信息科学是以信息的运动规律和利用信息的原理作为主要研究内容的科学，信息技术则是扩展人的信息功能的技术，它主要包括了传感技术、通信技术和计算网络技术。

计算机不仅能自动、高速地进行大量精确、复杂的数值计算，而且还具有对信息进行采集、加工、生产、存储、传递的能力，所以它是进行信息处理的有力工具。在信息社会里，人人都应学会使用计算机进行工作。

1.2　计算机的特点

计算机之所以获得这样广泛的使用，发挥巨大的威力，这是因为它具有处理速度快、存贮容量大、计算精度高、有逻辑运算能力、运行自动化、可靠性高六大特点。

1．处理速度快

计算机的处理速度，通常是用每秒钟能做多少次运算来表示。1946 年的第一台计算机的运算速度是 5000 次/秒，现在一般的微型机的运算速度为几亿次/秒。现代的巨型机的运算速度已高达 1000 亿次/秒。

处理速度也可以用每秒钟执行了多少条指令来表示，如常用 MIPS 即 Millions of Instructions Per Second，每

秒一百万条指令作单位，或者用执行一条指令所需的时间来表示。

计算机性能越高，解决复杂问题的能力就越强：处理速度愈快，单位时间完成的工作量就愈大，这就意味着工作效率的大幅度提高。现在计算机的处理速度还在不断提高。

2．存贮容量大

计算机具有存贮能力。计算机中具有这种功能的装置称为存贮器。计算机能把大量信息（如数据、文字、图形、图像等）保存在存贮器中，并能从存贮器中取出来进行查找、排序、分类等处理，从而大大方便和加快了信息的利用。现代微电子技术的飞跃发展，使存贮器的容量愈来愈大，而价格愈来愈低。有关存贮器的特性，将在后面作进一步介绍。

3．计算精度高

科学和工程计算对结果的精度有较高的要求。一般计算机最低也能达到 16 位有效数字的精度，完全能满足科学和工程计算的需要。实际上，计算机可以满足更高精确度的要求，例如人们对圆周率π值的计算，我国南北朝的数学家祖冲之算到 3.14159269～3.1415927，精确到小数点后 7 位，日本有人用计算机，在 1988 年算到了小数点后 20132.6 万位，打印这一结果用了 40266 张打印纸。

4．有逻辑运算能力

计算机不仅能进行算术运算，如加、减、乘、除等的计算，而且能进行逻辑运算。在处理过程中，能够对数量的大小、符号的异同进行比较判断，从而能够帮助人们完成分类、检索、推理、定理证明、模式识别等工作。因此计算机正在朝着智能化方向发展。

5．运行自动化

现在的计算机是采用存贮程序的原理工作的，即把计算机要做的事用计算机能懂的语言写成程序，送到它的存贮器中存起来，然后计算机再执行程序。这时，只要人们给出了运行的命令，计算机就能自动连续地执行程序，直到程序执行完毕。

6．可靠性高

随着电子技术的发展和计算机应用的需要，计算机的可靠性也日益提高。从元器件到系统，从硬件到软件都使用了很多先进的技术，如错误检测码、指令重试、诊断技术、容错技术等，从而实现高度的可靠性。

人们常常用 RAS 来说明可靠性等概念，所谓 RAS 即可靠性（Reliability）、有效性（Availability）、可用性（Serviceability）3 个英文单词的缩写。

1.3 计算机的应用

计算机已在各行各业广泛使用，据统计使用计算机的行业和用途已达数千种，从主要用于数值计算发展到主要用于非数值计算，以下对计算的应用领域作简要介绍。

1．科学与工程计算

这是计算机应用最早也是最成熟的领域。1946 年的第一台 ENIAC 计算机就是用于火炮试验场的弹道轨迹的计算。人们对客观世界的认识逐步深化，越来越多的从定性描述走向定时描述，建立数学模型，从而产生许多复杂的计算问题。如果用手工计算，那将占用许多的人力花费很长的时间，有的甚至还是不可能实现的。计算机一出现，一切为之改观。现在，科学技术和工程设计的大量计算都要借助计算机来完成。例如人造卫星轨道的计算，宇宙飞船的制导，天文学中星体溶化形态学的研究，天文年历的编制，高能物理方面可控热核的反应的研究，反应堆的控制，生物学遗传工程中核糖酸及一些人工合成的计算，农业方面的生态系统模拟、水利设施的设计、气像预报等等。由于计算机速度快、精度高，因而可以缩短计算机周期，节省大量人力物力，大大促进科学研究与国民经济的发展。反过来，科学研究和国民经济的发展又向计算机提出了大量的新课题，成为研制功能更强更完善的计算机的推动力。

2．信息处理

信息处理就是对信息进行的采集、加工、存贮、传递等的处理。这已成为信息社会的主要特征。在人类历史上，早已存在信息处理，只不过是用手工方式进行，规模也小，而且对信息资源的重要性和价值的认识还不是那样深刻，信息的开发利用也就不那么充分。计算机的出现，很快就用于信息处理，给社会生活带来了广泛

而深刻的变化。例如，我国人口普查，每个人要登记 10 多项数据，全国汇总起来就是上百亿个的数据，对这些数据进行分类统计，用人工进行又慢又费人力，还易出错，用计算机处理，又快又精确，而且能提供各种统计分析，及时为国家经济建设的决策提供有用的信息。现在，用于企事业管理的各种管理信息系统（如财务、计划、物资、人事等的管理），用于文字处理的编辑排版系统和办公室自动化系统，用于图形图像处理的图像信息系统，用于图书资料查询的情报检索系统等等，不胜枚举。随着信息化社会的发展，人们已经认识到信息是社会发展的重要资源，信息就是财富，信息量与日俱增，信息处理必然会得到更广泛而深入地发展。

3．过程控制

计算机用于控制各种自动装置、自动仪表、机床工具的工作过程，就称过程控制，或称实时控制。所有的生产过程、科学实验、生活用具都可以实现自动控制。在工业生产过程中，广泛利用它实现生产过程的自动化，比如巡回检测、自动记录、监视报警、自动启停，以及自动调节和控制生产过程等。这样可以提高产量、降低能源消耗、节约劳动力、减轻劳动强度，从而带来巨大的经济效益。此外，交通运输方面的行车高度，农业方面的人工气候箱的温湿度控制，食品仪表的智能化，家电方面的全自动洗衣机，电视机自动选台、录音机自动选曲等等都广泛地应用到计算机的过程控制。

4．辅助工程

在工程设计制造领域应用了计算机就形成了计算机辅助工程 CAE（即 Computer Aided Engineering），它包括最先应用于辅助设计的 CAD（Computer Aided Desingn），后来又扩展到用于辅助制造 CAM（Computer Aided Manufacturing）、辅助测试 CAT（Computer Aided Test）等。

CAD 技术提高了质量和自动化程序，加快了新产品的设计和试制速度。比如飞机设计，过去从制定方案到产生全套图纸，花费大量人力物力，要用两年半到三年的时间，采用 CAD 后，只需 3 个月就可完成。工效大幅度提高。而像大规模集成电路的设计，没有 CAD，那简直是不可能做的事情了。

辅助工程还扩展到辅助教学 CAI（Computer Aided Institute），它使教学内容多样化、形像化、规范化，易于理解，便于学习。从而改善和提高教学质量。

5．人工智能

这是计算机应用的一个新领域。给它下一个准确定义是困难的。人工智能就是用计算机执行某些与人的智能有关的复杂功能。如判断、图像识别、理解、学习、规划和问题求解等。这种计算机，常称为智能机。当前，世界上正兴起第五代计算机的研究热潮，其目标就是使计算机实现智能化。目前研究的范围包括模式识别、自然语言理解、自动定理证明、自动程序设计、知识表示、机器学习、专家系统、机器人等。

1.4 计算机的发展与种类

1.4.1 计算机的发展

1．计算机出现前的计算工具

计算工具是随社会生产的发展需要而不断改进和完善的。人类最早的计算主要是计数，计数的方法就用人自身的手指、脚趾、或者身边的天然物，如小石块、绳子等物体作为计数工具。后来就有了人造的计算工具，我国古代劳动人民最先创造了算筹。它是用竹子或者用木头或者用骨制成的小棍，小棍的不同摆法，表示不同的数，祖冲之就是用算筹算出π值在 3.1415926 和 3.1415927 之间，这一结果比西方早了近 1 千年。以后算筹又演变成了带珠的算盘，一直流传至今，不仅在我国广泛使用，而且流传到国外，对世界文明也作出了重大贡献。

在西方，16 世纪出现了对数计算尺，17 世纪，西欧一些国家，资本主义经济开始出现，由于航海业的发展，天文学的研究，从而提出了大量的计算问题，促使人们寻求新的计算工具。1642 年法国教学家帕兹（Gottfried Wilhclm Von Leibnitz）在 1673 年进一步发展为能做加、减、乘、除四则运算的机械计算器。

18 世纪中叶工业革命兴起，对计算工具的研究就更进一步了。值得一提的是英国天文学家巴贝奇（Charles Babbage），他 1822 年制作了自动计算导航的差分机，1830 年设计了分析机，它是带卡片存贮数据的自动通用计算机，当时未能实现，但成为现代计算机发展的基础。他的一个助手，伯爵夫人艾达（Augusta Ada Lovelace），在完善该设计中，提出了自动循环计算的概念，为后来计算机的程序设计奠定了基础。

2．计算机的诞生

20 世纪初，科学技术发技术很快，特别是电子技术的进步，如电子管的发明，数学理论取得的成果，计算工具已有的基础，使得计算机的研制就日趋成熟了。1925 年布什（Vannever Bush）领导制造了第一台模拟式计算机，30 年代艾肯（Howard Aiken）研制的 Mark I，是第一台用 3 千多个继电器做成的数字自动计算机。它可以自动按程序员编的一系列指令进行计算。1937 年阿塔纳索夫（Atanasoff）提出了用电子管制造计算机，但因战争，经费短缺，未能实现。

由于第二次世界大战爆发，战争对现代武器的需要，也迫切需要改进计算工具。当时美国宾夕法尼亚大学莫尔学院电工系阿伯丁弹道研究实验室共同负责为陆军每天提供 6 张火力表，每张表都要计算几百条弹道，用台式计算器计算一条飞行时间 60s 的弹道要花 20h，用 200 人同时计算，一张火力表也往往要算两、三个月，可见问题相当严重。于是研究电子数字计算机的计划提到了议事日程，在 30 多岁的物理学家莫克利（John W.Mauchly）和 24 岁的总工程师埃克特（J.Presper Eckert）的主持下，ENIAC（Electronic Numerical Integrater and Computer），即电子数值积分和计算机，1946 年正式投入运行。它的运算速度达 5000 次/秒，比已有的计算机快 1000 倍。

3．计算机的迅猛发展

ENIAC 是五个庞然大物，共用 19000 个电子管，30 吨重，占地面积达 170 平方米，耗电功率为 150KW。性能并不理想，运算速度 5000 次/秒，存贮字长为 10 位的十进制数 20 个，程序是用线路连接的方式来实现的，不便于使用。经过 40 年的努力，使用的器件已经历了电子管、晶体管、集成电路、大规模和超大规模集成电路 4 代的变化，性能提高了一百万倍，价格降为万分之一，世界上没有一项技术像它这样变化快。

现在计算机正向着巨型化、微型化、网络化、智能化方向发展。

1.4.2 计算机的种类

计算机的种类很多，有不同的分类标准。

1．按所处理的数据特点来分

（1）数字式计算机（Digital Computer）处理的是脉冲变化的离散量，如 0，1 之类的数字，它计算精度高，抗干扰较强。

（2）模拟式计算机（Analog Computer）处理的是连续变化的模拟量。如压力、温度等物理量的变化曲线。

（3）混合式计算机（Hybrid Computer）它是将数字式和模拟式结合为一体的计算机。通常说的“计算机”均指电子数字式的计算机。

2．按作用来分

（1）通用计算机

可用来完成多种不同的任务，如既可作科学计算，又可作各种事务管理，适用范围广。

（2）专用计算机

只用于完成某一专门任务，如控制炼钢轧钢的专用机、各种程控机床用的计算机。它的执行效率较通用机高。

3．按计算机处理速度、内存容量、配置规模来分

（1）巨型机和小巨型机

处理速度上千亿次，字长 256 位以上，内存千兆字节以上，如我国的“银河”系列巨型机。

（2）大、中型机

处理速度从数百万次到数千万次，字长 128 位以上，内存数千兆字节。

（3）小型机和超级小型机

处理速度数百万次，字长 64 位以上，内存数百兆字节。

（4）微型机和便携机

这是发展变化相当大的一类计算机。其中：高档机，字长 64 位，如用 PIII、P4 为 CPU 的工程工作站，处理速度、字长、内存均已达到或超过超级小型机的指标；中档机，字长 64 位或 32 位，时钟频率 500～800MHz，内存数百兆字节；低档机，字长在 32 位以下，时钟频率 500MHz 以下，内存 64MB 以下。

（5）单片机

是把计算机的基本组成部件如 CPU、存贮器、I/O 接口做在一张芯片上的计算机。它集成度高、体积小、功能强、速度快、价格低廉、可靠性强，在机电一体化和传统工业改造中广泛应用。

以上种类和界限是发展变化的，并无严格规定，昔日的大型机还赶不上如今的袖珍机，随着计算机技术的飞速发展，将来还可能发生巨大的变化。

1.5 电子计算硬件系统

计算机是一个复杂的系统，它由硬件（Hardware）系统和软件（Software）系统组成。硬件系统是指由电子元器件和机电结构等构成的各种的物理实体；软件系统是指在计算机中运行的各种程序。本节先介绍它的硬件系统，然后在了解了有关的概念的基础上，再在 1.9 介绍它的软件系统。

计算机的硬件系统是由多台不同功能的设备相互连接而成。根据计算机的处理能力和用途的不同，系统的大小和配置是各不相同的。一个大中型的计算机系统，可以由几十台设备组成，而一个微型机系统可以由 4、5 台设备组成。但是，无论系统大小，一个计算机的硬件系统都由以下几个基本部分组成。

1.5.1 主机

它包含中央机和主存贮器，是整个系统的核心。

1．中央处理机

常常用 CPU 表示，它是英文 Central Proccessing Unit 的缩写。中央处理机中，主要由控制器和运算器构成。

（1）运算器

运算器是执行算术运算和逻辑运算的电子部件，又叫 ALU，即英文 Arithmetic Logic Unit 的缩写。它和算盘的功用相似，算盘是用人工进行操作的计算工具，而运算器能自动连续进行运算，算盘只能做加、减、乘、除等基本的算术运算，而运算器不仅能做算术运算，它还能进行逻辑判断、逻辑比较等基本逻辑运算。

运算器是在控制器的指挥下，从主存贮器中取出数据进行运算，结果传送回主存贮器，形成运算器和主存贮器之间的数据流，在两者之间进行信息交换。

（2）控制器

它好比人的神经中枢，是整个系统的最高司令部，它指挥系统各个部分协调地工作，使系统自动地执行程序。

控制器是由复杂的逻辑电子线路组成。它按事先设计好的指令和程序发出各种控制用的命令，以便系统有条不紊地运行。比如，它从主存贮器中顺序地取出一条一条的指令，每取一条，就要分析这条指令（称为译码），根据这条指令的功能向各功能部件发控制信息，执行这条指令规定的任务。一条指令执行完毕，控制器将获得执行情况的反馈信息，以后又自动地去取下一条指令，又进行译码、执行，重复这一过程，直至程序结束。

2．主存贮器

简称 MS，即英文 Main Storage 的缩写。又叫内部存贮器，或称内存。它和人的大脑相似，用来存放计算机运行时的程序和数据。

具有记忆能力的器件种类很多，有半导体存贮器、磁介质存贮器、光存贮器等。现在的计算机中都是用半导体存贮器作为主存贮器。半导体存贮器是由若干集成电路芯片组成，一块芯片上有成千上万个的存贮单元电路，就像构成生物组织的细胞一样。这种单元电路具有两种不同的状态，常称之为双稳态电路。因此可以用 0 和 1 来表示这两种状态。这个单元电路是存贮信息的最小单位，称为一个位（即 bit，常译成“比特”）。这个单位很小，常常把 8 个位合起来称为 1 个字节（Byte）。即 1Byte=8bits。

实际上，计算机的存贮容量很大，常常还用 KB、MB、GB 作单位。

1KB＝1024Bytes

$1MB=10^3KB=10^3\times1024Bytes$

$1GB=10^3MB=10^6KB=10^6\times1024Bytes$

根据功能的不同，半导体存贮器可分为以下两类：

（1）随机存贮器

又叫 RAM（Random Access Memory），即读写存贮器，既可以向它存入信息，又可以从它取出信息。我们称“存入”的动作叫做写（Write）操作，“取出”的动作叫“读”（Read）操作。RAM 是读、写均可的。它有“取之不尽，去旧纳新”的特性。取之不尽的意思是：从 RAM 某单元中读出了信息之后，该单元的信息仍保持不变；去旧纳新的意思是：向某单元写入信息之后，该单元原有的信息将被写入信息代替，即原来的信息不再保留。RAM 只有当电源接通的状态下能存贮信息，一旦断电，存贮的信息就要全部丢失。所以，它只供计算机运行时存放当前的程序和数据，不便于长期保存信息。

（2）只读存贮器

又叫 ROM（Read Only Memory）。和 RAM 不同，只能从它读出信息，一般不能向它写入信息（只能用专门的写入器可以向它写入程序）。一旦写入了程序，无论电源接通或断开，写入的信息都不丢失。因此，常常把一些系统程序写在 ROM 里，供系统起动或运行时使用，这种 ROM，就成了固化软件，又称固体（Firm ware）。

1.5.2 外部存贮设备

对于大批量的信息存贮，只有半导体存贮器是不够的。常常在主存之外配备外部存贮器，简称外存，以辅助主存之不足，故又称为辅助存贮器，简称辅存。磁盘机、磁带机和光盘就是现在广泛使用的外存设备。

磁盘机、磁带机都是根据磁感应原理制成的。它们的结构、特点及用作简述于下：

1．磁盘机

有磁性介质的圆盘绕转轴旋转，活动的读写头在径向作直线运行，从而在盘面上形成若干同心圆，构成磁道，信息记录在磁道上，磁盘机又分两类。

硬磁盘机（Fixed Disk）：将盘片密封在一个盒子内，以保持盘面的洁净，盘片数随容量大小而不同。如一般微机中的硬盘机容量有 10GB~20GB、40GB 的。

软磁盘机（Floppy Disy）：它的磁盘和普通唱片相似，盘片是可卸的，盘片用纸袋封住，一片一片单独使用，根据直径大小，有 5"、3"等几种。每张盘的容量不等，低密度盘能记录 360KB、高密度盘能记录 1.2MB、1.44MB。

2．磁带机

计算机专用的磁带机和普通家庭用录单机原理相同，但是在转速、记录密度、磁道数、容量等特性方面不一样，专用磁带机的要求要高些。在低档微机中也可用录音机作外存。在大中型计算机中使用专用磁带机作外存。

磁带上的信息，只能顺序写入和顺序读出，存取速度比磁盘机低，但它容量大、可以拆卸，价格比磁盘机低，所以暂时不用的信息，可以存在磁带上作为后备。

磁盘上的信息，可以随机地写入和读出，存取速度远比磁带机高，虽不及半导体存贮器的速度快，但是断电后不会丢失信息。因此，常常用它存贮重要的程序和数据，和主布配合，使计算机更有效的运行。

磁盘机和磁带机既可从主机输出信息，写在盘上，又可以从它把信息读出送入主机，因此，相对主机而言，它既是输出设备，又是输入设备。

近几年来发展起来的光盘，也是大容量的外存设备，已经普遍使用。

3．输入设备

人们使用计算机时候，就要把信息告诉计算机。把程序和数据送到计算机中去的装置就是输入设备。键盘（Keyboard）是最常用的输入设备，还有软磁盘输入机、卡片读入机、纸带读入机、手写笔数字化仪器等。此外，光学符号阅读机、语音识别输入设备也已取得很大进展。

4．输出设备

计算机处理的结果要返回给人们，这就要用输出设备。常用的有阴极射线管显示器（CRT）、打印机、绘图仪等等。

极射线管显示器，就是常说的显像管，有单色和彩色两类。分辩率有高、中、低 3 档。在荧屏上，既可以显示计算机处理的结果，又可以看到内存中的程序和数据。既可以显示字符（英文或汉文）又可以显示复杂的图形。

打印机的种类很多，常用的有点阵式打印机、宽行打印机、激光打印机等。不仅可以用来作源程序清单和计算结果的输出，而且还可以用它输出表格、图形、编排书刊。

综上所述，控制器、运算器、存贮器、输入设备、输出设备等 5 个部分以一定方式相互连接成为一个和谐

的整体。

1.6 计算机中数的表示与编码

1.6.1 二进制数

在数学和日常生活中，我们最熟悉的是逢 10 进位的十进制数。它有两个基本特征：

（1）数位从右至左，每位代表的数值是按 10 的指数增加的，如个、十、百、千…等，这个数值，我们常称为位权值。

例如：2184 这个数可以用多项式表示为：

$$2184=2\times10^3+1\times10^2+8\times10^1+4\times10^0$$

千 百 十 个　千　百　十　个

位 位 位 位　位　位　位　位

可见，十进制数的位权值是以 10 为底的幂，因此任何一个十进制数都可以用一个多项式表示：

$$(N)_{10}=a_n\cdot10^n+a_{n-1}\cdot10^{n-1}+\cdots+a_1\cdot10^1+a_0\cdot10^0+a_{-1}\cdot10^{-1}+\cdots+a_{-m}\cdot10^{-m}$$

——————整数部分——————　　———小数部分————

（2）每一位数上要有 10 个不同的符号：0、1、2、3、4、5、6、7、8、9。在多项式中的 a_n、a_{n-1}…，可以是这 10 个中的任意一个。

我们用 10 个不同形状的字符表示 10 个数，但是在计算机的硬件中，还不可能找到 10 种不同状态的物理量来表示 10 个数。因此，数制必须修改。我们知道，计算机是基于电子器件的特性来工作的，这些电子线路处理的是数字信号，它们表现为电位的高与低，电路的接通与断开等两种状态，我们用“0”、“1”来描述。这样就可以用 0、1 两个状态表示数，以 2 为底的幂作位权值，建立逢 2 进位的进进制数。例如：

$$(1101)_2=1\cdot2^3+1\cdot2^2+0\cdot2^1+1\cdot2^0$$

表 1-1 列出从 0～15 的二进制数的表示。

表 1-1　各种数制对应表

数	十进制	二进制	八进制	十六进制	数	十进制	二进制	八进制	十六进制
零	0	0	0	0	八	8	1000	10	8
一	1	1	1	1	九	9	1001	11	9
二	2	10	2	2	十	10	1010	12	A
三	3	11	3	3	十一	11	1011	13	B
四	4	100	4	4	十二	12	1100	14	C
五	5	101	5	5	十三	13	1101	15	D
六	6	110	6	6	十四	14	1110	16	E
七	7	111	7	7	十五	15	1111	17	F

任何一个二进制数都可以用以下多项式表示

$$(N)_2=a_n\cdot2^n+a_{n-1}\cdot2^{n-1}+\cdots+a_1\cdot2^1+a_0\cdot2^0+a_{-1}\cdot2^{-1}+\cdots+a_{-m}\cdot2^{-m}$$

——————整数部分————　　——小数部分——

式中，a_n、a_{n-1}、…是 0 或 1。

1.6.2 二进制和十进制间的转换

1．二进制数转换为十进制数

按位权值展开，求多项式之和，就得到十进制数。

为了区别不同的数制，我们把数值用圆括弧括起来，在右下角用小号字体的 2、10 等来注明，如 $(110101)_2$，表示是二制制数，$(216)_{10}$ 表示是十进制数。

（1）整数的转换

例如：

$(110101)_2 = 1\times2^5+1\times2^4+0\times2+1\times2^2+0\times1^1+1\times2^0$

$= 32+16+0+4+0+1$

$= (53)_{10}$

（2）小数的转换

例如：

$(0.101)_2 = 1\times2^{-1}+1\times2^{-2}+1\times2^{-3}$

$= 0.5+0+0.125$

$= (0.625)_{10}$

（3）既有整数部分又有小数部分的数，转换时，分别按整数和小数转换，再用小数点连起来。例如：

$(110101.101)_2 = (110101)_2 + (0.101)_2$

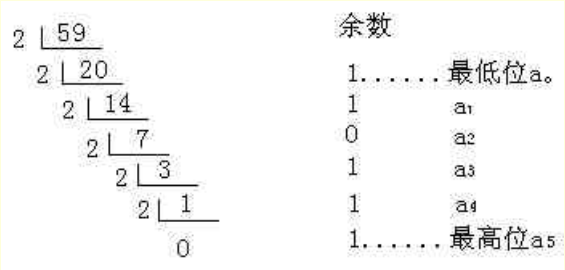
$= (53)_{10} + (0.625)_{10}$

$= (53.625)_{10}$

2. 十进制数转换为二进制数

（1）整数的转换

把十进制整数连续除以 2，记录其余数，就得到二进制数，这个方法简称为除 2 取余法。例如：



按由高位到低位写下来就得到：

$(59)_{10} = (111011)_2$

（2）小数的转换

把十进制小数连续乘以 2，取出其积的整数，就得到二进制数，这个方法简称为乘 2 取整法。

例如：

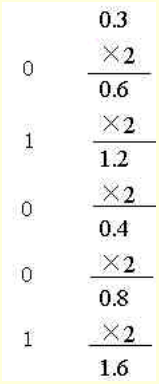
	乘积的整数部分 ↓	0.5625 ↓
a^{-1} 最高位→	1←	× 2 = 1.1250
a^{-2}	0←	× 2 = 0.2500
a^{-3}	0←	× 2 = 0.5000
a^{-4} 最低位→	1←	× 2 = 1.0000

于是得到：

$(0.5625)_{10} = (0.1001)_2$

小数换算中，不是都能得到小数部分乘积为零的结果，这时，只能按精度要求取若干位，作为它的近似值。

例如：



（0.3）$_{10}$＝（0.01001…）$_2$≈（0.01001）$_2$

（3）既有整数部分又有小数部分的数，转换时要把整数部分和小数部分分别计算，再把两部分加起来，就是一个二进制数。例如:

（59.3）$_{10}$＝（59）$_{10}$＋（0.3）$_{10}$

上例已求出（59）$_{10}$＝（111011）$_2$、（0.3）$_{10}$≈（0.01001）$_2$

所以（59.3）$_{10}$≈（$_{10}$111011.01001）$_2$

1.6.3 八进制和十六进制数

计算机只能识别二进制数，当数值愈大时，二进制数表示的位数就愈多，例如

（11011011011.001）$_2$＝（1755.125）$_{10}$

这个二进制数有 11 位，小数有 3 位。（可以表示到上千的十进制数），在读数和书写时，都极不方便。因此，在编写程序时又常常使用八进制数和十六进制数。

和十进制、二进制的规则相仿，八进制数采用 8 个符号（即 0、1、2、3、4、5、6、7），逢 8 进位。十六进制数采用 16 个符号（即 0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F），逢 16 进位。例如：

（16）$_8$＝$1\times8^1+6\times8^0$＝（14）$_{10}$

（25）$_{10}$＝$2\times16_1+5\times16^0$＝（37）$_{10}$

以下介绍各种数制之间的转换方法：

（1）八进制、十六进制转换为十进制

方法与上面二进制和十进制的转换方法相类似。如上例所示。

（2）八进制数转换成二进制数

把每位八进制数用所对应的 3 位二进制数表示，就转换为它的二制数。

例：

（3	2	6）$_8$
↓	↓	↓
（001）	（010）	（110）

即（326）$_8$＝（11010110）$_2$

（3）十六进制数转换成二进制数

把每位十六进制数用所对应的 4 位二进制数表示,就转换为的二进制。

例如：

（7	A	3）$_{16}$
↓	↓	↓
（0111）	（1010）	（0011）

即（7A3）$_{16}$ ＝（11110100011）$_2$

（4）二进制整数转换为八进制数

从最低位开始，向左每 3 位分为一组，不够 3 位的用 0 补足 3 位，将每组的二进制数按对应的八进制数写出，就转换成了八进制数；二进制小数，则从小数点开始向右每 3 位划为一组，不够三位，用 0 补充 3 位，写出其对应的八进制数，例如：

（11101001110.1101）$_2$

分组为: （011）（101）（001）（110）.（110）（100）

对应的八进制数为： ↓ ↓ ↓ ↓ . ↓ ↓

3 5 1 6 . 6 4

即（11101001110.1101）$_2$=（3516.64）$_8$

（5）二进制数转换成十六进制数

从最低位开始，向左每 4 位分为一组，不够 4 位的用 0 补足 4 位，将每组的二进制数按对应的十六进制数写出，就转换成了十六进制数，二进制小数，则从小数点开始向右每 4 位划为一组，不够 4 位用 0 补足 4 位，写出其对应的十六进制数。例如:

（1011011.01101）$_2$

分组为： （0101） （1011） （0110） （1000）

对应的十六进制数 ↓ ↓ ↓ ↓

5 B 6 8

即（1011011.01101）$_2$=（5B.68）$_{16}$

1.6.4 编码

我们用 0、1 来表示二进制，是数的一种编码表示。编码的方法在我们生活中也是常见的，如通信中的邮政编码、电话号码、电报中的莫尔斯码、身份证代码、学号代码、汉字的国际码、区位码等等。

在计算机中只给数的表示进行了编码还不够，要处理数的符号是正还是负怎样办？处理英文字符、乘除，以至汉字又怎么办？所以都得用 0、1 来给它们编码。现在最通用的一种给字符的编码是 ASCII 代码（即 American Standard Code for Information Interchange 的缩写），例如英文字母 A，用 8 位二进制数 01000001 表示，换成十六进制表示为（41）$_{16}$，各种字符的 ASCI 代码，列在附录一中。

计算机要做各种运算和处理，这些动作也需要编码，例如要它做 5＋6 的加法，就要有取数、进行加、送数等操作，对这些操作的说明也用 0、1 的符号串编码，这就形成了计算机的指令码。

总之，计算机只能识别 0、1 两个符号。和计算机沟通信息，就得要以这两个最简单的符号为基础，对各种信息形式进行变换，把计算机不能直接识别的信息变换为计算机能识别 的信息。

1.7 计算机的工作原理

1.7.1 计算机的指令系统

人们要计算机处理信息，就要给计算机规定一些最基本的操作，并用 0、1 来表示这些操作，这就构成为一条一条的指令。在设计的时候，就给它规定了一套指令，称之为指令系统（即 Instruction Set）。不同型号的计算机，指令系统也不相同。而一条指令由操作码（Opcode）和操作数（Oprand）两部分构成，例如在 Z80 中有这样一条指令：

11000110 00000110

操作码 操作数

操作码 11000110 表示做加法的操作，操作数是 00000110。这条指令的功能就是把操作数 00000110 与计算机的累加器中的数相加，相加的和仍放在累加器中，例如先累加器中放一个数 00000101，执行这条指令的过程如图 1-1 所示。这条指令用十六进制表示为：C6 06。

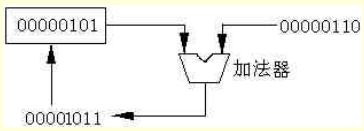


图 1-1 执行过程

1.7.2 计算机的算题过程

计算机算题要由要事先告诉它算题的方法和步骤，而且要把动作分解得非常之细，每一步都是计算机指令能执行的，它才能按照人们设计的步骤，一步一步地去执行。如果人们设计的步骤是正确的，计算机就能算得正确的结果，如果设计的步骤不正确，计算机就不能算出正确的结果，甚至没有结果。

以极简单的 5＋6 的加法为例。它的解题步骤如下：

（1）把数字 5 和 6 送到机的内存中存放起来；存贮单元都要有一个编号，称为地址。例如 43 号地址存入数 5，写为（43）←5，同样，（44）←6；

（2）把数 5 取出来，送到累加器；

（3）把数 6 取出来，与累加器中的数相加，结果放在累加器中；

（4）把累加器结果送回到内存的 45 号地址存放起来，即（45）←11；

（5）把结果输出到打印机或显示器上；

（6）结束。

这些解题步骤的集合，我们就称之为程序，第 1 步是数据的输入，第 5 步是数据的输出。2～4 是计算机内的处理，用某种机器指令写出 2～4 这一过程，有如下的形式：

存贮地址	机器指令
⋮	
00010000	00111011
00010001	00000000
00010010	00001011
00010011	00100001
00010100	00000000
00010101	00001100
00010110	10000110
00010111	00110010
00011000	00000000
00011001	00101101
⋮	
00101011	00000101
00101100	00000110
00101101	00001011

这些由机器指令构成的有序集合，就称为机器语言程序。计算机的工作就是按规定顺序地执行程序。人们使用计算机就要为它编制程序，我们称为程序设计。用机器语言编写程序很不直观，初学者看到这个程序就不知其所以然了。不必着急，看不懂没关系，这只是让你对机器语言有点感性认识。对于计算机来说，只有这样的机器语言，才能执行。

1.7.3 存贮程序原理

有了指令和程序的概念，就可以进一步了解计算机的工作原理，计算机是基于存贮程序的方法工作的。

首先，把程序和数据通过输入设备送往内存。一般的内存都是划分为很多存贮单元，每个存贮单元包括字节数随计算机的机型不同而不同，一个存贮单元通常称为字（Word），一个字的位（Bit）的多少，称为字长（Word Length），例如，中型机 M340，字长为 32 位（即 4 个 Bytes），微机字长有 8 位、16 位、32 位等。

每个存储单元都有地址编号，这样按一定顺序把程序和数据存起来，而且还把内存分为若干区域，比如有专门存放程序的程序区和专门存放数据的数据区。

其次，执行程序，必须从第一条指令开始，以后一条一条的执行。一般的情况下按存放地址号的顺序，由小到大依次执行，当遇有条件转移的指令时，才改变执行的顺序。每执行一条指令，都要经过三个步骤。第 1 步，把指令从内存中送往译码器，称为取指；第 2 步，译码器把指令分解成操作码和操作数，产生相应的各种控制信号送往各电路部件；第 3 步，执行相应的操作。这一过程是由电子线路来控制的，从而实现自动连续的

工作。

这一原理是计算机结构设计的基础，它是美籍匈牙利数学家冯·诺依曼（Von Neumann）1946 年提出并论证的，因此常把这一类型的计算机称为冯·诺依曼计算机，迄今为止，各种计算机仍属这一类型。

1.8 从机器语言到高级语言

1.8.1 机器语言

在上一节里已经看到了直接用“1”、“0”组成的机器指令编写的程序，这就是机器语言源程序。对计算机来说，这是它唯一能直接“听”得懂的语言。所以，常常称之为面向机器的语言。但是，对使用计算的人来说，这是十分难懂的语言，它难读、难记、难写，容易出错，不同机型又不通用。显然人和机器之间的通信存在巨大的鸿沟，只有填这补上这个鸿沟，使用的愈是方便容易，机器又能懂得，计算机才能发挥更大的作用。为此，人们研究了一种汇编语言。

1.8.2 汇编语言

把用二进制数表示的指令，用一些符号来表示，如用表示操作的英文缩写来代替指令代码，用 16 进制数表示数字，上一节中的几条指令就可写出为如下形式：

```
LD      A ,     (2BH)
LD      HL ,    2CH
ADD     A ,     HL
LD      (2DH ,  A
```

- 第 1 条 LD 即 Load 的缩写，表示“取数”的操作，A 表示累加器，(2BH) 括号内的十六进制数是内存地址。它的含义是把存放在内存第 43 号地址的数（已存放有数“5”）取出来，放到累加器 A 中。
- 第 2 条 LD 仍为取数，HL 表示一个暂时存放数据的寄存器名，它的含义是把内存地址号 44（表示为十六进制数 2CH），放在寄存器 HL 中。
- 第 3 条 ADD 是“加”的意思，操作数中指出把 A 和 HL 所指示的 44 号地址中存放的数相加（即将 5 和 6 相加，结果为 11），并把结果放在 A 中。
- 第 4 条 LD，送数，把计算结果 11 从 A 中送到内存地址为 45（表示为十六进制数 2DH）的存贮单元存放。

这种用符号代替后的指令，就叫汇编语言，又称符号语言，像 LD、ADD 等这类符号称为指令符号或助记符。用汇编语言编写的程序，称为汇编语言源程序，常简称为汇编语言程序。

这种语言，相对机器语言就容易读、容易写了。但是，机器却一点也不懂了。因此，计算机是无法直接执行的。一个不懂汉语的外国人到中国来要和中国人直接交谈，那是无法进行对话的，所以，只好求助翻译。在计算机中，也同样采取这种方法，人们编写程序用汇编语言，然后请一位翻译，把汇编语言程序翻译成机器难懂得的机器语言程序，这个翻译过程，叫做“汇编”。汇编后产生的机器代码称为目标程序。翻译可以由人手工完成，但做起来即繁琐单调，又容易出错。实际上，我们是让计算机来做。因而就研制出了担任翻译的程序，取名叫汇编程序。汇编过程如图 1-2 所示。

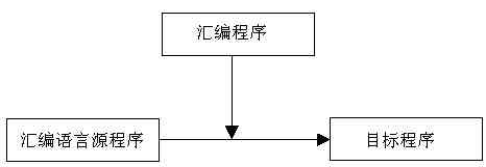


图 1-2 汇编过程

汇编语言使程序设计工作前进了一大步，但是仍然存在很多缺点，第一、不便于我们求解问题过程的描述，如一个数学公式，汇编语言的表达形式与人们的习惯表达形式差别很大；第二、它仍是面向机器的语言，不同机型，汇编语言也不一样，因而用它编制的程序，没有通用性。为了克服这些不足之处，人们进一步研制出了

高级语言。

1.8.3 高级语言

它是用更接近人的自然语言和数学表达式的一种语言，它由表达不同意义的“关键字”和“表达式”，按照一定的语法语义规则组成，完全不依赖机器的指令系统。这样的高级语言为人们编制程序提供了很大的方便，编制出来的程序易读易记，也便于修改、调试、大大提高了程序的效率，也大大提高了程序的通用性，便于推广交流，从而极大的推动了计算机的普及与应用。

我们看到高级语言离人们的理解愈加接近了，但离计算机的理解就越来越远了。计算机是不能直接理解那些英语单词、数学表达式的。所以，为了填补人机之间的鸿沟，还是得求助翻译。把高级语言翻译成机器能懂的机器语言，这一翻译过程，我们称为“编译”，如图 1-3 所示。用高级语言编写的程序称为高级语言源程序（Source）来做，这个编译程序也是人们设计并事先装入计算机的。没有这一程序，高级语言源程序就无法在计算机中执行。

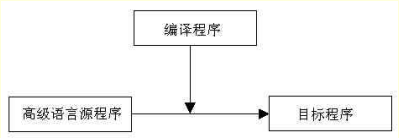


图 1-3 编译过程

因此，高级语言源程序从输入程序到执行程序都要经过编译，一般是把整个源程序编译成目标程序后，再执行目标程序。不过，有的高级语言与此稍有差别，如较早的 BASIC 语言，源程序是翻译下一个语句就执行一个语句，边翻译一个语句就执行一个语句，边翻译边执行，并不生成目标程序，这一过程取名叫解释，担任翻译工作的称为解释程序，这也是人们设计并事先装入计算机的。

世界上已有一百多种高级语言，最流行的有十几种，例如：

（1）FORTRAN（Formula Translator 的缩写）。它是世界上最早出现的高级语言，从 1954 年问世以来，经过几次大的发展，功能有很大的增强。现在流行的是 FORTRAN 77 版本。它特别适于作科学、工程计算。

（2）COBOL（Common Business Language 的缩写）。适于作非数值计算的商业、管理领域。

（3）PASCAL 语言是最早出现的结构化语言，适用于计算机教育。

（4）PL/I 语言是一种大型语言，功能强，数值计算和数据处理均适用。

（5）Ada 语言是一种工程化的大型语言，适合于大型软件工程。

（6）C 语言是广泛推广的结构化语言，适于编写系统软件。

（7）BASIC 语言是一种简单易学的会话式语言，在世界上应用最广泛。

1.9 计算机的软件系统

计算机的软件系统是计算机中各种程序的集合。如上节提到的各种高级语言的编译程序，都属于软件。只有这些软件还不够，为了扩充和增强计算机的功能，软件的种类和数量是很多的。一般分为系统软件和应用软件。

1.9.1 系统软件

（1）操作系统（Operating System）

它是软件系统的核心，无论是大中型还是微型机，操作系统都是必不可少的，它是对计算机系统的资源（包括硬件和软件的各种资源）进行管理和控制的程序。任何一个用户都是通过操作系统来使用计算机的。

不同的机型可能配备不同的操作系统。一般现在微型计算机用户使用的是 Windows 98 视窗操作系统，最新的版本是 Windows 2000，在出现图形界面操作系统以前流行的是一种叫 DOS（即 Disk Operating System）的磁盘操作系统。对于用户来说，并不需要了解操作系统的原理和细节，只要会使用操作系统就行了。除 Windows、DOS 而外，还有 UNIX、LINUX 等操作系统，也正逐步推广。

（2）各种语言处理程序，如 BASIC、FORTRAN、COBOL、PASCAL、C 等高级语言的编译程序、汇编程序等。

（3）各种面向用户的服务程序，如文本编辑程序、调试程序等。

（4）各种面向计算机维护的诊断、测试程序。

1.9.2 应用软件

它是面向各种应用领域的专用软件，是十分丰富的，有的具有通用性，很多行业都可使用，如科学计算程序包、统计分析程序包、各种数据库管理系统等等。

为了便于人们使用，充分发挥计算机的效率、尽量扩展计算机的功能，因此就需要有多种多样的软件，一台计算机不但要有性能良好的硬件系统，更要有丰富多彩的软件系统。随着计算机应用的日益广泛深入，软件的研究和开发越来越显示出它的重要性，并已形成软件产业，现在往往一个计算机系统的软件研制费用大大超过硬件的研制费用。

1.10 本章小结

1．计算机不仅仅是一个计算工具，更主要地它是信息处理的工具；它具有速度快、容量大、精度高、能逻辑判断、自动连续运行、可靠性好等特点，在国民经济领域以至在人们日常生活中得到了广泛的应用。电子信息技术将成为社会生产力发展的主要因素。

2．计算机是利用电子器件的特性来工作的，因此基于电子器件的开和关的两个不同状态而与二进制数学结下了不解之缘，因此，计算机能直接处理以 0 和 1 这两个简单的符号为基础的各种信息编码，如数的编码、符号的编码、指令的编码。由此而产生的更为复杂的各种语言、各种软件。最终都要归结到计算机能识别的 0、1 构成的符号串上。

3．对于一个完整的计算机系统的组成可见图 1-4 所示。

图 1-4 计算机系统组成

习　题

1．把以下十进制数换算成二进制数：

157　　21.125　　1106　　10.3　　216.005

2．把以下二进制数换算成十进制数：

0100　　1101　　1101001　　100111101.1111　　110100111.001

3．把以下十六进制数换算成二进制数：

E　　2C　　FFFF　　A21　　BC.2A

4．把以下二进制数换算成十六进制数：

1011　　11000101010　　100110001　　11001000.001　　1011110001.1

5．当你正在计算机的键盘上输入一批数据（或一段程序）时突然电源掉电，你的数据还会在内存中吗？为什么？

6．什么是汇编程序、编译程序、解释程序？分别说明它们的功能？

第二章　算法与程序设计

学完第一章后，知道了计算机所做的工作就是按指定的步骤执行一系列的操作，以完成某项特定的任务。因此，要计算机为我们做事，就必须事先为它设计出这一系列的操作步骤，并用计算机语言写成程序，这就是程序设计。解决问题的方法和步骤，称之为算法，它是程序设计的关键，因此，在学习用 Quick BASIC 语言进行程序设计之前，了解一点算法的知识是必要的，它是以后各章学习的基础。本章主要内容有：

- 算法的概念
- 用流程图表示算法
- 用结构化流程图（N-S 图）表示算法
- 结构化程序设计步骤

2.1　算法的概念

2.1.1　什么是算法

计算一个算术式，要先乘除后加减，这个规则就是算法。使用算盘，珠算口诀就是算法。在日常生活中做任何一件事情，都是按照一定规则，一步一步地进行，比如乐队演奏、厨师炒菜都有各自的操作步骤，这也是算法。在工厂中生产一部机器，先把零件按一道道工序进行加工，然后，又把各种零件按一定方法组装成一部完整机器，它们的工艺流程就是算法；在农村中种庄稼有耕地、播种、育苗、施肥、中耕、收割等各个环节，这些栽培技术也是算法。总之，任何这些数值的计算或非数值计算的过程中的方法和步骤，都称之为算法。

计算机解决问题的方法和步骤，就是计算机的算法。计算机用于解决数值计算，如科学计算中的数值积分、解线性方程等的计算方法，就是数值计算的算法；用于解决非数值计算，如用于管理、文字处理、图形图像等的排序、分类、查找，就是非数值计算的算法。

下面举几个简单的例子，以说明计算机算法。

2.1 2　算法举例

[例 2-1]　将两个变量 X 和 Y 的值互换。设 X＝5，Y＝10。

问题分析：两人交换座位，只要各自去坐对方的座位就行了，这是直接交换。一瓶酒和一瓶醋互换，就不能直接从一个瓶子往另一个瓶子倒。必须借助于一个空瓶子，先把酒倒入空瓶，再把醋倒入空的酒瓶，最后把酒倒入已倒空的醋瓶，这样才能实现酒和醋的交换，这是间接交换。

计算机中交换两个变量的值不能用两个变量直接交换的方法，而必须采取间接交换的方法。因此，需设一中间变量为 Z。

算法表示如下：

S1　将 Y 值存入中间变量 Z：

S2　将 X 值存入变量中：X→Y

S3　将中间变量 Z 的值存入 X 中，Z→X

这里 S1、S2…即 Step1，Step2…的简写，用以标识执行的步骤，以后的例中均用此符号，不再说明。用一个示意图说明此算法，如图 2-1 所示。

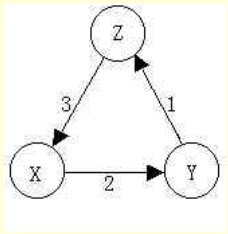


图 2-1　例 1 图

[例 2-2]　求 5 个自然数的和$\sum_{n=1}^{5} n$（即 S＝1＋2＋3＋4＋5）

问题分析：该题有两个特点：

（1）重复执行加法，每加一个数，总和的值都在变；

（2）加数是一个有规则的等差数列，第 1 项是 1，以后，每加一次，加数就增加 1。因此，可以用以下算法实现。

算法 1：

S1　设一存放累加和的变量 S，初值置 0，即 0→S；设一个数变量 N，初值 0，即 0→N。

S2　计算和 S＋N→S，并把计数变量增值 N＋1→N.

S3　判断：当 N≤5 时，返回第 2 步 S2，再次求和，当 N＞5 时，顺序执行下一步 S4。

S4　输出结果，S 为所求之和。

算法 2：

S1　0→S，1→N（同算法 1）

S2　判断：当 N≤5 时，顺序执行下一步 S3；当 N＞5 时，跳过 S3、S4，执行第 5 步 S5。

S3　S＋N→S，N＋1→N（同算法 1）

S4　返回第 2 步 S2。

S5　输出结果。（同算法 1）

在这个算法里，出现了重复求和的操作，称为循环，这种循环必须用条件加以限制，让它进行有限次数就停止，否则，成为死循环。上述算法 1 的 S3 是条件判断，求和的操作是先执行，后判断：算法 2 的 S2 是条件判断，求的和操作是先判断，再执行。

[例 2-3]　计算 N！(即求 1×2×3×…×N）的值。

问题分析：该题也有两个特点：(1）重复执行乘法的操作，每乘一次，积都在变；(2）乘数是一个有规则的等差级数，后项是前项加 1，因此，可用以下算法实现。

S1　指定一个具体的 N 值，如 5→N。

S2　设一累乘变量 M，初值置 1；设一计数变量 P，初值置 1。

S3　求积 M×P→M，计数变量 P 增值 P＋1→P。

S4　判断：当 P＜N 时，返回第 3 步 S3；否则，往下执行 S5。

S5　输出结果，即 M 为所求之和。

[例 2-4]　求两个自然数 M 和 N 的最大公约数。

问题分析：最大公约数就是能同时整除 M 和 N 和最大正整数，这是一个古老的算术问题，我国宋代数学家秦九韶，希腊数学家欧几里得（Euclid）提出了各自不同的算法。

算法 1：秦九韶算法

S1　输入两个自然数

S2　当 m≠n 时，顺序执行第 3 步，当 m=n 时，跳到第 5 步 S5。

S3　当 m＞n 时，m-n→m，否则 n-m→n

S4　返回第 2 步 S2。

S5　输出结果，n 为所求最大的公约数。

算法 2：欧几里得算法

S1　输入两个自然数

S2 求余数：M/N，得到余数 R（0＜R＜N）

S3 置换：N→M，R→N

S4 判断：当 R≠0，返回第 2 步 S2；当 R＝0 时，顺序执行第 5 步 S5。

S5 输出结果，n 为所求最大分约数。

综合上述算法示例看出，算法是解题方法的精确描述。算法并不给出问题的精确的解，只是说明怎样才能得到解。第一个算法都是由一系列的操作组成的。这些操作包括加、减、乘、除、判断、置数等，按顺序、分支、重复等结构组成。所以研究算法的目就是研究怎样把各种类型的问题的示解过程分解成一些基本的操作。

算法写好之后，要检查其正确性和完整性，再根据它编写出用某种高级语言表示的程序。程序设计的关键就在于设计出一个好的算法。所以，算法是程序设计的核心。

2.1.3 算法的特性

从上面的例子中，可以概括出算法的 5 个特性：

（1）算法中执行的步骤总是有限次数的，不能无休止地执行下去。这称之为有穷性。例如计算圆周率，可用如下公式：

$$\pi = (1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots)$$

这个多项式的项数是无穷的，它是一个计算方法，不是算法。我们要计算 π 的值，只能取有限个项数。例如，取精确到第 5 位，那么，这个计算就是有限次的，因而才能称得上算法。

（2）算法中的每一步操作的内容和顺序必须含义确切。这称之为确定性。不能有二义性。

（3）算法中的每一步操作都必须是可执行的，这称之为有效性。

（4）要有数据输入。算法中操作的对象是数据，因此，应在进行操作前提供数据。

（5）要有结果输出，算法的目的是用来解决一个给定的问题，因此，它应向人们提供算法产生的结果，否则，就没有意义了。

了解算法的这些特性，对我们在程序设计中构造一个好的算法，是有重要指导意义的。

2.2 用流程图表示算法

描述算法有多种不同的工具，采取不同描述算法的工具对算法的质量有很大的影响。如上节中例 1 至 4 是用自然语言——汉语描述的算法。使用自然语言描述算法的最大优点是，人们比较习惯，容易接受，但也确实存着很多的缺点，一是容易产生二义性，例如，中文“走火”二字，既可以表达子弹从枪膛射出，又可表示电线上漏电，也可表示说话漏了嘴。二是比较冗长，三是在算法中如果有分支或转移时，用文字表示就显得不够直观，四是目前计算机不便于处理，所以自然语言不适合描述算法。在计算机中常用流程图、结构化流程图、计算机程序设计语言等类型的描述工具来表示算法。

流程图（Flowchart），亦称框图，它是用一些几何框图，流向线和文字说明表示各种类型的操作。流程图中的基本图形、图形意义和长度比例都有国家颁布的标准（GB ISO5807-85）。如图 2-2 所示。

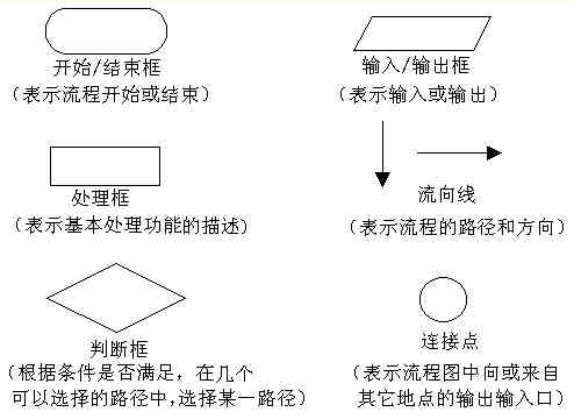


图 2-2 流程图的符号

处理框亦称矩形框有一个入口，一个出口。判断框亦称菱形框有一个入口，两个出口。在框内写上简明的文字或符号表示具体的操作，用带箭头和流向线表示操作的先后顺序。

流程图是人们交流算法设计的一种工具，不是输入给计算机的。只要逻辑正确，人们都能看得懂即可以了，一般是由上而下地按执行顺序画下来，下面是用流程图表示例 1～4 的算法。

从这 4 个例子看出，流程图的优点是形象直观，清晰明了，所以它很早就被广泛用在各种高级语言的程序设计中，常常称之为传流的流程图。

流程图可粗可细。粗流程图集中地概括了算法的总体逻辑结构，细流程图对算法的描述则比较细致而具体，其特点是详尽到足以直接凭借它就能用某种计算机语言顺畅地编写出相应的程序。

但是它也有不足之处，对于较复杂的问题，要详细地画出其过程，图形就显得特别长，占篇幅比较大。前后的流向线就不容易清楚表示。

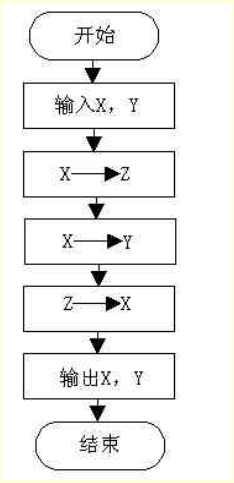


图 2-3　交换变量流程图

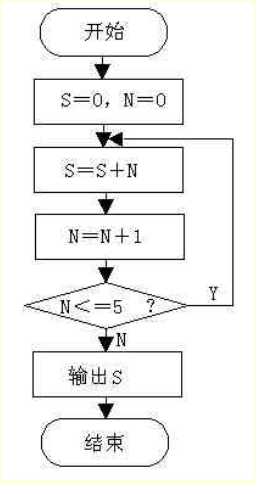


图 2-4　求 5 个自然数和流程图

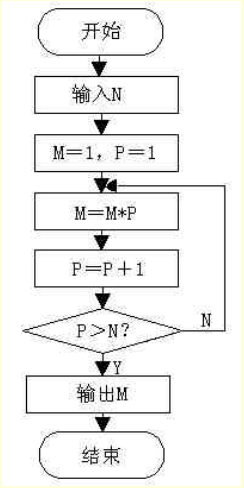


图 2-5　求 N! 流程图

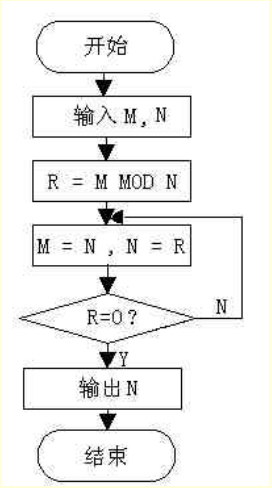


图 2-6　求最大公约数流程图

2.3　用结构化流程图（N-S 图）表示算法

2.3.1　结构化程序的三种基本结构

1．什么是结构化程序

随着计算机的发展，编制的程序越来越复杂。一个复杂程序多达数千条语句，而且程序的流向也很复杂，

常常用无条件转向语句去实现复杂的逻辑判断功能。因而造成质量差，可靠性很难保证，程序也不易阅读，维护困难，20 世纪 60 年代末期，国际上出现了所谓“软件危机”。

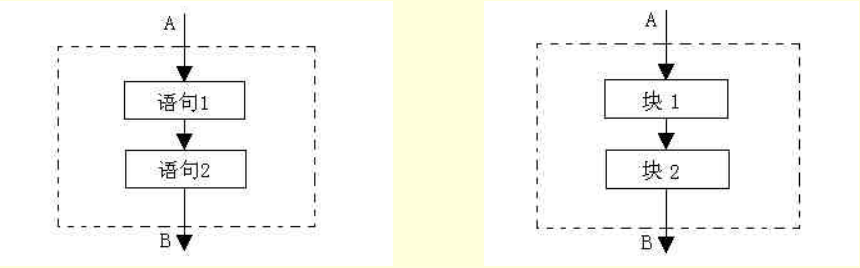
为了解决定这一问题，就出现了结构化程序设计，它的基本思想是象玩积木游戏那样，只要有几种简单类型的结构，可以构成任意复杂的程序。这样可使程序设计规范化，便于用工程的方法来进行软件生产。基于这样的思想，1996 年意大利的 Bobra 和 Jacopini 提出了 3 种基本结构，由这 3 种基本结构组成的程序，就是结构化程序（Structured Program）。

2. 三种基本结构

（1）顺序结构

程序流程是沿着一个方向进行的有一个入口（A）、有一个出口（B），它包含两种情况：

- 简单的顺序结构，由程序的基本单元——语句组成的块，它没有分支，如图 2-7（a）所示。
- 复合的顺序结构，它可以由若干基本结构的程序块组成，各块之间的关系是顺序走向的，没有分支的（这时，各块内可能有分支的情况）。如图 2-7（b）所示。



（a） （b）

图 2-7 顺序结构

（2）选择结构

程序的流程发生分支，根据一定的条件选择其中之一，这就是选择结构。它也有一个入口（A），一个出口（B）。简单的选择结构是只有两个分支的情况，如图 2-8 所示。

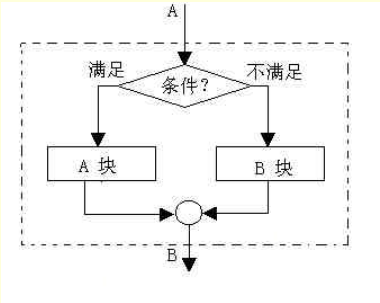


图 2-8 选择结构

（3）循环结构（亦称重复结构）

程序流程是有限次重复执行某一块程序。再退出循环。循环结构有两种类型：

- 当型（WHILE）循环结构，先判断条件是否满足，若满足时就执行循环体，如条件不满足时就不执行循环体，并转到出口。如图 2-9（a）。
- 直到型（UNTIL）循环结构，它是后执行，先判断。当条件不满足时继续执行循环体；当条件满足时，停止执行，并转到出口。如图 2-9（b）。

循环结构也有一个入口（A），一个出口（B），它应当使重复处理为有限次，不能无限制地下去。

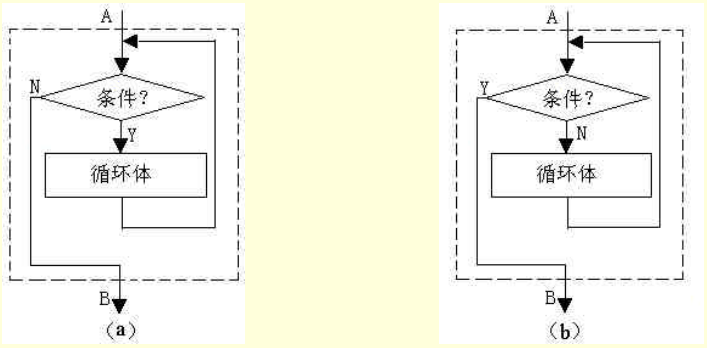


图 2-9　循环结构

这种结构化程序具有以下特点：

- 只有一个入口和一个出口。
- 无死语句，既没有永远都执行不到的语句。
- 无死循环。即永远执行不完的循环。

2.3.2　结构化流程图（N-S 图）

前面的图 2-7、图 2-8 和图 2-9，用框图表示了结构化程序的 3 种基本结构。看起来还清楚，但情况复杂时，图形中的流向线多，仍不便于阅读。所以美国人 I.纳斯（I.Nassi）和 B.施内德曼（B.Schneiderman）提出了一种的绘制流程图的方法。由于他二人的名字以 N 和 S 开头，故把这种流程图称为 N-S 图。这种结构化流程图，完全去掉了在描述中引起混乱的带箭头的流向线，全部算法由 3 种基本结构的框表示。

3 种基本结构框的画法规定如下：

（1）顺序结构（Sequence Structure）

它由若干个前后衔接的矩形块顺序组成。一个顺序结构，如图 2-10，先执行块 A，然后执行块 B。各块中的内容表示一条或若干条需要顺序执行的操作。

（2）选择结构（Choice Structure）

见图 2-11 所示，在此结构内有两个分支，它表示当给定的条件满足时 A 块的操作，条件不满足时，执行 B 块的操作。

图 2-10　顺序结构流程图

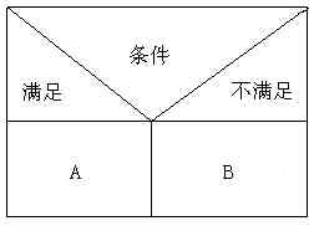


图 2-11　选择结构流程图

（3）循环结构（Repetition Structure）：

① 当型（WHILE）循环结构，见图 2-12（a）。先判断条件是否满足，若满足就 A 块（循环体），然后再返回判断条件是否满足，如满足 A 块，如此循环执运，直到条件不满足为止。

② 直到型（UNTIL 型）循环结构。见图 2-12（b）。它先 A 块（循环体）然后判断条件是否满足，如不满足则返回再 A 块，若满足则不再继续执行循环体了。

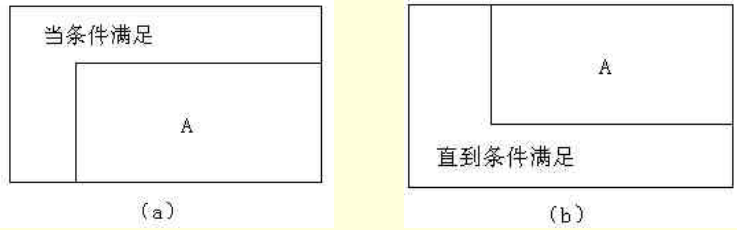


图 2-12　循环结构

用 N-S 图表示 1 至例 4 中算法，见图 2-13 至图 2-16。

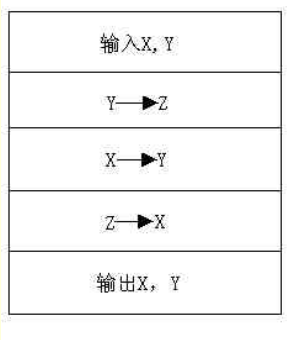


图 2-13　交换变量

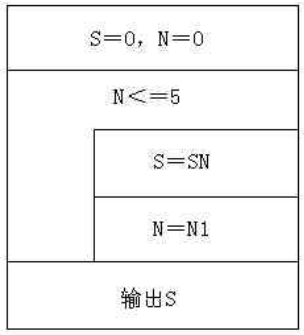


图 2-14　求 5 个自然数的和

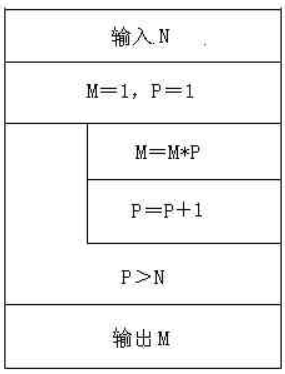


图 2-15　求 N！

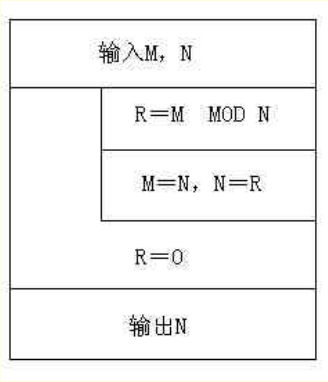


图 2-16　求最大公约数

将图 2-13 至图 2-16 的 N-S 图与图 2-3 到图 2-6 的传统流程图相比较，从中明显看出：N-S 流程图是由基本结构单元组成的，各基本结构单元之间是顺序执行关系，即从上到下，一个结构一个结构地顺序执行下来。这样的程序结构，对于任何复杂的问题，都可以很方便地用以上 3 种基本结构顺序地构成。因而它描述的算法是结构化的，这是 N-S 图的最大优点。

和 N-S 图表示算法，思路清晰，阅读起来直观、明确、容易理解，大大方便了人们对结构化程序的设计，并能有效地提高算法设计的质量和效率。对初学者来说，使用 N-S 图还能培养良好的程序设计风格，因此本书在表示算法时，主要地采用了这种 N-S 图。

2.4　结构化程序设计步骤

在学习编写程序之前，对结构化程序设计的全过程有个全面的了解，从中可以知道用机器语言编写程序在全过程中的地位，这对培养和提高程序设计的能力很有好处。

完成一个正确的程序设计任务，一般可分为以下几个步骤进行，如图 2-17 所示。

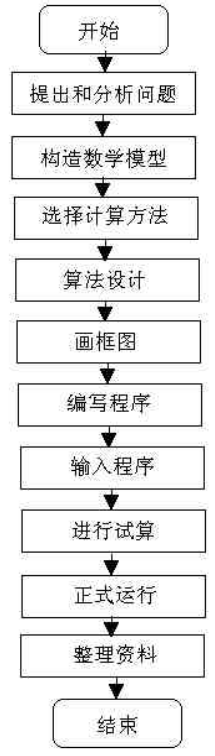


图 2-17　设计程序步骤

（1）提出和分析问题

即弄清提出任务的性质和具体要求。例如，提供什么数据，得到什么结果，打印什么格式，允许多大误差，都要确定。若没有详细而确切的了解，匆忙动手编程序，就会出现许多错误，造成无谓的返工或损失。

（2）构造模型

即把工程或工作中实际的物理过程，经过简化，构成物理模型，然后，用数学语言来描述它，这称为建立数学模型。

（3）选择计算法

即选择用计算机求解该数学模型的近似方法。不同的数学模型，往往要进行一定的近似处理。对于非数值计算则要考虑数据结构等问题。

（4）算法设计

即制订出计算机运算的全部步骤。它影响运算结果的正确性和运行效率的高低。

（5）画流程图

即用结构化流程图把算法形象地表示出来。

（6）编写程序

即根据流程图用一种高级语言把算法的步骤写出来，就构成了高级语言源程序。

（7）输入程序

即将编好的源程序，通过计算机的输入设备送入计算机的内存存贮中。

（8）进行试算（调试）

即用简单的、容易验证结果正确性的所谓“试验数据”输入到计算机中，经过执行、修改错误、再执行的反复过程，直到得出正确的结果为止。

（9）正式运行

即输入正式的数据，以得到预期的输出结果。

（10）整理资料

即写出一份技术报告或程序说明书，以便作为资料交流或保存。

2.5 本章小结

本意主要介绍了算法的概念和表示方法。

（1）算法就是为解决一个问题而采取的方法和步骤，程序设计的关键就是选择正确的、高效率的算法。计算机中的算法必须是计算机能执行的、无二义性、有限次执行后能停止的。

（2）算法的表示方法有很多种，本章仅介绍了传流程图 N-S 结构化流程图两种。在程序设计中，应重视用流程图表达算法。

（3）学习程序设计的过程，就是不断地讨论算法、积累算法，并用流程图和程序设计语言描述算法的过程，因此，在以后的学习中，应着重对问题进行分析，明确解题的思路和方法。

习题

1．什么是算法？它有何特征？为什么要研究它？

2．结构化程序设计有哪 3 种基本结构？用 3 种基本结构进行程序设计有何好处？

3．对下列各题写出算法，并画出 N－S 流程图。

（1）求 n 个数的平均值。

（2）求 1 至 100 之间全部奇数之和。

（3）求 $1+\frac{1}{3}+\frac{1}{5}+\cdots+\frac{1}{99}$ 之和。

（4）有 10 个数：$a_1,a_2,\cdots\cdots a_{10}$,找出其中的最大数。

第三章 Quick Basic 程序的运行环境

与其他高级语言不同，学习 Quick Basic 不是从程序设计开始，而是首先要熟悉它的编程环境。Quick Basic 把程序的编辑、编译、连接、调试等步骤集成在一个视窗环境中，并具有较强的联机帮助。编程集成环境是 Quick Basic 的重要组成部分，也是区别于其他高级语言的一个特点，我们只有在初步熟悉编程环境之后，才能进一步学习程序设计。

在本章中，我们将学习到：

- Quick Basic 的启动和退出
- Quick Basic 的菜单的使用功能

本章的重点是熟悉 Quick Basic 的编程环境。这是基础性内容，一定要熟练掌握。

3.1 Quick Basic 的启动和退出

3.1.1 Quick Basic 的启动

如果用户安装的是 DOS 磁盘操作系统的话，在确认磁盘上安装有 Quick Basic 后，并设置好路径，用户就可以在硬盘或软盘中的工作目录下启动 Quick Basic。启动的格式很简单，只要在进入 Quick Basic 所在的子目录后键入：

>QB↙

回车后就进入了 Quick Basic 的初始画面后，按下 Esc 键就进入了 Quick Basic 的编辑界面，用户可以在 Quick Basic 的编辑视窗中完成新建源程序、调入源程序、编辑、调试和运行等操作。

如果用户安装的是 Windows 95 或 Windows 98 的多任务窗口操作系统，可直接运行 Quick Basic 的编辑界面。

3.1.2 Quick Basic 的退出

Quick Basic 的退出也很简单，可以使用 Alt+F 组合键激活菜单条，在 File 菜单中选择 Exit 命令就会退出 Quick Basic，返回 DOS 或 Windows。当然，用户也可以使用鼠标点出 File 菜单中的 Exit 退出 Quick Basic。

3.2 Quick Basic 的功能菜单操作

从图 3-1 中的编辑界面中，我们可以看到 Quick Basic 的主菜单包含 8 个菜单项。下面逐一作简要介绍。

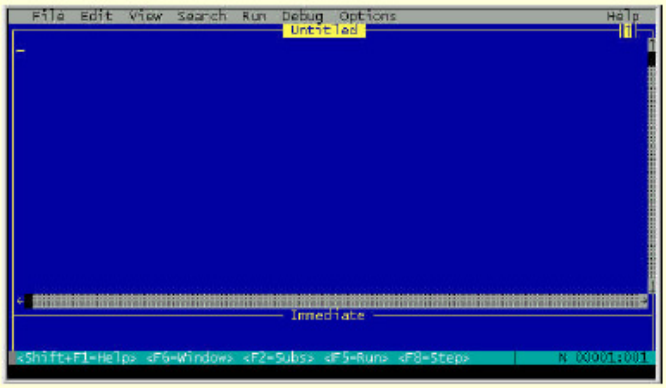


图 3-1 Quick Basic 的编辑界面

3.2.1 File 菜单命令介绍

当我们激活 Quick Basic 主菜单的 File 菜单标题时，就会弹出 File 菜单标题的菜单条。File 菜单中包括 New program、Open program、Save As、Print 和 Exit 等菜单项，如图 3-2 所示。

图 3-2 File 菜单

（1）New program

该菜单项命令的功能是编辑一个新文件，文件名预置为 Untitled。如果在编辑了其他文件后再选择此菜单项，系统将把原来编辑的文件卸掉，刷新编辑窗口，将文件名重置为 Untitled。如果执行该菜单项命令，而前面编辑的文件没有存盘，系统将会弹出提示“File not save.Save(Y/N)?”询问用户是否存盘，确定后将自动进行相应的操作。

（2）Open programe

该菜单项命令的功能是打开一个已经存在的 Quick Basic 源程序文件。选择该选项将会弹出一个对话框如图 3-3 所示，用户可以在对话框中使用 Tab 键或鼠标，选择用户需要打开的文件的路径。

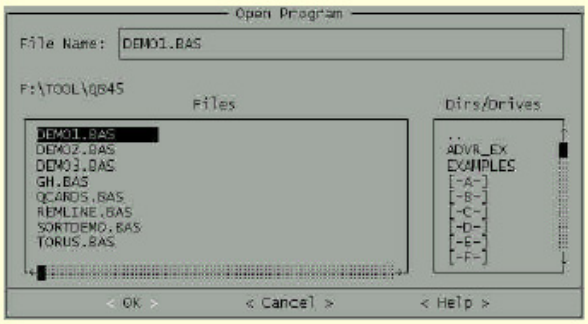


图 3-3 Open programe 对话框

（3）Save As

该菜单命令的功能是把当前文件另存为一个副本到指定的磁盘中。

（4）Print

该菜单项命令的功能是将当前的源程序文件在打印机上打印出来。选择该命令，屏幕将弹出一个对话框如图 3-4 所示。用户可以根据需要在对话框中选择不同的打印方式，打印方式有 4 种：Selected Text（选中模块）、Acitve Window（活动窗口）、Current Windows（当前窗口）。

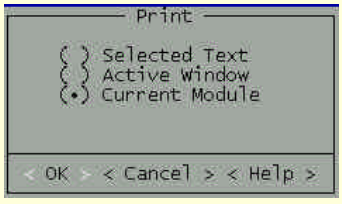


图 3-4 选择打印方式

（5）Exit

该菜单项命令的功能是退出 Quick Basic。

3.2.2 Edit 菜单命令介绍

Edit 菜单标题中包括 Cut、Copy、Paste 等选项，如图 3-5 所示。下面介绍它们的功能。

图 3-5 Edit 菜单

（1）Cut

剪切选中的文本，并将剪切的文本保存到文本缓冲区（剪切板）中。

（2）Copy

将选中的文本拷贝到剪切板中。

（3）Paste

将剪切板中保存的文本粘贴到光标当前位置。

3.2.3 View 菜单命令介绍

View 菜单标题命令或以完成观察程序和模块的操作。包括 SUBs、Output Screen、Incuded Lines 等命令，如图 3-6 所示。

图 3-6 View 菜单

（1）SUBs

选择该命令选项将弹出一个对话框，该对话框中显示了程序所包含的所有模块。

（2）Output Screen

该命令选项的功能是暂时屏蔽 Quick Basic，显示当前输出的屏幕内容。

（3）Included Lines

该命令选项的功能是显示或隐藏标尺。

3.2.4 Search 菜单命令介绍

Serch 菜单标题命令能够完成字符或字符串的查找和替换操作。包括 Find、Change 等命令。

（1）Find

选择此命令将弹出 Find 对话框，如图 3-7 所示。用户可以在上面的文本框中键入需要查找的字符或字符串。

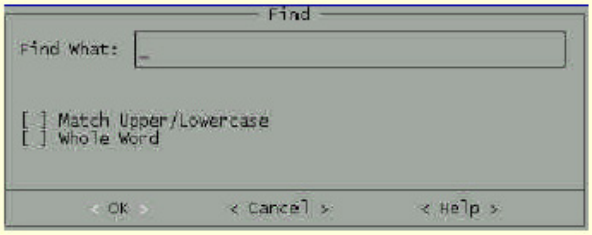


图 3-7 Find 对话框

（2）Change

在查找到相关内容后，选择该命令将弹出一个对话框，在对话框中用户可以在下面的文本框中键入需要替换的内容来替换查找的内容。

3.2.5 Run 菜单命令介绍

Run 菜单标题命令为用户提供了各种执行程序的方法，它包含 Start（快捷键 Shift+F5）、Restart、Continue（快捷键 F5）、Make EXE Flie，如图 3-8 所示。

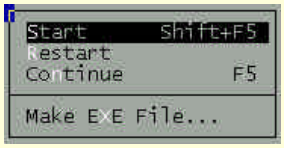


图 3-8 Run 菜单命令

（1）Start

该命令的功能是执行正在编辑的程序并检查执行情况。在程序的执行过程中，如果程序有错误，系统将中断执行，返回到编辑窗口，此时光标停在出错语句上并指出错误号。用户可以根据提示修改错误，直到没有错误时为止；如果用户在程序中设置了断点，系统将会分段执行程序。

（2）Restart

该命令选项的功能是将程序中的变量重新设置为 0 或空字符串，重执行正在编辑的程序。

（3）Continue

该命令选项可以从上一次执行的断点处继续执行正在编辑的程序。

（4)Make EXE Flie

该命令选项可用于生成可执行文件。

3.2.6 Debug 菜单命令介绍

该菜单标题包含的选项可以完成 Quick Basic 程序的调试工作。我们可以在程序中设置断点来检查程序运行到此时的状态和各变量的值。该菜单标题包含的选项有 Add Watch、Instant Watch、Delete Watch、Toggle Breakpoint、Clear All Breakpoints 等，如图 3-9 所示。

图 3-9 Debug 菜单

（1）Add Watch

该命令选项用于在程序调试过程中，增设一个观察窗口，用于观察变量和表达式的值。

（2）Instant Watch

该命令选项用于在程序调试过程中，增设一个即时观察窗口，用于动态观察变量和表达式的值。

（3）Delete Watch

用于清除所有观察窗口。

（4）Toggle Breakpoint

这是一个管理断点的命令，用户可以使用该命令来完成设置或取消断点的操作。我们在设置断点时，一般把断点设置在控制流程中需要挂起的地方。

（5）Clear All Breakpoints

该命令选项的功能是清除原来设置的所有断点。

3.2.7 Option 菜单命令介绍

选择该菜单标题中的选项可以完成设置屏幕状态、设置帮助路径，或是否打开语法检查器。它包含 Display、Help Path 和 Syntax Checking 等选项，如图 3-10 所示

图 3-10 Option 菜单

（1）Display

选择该命令选项将显示颜色设置对话框，如图 3-11 所示。在该对话中，用户可以完成对屏幕颜色状态的设置。选择左边的单项按钮选项可以完成不同选项的前景和背景设置，其中有包括正文（Normal Text）、当前语句（Current Statement）和断点行（Breakpoint Lines）的前景与背景的设置。用户还可以在该对话框中确定是否选择滚动条（Scroll Bars）等。

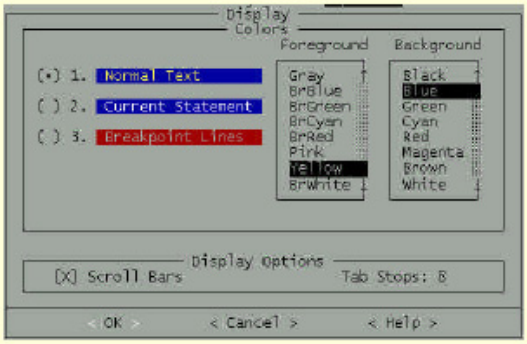


图 3-11　设置屏幕状态

（2）Set Path

选择该命令选项将弹出 Set Path 对话框，如图 3-12 所示。

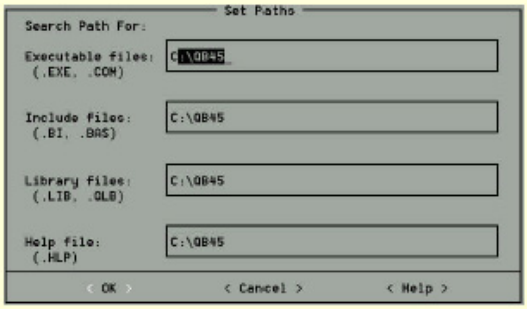


图 3-12　Set Path 对话框

用户可以在该对话框上面的文本框中输入*.EXE、*.COM、*.BI、*.BAS、*.LIB、*.QLB、*.HLP 文件所在的路径来完成路径的设置。当然，如果用户的文件就在默认的目录下，则不必对此项进行设置。

（3）Full Menus

该选项用于设置菜单的显示或隐藏。

3.2.8　Help 菜单命令介绍

Help 菜单标题为用户提供了各种联机帮助，用户可以根据不同的需要选择自己想要查询的帮助信息。该菜单标题包含 Index、Contents、Topic、Help on Help 等选项，如图 3-13 所示。

图 3-13　Help 菜单

（1）Index

选择此选项将弹出一个按照英文字母顺序排列的 Quick Basic 语言帮助信息索引表，如图 3-14 所示。其中包含 Quick Basic 的所有的关键字。用户可以首先按下某个关键字的第一个字母键，用方向键或鼠标选定想得到帮助的关键字上，按 F1 或回车或双击鼠标左键，屏幕将显示该关键字的主要功能、语法和实例。

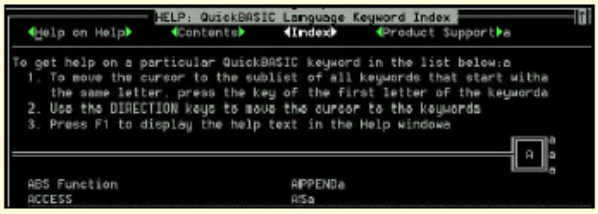


图 3-14　Quick Basic 语言帮助信息索引表

（2）Contents

选择该命令选项系统将弹出 Quick Basic 的详细帮助功能内容的目录列表，如图 3-15 所示。用户需要哪些帮助，只要用方向键或鼠标选定想要得到帮助的字段上，按 F1 或回车或双击鼠标左键，屏幕将显示相关的帮助信息。

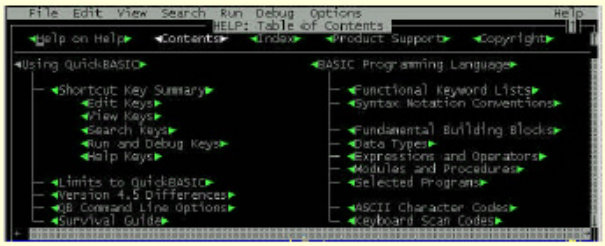


图 3-15　帮助目录列表

（3）Topic

该命令选项提供了 Quick Basic 所有关键字的帮助。

（4）Help On Help

帮助菜单的帮助，用于说明怎样使用菜单。

3.3　本章小结

通过本章的学习，读者应该熟练地掌握在不同系统环境中如何启动和退出 Quick Basic，最主要的是熟悉 Quick Basic 的编程环境。

习　题

按照本章讲述的方法，启动 Quick BASIC 进入编辑界面，熟悉 Quick BASIC 运行环境下的各个菜单标题中常用命令选项的功能。

第四章 Quick BASIC 的基本概念

上一章介绍了 Quick BASIC 语言的运行环境。在本章中将学习到：

- Quick BASIC 的程序结构
- Quick BASIC 的数据类型
- 常量和变量的概念
- 变量的命令规则
- 变量的类型说明
- 运算符和运算规则
- 表达式
- Quick BASIC 的标准函数

本章的重点讲述 Quick BASIC 语言的数据类型，常量和变量的概念，变量的命名规则，变量的类型说明，运算符和运算规则中的算术运算、关系运算、逻辑运算、字符运算，表达式中的算术表达式、关系表达式、逻辑表达式、字符表达式及求值，以及常用的标准函数等内容，逐一分析、理解和掌握。本章的难点在各种变量之间的关系、运算和表达式部分。

4.1 Quick BASIC 的程序结构

Quick BASIC 的程序结构是模块化结构。在 Quick BASIC 程序中，模块是可以独立编译的源文件，在模块中可以包含说明语句、可执行的语句和其他 Quick BASIC 语句。一个模块中的代码可以分为两极，过程（Procendure）之外的语句称为模块级代码，过程中的语句称过程级代码，这些代码通常以 SUB 或 FUNCTION 开头，以 END SUB 或 END FUNCTION 结束。一个 Quick BASIC 程序调入内存可以包含一个或多个模块，这样就组成了一个程序。模块中的过程（Procedure）可以被其他程序调用。下面介绍一下 Quick BASIC 源程序的模块化结构和程序设计语法规范。

4.1.1 Quick BASIC 源程序结构

我们首先通过一个简单的实例来介绍一下 Quick BASIC 程序的模块化结构。

［例 4-1］ 计算表达式 b=4+3×a 的值。

```
REM 计算表达式的值
INPUT a
b=4+3*a
PRINT "b";b
END
```

程序的第一行为注释语句，用户可以通过它对程序作简短的解释说明；第二行为输入语句，用于把输入的数值送给变量；第三行为赋值语句，计算出赋值号右边表达式的值后将计算结果赋给左边的变量；第四行为输出语句，用于此变量的值从外部设备输出；END 为结束语句，用于终止程序的执行。

在程序中，最基本的成分是字符。从上面这个程序，我们可以知道 Quick BASIC 程序由若干个程序行组成，每行由相关的语句组成，语句由特定的关键字组成，即“ 语句定义符 ”。有关字符集、程序行和语句的有关内容，我们将在 4.1.2 小节中介绍。

4.1.2 Quick BASIC 程序设计语法

Quick BASIC 所使用的字符集与其他语言大致相同，共用 90 个，可以分为字母、数字和专用字符三类。

- 字母：大写英文字母 A~Z，小写英文字母 a~z。
- 数字：0~9，字母 A~F 和 a~f 也可以作为十六进制数字的一部分。
- 专用字符：共 28 个，包括表 4-1 所列字符。

表 4-1 专用字符列表

字 符	含 义	字 符	含 义
＜CR＞	回车键	.	句号（小数点）
	空格	/	斜杠
!	叹号（单精度类型数据说明符）	：	冒号
#	镑号（双精度数据类型说明符）	；	分号
$	美元符号（字符串数据类型说明符）	＜	小于号
%	百分号（整型数据类型说明符）	＝	等于号
&	和号（长整型数据类型说明符）	＞	大于号
'	单引号（撇号）	？	问号
（	左圆括号	@	At 号
）	右圆括号	［	左方括号
*	星号（乘号）	］	右方括号
+	加号	\	反斜杠（整除号）
，	逗号	^	向上箭头（乘方）
-	减号	_	下划线（续行符）

Quick BASIC 的程序行一般格式为：

［行标识符］［语句］［：语句］…［语句］［'注释］

如果使用 Quick BASIC 内置的编辑程序来输入源程序，则每个程序行不能超过 256 个字符，而且不能把下划线作为行接续符。如果使用的其他编辑程序，则可以使用下划线作为接续符，把一个程序行放到几个物理行中。例如：

```
IF（Student $ = ""OR Student $ = "."AND Linenumber<18_
AND NOT EOF(Filenumber)THEN_
……
```

在装入程序时，Quick BASIC 将会自动去掉下划线，把各接续行连成一行，从而放宽行的长度。但要注意不能用下划线接续 REM 或 DATA 语句。

Quick BASIC 行标识可以分为两种类型：行号和行标号。

- 行号：Quick BASIC 的行号从 0 到 65529 的整数。虽然 0 可以作为有效的行号，但在有些情况下可能会产生错误，所以，最好不要用 0 作行号。
- 行标号（标号）：行标号由字母和数字组成，最多不超过 40 个字符。它必须以字母开头并以冒号结束，不能用 Quick BASIC 保留字作为行标号。下面的行标号有效：

```
ScreenSub:
Test2B:
A1：
```

此外，一行只能有一个行标号。在一个源程序中，行号和标号可以混合使用。用户也可以用标号代替原来程序中的目标行号。但是在 IF…THEN 语句中，除非明确地使用 GOTO 语句，否则只能用行号作为转移目标。例如：

```
IF A＝0 THEN 100
IF A＝0 THEN GOTO InData
```

是合法的。

在 BASIC 及其他版本中，行号决定语句执行的先后顺序，而在 Quick BASIC 中并非如此，例如：

```
50 PRINT"Welcome"
20 PRINT"Glad to meet you"
```

```
10 PRINT"See you tomorrow"
```

在其他 BASIC 版本中执行顺序为 10、20、50，而在 Quick BASIC 中执行顺序为 50、20、10。

Quick BASIC 中的语句可以分为两类：执行语句和非法行语句。执行语句可以完成某中指定的操作，例如：读、写、计算、赋值、打开文件、转移控制等，它可以把程序的逻辑流程向前推进；非执行语句只是执行诸如为变量分配存储单元、说明和定义变量类型、指明源文件中被所有过程共享的变量等任务。在 Quick BASIC 中，非执行语句包括如表 4-2 所示语句。

程序行最后的注释是非执行语句，用来说明程序的操作和使用。注意，注释只能以单引号或 REM 开头。

表 4-2 Quick BASIC 非执行语句

REM 或'（注释开始，可以含有任何字符）	DIM（常量用于定义静态数组）
COMMON	OPTION BASE
CONST	SHARED
REDIM（重新定义动态数组大小）	DIM（变量用于定义动态数组）
DATA	STATIC
DECLARE	TYPE…END TYPE
DEF type	

4.2 Quick BASIC 中的数据和数据类型

与其他 BASIC 版本不同，Quick BASIC 不但有系统定义的基本数据类型，而且用户还可以自己根据需要定义数据类型。我们首先介绍一下数据的概念，然后再介绍基本数据类型和如何自定义数据类型。

4.2.1 数据概念

数据的概念虽然有各种说法，但都大同小异。数据是信息的载体，是描述客观事物的数、字符，以及所有能输入到计算中，并能被计算机程序加工处理的符合集合。计算机中的数据，并不单单指数值，也包括字符串等。

4.2.2 数据类型

数据类型是简单数据的基本属性，数据类型是确定的一个值的集合。计算机中，只有相同或相容的类型的数据之间才能进行操作。

1．基本数据类型

基本数据类型可以分为字符串型数据和数值型数据。

（1）字符串

变长字符串　变长字符串的长度为 0~32767 个字符。组成它的字符可以是代码为 0~127 的标准 ASCII 字符，也可以是代码为 128~255 的扩展 ASCII 字符。（参见附录 C）

定长字符串　定长字符串含有确定个数的字符，但最大长度不超过 32767 个字符。定长字符串中字符的代码范围也为 0~255。

（2）数值

Quick BASIC 的数值型数据可以分为整型数和浮点数两类。其中整型数分为常规整型数和长整型数，浮点数分为单精度浮点数和双精度浮点数。下面分别予以介绍：

整型数：整型数是不带小数点和指数符号的数。在计算机内部，它以二进制补码形式表示。例如，整数 8 和-8 分别以下在的形式存放：

```
0000000000001000
1111111111111000
```

整型数中，整型（常规整型）在计算机内部，以两个字节（16 位）的二进制码表示和参加运算，它的取值范围从-32768 到+32767。在 Quick BASIC 中，增加了长整型数。长整型数以带符号的四字节（32 位）二进制存储，它的取值范围从-2147473648 到+214743647。

浮点数：Quick BASIC 使用 IEEE（Institute of Electrical and Electoronice Engineers,Inc）格式的浮点数，而不

是像早期版本那样使用 Microsoft 二进制格式。IEEE 格式给出的结果更精确，并能使用 8087 或 80287 数学协处理器。

浮点数由三部分组成：符号、指数和尾数，其指数分别用 E 和 D 表示。其中的单精度浮点数占 4 个字节，符号占 1 位，指数占 8 位，其余的 23 位表示尾数，此外还有一个附加的隐含位。单精度数可以精确到 7 位十进制数，其负数的范围为-3.402823E+38 到-14.40129E-45，正数的范围为 1.40129E-45 到 3.402823+38。双精度浮点数占 8 个字节，符号占 1 位，指数占 11 位，其余 52 位用来表示尾数，此外还有一个附加的隐含位。双精度数精确到 15 或 16 位十进制数。其负数的范围为-1.797693134862316D+308。除去整型数和长整型数外，Quick BASIC 中还使用十六进制和八制数，还要用到逻辑数据。逻辑型数据只取“真”和“假”两种值，通常情况下把逻辑型数据也看成数值型数据。

2．用户定义的数据类型

除了系统定义的基本数据类型外，用户还可以利用 TYPE 语句定义自己的数据类型，它的格式为：

```
TYPE 定义的数据类型名
数据类型元素名 AS 类型名
数据类型元素名 AS 类型名
……
END TYPE
```

其中的“定义的数据类型名”的命名规则与变量的命名规则相同，有关变量内容我们将在 4.3 节中介绍。“数据类型元素名”也遵守同样的规则，注意，数据类型元素名不能为数组名。

“类型名”可以是除变长字符串以外任何数据类型，也可以是用户定义的类型。

［例 3-2］　定义日期（MyData）记录数据类型。

```
TYPE MyData
Year AS INTEGER
Month AS STRING*3
day AS INTEGER
END TYPE
DIM today AS mydate
today.year = 2001
today.month = "April"
today.day = 18
PRINT "today is";today.year;today.month;today.day
END
```

运行菜单选项"RUN"，输出结果：

```
today is 2001 Apr 18
```

其中的 MYDATE 为用户自定义的类型。它包含三个元素：Year、Month 和 Day。Year 和 Day 为整型，Month 为定长字符串，取 3 个 ASCII 字符。

我们在使用 TYPE 语句时，应注意以下几点：

- 用户自定义类型中的字符串必须是定长字符串，它的长度用一个星号和一个常数指明。
- 在使用用户自定义的类型之前，必须用 TYPE 语句在模块代码中加以定义。而用户自定义类型的变量可以出现在模块的任意地方，包括在 SUB 或 FUNCTION 过程中。
- 用户自定义的类型中不能含有数组。

4.3　Quick BASIC 中的常量和变量

使用计算机能够实现的运算可以分为两类：数值运算和非数值运算。数值运算的运算对象主要是数值量，当然，得到的结果也为数值量，比如 3*2+4 属于数值运算。非数值运算的运算对象主要不是数值量，而是非数

值量，比如把全班同学的姓名按姓氏的英文字母顺序排序；查询 3 月 8 日出生的女士；打印全班前十名同学的姓名等，都属于非数值运算。程序处理的对象是数据，数据可以分为数值型数据和非数值型数据，参加运算的数据就是运算量或操作数。本节主要介绍常量和变量。

4.3.1 常量

常量是指程序运行过程中不能改变的数据量。如 3.4，78，-1 等都是常量。在 Quick BASIC 中，常量可以分为数值常量和字符串常量。有时也把常量分为文字常量和符号常量。

1．数值常量

数值常量又可分为整型常量和实型常量两类。

（1）整型常量（又称为整数）

整型常量有若干个数字组成的序列，它可以带有数值符。整型常量又可分为常规整型数和长整型数两类。

常规整型数：常规整型数的取值范围从-32768 到+32767。它不包含小数点，常规整型数按两个字节存储。Quick BASIC 中用“%”作为常规型数的符号，比如 76、76%指的是同一个整型数。但要注意，如果我们所取的数超过了规定的范围，如果仍然在数值后加“%”，则 Quick BASIC 会出错。如 32768%不合法。

长整型数：长整型数是指超过-32768 到+32767 范围而不超过-2147483648 到+214783647 范围内不带小数点的整数。在 Quick BASIC 中，长整型数可以数的末尾加一个“&”表明是长整型，比如 32777 和 32777&指同一个数。所有被认为或被定义为长整型的整数，不管大小均按 4 个字节存储。比如 8&，虽然数值很少，因为被定义为长整型数，Quick BASIC 仍然会按 4 个字节存储。

（2）实型常量（又称实数）

实型常量可以分为单精度实数和双精度实数，在 Quick BASIC 中，有两种计数方法可计实数，它们是日常计数法和科学计数法。

日常计数法：日常计数法是指日常生活中所使用的十进制小数形式计数。它们由若干个数字序列组成，可以包括小数点和数值符号。比如-3.4，0.786，0.0，+9.485 等。因为这些实数的小数点的位置是固定不变的，所以又被称为“定点实数”。比如 743.45 中的小数点的位置在 3 后面，我们不能随便移动它的位置，因为移动它将会改变它实际表示的数值。

科学计数法：科学计数法就是用指数的形式来表示实数。在数学中，我们常用幂（幂的底数为 10）的形式表示，如 743.45 可表示为 743.45×10^{0}、7.4345×10^{2}、74.345×10^{1} 等数值常量。因在为在输入这些数据时，无法输入上角标和下角标，所以 Quick BASIC 规定用字母“E”或“e”来表示以 10 为底的幂数。比如上例可表示为 743.45E0、7.4345E2、74.345E1。

在 Quick BASIC 中，一个指数形式的常量是有数字部分和幂（指数）部分组成时，一个实数可以表示为多种指数形式，如上例的 743.45E0，还可以表示成 7.4345E2、74.345E1 等形式。因为用指数形式表示一个实数时，其实数中的数字部分内的小数点的位置可以随着指数的变化而变化，故又称“浮点型实数”。

在 Quick BASIC 中，为了区别单精度实数和双精度实数，又增加了有关单精度和双精度实数的特殊表示方法：单精度数用末尾带有“!”的十进制定点数表示，也可以用“E”代替指数的底。比如 43.21 可以表示为 43.21！或 4.321E1，二者等价。因为我们经常在程序中使用单精度数，为了简化起见，实数中的“!”可以省略不写，上例的 43.21！和 43.21 是等价的；双精度实数用末尾带“#”的十进制定点数来表示，也可用“D”代表指数的底，比如双精度数 876.098098098 可以表示为 876.098098098#或 8.76098098098D2。其中的“D”和科学计数法中的“E”等价，只是为了区别起见才这样写。

注意：当我们用指数形式表示一个实数时，指数部分由字母“E”、指数符号（正数可以默认）和不超过 3 位的整数组成。比如 1234.567E-3、5879.55E3、787.23E2 都是合法的，而 1234.567E-3.2、5879.55E-4321、787.23E22.3 均不合法。

计算机中，不同类型的数的存储方式不同，前面我们提到了不同类型的数所占的字节数不一样，经如 20、20.00、20.00#数值虽然相同，但 20 为整型数，而 20.00 为实型数，20.00#为双精度数。故在存储时所占的存储空间就不一样。20 在存储时占用 2 个字节（整型数存储占 2 个字节 16 位），而 20.00 在存储时计算机会当作单精度实型数处理，存储占 4 个字节，20.00#在存储时计算机会当作双精度数，存储占 8 个字节。

程序中若没有特殊说明某实型数是单精度还是双精度数时，可以参照以下原则加以区别：

不超过 7 位并且带小数点的数，或末尾加“ ！”的不超过 7 位数字的十进制小数，或用“ E ”（或“ e ”）表示指数部分的浮点实数，在计算机中均以 4 个字节按单精度实型数存储。

多于 7 位并且包含小数点的数，或末尾加“ # ”的实数，或用“ D ”（或“ d ”）表示指数部分的浮点实数，在计算机中均以 8 个字节按双精度实型数存储。

2．字符串常量

字符串常量是指用一对双引号括起来，包括所有可以打印的 ASCII 字符和国标码字符的字符串。如"OK"、"Quick BASIC"、"Welcome"等都是合法的字符串常量。应当注意的是：双引号只是字符串的“ 定界符 ”，只起“ 定界 ”的作用，不算做常量本身的内容。

3．符号常量

符号常量是一种用来代替某数值常量或字符串常量的标识符。此常量在使用之前必须用 CONST 语句加以说明。在 Quick BASIC 中，可以定义符号常量，用来代替数值或字符串。定义的一般格式为：

CONST 常量名＝表达式［，常量名＝表达式］…

其中的“ 常量名 ”为一个名字，按变量名的规则定义，可以加类型说明符；“ 表达式 ”由文字常数、算术运算符（除乘方指数运算符 ^ 外）、逻辑运算符组成，也可以使用诸如“ File already exist ”之类的字符串，但是不能使用字符串连接、变量和用户定义的函数或内部函数。例如：

```
CONST Maxchars%=254,Maxbue%=Maxchars+1
DIM Buffer%(Maxbue%)
```

其实，符号常量完全可以通过对变量赋值来实现，但使用符号常量有一定优点：

- 在整个程序模块中经常用到的常量只需定义一次。
- 常量不会因疏忽而改变。
- 在独立的程序中，使用常量比使用变量能产生更为有效的代码。
- 使用常量使程序更容易修改。

同时在使用符号常量时应当注意：

（1）如果使用了类型说明符，那么在该常量被引用时可以省略。例如：

```
CONST Maxdim%=20
……
DIM AccountName$(Maxdim)
```

在省略了类型说明符后，常量的类型将取决于 CONST 语句中表达式的类型。字符串表达式总是产生字符串常数，对于数值型表达式来讲，会将该表达式求值并按照最简单的类型来表示此常数。经如，当表达式的结果为整数时，该常数就会确定为最简单的整型常数。

（2）符号常量的类型由类型说明符或表达式的类型决定，并不受 DEF type 语句的影响。

（3）在过程之外说明的常量可以在本模块中使用，可以在任何使用表达式的地方使用符号常量。

（4）因类型说明符不是符号常量的一部分，故在定义符号常量后，有些变量名在使用时要小心。例如：

```
CONST A＝5.678
```

那么此时 A！、N#、N%和 N＆就不能再用作变量名。

4.3.2 变量

变量是指程序运行过程，其值可重新赋值的量，亦即在程序执行过程可以发生变化的量。下面将介绍变量的命名规则和变量的类型。

1．变量的命名规则

变量名是指表示变量值的名称，主要用于代表一个存储单元。在 Quick BASIC 中，变量的命名由字母、数字和小数点组成。它命名的规则如下：

（1）变量名由字母、数字和小数点组成，必须以字母开头。如：A1、ConNT.add、B 等都是合法的。

（2）变量名由 1 到 40 个字符组成，在变量名中只能出现字母、数字、小数点和用于表明变量类型的符号。

（3）表示变量类型的符号（！ #、%、$、＆）必须是变量名的最后一个字符。

（4）不能使用 Quick BASIC 中的保留字作为变量名。例如变量 DATA 或 DATA$非法，但变量 DATA 合法。

（5）变量名中不能出现空格符号。

（6）变量名不能与过程名或符号常量同名。

（7）在变量名中出现的字母大小写是等效的（要区别于 C 语言等）。

2. 变量类型的说明

变量是用来存储数据的，因为数据可以分为不同的类型，所以存放数据的变量也应当有相应的类型，用来存放整型数、长整型数、单精度实数、双精度实数和字符串的变量，依次被称为整型变量、长整型变量、单精度变量、双精度变量和字符串变量。变量在内存中所占的存储空间由相应的数据类型所占空间决定，整型变量占 2 个字节、长整型变量占 4 个字节、单精度变量占 4 个字节、双精度变量占 8 个字节，字符串变量由字符串的长度决定，字符串中每个字符占 1 个字节。下面我们介绍一下如何定义变量应注意的事项。定义变量的方法大致有三种：

（1）用类型说明符定义变量

把类型说明符放在变量的末尾，可以定义不同的变量类型。其中“ % ”表示整型、“ & ”表示长整型、“ ! ”表示单精度型，“ # ”表示双精度型，“ $ ”表示字符串型。如：RESULT%、RESULT&、RESULT!、RESULT#、RESULT$等。

（2）用 declare 变量名 AS 类型格式来定义变量

其中的“ declare ”可以是 DIM、COMMON、REDIM、SHARED 或 STATIC。“ 类型 ”可以是 INTEGER（整型）、LONG（长整型）、SINGLE（单精度型）、DOUBLE（双精度型）、STRING（字符串型）或用户自定义的类型。例如：

```
DIM A AS LONG
```

是把 A 定义为长整型变量。

使用 AS STRING 子句可以说明变长字符串，也可以说明定义长字符串。变长字符串的长度取决于赋给它的字符串常量的长度。定长字符串的长度通过“ *数值 ”来确定。比如：

```
DIM STRING1 AS STRING
DIM STRING2 AS STRING *6
```

记录类型可以使用 TYPE 语句定义，也可以使用记录类型定义变量。

［例 3-3］　定义记录类型。

```
TYPE InventoryItem              '记录类型名定义
Description AS STRING *20       '该数据项（域）为字符串型
Number AS STRING *10
Quantity AS LONG                '该数据项（域）为长整型
Orderpoint AS LONG
END TYPE
DIM CurrentItem AS InventoryItem,
PreviousItem AS InventoryItem
```

为了程序的易读性和指定记录类型变量中的某个元素，需要使用“ 变量名.元素名 ”这样的格式。当使用 AS 子句说明了一个变量时，则变量的每次说明都必须使用 AS 子句。例如：

```
CONT MAXEMPLOYEES＝300
DIM EmpNames(MAXEMPLOYES)AS STRING
COMMON EmpNames()AS STRING
……
```

（3）使用 DEF type 语句定义变量类型

格式：

```
DEF type［-字母］［,字母［-字母］］……
```

其中的“ DEF ”是保留字；type 是类型标志，它可以是 INT、SNG、DBL、LNG 或 STR，分别表示整型、单精度型、双精度型、长整型和字符串型。注意在 DEF 和类型标志之间不要包含空格。其中的“ 字母 ”可以是英文字母的 A~Z 中的任意一个（不区分大小写）。例如：

DEFDBL L-P

其中，L、M、N。O、P 都可以作为双精度变量而 LIMIT、Number Pointer 等也同样是双精度变是名。在类型说明语句中定义的字母可以作为该类型的变量名，而且以该字母开头的变量名，也对应那种类型的变量。

类型说明语句一般放在程序或模块的开头。当整个模块的变量都使用整型时，可以这样说明：

DEFINT A-Z

Quick BASIC 在执行程序时只扫描一次文本，当变量在程序中出现时，它的类型就不再改变了。以上介绍了三种定义变量类型方法，表 4-3 列出了各种变量类型的定义方法和存储要求。

表 4-3 各种变量类型的定义方法和存储要求

变量类型	类型说明符	AS type 名	占用内存（字节）
整型	%	INTEGER	2
长整型	&	LONG	4
单精度型	!	SINGLE	4
双精度型	#	DOUBLE	8
变长字符串	$	STRING	实际字符个数
定长字符串	$	STRING*NUM	NUM

在定义变量类型时，我们应当注意：

- 变量名相同但类型说明符不同时，所表示的是不同的变量，如 ZH%、ZH＆、ZH#是三个不同的变量。
- 当一个变量没有类型说明符时，将默认是单精度变量。
- 在实际应用中，应当根据需要设置变量的类型。但应在达到自己目的情况下尽量节约内存空间，以提高处理速度。
- 类型说明符（%、＆、!、#、$）总比 DEF type 语句优先起作用。

4.4 运算符和运算规则

Quick BASIC 的运算符大致可以分为算术运算符、关系运算符、逻辑运算和字符串运算符。下逐一介绍各运算符及其运算规则。

4.4.1 算术运算

Quick BASIC 的算术运算符主要有加法、减法、乘法、除法、整数除法、浮点除法、取模、取负和指数等，表 4-4 中列出了各算术运算符及其在运算中的优先级（由高到低）。

因为我们对于 Quick BASIC 的算术运算已经很熟悉了，在这里我们只向读者介绍整数除法与取模运算。

1．整数除法（\）

当操作数带有小数时，首先被四舍五入为整数或长整数，然后再进行整除运算。但操作数必须在-21417483648.5 到+2147483647.5 的范围内，其运算结果被截断为整数（INTEGER、LONG）不进行舍入处理。例如：

X＝14\6

Y=68.56\3.47

运算的结果 X 为 2；Y 值的运算首先将 68.56 变为 69，再把 3.47 变为 3 后进行整除，结果为 23。

表 4-4 算术运算符及在运算中的优先级

含义	运算符	表达式举例
指数	^	a^b
取负	-	-a
乘法	*	a*b
浮点除法	/	a/b
整数除法	\	a\b
取模	MOD	a MOD b
加法	+	a+b
减法	-	a-b

2．取模运算（MOD）

取模运算的结果是一个整型数值，此值为整数除法的余数。例如：

```
X＝8 MOD 3
```

结果为 2。

```
Y＝26.782 MOD 4.586
```

的结果为 1（首先将 26.482 变为 26，再将 4.586 变为 5，然后进行 26/5 运算结果为 5 余 1，所以结果为 1）。

4.4.2 关系运算

在 Quick BASIC 中，关系运算可以进行对象的大小比较，使用关系运算符既可以进行数值的比较，也可以进行字符串的比较。有关 Quick BASIC 中的关系运算符及其含义见表 4-5 所示。数值比较一般是对两个算术表达式的值进行比较。例如：

```
a+b>c/d
```

如果 a+b 的值大于 c/d 的值，则表达式的值为真（-1），否则为假（0）。

在编程时，关系运算常常作为条件判断使用。

［例 3-4］ 关系运算作条件判断。

```
a=100
IF a<>100 THEN?"Not Equal"ELSE?"Equal"
END
```

因为 a 等于 100，所以招待后输出“ Equal ”

表 4-5 关系运算符及其含义列表

运算符	测试关系含义	a=b
=	相等	a=b
< >或><	不相等	a< >b 或 a><b
<	小于	a<b
>	大于	a>b
<=	小于或等于	a<=b
>=	大于或等于	a>=b

4.4.3 逻辑运算（布尔运算）

逻辑运算一般用逻辑运算符连接两个或多个关系式，组成一个布尔表达式，其结果也为一个逻辑值。在 Quick BASIC 中的逻辑运算符主要包含：

NOT 逻辑非 XOR 异或

AND 逻辑与 EQV 等价

OR 或 IMP 蕴含

逻辑运算的规则如表 4-6 所示：

表 4-6 逻辑运算规则（T 为真，F 为假）

A	B	NOT A	A AND B	A OR B	A XOR B	A EQV B	A IMP B
T	T	F	T	T	F	T	T
T	F	F	F	T	T	F	F
F	T	T	F	T	T	F	T
F	F	T	F	F	F	T	T

与关系运算一样，逻辑运算也可以用来判断程序的流程。比如：

```
IF X＞100 AND Y＜200 THEN……
```

当对数值进行逻辑运算时，操作数必须在-2147483648 到+2147483647 的范围内。如果超出此范围，就会产生溢出错误，如果操作数为负值，则以它对应的补码形式表示。参加运算的数都要置换成整型（16 位）或长整型（32 位）二进制数。

4.4.4 字符串运算

字符串运算只有一个，即“十”，其作用是连接一个或多个字符串型常量、变量或函数，形成新的字符串。Quick BASIC 有大量的内部函数，下面列出字符串函数及其功能，见表 4-7。

表 4-7 字符串函数及其功能列表

函 数	功 能
LEFT$（x$,n）	取从 x$左边开始的 n 个字符
RIGHT$(x$,n)	取从 x$右边开始的 n 个字符
MID$(x$,n,m)	从 x1$的第 n 个字符开始，取 m 个字符
INSTR(n,xl$,x2$)	从 xl$的第 n 个字符开始，查找 x2$d x1$中的位置
CHR$(x)	把 ASCII 码 x 转换为数值
VAL(x$)	把数字字符串转换为数值
STR$(x)	把 x 转换为数字字符串
UCASE$(x$)	把 x$的所有字母都变为大写
LCASE$(x$)	把 x$的所有字母都变为小写
LEN(x$)	返回 x$中字符的个数
STRINGe(n,m)	返回长度为 n 的一个字符串
DATE$	返回系统日期
TIME$	返回系统时间
LTRIM$(x$)	去掉 x$的前面空格
RTRIM$(x$)	去掉 x$的末尾空格

4.5 表达式

学到这里，读者对表达式应该不陌生了，因为前面已经几次提到过并在例题中使用过表达式。Quick BASIC 表达式是指利用运算符将若干运算量或操作数，包括常量、变量、函数、数组元素等连接起来的式子。表达式可以分为算术表达式、关系表达式、字符运算表达式等。在 4.4 节中已经提到过各类表达式，本节进一步介绍表达式求值的执行顺序。

因为表达式可能饮食多种运算，计算机将按一定的顺序对表达式进行求值。求值顺序通常按下述顺序执行。

（1）首先进行函数运算。

（2）其次进行算术运算，算术运算顺序见表 4-4。

（3）接着进行关系运算。

（4）最后进行逻辑运算，逻辑运算的顺序为：

NOT AND OR XOR EQV IMP

但要注意，当指数和负号相邻时，负号优先，比如：

？5 ^ -2 的执行结果为 0.04（1/25），而不是-25。

连加或连乘没有固定的执行顺序，但当连续调用 FUNCTION 过程就会出错，比如：

```
A＝B(x)+G(x)+D(x)
```

如果上述三个过程中的一个改变了 x 或共享变量的值，运算顺序可能会对运算结果产生影响。为了防止此类问题的发生，我们可以首先把 FUNCTION 的调用结果赋给临时变量，然后再相加。可以把上例改为：

```
M＝B(x):N=C(x):P=D(x)
A=M+N+P
```

我们在书写表达式时，应当注意：

（1）乘号（*）不能省略，也不能用“.”或“×”代替。

（2）通常情况下，不允许两个运算符相连，要用括号隔开。

（3）乘幂 ^ 表示自乘，如 M ^ N 表示 N 个 M 连乘，当 M 或 N 不是单个常量或变量时，要用圆括号括起来，比如：(M+N)^(P+6)。

4.6 Quick BASIC 标准函数

在 Quick BASIC 中，系统为用户提供了一些常用的函数，这些函数被编成子程序，当用户需要使用它们时只要直接引用就可以了。这些由系统事先确定的能够完成常用运算或某种功能的子程序称为标准函数。比如要求 X 的自然对数，只要直接使用 LOG（X）即可。其中的 X 为自变量（或形式参数），自变量 可以是一个常量、变量或表达式。例如：

SQR(34.4*3/9)

ABS(X+Y)

LOG(8*X)

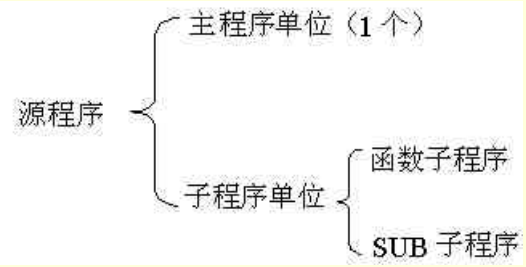
等都是合法的。表 4-8 列出了通用的数值标准函数。

表 4-8 Quick BASIC 的数值函数列表

函 数	函 数 值	函 数	函 数 值
ABS(X)	X 的绝对值	SIN(X)	X 的正弦值
FIX(X)	对 X 截断取整	COS(X)	X 的余弦值
INT(X)	不大于 X 的最大整数	TAN(X)	X 的正切值
CINT(X)	把 X 转换为整型	ATN(X)	X 的反正切值
SGN(X)	1 X>0 0 X＝0 -1 X＜0	EXP(X)	e 的 X 次幂
		LOG(X)	X 的自然对数
		SQR(X)	X 的平方根
DEBL(X)	把 X 转换为双精度数	FRE(X)	空闲内存的字节数
CLNG(X)	把 X 转换为长整数	ASC(X$)	X$的第一个字符的 ASCII 码
CSNG(X)	把 X 转换为单精度数	RND［(X)］	0~1 之间的随机数
OCT$(X)	把 X 转换为八进制字符串	TIMER	从午夜开始计算的秒数
HEX$(X)	把 X 转换为十六进制字符串		

4.7 本章小结

1．Quick BASIC 源程序的结构是模块化的。



2．Quick BASIC 所使用的字符集是 Quick BASIC 语言最基本的元素，只有使用这个字符集内的字符才是合法的。

3．数据是 Quick BASIC 语言中运算的操作对象，是构成语句体的要素，本章介绍了如下数据类型：

表 4-9 数据类型

类型	常量	变量	占用内存（字节）
整型	＋53，－30A%2	A%	2
长整型	162，－87	B&	4
单精度型	1.52，J.13E+2	C!，N	4
双精度型	1.56D－3	D#	8
变长字符串	"STUDENT"	E$	实际字符个数
定长字符串	'TEATHER"	F$	NUM

4．算术表达式包含了运算操作的符号（算术运算符和括号）和操作的对象（数据）两大要素。

5．标准函数见表 4-8，可以把它看为支持简化运算的函数运算符，注意引用函数的规定。

习 题

1．判断以下常数，哪些是 BASIC 中的合法常数？哪些是不合法的？

11，021　　-15.5e3　　11.08×108　　E-5　　“follow Me”

5678　　‘ON It’　　5/2　　1El/30.5D-20

2．判断下列字符序列，哪些可以作变量名？哪些不可以？

-X1　　Average　　δ　　MIBF　　e×3-1　　LOG1

BegPos　　D・D・T・/1　　Y1,3　　Customer　　Address

3．把下列定点数写成规格化的单精度浮点数

8439.76　　214500000　　0.00000004　　0.1734

4．把下列浮点写成定点数

6.7856E　　0.00156E-3　　3.1415928E+4　　2.128E-6

5．将下列 BASIC 算术表达式改写成数学代数式

①a*Log(X+Y)-SQR(sin(X-Y)/X*Y)

②a+b/(c+d)/(e+f/g)/a-b

③(((X-5)*X+2.7)*X-92.1)*X+10.1

④1/SQR(2*PI)*EXP(-X*X)

6．指出下列代数式转换成 BASIC 算术表达式中的错误

代数式	BASIC 算术表达式
（1）a+bX+cX2+dX3	a+bX+cX*X+dX∧3
（2）$\frac{X+2}{Y+4}$	X+2/Y+4
（3）$(\frac{X}{Y})^{n-1}$	(X/Y)∧n-1

第五章　顺序结构

一般程序设计语言都提供了顺序结构、选择结构和循环结构这 3 种基本结构。其中顺序结构是最简单的一种。在该结构中，组成程序的各个语句或语句块依次顺序执行。也就是说，它的流程是单方向直线进行的。本章将介绍在这种结构中用到的数据输入、数据运算、数据输出等语句。涉及的主要内容有：

- 变量的赋值 LET 语句
- 键盘输入 INPUT 语句
- 读数 READ 语句和置数 DATA 语句
- 数据输出语句 PRINT 语句和 PRINT USING 语句
- 结束 END 语句和暂停 STOP 语句
- 注释 REM 语句

上述语句中的 LET 和 PRINT 都兼有数据运算功能，所以也是最基本的数据运算语句。

5.1　引例

例如　计算一个梯形的面积。

算法：

先向计算机输入梯形的上底、下底和高这 3 个参数；然后计算机将输入的这些数据代入梯形面积公式进行计算；最后输出计算结果。这是一个典型的顺序结构程序，用 N－S 图表示，如图 5-1 所示。其中第 1 个块即为“数据输入”部分；第 2 个块对应“数据运算”部分；第 3 个块就是“数据输出”部分。

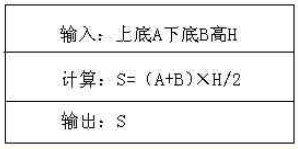


图 5-1　计算梯形面积的结构流程图

源程序：

```
LET A＝6
LET B＝10
LET H＝4
LET S＝（A+B）* H/2
PRINT S
END
```

程序按照语句的行号，由小到大依次执行。

第 1、2、3 行语句是赋值语句，执行该语句时，计算机将上底、下底和高的值，依次送给变量 A、B、H。这是数据输入。

第 4 行语句也是赋值语句。执行该语句时，计算机根据赋值号“＝”右端的表达式，对有关的数值进行运算，并将计算结果赋值给变量 S。这是数据运算。

第 5 行语句是输出语句，其功能是将变量 S 中存放的计算结果，通过屏幕显示给用户。这是数据输出。

程序中的第 6 行语句是结束语句，表示程序终止。

通过此例我们可以看到：

- 组成一般源程序的 3 大部分：数据输入、数据运算、数据输出。
- 顺序结构程序的模式：各语句按排列顺序，依次执行，中间没有分支或循环。
- 设计程序的主要过程：先分析问题、再确定算法、然后画流程图、最后编写程序。

此外的“输入”与“输出”，是相对于计算机中的内存贮器而言的。凡数据从内存贮器之外进入其中，称为“输入”，反之称为“输出”。

5.2 赋值语句（LET）

5.2.1 语句介绍

1. 格式

［LET］〈变量名〉＝{〈常量〉丨〈变量〉丨〈表达式〉}

例：
```
LET A＝5                ′用常量赋值
LET B＝A                ′用变量的值赋值
LET S＝A+B              ′用表达式的值赋值
```

其中

LET——赋值语句的关键字；

＝——赋值号。与数学中等号的概念截然不同，不可混淆；

［ ］——方括号表示该部分可以省略；

〈〉——尖括号用来说明内容。

后面将会遇到的符号还有：

…——表示可重复多个；

{}——花括号表示该部分内容中选择一个，各项之间用“丨”分隔。

2. 功能

使赋值号右边表达式的值赋值给左边的变量。它主要用于数据运算，但也可用于简单的数据输入。

3. 使用说明

（1）使用时要求赋值号右边的数据类型与左边的变量类型必须一致。即数值变量只能用数值赋值；字符串变量只能用字符串赋值。

（2）如果赋值号右边只是一个常量或一个变量，执行时把该常量或变量的值直接接赋值给左边的变量。

（3）如果赋值号右边是一个算术表达式，执行时先进行右边表达式的计算，然后把得到的值再赋值给左边的变量。即先计算，后赋值。

5.2.2 应用举例

[例 5-1] LET 语句的几种赋值形式。源程序：

```
REM EXAMPLE 1
LET A＝5
LET B＝A
LET A＝A+1
LET A $＝“GOOD！”
PRINT A，B，C，A$,B$
END
```

运行结果：

```
6          5          0          GOOD！
```

程序说明：

第 1 行语句是注释语句，说明该程序是例 5-1。注释语句的有关内容，参阅 5.5.3。

第 2 行语句是将一个数值常量赋值给一个数值变量。

第 3 行语句是将一个变量的值赋给另一个变量。即将变量 A 中的数值 5 赋值给变量 B，而 A 中的值保持不变。

第 4 行语句是将一个变量的值，按算术表达式运算后，再赋以新值。即将原 A 中的 5 加 1 后得 6，再赋值给 A。因此，一个变量若被多次赋值，只保留其最后一次所赋的值。

第 5 行语句是将一个字符串常量 GOOD！赋值给一个字符串变量 A$。（字符串常量应用双引号括起来）。

第 6 行语句是输出语句，依次显示变量 A、B、C、A$、B$中的值。从运行结果看，其中数字变量 C 因没赋值，则自动取零；字符串变量 B$因没赋值，则自动取空串。

[例 5-2]　交换两量中的数值。

算法见第二章的例 2-1。

源程序：

```
REM EXAMPLE 2a
LET A＝5                    ′变量 A 赋初值
LET B＝8                    ′变量 B 赋初值
PRING A，B                  ′输出 A，B 值
LET C＝A                    ′A 值暂存于变量 C
LET A＝B                    ′B 值送给 A
LET B＝C                    ′暂存 C 的 A 值送给 B
PRINT A，B                  ′再输出 A，B 值
END
```

运行结果：

```
8              5
```

5.2.3　变量数值交换语句（SWAP）

在列 2 中两个变量交换数值的方法是引入中间变量，此时需用 3 个赋值语句才能完成任务。对于此项工作使用专用的变量数值交换语句 SWAP，可以大大简化程序。

1．格式

SWAP〈变量名 1〉，〈变量名 2〉

2．功能

交换两个类型相同的变量值。

用此语句可将例 5-2 改写如下：

```
REM EXAMPLE 2b
A＝5：B＝8                      ′变量 A，B 赋值
PRINT A，B
SWAP A，B                       ′变量 A，B 交换数值
PRINT A，B
END
```

5.3　输出语句（PRINT）

数据输出到不同的输出设备，用不同的 BASIC 语句，如果需要输出到显示器上，所用的语句为 PRINT 和 PRINT USING。如果需要输出到打印机上，所用的语句为 LPRINT 和 LPRINT USING。虽然两个语句面向的输出设备不同，但用法基本一样，因此本教材只介绍 PRINT 和 PRINT USING 语句。

PRINT 语句在 BASIC 程序中使用极为频繁，它除了输出程序的运行结果之外，还能对字符串原样照印和输

出空行。字符串的原样照印可以用来给输出结果添加说明字符；输出空行可以将多行输出的数据用空行拉开间隔，这样使输出结果清晰明了，便于阅读。

PRINT USING 语句是自选格式打印语句，它具有丰富多变的输出格式，由用户自由选定，可使输出数据排列美观整齐，满足用户的各种需要。读者应该逐步地掌握它们。

5.3.1 语句介绍

1. 格式

PRINT［〈输出项 1〉］［{，｜；}〈输出项 2〉］…{，｜；}

例：

```
PRINT 56,X,A*B
PRINT "X="；X
PRINT
```

其中

“，”——输出项之间的分隔符，表示按标准格式输出。

“；”——输出项之间的分隔符，表示按紧凑格式输出。

〈输出项〉——可以是常量、变量、表达式。

2. 功能

把输出项的值显示到屏幕上。输出项是常量、变量、表达式时，输出它们的值；输出项是用双引号括起来的字符串时，原样输出字符串；缺省输出时，输出空行。

3. 使用说明

（1）一个 PRINT 语句可以输出一个或多个输出项，当多项输出时，中间用分隔符“，”或“；”分隔。例如：

```
PRINT 5，9，21
PRINT A；B；A+B
```

（2）PRINT 语句用以原样照印字符串时，字符串要用双引号引起。例如：

```
PRINT“GOOD！”
```

（3）当输出项为表达式时，PRINT 语句具有计算功能。能先计算后输出。

（4）为了使输出结果清晰，人们对输出格式常有不同要求，常用输出格式可以用输出项之间的分隔符来指定，所用分隔符有逗号“，”和分号“；”两种；还可用专门的函数来指定，专用函数有输出定位函数 TAB 和输出空格函数 SPC。

5.3.2 标准格式输出

当 PRINT 语句输出项之间用逗号分隔时，称为标准格式输出。不同 BASIC 版本其标准输出格式略有差异，Quick BASIC 是将整个 80 列宽的屏幕分成 5 个显示区域，每个显示区域的宽度是 14 个字符，如图 5-2 所示。

图 5-2 显示区域划分

标准格式输出规则举例说明如下：

（1）PRINT 语句中用逗号分隔的各输出项按顺序输出在各显示区域上，请看例 5-3。

[例 5-3] 源程序：

```
REM EXAMPLE3
PRINT "1","2","3","4","5"
PRINT 10,31,-8,12,-23
```

```
PRINT 1,2,3,4,5,6,7,8
END
```

运行结果：

```
1       2       3        4        5
10      31      -8      12      -23
1       2       3       4       5
6       7       8
```

程序说明：

①第 2 行语句按标准格式输出字符串，用于标注各显示区域的第一列位置。

②第 3、4 行语句都是按标准格式输出数值量，只是输出项多少不同而已。

③第 4 行语句中的输出项多于 5 个时，则第 5 个输出项后自动换行，再从下一行的第一个显示区域继续输出，依此类推。

④输出字符串时，从该显示区域的第一列开始输出；输出数值量时，各显示区域的第一列留给数值的符号位。如果输出的是 0 或正数，符号位为空格；如果输出的是负数，则符号位显示“ - ”号。

（2）PRINT 语句中两个逗号之间如果没有输出项，则意味着输出时跳过一显示区域，请看例 5-4。

[例 5-4] 源程序

```
REM EXAMPLE 4
PRINT "1","2","3","4","5",
PRINT 15,,12.86,,-20
END
```

运行结果：

```
1       2       3       4        5
15              12.86           -20
```

程序说明：

①第 2 行语句输出的字符串，用于标注各显示区域位置。

②第 3 行语句要求跳过第 2 和第 4 显示区域，与运行结果是一致的。

（3）PRINT 语句中某输出项的长度超过 14 列时，可以连续占用两个或更多的显示区域。请看例 5-5。

[例 5-5] 源程序：

```
REM EXAMPLE 5
PRINT "123456789A123456789B123456789C123456789D"
PRINT "First"，“ Second ”,"Third"
PRINT "abcdefghijklmnopqrstuvwxyz","ABCDE"
END
```

运行结果：

```
1 2 3 4 5 6 7 8 9 A 1 2 3 4 5 6 7 8 9 B 1 2 3 4 5 6 7 8 9 C1 2 3 4 5 6 7 8 9 D
First                Second               Third
a b c d e f g h i j k l m n o p q r s t u v w x y z        ABCDE
```

程序说明：

①第 2 行语句输出的字符串，用于标注各列列号，便于读者对照各输出项的输出位置。因其长度超过 14 例，所以占用了后面的显示区域。

②第 3 行语句按标准格式输出的字符串，用于标注显示区域。

③第 4 行语句的第 1 个输出项长度超过 14 列，输出结果，占用两个显示区域；第 2 个输出项从第 3 个显示区域开始。

5.3.3 紧凑格式输出

当 PRINT 语句输出项之间用分号分隔时，输出数据一个紧挨一个，称为紧凑格式输出。

紧凑格式输出规则举例说明如下：

（1）如果输出的是数值量，则两项之间空一格，而且每个数的前边还留一个符号位；如果输出的是字符串，则两项之间不留空格。请看例 5-6。

[例 5-6] 源程序：

```
REM EXAMPLE 6
PRINT "123456789A123456789B"
PRINT 8；0；-10；18；-12.6
PRINT "Q-"；"BASIC"
END
```

运行结果：

```
1 2 3 4 5 6 7 8 9 A 1 2 3 4 5 6 7 8 9 B
8      0      -10      18      -12.6
Q-BASIC
```

（2）在一个 PRINT 语句中，分号、逗号可以混合使用。请看例 5-7。

[例 5-7] 源程序：

```
REM EXAMPLE 7
A＝5：B＝-2
PRINT "First","Second","Third"
PRINT "A="；5，"b="；-2，"A+B="；A+B
END
```

运行结果：

```
First        Second        Third
A = 5          B = -2          A+B ＝ 3
```

程序说明：

①第 2 行语句是赋值语句，给变量 A、B 赋初值。省略了语句关键字 LET。

②第 4 行语句中，分号、逗号混合作用。用分号分隔的项按紧凑格输出；用逗号分隔的项按标准格式输出。利用这一功能可以给输出数值添加提示符，便于输出数值的识别。

③第 4 行语句中的输出项 A+B 为表达式，运行时是先计算后输出。

（3）PRINT 语句输出项的最后，如果没有分隔符，输出后便自动换行。如果加了分隔符，此时不再自动换行。若加了分号，则按紧凑格式接着输出下一个 PRINT 语句的输出项；若加的是逗号，则按标准格式在下一个显示区域接着输出下一个 PRINT 语句的输出项。请看例 5-8。

[例 5-8] 源程序：

```
REM EXAMPLE 8
PRINT "3*5="；
PRINT 3*5
PRINT "1*1="；1*1，
PRINT "1*2"；1*2
END
```

运行结果：

```
3*5＝15
1*1＝1        1*2＝2
```

（4）当 PRINT 语句中没有任何输出项时，则输出一个空行，相当于增加了一次换行，请看例 5-9。

[例 5-9] 源程序：

```
REM EXAMPLE 9
PRINT 1，2，3
```

```
PRINT
PRINT 4，5，6，
PRINT
PRINT 7.8
END
```

运行结果：

```
1        2        3

4        5        6
7        8
```

程序说明：

在例 5-9 **中，第** 2 **行语句输出完数字** 3 **之外自换行，因此第** 3 **行语句的作用是空一行的作用。而第** 4 **行语句最后有逗号，输出完数字** 6 **之后不换行，因此第** 5 **行语句的作用相当于增加一次换行。**

5.3.4 定点格式输出

用分隔符指定输出格式虽然比较方便，但在使用中仍显得死板，满足不了用户某些特定输出格式的需要，为此 BASIC **提供了输出定位函数** TAB **和输出空格函数** SPC**，利用它们可以构成特殊的输出格式。**

1．输出定位函数 TAB

该函数在 PRINT **语句中使用，它的功能是将光标定位到** TAB **函数指定的列号上。**

函数格式：

TAB（〈算术表达式〉）

表达式的值必须是大于或等于零的整数（如果是小数，则按四舍五入取整），该整数即为指定输出位置的列号。请看例 5-10。

[**例** 5-10] **源程序：**

```
REM EXAMPLE 10
PRINT "123456789A123456789B123456789C"
A＝-23：B＝5
PRINT TAB（3）；"A+B="；TAB（8）；A；TAB（12）；"+"；
TAB（14）；B；TAB（17）；"="；TAB（20）；A+B
END
```

运行结果：

```
1 2 3 4 5 6 7 8 9 A 1 2 3 4 5 6 7 8 9 B 1 2 3 4 5 6 7 8 9 C
A+B＝     -23     +5     ＝      -18
```

TAB **函数的使用规则：**

（1）输出项跟在 TAB **函数后面，用分号“ ;”与之相连，输出项按** TAB **函数字位的列号输出。**

（2）TAB **函数定位的列号超过行宽时，则按行〈指定列号〉**MOD**〈行宽〉定位。例如行宽为** 80 **列，当** TAB **定位于** 90 **列时，则实际定位在** 10 **例。**

（3）TAB **函数后若没有输出项，则可作为后续** PRINT **语句的定位点。**

（4）在一个 PRINT **语句中，可以包含多个** TAB **函数。各个函数之间用分号“ ;”分隔。**

2．输出空格函数 SPC

该函数也在 PRINT **语句中使用，它的功能是跳过若干个空格。**

函数格式：

SPC（〈算术表达式〉）

表达式的值必须是大于或等于零的整数（如果是小数，则按四舍五入取整），该整数即为跳过的空格数。请看例 5-11。

[**例** 5-11] **利用** XPC **函数打印一个表框。源程序：**

```
REM EXAMPLE 11
PRINT "123456789A123456789B123456789C123456789D"
PRINT
PRINT SPC(12)；"* * * * * * * * * * * * * * * "
PRINT SPC(12)；"*"；SPC（12）；"*"
PRINT SPC（12）；"*"；SPC（4）；"LIST"；SPC(4)；"*"
PRINT SPC（12）；"*"；SPC（12）；"*"
PRINT SPC（12）；"* * * * * * * * * * * * * * * "
END
```

运动结果：

```
1 2 3 4 5 6 7 8 9 A 1 2 3 4 5 6 7 8 9 B 1 2 3 4 5 6 7 8 9 C 1 2 3 4 5 6 7 8 9 D
* * * * * * * * * * * * * * * * *
*                               *
*             L I S T           *
*                               *
* * * * * * * * * * * * * * * * *
```

程序说明：

①在一个 PRINT **语句中，可以包含多个** SPC **函数，各个函数之间用分号“ ；”分隔。**

②TAB **函数与** SPC **函数相比，其差别在于** TAB **函数所控制的是输出项的起始列号，**SPC **函数控制的是输出项之后跳过空格的个数。对于输出项不等长的内容，为使多个输出项排列整齐，使用** TAB **函数较方便，因为** SPC **函数对每个输出项都要计算空格数，较麻烦。**

5.3.5 自选格式输出

人们在实际工作中对输出数据格式的要求是多种多样的，例如：对输出数据要求安排固定的输出长度；多行排列的数值要求小数点纵向对齐；按科学表示法（指数形式）输出数据；有一定结构关系的数据排列成整齐合理的表格形式。

对于这些要求，前面介绍的几种 PRINT **输出格式是难以实现的。自选格式输出语句** PRINT USING **能满足用户的上述需要。**

1. 格式

PRINT USING“〈格式字符串表达式〉”；〈输出项 1〉［，〈输出项 2〉］…［{，｜；}］

其中

〈格式字符串表达式〉——用于定义输出项的输出格式。

2. 功能

按格式字符串表达式定义的输出格式输出字串或数值。

3. 使用说明

（1）一个 PRINT USING **语句中只能有一个格式字符串表达式。**

（2）表达式由格式代码及其他字符组成，表达式要用双引号括起来。

BASIC **的格格式代码共有** 14 **种，它们是：**

用于输出字串的 3 **种：!、** **、&。**

用于输出数值的 11 **种：**#、 ^ 、+、-、$$、* *、* * $、.、,、%、-。

文字字符可以是：数字 0~9**；大小写英文字母；某些键盘符等。**

（3）格式字符串的格式代码只用来确定输出项的输出格式，本身一般不被显示；文字字符则与输出项一起显示输出。各种格式代码的功能及用法较为繁杂，这里只介绍数值的格式代码“ # **”和“** . **”的用法，其余的请参阅附录二。**

（4）格式字符串表达式与输出项之间要用分号连接；多个输出项之间必须用逗号连接。此时逗号只起分隔

作用，不要理解成按标准格式输出。

（5）语句最后的分号或逗号可任选。两者都与 PRINT 语句最后的分隔号的用法相同。

4. “#”、“.”格式码的输出

（1）输出数字的长度及小数部分的位数完全由格式代码#号确定。

“######”：表示输出 6 位整数，长度为 6 列。

“####.#”：表示整数部分 4 位，小数部分 1 位，小数点占 1 列，总长度仍是 6 列。

“###.##”：表示整数部分 3 位，小数部分 2 位，小数点占 1 列，总长度仍是 6 列。

（2）数值指定的位数不得超过 24，否则将出现 lllegal function call 的出错信息。

（3）所有数字都是向右对齐，定义多余的整数部分补空格，多余的小数部分补 0，若不数部分定义不足，则尾数自动四舍五入。

（4）正数不显示符号，负数显示“-”号。

[例 5-12] 源程序：

```
REM EXAMPLE 12
A=126：B=21.2：C=-2.638：D=0.273
PRINT "123456789A123456789B123456789C"
PRINT USING"######"；A，B，C，D
PRINT USING"####.#"；A，B，C，D
PRINT USING"###.##"；A，B，C，D
PRINT USING"###.##"；B
PRINT USING"###.##"；C
PRINT USING"###.##"；D
END
```

运行结果：

```
123456789A123456789B123456789C
   126          21          -3           0
 126.0       2120       -2.6         0.3
126.00     21.20     -2.64      0.27
 21.20
-2.64
  0.27
```

应用 PRINT USING 语句，关键在于格式字符串的定义。虽然格式代码较多，使得格式字符串形式复杂、灵活、多变，但通过不断实践，必能逐步掌握它，从而使读者能够设计出自己理想的输出格式。

5.4 输入语句（INPUT、READ/DATA）

对于一般源程序，总不了数据输入，即通过各种上方式向有关变量赋值。最简单的方式是用 LET 语句在程序中直接给变量赋值。这种方法提供数据的能力十分有限，本节着重介绍程序中大量使用的另外两种数据输入语句：键盘输入语句 INPUT 和读数、置数语句 READ/DATA。

5.4.1 键盘输入语句（INPUT）

1. 格式

INPUT［;］［“〈提示字符串〉”{，｜；}］〈变量名 1〉［，〈变量名 2〉］…

例：

```
INPUT X，Y
    INPUT“N=“；N
```

INPUT；“A，B，C＝“，A，B，C

其中

［;］：任选。有此分号时，当用户输入完数据并回车后，光标仍停留在同一行上。没有此分号，则换行。

〈提示字符串〉：可选。用于说明输入的内容，必须用双引号括起来。

{；|，}：任选其一，当选择分号时，在显示出提示字符串之后，跟有"？"；当选择逗号时，提示字符串之后没有"?"。

2．功能

程序执行到此语句时暂停执行，屏幕显示“？”（如果语句中选有提示字符串，将原样显示字符串），表示向用户索要准备给变量赋值的数据。这时用户从键盘上输入数据并回车后，变量则被赋值。

3．使用说明

该语句的特点是在程序运行中，通过键盘给变量赋值。输入数据的规则是：

（1）对于一个包含有多个变量的 INPUT 语句，必须在一行将所有数据按变量的个数和排列的顺序依次输入，并用逗号分隔。数据个数多于或少于变量的个数都将产生错误信息：Redo from start，此时用户应重新输入数据。

（2）输入的数据必须与变量的类型一致。

（3）从键盘输入字符串时，不要求用双引号括起来。但是，当字符串本身包含有逗号或空格时，必须用双引号括起来。

（4）最后一个数据项输入完之后必须回车。

4．举例

[例 5-13]　任给三角形的三个边，求三角形的面积。

算法：

先求出三边和的半值：p=(a+b+c)/2

再计算三角形的面积：s=$\sqrt{(p(p-a)(p-b)(p-c))}$

源程序：

```
REM EXAMPLE 13
INPUT A，B，C
P＝（A+B+C）/2
S＝SQR（P*（P-A）*（P-B）*（P-C））
PRINT "A="；A，"B="；B，"C="；C。"S="；S
END
```

运算结果：

```
？3，6，2
A＝3          B＝6          C＝2          S＝9
```

显然，用 INPUT 语句给变量提供数据要比 LET 语句灵活得多。

[例 5-14]　任意输入某一同学的数学、物理、化学 3 门课的考试成绩，计算平均分。

源程序：

```
REM EXAMPLE 14
INPUT "MATHFMATICS:",A
INPUT "PHYSICS:          ",B
INPUT "CHEMISTRY:     ",C
T=(A+B+C)/3
PRINT "AVERAGE:"；SPC（5）；
PRINT USING"###.##"；T
EMD
```

运行结果：

```
MATHEMATICS：78
PHYSICS：        90
CHEMISTRY：      86
AVERAGE：        84.67
```

程序说明：

（1）第 2、3、4 行语句是有提示字符串的键盘输入语句。能提示用户输入相应的单科成绩。为使输出形式整齐美观，提示字符串用空格补齐长度，且提示字符串后用逗号，执行时不显示“？”。

（2）第 6 行语句中的 SPC（5）也是为了补齐空格，保持字符串等长。该语句最后加有分号，是为了使下一个 PRINT USING 语句的输出项接在后面。

（3）第 7 行语句采用了自选格式输出。

请读者仔细理解此程序的提示符设计技巧。

5.4.2 读数、置数语句（READ/DATA）

用键盘输入语句给变量赋值，虽然比较灵活，但每次输入时，计算机都处于等待状态，当输入数据较多时，占用时间较长，并且容易出错；用 LET 语句给一个变量赋值，输入数据多时，语句必然增多，使得程序很长，也不理想。现在，我们再介绍第 3 种输入数据的方法，即使用读数、置数语句。该方法是把输入数据按顺序放在程序的置数语句中，把被赋值的变量按与数据对应的顺序放在程序的读数语句中。程序运行时计算自动把置数语句中的数据按顺序赋值给读数语句中的变量。

1．格式

读数语句：READ〈变量名 1〉［，〈变量名 2〉］…

置数语句：DATA〈数据项 1〉［，〈数据项 2〉］…

2．功能

用存放在置数语句中的数据给读数语句中的变量赋值。

3．举例

[例 5-15] **源程序：**

```
REM EXAMPLE 15
READ A，B，C，D
READ E，F，G，H
RPINT A，B，C，D，E，F，G，H
READ X$,X,Y$,Y,Z$,Z

RPINT X$；X,Y$；Y,Z$；Z
DATA -1，8，0，6，4
DATA 3，-5，2
DATA X＝，8.26，，，Z＝，-4.3
END
```

（1）在程序中 READ 与 DATA 语句必须配合使用。READ 语句是执行语句；DATA 语句是非执行语句，放在程序中任何位置都可以。习惯上集中放在程序的最后，END 语句之前。

（2）无论程序中有多少个 DATA 语句，计算机都按 DATA 语句出现的先后顺序，以及在每个 DATA 语句中数据排列的顺序，依次把每一个数据置于内存贮器的一个数据区中，并设一数据指针，开始时向第一个数据，如图 5-3 所示。

图 5-3 中的数据

-1	8	0	6	4	3	-5	2	x=	8.26			z=	-4.3

内存贮器区

（3）当开始执行 READ 语句时，指针首先指向第一个数据，将该数据读入第 1 个变量中，然后指针后移，指向第 2 个数据，再把第 2 个数据读入第 2 个变量中，依此类推，直到所有 READ 语句中的所有变量被赋值。

（4）由于所有 DATA 语句中数据的存放是连续的，所以各个 READ 语句的变量相互之间也都是连续的。因此，数据与被赋值变量的位置顺序与类型要严格对应，但 DATA 语句与 READ 语句的个数不必一一对应。如

第 20 行的 READ 语句中有 4 个变量，而第 70 行的 DATA 语句中只有五个数据。

（5）DATA 语句中的数据只能是常量，不允许出现变量、表达式或函数；DATA 语句中的字符串常量可以不加双引号，但是，当字符串中包含有空格、逗号等一些特定字符时，应加双引号；DATA 语句中允许有空的数据项，如果空项被读入到数值变量中，则变量值为 0，如果空项被读入到字符串变量中，则变量为空字符串。

5.4.3 恢复数据指针语句

当用 READ 语句从 DATA 语句中读出一批数据后，数据指针也跟着移到这批数据的后面。如果需要将这些数据重新再赋值给另外一批变量，按照数据和变量顺序对应规则，应该把这些数据再用 DATA 语句重写一遍，并接在前一个 DATA 语句的后面。这样重复写数据非常繁琐。为了简化程序和节省内存，可利用恢复数据区语句。

1．语句介绍

（1）格式

```
RESTORE
```

（2）功能

将数据指针恢得到数据区的初始位置。即重新指向第 1 个 DATA 语句中的第 1 个数据。

2．举例

[例 5-16] 利用 RESTORE 语句重读数据。源程序：

```
REM EXAMPLE 16
READ A，B，C
RESTORE
READ P，Q，R
RESTORE
READ U，V，W
PRINT A，B，C，P，Q，R，U，V，W
DATA 7，11，15
END
```

运行结果：

```
7       11       15       7       11
15       7       11       15
```

程序说明：

第 2 行变量 A、B、C 读完数据 7、11、15 之后，第 3 行的 RESTORE 语句使数据指针又指向数据 7，于是第 4 行的变量 P、Q、R 仍读数据 7、11、15，如此重复下去。重复使用的数据越多，该语句的优越性就越明显。

[例 5-17] “虚读”方法。

有时重复使用的数据并不在数据区的开始位置，也就是需要跳过几个数据再进行重复。这时可使用“虚读”方法，即有意设置一些无意义的变量（称之为虚读变量）：去读那些不需要的数据。源程序：

```
REM EXAMPLE 17
READ A，B，C，D，E
RESTORE
READ Z，Z，P，Q，R
RESTORE
READ U，Z，Z，V，W
PRINT A，B，C，D，E，P，Q，R，U，V，W
DATA 7，11，15，21，30
END
```

运行结果：

```
 7    11    15    21    30
 15   21    30    7     21
```

程序说明：

第 4 **行和第** 6 **行中的变量** Z **即为虚读变量，其作用请读者自己理解。**

[**例** 5-18] **设某一水稻单株粒重的样本有** 5 **个观察值，以克为单位，其数为** 2、4、5、7、8，**求其平均数、中数和极差。**

算法：

平均数 资料中各个观察值的总和除以观察值个数所得的商。

AV＝（X1+X2+X3+X4+X5）/5

中数 资料内所有观察值从小到大依次排列居中间位置的观察值。

MP＝X3

极差 资料中最大观察值与最小观察值的差数。

SM＝X5-X1

源程序：

```
REM EXAMPLE 18
READ X1，X2，X3，X4，X5
AV＝（X1+X2+X3+X4+X5）/5
MP＝X3
SM＝X5-X1
PRINT  "X="；
PRINT USING  "#####"；X1，X2，X3，X4，X5
PRINT USING  "__AV=####.##"；AV
PRINT USING  "__MP=#####"；MP
PRINT USING  "__SM=#####"；SM
DATA 2，4，5，7，8
END
```

运行结果：

```
X＝     2    4    5    7    8
AV=    5.20
MP=    5
SM=    6
```

程序说明：

（1）**在例** 5-18 **中，采用** READ、DATA **语句给** X1、X2、X3、X4、X5 **变量赋初值；用** LET **语句计算** AV、MP、SM；**用** PRINT USING **语句输出。**

（2）**值得注意的是给输出数据添加提示符的方法及输出数据对齐的方法。添加提示符时“** X＝ **”与“** AV＝ **”、“** MP＝ **”、“** SM＝ **”，采用的方法不同：“** X＝ **”使用** PRINT **语句的紧凑格式；后三者使用** PRINT USING **语句自选格式。为使输出数据对齐，采用自选格式较方便。**

思考：能否对"X＝"**也使用后者的方法呢？若这样，将产生什么样的输出形式，请读者考虑。**

（3）**格式字符串中下划线为格式代码，表示以下字符作为文字字符输出，即原样打印。**

5.5 END、STOP、REM 语句

5.5.1 END 语句

END **语句是程序的结束标志，每一个完整的** Quick BASIC **程序，在程序的最后必须有一个** END **语句，程**

序执行到 END 语句时便停止下来。该语句已在前面各例中用过了，此处不再重复。

5.5.2 STOP 语句

STOP 语句可以用来终止程序的执行。与 END 不同的是，STOP 语句可以在程序的任何地方出现，而且可以出现多次。利用 STOP 语句这一功能可进行程序调试。

5.5.3 REM 语句

REM 语句称为注释语句。它是一个非执行语句，该语句中的内容仅仅用作对程序进行注释或说明，以增加程序的易读性。当程序执行或编译时，这个语句中注释的内容都被跳过。当打印程序清单时，这个语句中的所有内容原样打印。

注释语句 REM 可以用一个单引号来代替，下面两个语句是等价的：

```
INPUT  "R="；R：REM  the  radius
INPUT  "R="；R       'the radius
```

注意：用单引号时，可省去语句间的冒号。

5.6 符号常数说明语句（CONST）

在 Quick BASIC 中允许使用符号常数，符号常数是用来代替数值或字符串值的。即用一个名字来代表一个常数。例如用 P1 代表圆周率 3.1416；用 BJ 代表"BEIJING"。所以符号常的作用和普通常数的作用相同。

符号常数要用符号常数说明语句进行定义。

1. 格式

CONST〈符号常数名 1〉＝〈表达式 1〉［，〈符号常数名 2〉＝〈表达式 2〉］…

其中

〈符号常数名〉——遵守变量名的规则。也可以在名字名附加类型说明（%、&、!、#或$），用以说明它的类型。

表达式——由数值常值、字符串常数、算术运算符组成，其中不能有变量或函数。

2. 功能

定义符号常数代替的数值或字符串值。

3. 使用说明

（1）常数的类型由类型说明符决定，如果类型说明符省略，则常数类型由表达式决定。

例如：

```
CONST MIN%＝10，MAX＝MIN+100
```

其中，符号常数 MIN 用类型说明符%说明其为整型；符号常数 MAX 省略了类型说明符，但对应的表达式的值为整型数，因此其类型由表达式确定为整型。

（2）CONT 语句一般放在程序的开始，因为符号常数必须先定义，而后才能引用。

（3）符号常数是一个固定数值的符号，不应当作变量用。一经定义，不能再变。重新定义或作为变量赋值都不允许的。

（4）在子程序中定义的符号常数是局部的，只在子程序中起作用；在过程外面说明的符号常数，则在整个模块起作用。

4.举例

[例 5-19] 根据圆的半径计算圆面积。源程序：

```
REM EXAMPLE 19
CONST PI＝3.1416，RS＝"radius:",AE＝"area:"        '定义符号常数
INPUT "R=",R                                        '输入半径
S＝PI * R * R
```

```
PRINT   RS；R，AE；S
END
```

运行结果：

```
R=5
radius:5        area:78.54
```

使用符号常数可使修改程序变得容易。例如且个常数要在程序中多次使用，先用符号常数定义它，以后需要改变该常数值时，只要改变 CONST 语句符号常数代表的数值就成了，完全没必要去改变程序中多处用到的常数值。

本章小结

本章重点介绍了 Quick BASIC 语句中的数据的输入、数据运算和数据输出语句。这是程序中必不可少的部分。

1．LET 语句可经给变量赋以常量、变量或表达式的值。在 LET、INPUT、READ 三种给变量赋值的语句中，只有它能进行计算，并能把结果保存下来，所以它除用作输入数据外，更多的是用于运算。当它作为输入语句使用时，由于一个常量只能给一个变量赋值，它只适用于数据不多的简单程序。

2．INPUT 语句能“人机对话”方式灵活地输入数据。其缺点是执行该语句时，程序处于等待状态，而且键盘输入数据较慢，影响运行速度；重复执行程序时，必须重输入数据。所以，它多用在数据需要人工调节、输入数据较少的程序。

3．READ/DATA 语句能给大批变量赋值，而且利用 RESTORE 语句使有些数据能够重复使用。由于数据保存在内存中，所以运行速度也快。重复执行程序时，不必重复输入数据。该语句适用于处理数据量大的程序。

4．PRINT 语句的使用，重点放在输出格式上。它的输出格式有标准格式和紧凑格式两种。此外，与 PRINT 语句配合使用的还有输出定位函数 TAB 和输出空格函数 SPC。使用 PRINT 语句的两种输出格式，并配合 TAB 和 SPC 两种函数，基本能满足一般输出格式的要求。

5．PRINT USING 语句的使用，重点要放在格式字符串的设计上。为此，必须对组成格式字符串的几种主要格式代码的功能应有清楚的了解。PRINT USING 语句的自选输出格式，具有很强的输出功能，能满足用户各种复杂输出格式的要求。

此外，本章还介绍了 Quick BASIC 扩充的符号常数说明语句 CONST。

习题

1．指出下列语句中的错误

（1）PRINT A＝3*4　　（2）PRINT "A=":A

（3）PRINT USING"# #",B

（4）PRINT USING"# #.#";A,"# #.# #";B#

（5）LET A＝8，B＝7　　（6）LET C＝GOOD

（7）LET X+2＝Y　　（8）LET A，B，C＝6，-1，6

（9）INPUT X；Y，Z　　（10）INPUT X+Y，C-D

（11）INPUT X＝；X　　（12）INPUT X，Y，Z

（13）READ A，B，C　　（14）READ X；Y；Z

（15）READ "P=",P　　（16）READ A，B，C+D

（17）DATA 8；9；　　（17）DATA 3，6

2．写出下列程序运行结果

（1）
```
A＝5
B=8
A=A+B
```

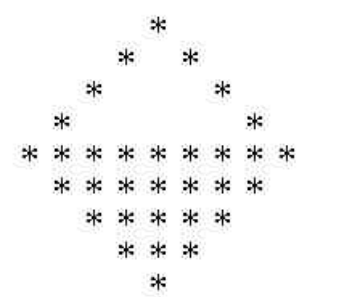
（2）
```
A$="GOOD"
B$="MORNING"
PRINT A$;" ";
```

```
B=A+B                    PRINT B$
PRINT A,B          END
END
```

（3）
```
S＝1
K＝0
READ A
K＝K+1
S＝S*A
READ A
K＝K+1
S＝S*A
PRINT K，S
DATA 10，20
END
```

（4）
```
READ A，B，C，D，E
    PRINT A，B，C
RESTORE
READ T，F，T，G
READ T，T，H，I
PRINT D，E，F，G，H，I
DATA 10，20，30
DATA 40，50，60，70，80
END
```

3．利用 TAB 函数或 SPACES 函数输出下列图案。



4．编制程序，要求程序运行时从键盘输入 3 种商品的单价、购买数量，并计算和输出所用的总金额。

5．班上集体购买课外读物，每册价格分别是 2.50 元 3.20 元和 3.90 元。输入每种书各买多少本，打印出共多少钱？

6．编制程序，完成摄氏温度向华氏温度的转换。（提示：设摄氏温度为 ss，华氏温度为 hs，则转换公式为：hs=ss*5/9+32）

7．鸡兔同笼，已知鸡兔总头数为 h，总脚数为 f，求鸡兔有多少只？（设 h=71,f=158)

8．根据说明写出程序。通过显示适当的提示，要求用户输入下列信息：姓名、性别、年龄、工作单位，然后将输入的数据按适当格式显示出来。

9．假设某储户到银行存款共计 x 元，试问银行出纳员应如何付款最佳？（提示：银行出纳员所付款的各种票额钞票总张数最少时就一定最简便，因而为最佳付款方案。在计算时为了不产生误差，可以使用长整型数据，单位为分）

第六章　选择结构

一个顺序结构程序，在 BASIC 中总是按语句行号的大小顺序执行；在 Quick BASIC 中是按语句的先后次序执行。也就是说，流程是单一直线进行的。它虽简单易懂，但功能有限，不能解决复杂的实际问题。本章将要介绍的选择结构，是要利用计算机的逻辑判断能力，根据确定的条件是否满足，有选择地执行某些语句或程序段，这种程序的执行顺序被改变了。所以选择结构的流程就不是单一直线的，而是发生了分叉，或称为分支。根据分支路的多少，可分为两路分支选择和多路分支选择两类。本章涉及的内容包括：

- 行 IF 语句
- 块 IF 结构
- SELECT CASE 结构
- 选择结构的嵌套

本章的难点在于选择结构的嵌套部分的学习和理解。

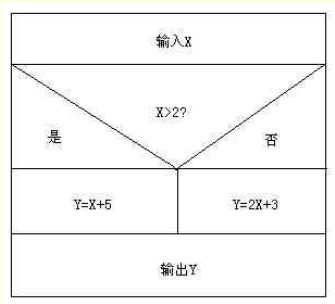
6.1　引例

例如　有一函数：

$$Y=\begin{cases} x+5 & (x+2) \\ 2x+3 & (x\leqslant 2) \end{cases}$$

要求输入一个 X 值，计算并打印大于 Y 的值。

算法：根据 X 的值是否大于 2，分为两种计算方法，这是一个最简单的两分支结构。N-S 流程图如图 6-1 所示。



如图 6-1　N-S 流程图

源程序：

```
INPUT“ X is ”；X
IF X<＝2 THEN Y＝2*X+3　ELSE Y＝X+5
PRINT “ Y＝ ”；Y
END
```

程序中第 2 行语句是一个行 IF 语句，其中 X<＝2 成立的话，即条件为真，那么就执行 THEN 后面的赋值语句，即 Y＝2*3X+3；否则如果 X<＝2 不成立的话，即条件为假，那么就执行行 ELSE 后面的赋值语句，即 Y

＝X+3。注意，两个分支只选择其中一个分支执行。

6.2 行 IF 的语句

6.2.1 语句格式

IF 〈条件〉 THEN {〈语句〉 | 〈行号〉} [ELSE{〈语句〉 | 〈行号〉}]

〈条件〉：用关系表达式或逻辑表达式表示

〈语句〉 | 〈行号〉：在 THEN 或 ELSE 后面可以是语句或行号。如果是语句的话，允许写多个语句，但各语句之间要有冒号分开；如果是行号的话，只允许写一个行号，这个行号必须是该程序存在的语句行号。建议初学者不使用行号。

6.2.2 功能

行 IF 语句的流程图如 6-2 所示。

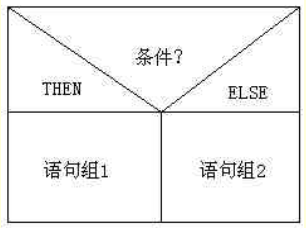


图 6-2 行 IF 语句的流程图

首先，对表示条件的关系表达式或逻辑表达式进行运算，根据其结果的逻辑值分别选择两路分支中的一支，即：如果逻辑值为“真”，条件成立，执行 THEN 后的语句，ELSE 后的语句不执行；如果逻辑值为“假”，条件不成立，执行 ELSE 后的语句，THEN 后的语句不执行。

6.2.3 举例

[例 6-1] 从键盘上输入两个数 M、N，如果 N 不等于 0，打印出商 M/N，否则打印出无商的信息。

算法：

1．输入被除数 M 和除数 N 的值；

2．判断 N〈〉0？按分支处理。

N-S 图如图 6-3 所示。

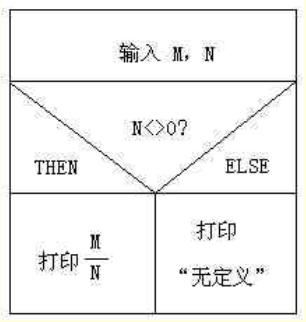


图 6-3 N-S 图

源程序：

```
REM EXAMPLE 1
IF N<>0 THEN PRINT "The Quotinet is":M/N
```

```
    ELSE PRINT "Quotinent undefined"
    END
```

[例 6-2] 从键盘上输入 3 个数，找出其中最大的数。

算法：1．输入 A、B、C3 个数；

2．A 存入一变量 BIG；

3．如果 B>BIG，把 B 存入 BIG；

4．如果 C>BIG，把 C 存入 BIG；

5．打印 BIG 的值。

N-S 图如图 6-4 所示。

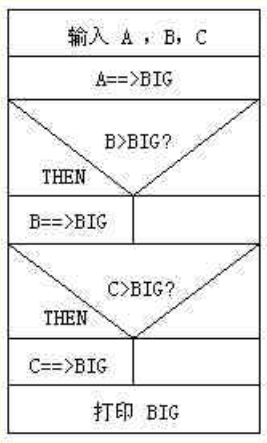


图 6-4 N-S 图

源程序：

```
    REM EXAMPLE 2
    INPUT A，B，C
    BIG＝A
    IF B>BIG THEN BIG＝B
    IF C>BIG THEN BIG＝C
    PRINT “The bigest is”；BIG
    END
```

程序说明：

在 IF-THEN-ELSE 结构中，允许 ELSE 部分省略，就成了 IF-THEN 结构，其 N-S 图如 6-4。由图中可以看出，当条件成立，关系表达式的值为真，就执行 THEN 后的语句；否则，条件不成立就什么也不处理，程序执行下一个语句。

6.3 块 IF

行 IF 语句的结构可以分为 3 个部分：IF〈条件〉部分：THEN 语句部分；ELSE 语句部分。在问题简单的情况下，3 部分写在一行还容易看懂，但当问题复杂的时候，3 部分写在一行里就不便于阅读了。Quick BASIC 中，对这个语句加以扩充，将这 3 个部分分成若干语句行，构成一段程序块，即块 IF，这样结构更清楚明了，功能更强，使用也更方便，而且行 IF 语句的功能，都可以用块 IF 实现。

6.3.1 块 IF 的格式

```
    IF 〈条件〉 THEN
    …}S1〈语句组 1〉
    ELSE
    …}S2〈语句组 2〉
    END IF
```

其中

〈条件〉——用关系表达式或逻辑表达式表示；

S1——条件成立时执行的语句组；

S2——条件不成立执行的语句组。

6.3.2 功能

如图 6-5 所示。如果 IF 语句中的条件成立的话，就执行 S1 的语句；S1 执行后，程序跳过 S2，转到 END IF 语句的下一个语句并继续执行；

图 6-5 IF 语句

如果 IF 语句中的条件不成立的话，就跳过 SI，执行 ELSE 后的 S2 语句，执行完 S2 后，转到 END IF 的下一个语句并继续执行。

总之，它对两个分支，只取其一执行。

6.3.3 举例

[例 6-3] **判断一个数是偶数还是奇数。**

源程序：

```
'EXAMPLE 3
DEFINT A-Z
INPUT Num
IF Num MOD 2=0   THEN
PRINT Num；"is a event number"
ELSE
PRINT Num；"is a odd number"
END IF
END
```

说明：

程序中的 DEFINT 是一个整型变量说明语句，凡是以 A~Z 开头的变量都是整型。

[例 6-4] **任意输入 3 个数 A、B、C，找出其中最大的数和最小的数。**

算法：

1．**先比较 A 与 B，大者赋值给 MAX，小者赋值给 MIN；**

2．**再将 MAX 与 C 比较，大者存入 MAX；**

3．**再将 MIN 与 C 比较，小者存入 MIN；**

4．**输出 MAX，MIN。**

N-S **图如图 6-6 所示。**

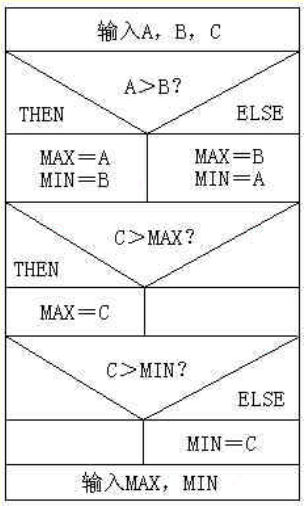


图 6-6 N-S 图

源程序：

```
‘ EXAMPLE 4
INPUT   "A,B,C=";A，B，C
IF A>B THEN
MAX＝A
MIN＝B
ELSE
MAX＝B
MIN＝A
END IF
IF C>MAX THEN
MAX＝C
END IF
IF C>MIN THEN
ELSE
MIN＝C
END IF
PRINT   "MAX=";MAX;"MIN=";MIN
END
```

程序说明：

块 IF **中的** S1 **和** S2 **可以同时存在，这是它的基本形式，但它可以缺省** S1，**或缺省** S2，**这就是它的特殊形式，如图** 6-6 **所示。该程序中包含了** 3 **个** IF **块：第** 1 **个是完整的** IF **块；第** 2 **个缺省** S2，ELSE **可省去不写；第** 3 **个缺省** S1，ELSE **语句不能省。**

注意：每个块 IF **必须以** END IF **结束，它不能省，而且** END IF **必须与** IF THEN **配对。**

6.4 选择结构的嵌套

6.4.1 行 IF 语句的嵌套

在行 IF **语句“** THEN **”后面的语句中或“** ELSE **”后面的语句中允许再包含行** IF **语句，这就叫嵌套，例如：**

```
IF〈条件〉THEN IF〈条件〉THEN〈语句〉ELSE〈语句〉ELSE〈语句〉
├————内层————┤
├————————外层————————┤
```

在 THEN **后嵌套了一个行** IF **语句，称它为内层。**

[**例** 6-5] **平面直角坐标系中，不同的象限内点的坐标的符号有以下** 4 **种情况：**

在 A **象限** X>0，Y>0

在 B **象限** X<0，Y>0

在 C **象限** X<0，Y<0

在 D **象限** X>0，Y<0

任意输入一对坐标值，找出它所在的象限。

算法：

1．**输入一对坐标值** X，Y；

2．**将** 4 **种可能的情况组成为嵌套的行** IF **语句。**

N-S **图如图** 6-7。

源程序：

```
REM EXAMPLE 5
```

```
INPUT X，Y
IF X>0 THEN IF Y>0 THEN PRINT "in A"
ELSE PRINT    "in D"ELSE IF Y>0 THEN PRINT "in B"
ELSE PRINT    "in C"
END
```

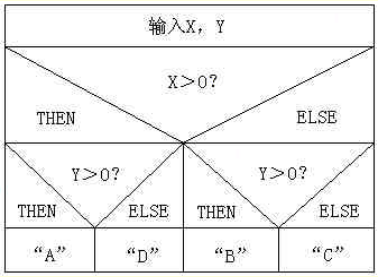


图 6-7　IF 嵌套

说明：

行 IF 的嵌套关系不容易看清楚，因此建议读者；

（1）嵌套不宜太深，尽可能不使用这种嵌套方法；

（2）嵌套的行 IF 语句不能省略 ELSE 部分。

（3）识别行 IF 语句嵌套的层次的方法是：首先扫描关键字 ELSE，从左至右将第一个 ELSE 与它左面的最邻近的 THEN 和 IF 配对，再依次往右找第二个 ELSE，与它后面最邻近的 THEN 和 IF 配对（已配过的 THEN 和 IF 除外）。

6.4.2　块 IF 的嵌套

在块 IF 结构中的两个块 S1、S2，每块中均可再包括块 IF 结构。

[例 6-6]　**将例 5 改用块 IF 的嵌套**

```
'EXAMPLE 6
INPUT X,Y
IF X>0 THEN
IF Y>0 THEN
PRINT "in A"
ELSE
PRINT "in D"
END IF
ELSE
IF Y>0    THEN
PRINT "in B"
ELSE
PRINT "in C"
END IF
END IF
END
```

6.4.3　应用举例

[例 6-7]　**把任意 3 个数按由大到小排列打印，用嵌套的块 IF 写程序。**

N-S **图如图** 6-8 **所示。**

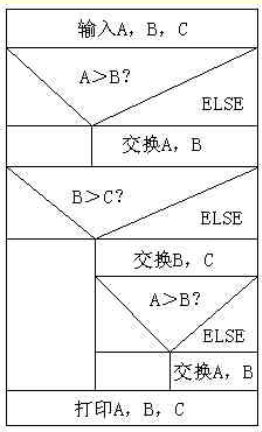


图 6-8　N-S **图**

源程序：

```
'EXAMPLE 7
INPUT A，B，C
IF A>B　THEN
ELSE
SWAP A，B
END IF
IF B>C THEN
ELSE
SWAP B，C
IF A>B THEN
ELSE
SWAP A，B
END IF
END IF
PRINT A，B，C
END
```

[**例** 6-8]　**编写求一元二次方程式** $Ax^2+Bx+C=0$ **的通解的程序。**

算法：

本题要求通解就要对 3 **个系数的任意取值进行判断，在各种情况下都能给出正确的结果。**

顶层设计：判断 A **是否为零，如图** 6-9（a）**。**

（1**）如果** A＝0**，则方程变为解一元一次方程** Bx+C=0**。**

（2**）如果** A≠0**，根据判别式** $disc=b^2-4ac$ **计算方程的根。**

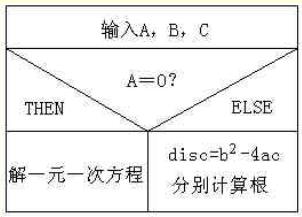


图 6-9　**顶层设计**

第二层设计：

（1**）**A＝0 **解一元一次方程，如图** 6-9（b）**所示。**

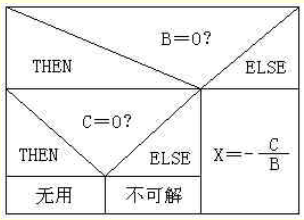


图 6-10 第二层设计

① 判断 b 是否为零，如果 b 不为零则 x=-c/b.
② 如果 b 为零，进一步判断 c 是否为零。
③ 如果 c=0,则变得没有用处，显然，任何 x 值都是该方程的解；
④ 如果 c≠0,则不可能求解。
（2）A≠0，如图 6-9（c）所示。

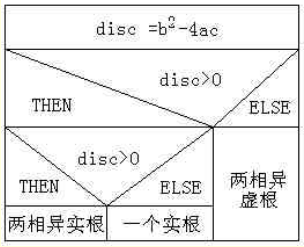


图 6-11 A≠0 时 N-S 图

① disc>0 有两相异实根
② disc=0 有一个实根
③ disc<0 有两相异虚根
N-S 图如图 6-9（d）

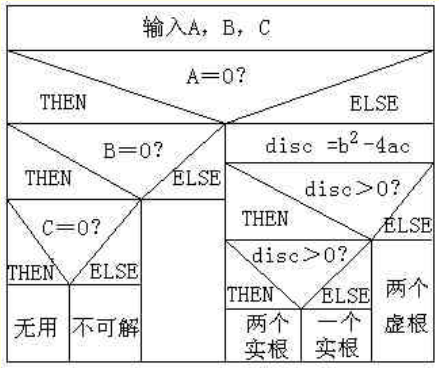


图 6-12

源程序：

```
'EXAMPLE 8
’Solve the Quadratic Equation
INPUT   "A,B,C：";A，B，C
IF   A＝0 THEN
IF B＝0 THEN
IF C＝0 THEN
PRINT   "It is trivial"
ELSE
PRINT   "It is impossible"
```

```
END IF
ELSE
PRINT   "Has one solution："
PRINT   "X="；-C/B
END IF
ELSE
DISC＝B*B-4*A*C
R＝-B/（2*A）
P＝SQR（ABS（DISC））/（2*A）
IF DISC>=0 THEN
IF DISC>0 THEN
PRINT   "Has two real roots；"
PRINT   X1＝"；R+P
PRINT   X2＝"；R-P
ELSE
PRINT   "Has one real roots："
PRINT   "X="；R
END IF
ELSE
PRINT   "Has two complex roots:"
PRINT   "X1="；R；"+"；P；"i"
PRINT   "X2"；R；"-"；P；"i"
END IF
END IF
END
```

6.5 多路选择结构

6.5.1 多分支的块 IF

在块 IF 的结构中，增加一个 ELSE IF 语句，就扩充成多分分支的块 IF 结构。

1．语句格式

```
IF〈条件 1〉THEN
ELSE IF〈条件 2〉   THEN
…}S2（块 2）
 ELSE IF〈条件 3〉    THEN          可以包括多个 ELSEIF 语句
…}S3（块 3）           和相应的语句块
…
  ELSE IF〈条件 n〉THEN
…}Sn（块 n）
ELSE
…}Sn+1（ELSE 块）
END IF
```

2．功能

当条件 1 的逻辑值为真时，就执行语句块 1（即 S1），执行完后就退出 S1，转到 END IF 语句的下一个语句

继续执行；

如果条件 1 的逻辑值为假时，就跳过 S1，执行 ELSE IF〈条件 2〉THEN 语句，当条件 2 的逻辑值为真时，就执行 S2，执行完后，退出 S2，转到 END IF 语句的下一个语句：

如果条件 2 的逻辑值为假时，就又跳过 S2，执行下一个 ELSE IF 语句。依此类推，有多少个 ELSE IF 语句，就有多少个分支，选择其一执行相应的语句块，然后退出该语句块。

如果所有条件都不满足，则程序执行就跳到 ELSE，执行 ELSE 后的语句，然后退出该语句块。

3. 举例

[例 6-9] 计算个人所得税，假设按以下规定征收：以月收入计征。

表 6-1 个人征税率

收 入 金 额 （元）	征 税 率
500 以下	免征
500~1000 以下	5%（扣除免征部分）
1000~5000 以下	10%（扣除免征部分）
5000~10000 以下	20%（扣除免征部分）
10000~20000 以下	30%（扣除免征部分）
20000 以下	45%（扣除免征部分）

算法：

6 种不同的收税标准构成 6 个条件的多路选择，用多分支的块 IF 就可写出程序。

N-S 图，如图 6-10。

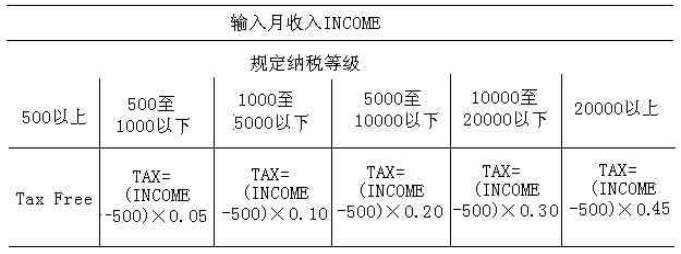


图 6-13 N-S 图

源程序：

```
'EXAMPLE 9
'INCOME TAX COMPUTATION
INPUT     "Get the income："；INCOME
IF INCOME<500 THEN
PRINT    "Tax Free"
ELSE IF INCOME<1000 THEN
Tax=(INCOME-500) * 0.05
ELSE IF INCOME<5000 THME
Tax=(INCOME-500) * 0.10
ELSE IF INCOME<1000 THEN
Tax=(INCOME-500) * 0.20
ELSE IF INCOME<20000THEN
Tax=(INCOME-500) * 0.30
ELSE
Tax=(INCOME-500) * 0.45
END IF
```

```
PRINT    "TAX="；TAX
END
```

6.5.2 SELECT CASE 语句

本语句用于执行从属于一个表达式的多个分支语句块中的一个。

1．语句格式

```
SELECT CASE〈测试表达式〉
CASE〈情况 1〉
S1（语句块 1）
CASE〈情况 2〉   可以有多个 CASE〈情况〉
S2（语句块 2）          及相应的语句块
…
CASE ELSE
Sn（语句块 n）
END SELECT
```

说明：

（1）〈测试表达式〉：可以是数值表达式或字符串表达式；

（2）〈情况 1〉、〈情况 2〉…，可以是以下格式中的任意一种：

①〈表达式〉［，〈表达式〉］…

②〈表达式〉TO〈表达式〉

③ IS〈关系运算符〉〈表达式〉

以上的〈表达式〉可以是数值类型的或字符串类型的，但它们要与〈测试表达式〉的类型一致。

2．功能

（1）SELECT CASE〈测试表达式〉是该选择结构的第一个语句，它要说明被测试的对象，用表达式表示，比如例 5.9 中，纳税人的收入多少是被测试的对象，收入多少不同，纳税就有六个等级，因此，就构成六个〈情况〉。

（2）CASE〈情况 1〉语句，就是反映测试对象的各种可能情况，当测试表达式的值，满足 CASE〈情况 1〉的条件时，就执行相应的语句块 S1，执行完后，跳到 END SELECT 的下一语句继续执行。不满足时，依次判断下一个。

在所有 CASE〈情况〉语句，如有两个及两个以上的〈情况〉条件都满足，则只执行最先满足〈情况〉条件的分支。

（1）CASE ELSE 当所有 CASE〈情况〉的条件都不满足时，执行后面的语句块，执行完后退出 END SELECT 结构，往下继续执行。

（2）END SELECT 语句，是多路选择结构的结束标帜，不产生任何操作，但它提供一个出口以便继续执行行其后的语句，每个多路选择结构有一个 SELECT CASE 语句，必须有一个 END SELECT 语句与之配对。

3．应用举例

[例 6-10] 某幼儿园儿童按年龄编班，2~3 岁的为小班，4 岁的为中班，5~6 岁的为大班，输入某儿童的年龄，计算机给出应编进哪个班级。

算法：

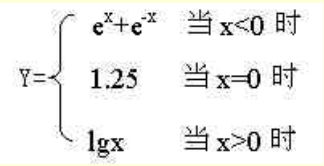
把入园儿童的年龄设为 OLD，以该变量为测试表达式，按 2~3，5~6 三种情况来应用 SELECT CASE 语句写程序。

源程序：

```
’EXAMPLE 10
INPUT How old are yor?”；OLD
SELECT CASE OLD
CASE 2，3
```

```
PTINT   "Enter lower class"
CASE 4
PRINT   "Enter Middle class"
CASE 5 TO 6
PRINT   "Enter higher class"
CASE ELSE
PRINT   "Can not enter!"
END SELECT
END
```

[例 6-11] **求函数 Y 的值：**



源程序：

```
'EXAMPLE 11
INPUT   "X="；X
SELECT CASE X
CASE IS＜0
Y＝EXP（X）+EXP（-X）
CASE 0
Y＝1.25
CASE IS＞0
Y＝LOG（X）/LOG（10）
CASE ELSE
PRINT   "It is wrong!"
END SELECT
PRINT   "Y="；Y
END
```

6.6 无条件转移语句（GOTO）

6.6.1 语句格式

GOTO〈行号丨标号〉

在 BASIC 中，只能指定一个有效的〈行号〉

在 Quick BASIC 中，还可以指定一个有效的〈标号〉

例：

```
GOTO 50（该行号应是本程序单元内存在的）
GOTO start（start 是一个标号，它应是本程序单元内存在的）
```

6.6.2 使用说明

用无条件转移语句 GOTO,可由编程人员强行指定程序的执行转移到程序中的相应位置去，这个用起来简单方便，但用得不好，容易破坏程序结构，引起混乱，不便阅读，因此，建议尽量少用，能用其他方法代替的，就不用 GOTO 语句，但是它和条件语句配合使用，构成一个循环结构，还是可以的。

6.6.3 应用例

[例 6-12] 计算某班科考试成绩的平均分

算法：

（1）用 READ 语句读入学生的考试成绩，将考试分数放在 DATA 语句中。

（2）建立一个终止标志，方法是在 Data 语句中，写完所有的分数之后，加一个“-1”，它和分数有根本区别，因为分数不可能为负数。设立终止标志的作用是控制读数操作的结束，以改变程序的流程。

（3）当读入一个数据之后，都要检查是不是-1，如果是-1，就表示已没有数据可读了，因此程序不再进行累加；如果不是-1，就表示读入的是分数，应该进行累加处理。

（4）用 GOTO 和条件语句构成循环，进行累加。

（5）全部累加完后，再求平均，打印结果。

源程序：

```
REM EXAMPLE 12
SUM＝0：N＝0
start; READ SCORE
IF SCORE＝-1 THEN 80
SUM＝SUM+SCORE
N＝N+1
GOTO start
MEAN＝INT（SUM/NJ+0.5）
PRINT    "MEAN="；MEAN
DATA 89，76，94，51，65，86，-1
END
```

这种反复执行一段程序的过程就称为循环，下一章，将进一步讨论循环结构。

6.7 多分支转移语句（ON GOTO）

6.7.1 语句格式

ON〈算术表达式〉GOTO〈行号 1〉，〈行号 2〉，…

说明

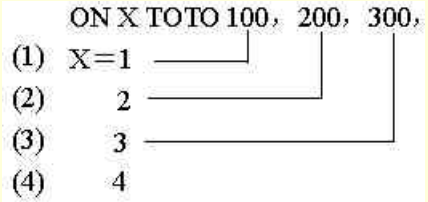
〈算术表达式〉：其值必须在 1~255 之间，若值为小数则按四舍五入取为整数；如果其值为负或大于 255，将出现错误信息。

〈行号 1〉，〈行号 2〉，…分别代表每一分支的入口行号。

6.7.2 功能

首先计算算术表达式的值，得到一个在 1~255 的整数。这个整数就是要转移到某分支行号的位置。

例：



当 X＝4 时，由于位置上没有行号，程序将转移到下一个 BASIC 语句去。

6.7.3 使用说明

ON GOTO 就象一个多路开关，按顺序选通某一通路。如图 6-11 所示。

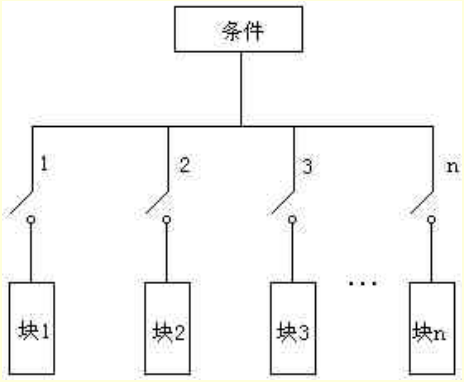


图 6-11 多分支语句

4．举例

[例 13] 在（0，60）区间有一连续函数，当给出任一个 X 值时求 Y 值，函数为：

```
Y＝30          0≤X＜10
   40-X        10≤X＜20
   20          20≤X＜30
   50-X        30≤X＜40
   10          40≤X＜50
  60-X         50≤X＜60
```

算法：

用多分支的 ON GOTO 语句处理；

源程序：

```
REM EXAMPLE 13
INPUT X
ON X/10+1 GOTO 60，70，80，90，100，110
PRINT    "X<0 OR X>60"
GOTO 130
Y=30:GOTO 120
Y=40-X:GOTO 120
Y=20:GOTO 120
Y=50-X:GOTO 120
Y=10:GOTO 120
Y=60-X
PRINT  "X="；X，"Y="；Y
END
```

6.8 本章小结

1．选择结构的功能是控制程序执行的流程。当流程发生分支时，根据一定的逻辑条件选择其中之一作为执行流程。基本的流程分支可分为两分支和多分支两类。

2．为了实现基本的两分支流程，BASIC 提供了行 IF 语句，Quick BASIC 扩展成为块 IF。

（1）行 IF 语句的格式是：

IF〈条件〉HTEN〈语句丨行号〉［ELSE〈语句丨行号〉］

它包括 3 个部分：（1）条件部分；

（2）THEN 语句部分；

（3）ELSE 语句部分。

当条件为“真”，执行 THEN 语句部分

当条件为“假”，执行 ELSE 语句部分

（2）块 IF 由 3 个语句两个组成，其格式是：

```
IF〈条件〉HTEN
S1
ELSE
S2
END IF
```

3．使用行 IF 语句时的注意事项

（1）在〈语句丨行号〉参数中，尽量不要使用〈行号〉这样可以避免发生结构上的混乱。

（2）行 IF 语句的 ELSE 部分是可选的，可以省去这一部分，这是，只对条件为真时进行处理，条件为假时，不执行该语句，程序执行转到它的下一个语句。

（3）行 IF 语句的长度不要超过 255 字符，最好写得简短些，不宜写得太长。

（4）允许行 IF 内嵌套，但不宜太深。

4．块 IF 是功能最强、结构最清楚的选择结构，因而也是一个最基本的重要结构。使用时应注意以下几点。

（1）IF 条件语句和 END IF 语句必须配对；

（2）ELSE 语句是可选的，当语句块 S2 省略时，可以省略该语句；

（3）它的两个语句块 S1、S2 可以同时出现，也可以省略其中任意一个；

（4）不允许用 GOTO 语句从块外转到 IF 块内来；

（5）允许块 IF 内再嵌套 IF 语句。

5．在条件语句中，条件是用关系表达式或逻辑表达式来表示的。条件较简单的可以只用关系表达式，条件较复杂的（如复合条件）可以用逻辑表达式，最常用的逻辑运算有 NOT、AND、OR。

6．为了实现多分支的流程，Quick BASIC 提供了一个专门处理多种情况的 SELECT CASE 语句和扩充的块 IF，在 BASIC 中则有 ON GOTO 语句。

实际上利用行 IF 语句和块 IF 的嵌套也能实现多分支的流程。

7．无条件转向语句 GOTO，在不破坏程序的结构化原则下可以使用，如它和行 IF 语句配合构成循环结构。但不提倡到处都用 GOTO 转移语句。

习　题

1．如果 X＝1，Y＝2，Z＝3 计算下列表达式的逻辑值。

（1）X＜0・1　　（2）X＞0 AND Y＞Z

（3）Z＝X+Z　　（4）SIN（X）＝COS（X）*Y

（5）4*Y/Z＜＞X　　（6）X*Y*Z＞Y∧3

（7）X+Y+Z＞0　OR　1-COS（Z）＞0

（8）Z*2＞X AND　Z+2＜Y　OR　Z/2＞X AND Z-2＜Y

2．把下列每个条件写成 BASIC 的逻辑表达式（或关系表达式）

（1）0＜X＜10　　（2）X 小于等于 Y

（3）X 和 Y 都不大于 100　　（4）X 不在 a 和 b 之间

（5）X 不等于 Y　　（6）X 不大于 Z 或不小于 Y

（7）N 大于 100 或 N＜0　（8）X＝Y＝Z

3．输入 x，编制程序计算 y 的值。

$$y=\begin{cases}1+x & x\geqslant 0\\ 1-2x & x<0\end{cases}$$

4．输入 a、b、c 三个数，要求将绝对值最大者打印出来。

5．输入 x，计算 y 的值。

$$y=\begin{cases}\ln(-x) & x<0\\ x+\ln(5+x^3) & 0\leqslant x<5\\ e^x & x\geqslant 5\end{cases}$$

6．假设邮局规定为

（1）凡不超过 20kg 的邮包，按 0.85 元/kg 计算邮费；凡超过者，按 0.15 元/kg 增收；

（2）每个邮包收手续费 0.50 元；

（3）需作快件投递者，增收平常邮费的 20%。

7．编写程序，从键盘输入 3 个不同的数，按从大到小的顺序输出。

8．编一程序，从键盘上输入两个操作数和一个运算符，，由计算机输出运算结果（运算符为：+、－、*、/）。

9．某学校评选一等奖学金的条件有以下两种：

（1）每门课程成绩在 85 分以上。

（2）所考 6 门课程成绩总分在 520 分以上。

请输入某学生 6 门课程的成绩 cj1、cj2、cj3、cj4、cj5、cj6，问他能否获得一等奖学金。

第七章　循环结构

第六章中的例 6-12 是一个由条件语句实现的循环。本章将介绍专门用于实现循环结构的 3 个语句，涉及的主要内容包括：

- 循环的概念
- WHILE 循环结构
- WHILE 循环结构的执行过程
- FOR-NEXT 循环结构
- FOR-NEXT 循环的执行过程
- DO 循环结构
- 循环结构的嵌套

循环结构是结构化程序 3 种基本结构之一，它可以充分利用计算机处理速度快的特点，把许多复杂问题的求解步骤变为计算机重复执行某些简单的操作。因此，我们必须很好的掌握它们的应用技巧。本章的难点是对于循环结构的嵌套的理解和学习。

7.1　引例

例如：求 $S=\sum_{n=1}^{100} n$ ＝1+2+3+…+100。

算法：

本题的算法已在第二章例 2-2 进行过分析，它就是把加法的操作重复一百次，这就构成了循环。现在分别用条件语句和循环语句编写程序。

源程序 1：

```
N＝0：S＝0
N＝N+1
S＝S+N
IF N＜100 THEN 20
PRINT "S="；S
END
```

源程序 2：

```
S＝0：N＝0
WHILE N＜100
N＝N+1
S＝S+N
WEND
PRINT "S="；S
END
```

源程序 3：

```
S＝0
FOR N＝1 TO 100
 S＝S+N
NEXT N
  PRINT "S="；S
END
```

程序说明：

以上 3 个程序都是完成同一算法的 3 种不同表现形式。其中，第 1 个程序是用条件语句实现的直到型循环。

第 2、3 个程序是用 WHILE-WEND 和 FOR-NEXT 实现的当型循环。

7.2 WHILE 循环语句（WHILE-WEND）

7.2.1 语句格式

WHILE〈条件〉
〈循环体〉
WEND

其中：〈条件〉：是关系表达式或逻辑表达式；
〈循环体〉：是一语句组，要重复执行的内容。

7.2.2 使用说明

（1）WHILE 循环的执行过程如图 7-1 所示。WHILE 语句首先判断条件：当条件成立时，则执行 WHILE 和 WEND 之间的循环体，然后再转到 WHILE 语句重新进行判断；当条件不成立时，就退出循环转到 WEND 的后续语句。

（2）WHILE 和 WEND 必须配对使用。

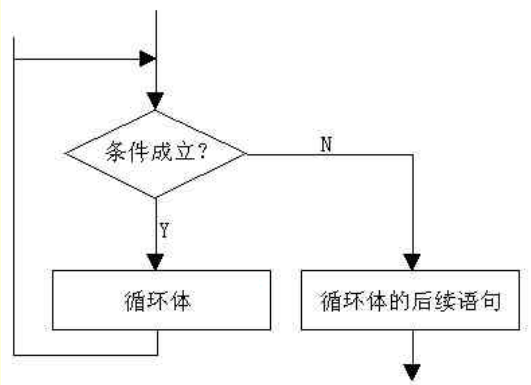


图 7-1 WHILE 循环的执行过程

7.2.3 举例

[例 7-1] 编程计算 N！的值。

源程序：

```
REM EXAMPLE 1
INPUT N
T＝1
WHILE 1＜N
1＝1+1
T＝T*H
WEND
PRINT "T="；T
END
```

运行结果：

？5

T＝120

程序说明：

当条件 1＜N **为真时进行循环。**5 **行语句为计数器，统计循环次数。**6 **行语句为累乘器，将阶乘的值存放在** T **中。**

[**例** 7-2] **有一阶梯，若每步跨** 2 **阶，则最后剩下** 1 **阶；若每步跨** 3 **阶，则最后剩下** 2 **阶；若每步跨** 5 **阶，则最后剩** 4 **阶；若每步跨** 6 **阶，最后剩下** 5 **阶；只有每步跨** 7 **阶时，最后才正好走完，**1 **阶不剩。请问这条阶梯有多少阶。**

算法 1：

由于阶梯数被 2 **除后余** 1，**所以不妨设初值** X＝3，**然后用** 3 **除** X，**若不余** 2，**则把** X **每次增加** 2，**直到最后余** 2 **为止；再用** 5 **除，若不余** 4，**则每次增加** 6（6 **是** 2 **与** 3 **的最小公倍数），直到余** 4 **为止；再用** 6 **除，若不余** 5，**则每次增加** 30（30 **是** 2、3、5 **的最小公倍数），直到余** 5 **止；再用** 7 **除，若不能整除，每次增加** 30（30 **是** 2、3、5、6 **的最小公倍数），直到整除为止。此时** X **中放的数就是要求的阶梯数，**N-S **图如图** 7-2 **所示。**

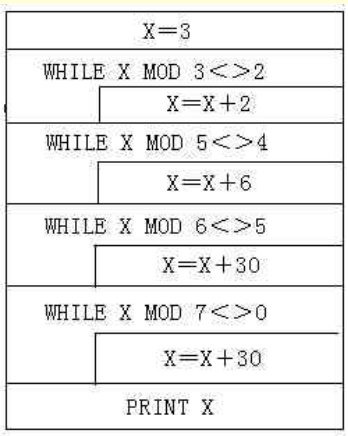


图 7-2 N-S **图**

源程序：

```
REM EXAMPLE 2A
X＝3
WHILE X MOD 3＜＞2
X＝X+2
WEND
WHILE X MOD 5＜＞4
X＝X+6
WEND
WHILE X MOD 6＜＞5
X＝X+30
WEND
WHILE X MOD7＜＞0
X＝X+30
WEND
PRINT  "X="；X
END
```

运行结果：

X＝119

算法 2：

根据题意，阶梯数能被 7 整除，设初值 X＝7，判断 X 是否满足题中的要求，如果满足，则打印 X，否则 X 再增加 7，再去判断，以此类推，直到满足要求，打印出 X。

源程序：

```
REM EXAMPLE 2B
X＝7
WHILE X MOD 2<>1 OR X MOD 3<>2
OR X MOD 5<>4 OR X MOD 6<>5
X＝X+7
WEND
PRINT    "X="；X
END
```

两种算法结果相同，算法 2 比算法 1 简单易懂。

7.3 FOR 循环语句（FOR－NEXT）

7.3.1 格式

FOR〈循环变量〉＝〈初值〉 TO 〈终值〉 [STEP 〈步长〉]
〈循环体〉
NEXT 〈循环变量〉

其中：

〈循环变量〉：又称循环控制变量，是简单变量。

〈初值〉，〈终值〉，〈步长〉：可以是整型实型的常量、变量或算术表达式。

当步长为 1 时，[STEP〈步长〉] 可以省略。

说明：

（1）FOR 语句称为循环说明语句，也是循环的入口。它的含义是循环变量的取值从初值开始，每循环一次增加一个步长，直到超过终值为止，在语句中由此决定循环的执行次数。循环执行的次数可以用下面的公式计算出来。

循环次数＝INT（（终值-初值）/步长）+1。

凡是计算出的循环次数小于等于 0 的，循环次数为 0。

（2）NEXT 语句称为循环结束语句，是循环的出口。当执行该语句时，循环变量取下一个值，即把控制循环变量的当前值增加一个步长。

（3）FOR 和 NEXT 之间的语句称为循环体，它是循环语句的主体，是需要多次重复执行的部分，可以由一个或多个语句组成。

7.3.2 执行过程

执行过程如图 7-3 所示。

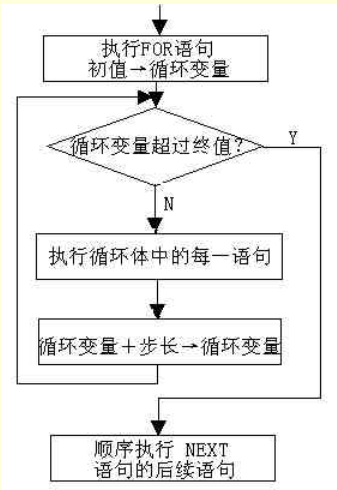


图 7-3 FOR 循环执行过程

（1）在执行 FOR 语句时，计算机首先把初值赋给循环变量，并自动记下终值和步长。

（2）将循环变量的值与终值比较，如果循环变量的当前值超过终值，则不执行循环体，执行 NEXT 的下语句。

（3）若循环变量的值未超过终值，则执行循环体。

（4）执行到 NEXT 语句，循环变量的当前值增加一个步长值再赋给变量，然后转到第 2 步。

7.3.3 使用说明

（1）FOR 语句和 NEXT 语句必须配对，而且同一循环中 FOR 语句和 NEXT 语句中的循环变量必须同名，否则就会出错。

（2）步长值可以是整数，也可以是小数，可以是正值，也可以是负值。

- 当步长值正时，在循环的执行过程中，循环变量按步长递增，直到循环变量的值变化到大于终值时，循环停止。例如：

```
FOR I＝1  TO 10  STEP  2.5
```

- 步长值为正时，终止循环的条件是终值必须大于初值。
- 如果步长值为负，在执行循环的过程中，循环变量按步长递减，当循环变量值变化到小于终值时，循环才会停止。例如：

```
FOR X＝5 TO 1 STEP -1
```

- 步长值为负时，终止循环的条件是终值必须小于初值。
- 当步值长为小数时：

```
FOR X＝1 TO 3+0.5/2 STEP 0.5
```

当步长值为小数时，实数计算会产生一定误差，为确保循环的预定次数，建议终值加上半个步长，如例 3 所示。

（3）循环变量的初值，终值和步长的选择方法。

- 如果循环变量不出现在循环体内，仅仅用来控制执行循环次数，则可随意选择初值，终值和步长，只要保证完成需要的循环次数即可。为了提高速度，最好用整数。如要循环 N 次，则用 FOR I ＝1 TO N
- 如果循环变理出在循环体内并参与运算，则应根据题目要求来选择初值，终值和步长。如例 7-3。

[例 7-3] 打印-π到π之间的正弦曲线，每隔 10 ° 打一“＊”号。

源程序：

```
REM EXAMPLE 3
PI＝3.14159
FOR X＝-PI  TO  PI+（PI/18/2） STEP  PI/18
PRINT INT（X* 180/PI）；
PRINT TAB（32+20 * SIN（X））；"*"
```

```
    NEXT X
    END
```

程序说明：

（1）根据该题要求，选择初值为-3.14159、**终值为** 3.14159+（3.14159/18/2）、**步长定为** 10 °。**终值加步长的一半是为了确保循环的正常次数。**

（2）第 5 **行语句是指定适当位置打印‘** * **’号，因为** sinx **的值是小于** 1 **的，取整后都为** 0，**所以放大** 20 **倍，这样它的坐标值才能打印；为了使图形完整地显示在屏幕中部，故将正弦曲线的** X **轴线平移** 32 **格。**

7.3.4 循环的嵌套（多重循环）

1. 嵌套的概念

循环语句的循环体中包含循环语句，称为“循环嵌套”，外层的叫外循环，在内层的叫内循环。在一个循环内包含一个循环语句的为双重循环，内循环体又可包含循环语句，构成多重循环。多重循环增加了计算处理的能力，能解决一些比较重杂的问题。

双重循环的结构为：

```
FOR 〈循环变量 1〉＝〈初值 1〉 TO 〈终值 1〉 [STEP 〈步长 1〉]
FOR 〈循环变量 2〉＝〈初值 2〉 TO 〈终值 2〉 [STEP 〈步长 2〉]
    〈循环体〉
NEXT 〈循环变量 2〉
NEXT 〈循环变量 1〉
```

[**例** 7-4] **打印九九乘法表。**N-S **图如图** 7-4。

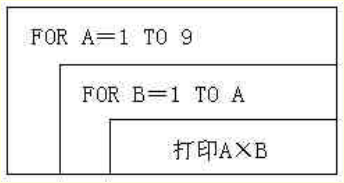


图 7-4 N-S **图**

源程序：

```
REN EXAMPLE 4
FOR A＝1 TO 9
FOR B＝1 TO A
PRINT USING "#"；A；
PRINT "*"
PRINT USING "-#=##"；B，A * B；
NEXT B
PRINT
NEXT A
END
```

运行结果：

```
1*1＝1
2*1＝2 2*2＝4
3*1＝3 3*2＝6 3*3＝9
…
9*1＝9 9*2＝18 9*3＝27 9*4＝36 9*5＝45 9*6＝54 9*7＝63 9*8＝72 9*9＝81
```

2. 嵌套的使用注意事项

（1）多重循环嵌套，一定是外循环套内循环，内外循环不能交叉，如例（a**）是正确的，（**b**）是错误的。**

内外层的变量不能同名。

（2）在同一外循环体中，可以有并列的内循环出现。对于并列循环的变量名可以相同也可以不相同，如例（c）。

```
FOR X…          FOR X…          FOR X…
FOR Y           FOR Y           FOR Y
NEXT Y          NEXT Y          NEXT Y
NEXT X…         NEXT Y…         FOR Y
                                NEXT Y
                                NEXT X…
```

（a）正确的嵌套　　（b）错误的嵌套　　（c）内循环并列的情况

7.3.5 应用举例

[例 7-5] 用多项式计算自然数 e 的近似值，用公式

$$e=\sum_{n=0}^{n}\frac{1}{n!}=1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{N!}$$

算法 1：

1．当 N 给定后，用内层循环计算每一项的分母的值；

2．用外层循环计算 N 项的和。

源程序：

```
REM EXAMPLE 5A
S＝1
INPUT    "N="；N
FOR I＝1 TO N
T＝1
FOR J＝1 TO I
T＝T * J
NEXT J
S＝S+1/T
NEXT I
PRINT   "E="；S
END
```

算法 2：

将多项式改写如下：

$$e=1+\frac{1}{1}(1+\frac{1}{2}(1+\frac{1}{3}(1+\cdots\frac{1}{N-1}(1+\frac{1}{N}))\cdots))$$

由内向外不断计算（$1+\frac{1}{N}$），其中 N 的值由 N 减到 1，从而构成一个循环。

源程序：

```
REM EXAMPLE 5B
INPUT    "N＝"；N
S＝1
FOR I＝N TO 1 STEP -1
S＝1+S/1
NEXT 1
PRINT    "E="；S
```

```
END
```

[例 7-6] **我国古代数学张邱建在《算经》里提出了一定方程问题。即“百鸡问题”：公鸡每只值 5 元，母鸡每只值 3 元，小鸡 3 只值 1 元，1 百元钱买 1 百只鸡。问公鸡、母鸡、小鸡各可买多少？**

算法：

设公鸡、母鸡、小鸡各为 x、y、z,可得方程：

$$\begin{cases} 5x+3y+\dfrac{z}{3}=100 & (1) \\ x+y+z=100 & (2) \end{cases}$$

这个不定方程式有多个答案，我们用穷举法求解。

所谓穷举法，就是对所有可能的答案一一列举，从中判断哪些答案是符合题设条件的。分析一下 X、Y、Z 的可能取值范围：公鸡最少 1 只，最多只能买 20 只，故 1＜X＜20；母鸡 1＜Y＜33；只要 X、Y 值一确定，Z＝100-X-Y。

在各种 X、Y 值的情况下，一一列举出 X、Y 所能取值的情况。因此用双层循环编写程序。外层循环控制 X（即 FOR X＝1 TO 19），内层循环控制 Y（即 FOR Y＝1 TO 33）。

源程序：

```
REM EXAMPLE 6
PRINT "'Cock","Hen","Chicken"
FOR X＝1 TO 19
FOR Y＝1 TO 33
Z＝100-X-Y                                    '满足百鸡条件
IF 5*X+3*Y+Z/3＝100 THEN PRINT X，Y，Z    '满足百鸡条件
NEXT Y
NEXT X
END
```

运算结果：

Cock	Hen	Chicken
4	18	78
8	11	81
12	4	84

[例 7-7] **计算：**

$$\sin X \approx X-\frac{X^3}{3!}+\frac{X^5}{5!}-\frac{X^7}{7!}+\frac{X^9}{9!}-\cdots\frac{X^{4n-1}}{(4n-1)!}+\frac{X^{4n+1}}{(4n+1)!}$$

算法 1：

上式可化为

$$\sin X \approx (X+\frac{X^5}{5!}+\frac{X^9}{9!}+)\cdots+\frac{X^{4n+1}}{(4n+1)!})-(\frac{X^3}{3!}+\frac{X^7}{7!})$$

因此可分别计算

$$y_1=(x+\frac{x^5}{5!}+\frac{x^9}{9!}+\cdots+\frac{x^{4n+1}}{(4n+1)!})$$

$$y_2=(\frac{x^3}{3!}+\frac{x^5}{7!}+\cdots+\frac{x^{4n-1}}{(4n-1)!})$$

分析上面两式，有共同的规律可循，即

（1）分子的指数和阶乘数相同；

（2）多项式各项指数的值是 4 的级差：y_1 的变化范围是（1~4n+1），y_2 的变化范围是（3~4n-1）。

这样，我们可以用双层循环：

（1）内层，计算每一项的分子分母；

（2）外层，计算各项之和；

（3）最后，sinX＝y_1-y_2。

N-S **图如图** 7-5 **所示。**

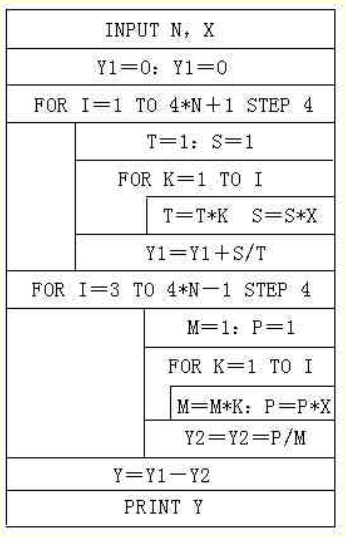


图 7-5 N-S **图**

源程序：

```
REM EXAMPLE 7A
INPUT N，X
Y1＝0：Y2＝0
FOR I＝1 TO 4*N+1 STEP  4
T＝1：S＝1
FOR I＝1 TO 1
T＝T * K：S＝S * X
NEXT K
Y1＝Y1+S/T
NEXT I
FOR I＝3 TO 4*N-H STEP  4
M＝1：P＝1
FOR K＝1 TO I
M＝M * K：P＝P * X
NEXT K
Y2＝Y2+P/M
NEXT I
Y＝Y1-Y2
PRINT  "Y="；Y
END
```

运行结果：

```
？4，5
Y＝-.9587822
？5，4
Y＝-.7568007
```

算法 2：

观察该多项式，我们会发现：①各项的符号，奇数项为正，偶数项为负；②分子的指数和阶乘数相同，各

项指数之间，都相差为 2。因此，可以把每项的计算用内层循环实现，各项求和用外层循环实现。求和时，每项乘一个符号系数 S＝（-1）*S，设初值 S＝1。这样，每执行一次，S 的值就交替为-1 和+1，从而把奇数项置成正，偶数项置成负。

源程序：

```
REM EXAMPLE 7B
INPUT   "X,N="；X，N
Y＝X；S＝1
FOR I＝2 TO N
P＝1；K＝1
FOR J＝1 TO 2*I-1
P＝P * J：K＝K * K
NEXT J
S＝（-1） * S
Y＝Y+S * P/K
NEXT I
PRINT   "sin("；X；")＝"；Y
END
```

7.4 DO 循环语句（DO－LOOP）

7.4.1 格式

DO 循环可分为“当型循环”与“直到型循环”两种，而在当型和直到型中又可分为先进行条件判断再执行循环体和先执行循环体再进行条件判断两种类型。由此构成 4 种格式如下：

表 7-1 循环类型

类 型	先判断后执行	先执行后判断
WHILE（当型）	DO WHILE－LOOP	DO－LOOP WHILE
UNTIL（直到型）	DO UNTILE－LOOP	DO－LOP UNTIL

格式 1：

```
DO WHILE〈条件〉
〈循环体〉
LOOP
```

说明：当条件成立时，执行循环体；当条件不成立时，一次也不执行循环体。

格式 2：

```
DO
〈循环体〉
LOOP WHILE〈条件〉
```

说明：首先执行循环体，再判断条件是否成立。如果条件成立，再执行循环体；如果条件不成立，就退出循环体。

格式 3：

```
DO UNTIL〈条件〉
〈循环体〉
LOOP
```

说明：当条件不成立时，执行循环体，直到条件成立时，退出循环体。

格式 4：

```
DO
〈循环体〉
LOOP UNTIL〈条件〉
```

说明：首先执行循环体，再判断条件是否成立。如果条件不成立，再执行循环体；如果条件成立，就退出循环体。

上述 4 种格式中的〈条件〉为关系表达式或逻辑表达式。

7.4.2 举例

[例 7-8] **求自然数的平方和 S，S＜N。**

源程序 1

```
'EXAMPLE 8A
INPUT N
I=0
S=0
DO WHILE S<N
I=I+1
S=S+I*1
LOOP
PRINT   "S="；S
END
```

源程序 2

```
'EXAMPLE 8B
INPUT N
I=0
S=0
DO
I=I+1
S=S+I*1
LOOP WHILE S<N
PRINT   "S="；S
END
```

源程序 3：

```
'EXAMPLE 8C
INPUT N
I=0
S=0
DO UNTIL S>=N
I=I+1
S=S+I * I
LOOP
PRNIT   "S＝"；S
END
```

源程序 4：

```
'EXAMPLE 8D
INPUT N
I=0
S=0
DO
I=I+1
S=S+I * I
LOOP UNTIL S>=N
PRINT   "S="；S
END
```

运行结果：

```
？1000
1          5      14      30      55
91         140    204     285     385
506        650    818     1015
S＝1015
```

程序说明：

该例的 4 个程序的打印结果完全相同。例 7-8A、7-8C 都是循环执行前先进行条件判断，但例 7-8A 是当条件为真时，执行循环体，而例 7-8C 是当条件为假时，执行循环体。例 7-8B、7-8D 都是执行循环体后，再进行条件判断，但例 7-8B 是当条件为真时执行循环体，而例 7-8D 是当条件为假时执行循环体。

7.4.3 DO 循环的使用说明

1．DO 循环的格式 2、3 在程序设计中可以互换，但条件要相反，功能是相同的

如例 8B~8C。

2．DO 循环的嵌套和 FOR 循环相同

[例 7-9] 求两个正整数 M、N 的最大公约数。

算法：

见第二章例 4，用欧几里得的辗转相除法求解。

源程序：

```
’EXAMPLE 9
INPUT M，N
DO WHILE M＞O AND N＞O
R＝M MOD N
DO UNTIL R＝O
M＝N
N＝R
LOOP
PRINT  "The greatest common divisor is"；N
LOOP
END
```

运行结果：

```
？84，64
The greatest common divisor is 4
```

程序说明：

（1）外层循环是一个 DO WHILE-LOOP 循环，只有 M、N 都为正数时，才进入循环体，否则停止运行。

（2）内层循环是一个 DO UNTIL-LOOP 循环，只有 R≠0 时才执行内循环体。因此，进入循环前，必须先判断一次 R，如果 R＝0，已经找到了最大公约数，就不进入内循环。

3．DO-LOOP 格式中的〈条件〉可以缺省

这时，如果循环体内无控制语句转向循环体外，将出现“死循环”。要停止死循环只有通过人工干预的办法，方法是同时按下 CTRL+BREAK 键来打断循环。程序设计中，应避免这种情况发生。

4．DO 循环和 FOR 循环一样，在使用转移语句时，遵循“许出不许入”原则

例如：

```
GOTO start
DO WHILE X＝1
X＝1
start: PRINT X
LOOP
END
```

第 1 行语句转到 DO 循环中的第四行语句，这是不合法的，若运行此程序，会显示出错信息。

5．从循环中途退出的方法

（1）用 EXIT DO 语句从 DO LOOP 中退出。

在执行 DO 循环时，除了根据循环条件的成立或不成立而退出循环外，可以用控制语句 EXIT DO 中途跳出循环。EXIT DO 为循环的出口语句。当程序执行到 EXIT DO 语句时，立即转到 LOOP 语句的后续语句继续执行。EXIT DO 语句一般要和 IF 语句一起联用。

[例 7-10] 在 100~300 之间找一个“水仙花”数。所谓水仙花数，是指一个 3 位数，各位数字的立方和等于该数的值，如 $371＝3^3+7^3+1^3$。

源程序：

```
‘EXAMPLE 10
DEFINT A－Z
N＝100
```

```
DO WHILE N＜＝300
I＝N\100                    '百位上的数字
J＝（N-I * 100）\10         '十位上的数字
K＝N-（N\10） * 10          '个位上的数字
IF IA3+JA3+KA3＝N  THEN
PRINT  "N="；N
EXIT DO                     ’找到水仙花后退出循环
END IF
N＝N+1
LOOP
END
```

（2）用 EXIT FOR 语句从 FOR NEXT 中退出。

在 FOR 循环中，除正常的通过控制循环次数来终止循环外，也可以控制语句 EXIT FOR 在 FOR 循环完成之前跳出循环，而执行循环之外的后续语句。

[例 7-11] **求立方大于 5000 的最小整数。**

源程序：

```
'REM EXAMPLE 11
FOR N=1 TO 100
IF N^3>5000 THEN EXIT FOR
NEXT N
PRINT "N^3="；N^3
END
```

运行结果：

```
N^3＝5832
```

7.4.4 应用举例

[例 7-12] **假设某乡镇企业现有产值 10000 元，如果保持年增长率为 11.25%，试问多少年后该企业的产值可以翻一番。**

算法：

本题可用复利公式计算。设现有产值为 P（Principal），增长后的产值为 V（Value of priducts），年增长率为 R（Rate），时间以年为单位，设为 Y（Years）。计算公式如下：

$$V＝P（1+R）^Y \text{ 或 } V＝P（1+R）（1+R）\cdots（1+R）\quad（Y 个）$$

由此可以构成循环结构，以 V＞＝2P 为条件，每循环一次，时间增加一年。循环结束，Y 的值便是产值翻一番所需的时间。

N-S 图如图 7-10 所示。

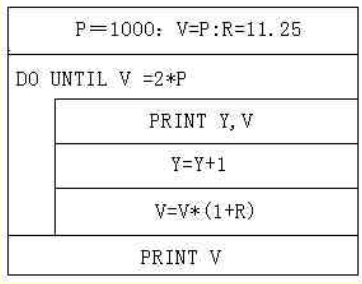


图 7-10 N-S 图

源程序：

```
'EXAMPLE 12
P=10000
V=P
R=11.25/100
PRINT   "YEARS"，"VALUES"
DO UNTIL V>=2 * P
PRINT Y,V
Y=Y+1
V=V*(1+R)
LOOP
PRINT   "IN"；Y；"YEARS IT LL HAS"；V
END
```

运行结果：

```
YEARS            VALUES
0                10000
1                11125
…                …
6                18958.33
IN 7 YEARS IT LL HAS 21091.14
```

若将条件改为“ DO WHILE V＜2 * P **”，将会出现什么结果。请读者自己分析。**

[**例** 7-13] **猜数游戏。由随机数发生器产生** 100 **以内的整数，用户从键盘上猜一个数，计算机会提示大了还是小了，直到猜准为止。**

源程序：

```
'EXAMPLE 13
'Number quesssing game
RANDOMIZE
NUMBER=INT(100*RND+1)
COUNT=0
DO
PRINT   "YOUR   QUESS"；
INPUT QUESS
COUNT＝COUNT+1
IF QUESS＞NUMBER THEN
PRINT   "TOO BIG"
ELSE IF QUESS<NUMBER THEN
PRINT   "TOO SMALL"
ELSE
PRINT   "THAT'S RIGHT!"
PRINT   "YOU NEEDED"；COUNT；"QUESSES"
END IF
LOOP UNTIL QUESS=NUMBER
END
```

[**例** 7-14] **用牛顿迭代法求代数方程** e^{-x}-x=0 **在** x_0=-2 **附近的一个实根，直到满足**｜X_1-X_0｜< **ε** =10E-06 **为止。方程的曲线如图** 7-11 **所示。**

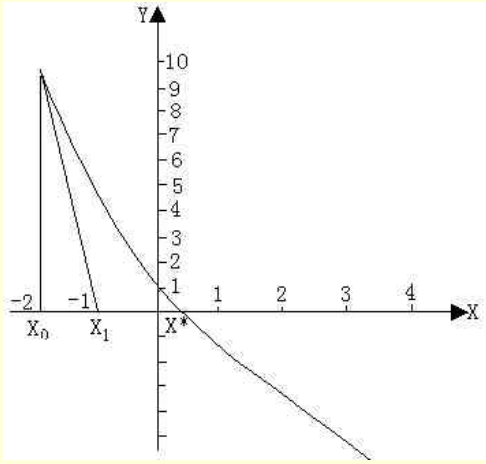


图 7-11 方程的曲线算法：

方程 f=(x)=0 的实根在几何图形上为曲线 f(x)与 x 轴的交点 x^*,可用对 f(x)求导的方法，从 x_0 开始，使方程的根逐渐逼近 x^*，这就是牛顿迭代法的基本思想。

所谓迭代（Iteration）就是重复执行一组指令，如：T＝T*I，S＝S+T，这组指令每执行一次，变量 S 就在原来的基础上增加一个 T 值。这种方法称迭代法。T、S 为迭代变量。所以，用迭代法解题，其方法是不断用新值去迭代原值，直到满足条件为止。牛顿迭代法解题的步骤如下：

（1）求 f(x)曲线在 x_0 点处的切线斜率（即 f'(x)）：

$$f'(x_0)=\frac{f(x_0)}{x_0-x_1}$$

x1o $f'(x_0)$与 x 轴的交点

则建立迭代式为：

$$x_1=x_0-\frac{f(x_0)}{f'(x_1)}$$

取 x_1 为新的近似根。

（2）判断条件｜X_1-X_0｜< ε ？，从前一个近似根可以推出下一个近似根，直到｜X_1-X_0｜< ε 为止。

变量设置：

F＝F（x）=e^{-x}-x

F1=f'(x)=-e^{-x}-1

X0＝-2

X1＝X0-F/F1

N＝N+1

E＝e＝1E-6

源程序：

```
’EXAMPLE 14
INPUT   "X0.E=";X0，E
DO
X0＝X1
N＝N+1
F＝EXP（-X0）-X0
F1＝-EXP（-X0）-1
X1＝X0-F/F1
PRINT   "X（";N；"）＝";X1
```

```
    LOOP UNTIL ABS （X1-X0）＜＝E
    END
```

运行结果：

```
X0，E＝？ -2，1E-06
X（1）＝-.80791
…
X（5）＝.5671433
X（6）＝.567143
```

7.5 本章小结

1．本章介绍的主要内容

（1）WHILE-WEND 语句

（2）FOR-NEXT 语句

（3）DO{WHILE 丨 UNTIL}-LOOP 语句

DO－LOOP{WHILE 丨 UNTIL}语句

同时还介绍了两种中途退出循环的语句：

（1）EXIT FOR 语句

（2）EXIT DO 语句

2．3 种循环结构的使用比较

（1）3 种循环结构的共同点是完成数据的重复计算和重复处理。

（2）3 种循环结构的区别在于：

①FOR 循环结构用于循环次数已知的情况。以指定的次数重复执行 FOR 与 NEXT 之间的循环体。

②WHILE 循环和 DO 循环为条件循环结构。以某一条件是否成立来控制是否重复执行循环体。

WHILE 循环语句用法较为单一，当条件为真时，执行循环体，当条件为假时，退出循环。而 DO 循环的功能更强，用法更灵活。DO 循环有当型和直到型两种：

当型 DO WHILE-LOOP：当条件成立时，执行循环体，否则退出循环。

直到型 DO UNTIL-LOOP：当条件不成立时，执行循环体，否则退出循环。

读者可根据不同情况选用不同的循环类型，以满足实际应用的需要。

习 题

1．写出下列程序的运行结果：

（1）

```
A＝3
B＝6
FOR Y＝B-A TO A+B    STEP    B/A
PRINT A,B
NEXT Y
END
```

（2）

```
    N＝1
    FOR K＝1 TO N
PRINT    " * ";
NEXT K
    PRINT
    N=N+1
        IF N<=5    THEN    20
        END
```

2．打印出 3~100 之间的所有奇数，奇数之和。

3．任意输入若干个非零，分别统计其中正、负的个数，并分别求出它们的和，输入零则结束程序。

4．编制程序打印下面的图案。

```
    *
   ***
  *****
 *******
*********
```

5．编制程序，打印输出下面的图形。

```
******
*****
****
***
**
*
```

6．编制程序，打印输出下面的图形。

```
    *
   ***
  *****
 *******
  *****
   ***
    *
```

7．猴子吃桃问题。猴子第一天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个。第二天早上又将剩下的桃子吃掉一半，又多吃了一个。以后每天早上都吃了前一天剩下的一半零一个。到第 10 天早上想再吃时，见剩下一个桃子了。试求第一天共摘多少桃子？

8．我国古代数学家张丘建“算经”里提出一个世界数学史上有名的百鸡问题：鸡翁一，值钱五，鸡母一，值钱三，鸡雏三，值钱一，百钱买百鸡，问鸡翁、母、雏各几何？

9．找出 1~100 之间的所有，“同构数”是这样的数：它出现在它的平方数的右端。例如 5 的平方数是 25，5 是 25 右端的数，5 就是同构数；25 的平方数是 625，因此 25 也是一个同构数。

10．编一程序，输出斐波那（Fibonaccii）数例 1，1，2，3，5，8，13，…前 20 项。

第八章　数　组

计算机应用在数据处理方面占有相当大的比重，而数据处理类的问题往往都有两个特点，一是数据量大；二是这些数据之间都在某些方面存在一定的内在联系。在前面章节中，由于解决问题所用的数据不多，用简单变量就可以进行存取和处理。但是在实际工作中，常常遇到大批的数据需要处理，例如学生成绩的统计、人口普查的数据处理、农业实验观测值等等。如果仍然用简单变量来存取和处理，很不方便，甚至是不可能处理的。为了解决这一问题，需要引入一个重要的数据结构—数组。本章内容：

- 数组和数组单元的基本概念
- 定义数组语句 DIM
- 静态数组和动态数组
- 利用数组进行查找数据

本章内容在的实际运用作用很突出，对于基本概念知识要熟悉。

8.1　数组和数组单元的基本概念

在现实生活中，有着各种各样的数据，这些数据在所讨论的问题中可分为两类：一类是仅与其取值有关，与其所在的位置无关；而另一类，不仅与其取值有关，并且与其所在的位置也密切相关。前面我们所讲过的变量都是简单变量，如 a，i，x 等，并可以给简单变量一个某种数据类型的数值，各个简单变量是各自独立的，与其所在的位置无关。利用简单变量可以解决不少问题，但是，如果在程序设计中仅使用简单变量，势必受到简单变量单独性和无序性的限制，而难于或无力解决那些数据不仅与取值有关，而且与其所在位置也有关的较复杂的问题，如体育比赛的成绩，就隐含着名次和成绩。要想方便地解决这类问题，通常要借助功能更强的变量—下标变量，即数组结构。

8.1.1.　数组

数组是一种最简单实用的数据结构。所谓数据结构，就是将多个变量（数据）人为地组成一定的结构，以便于处理大批量、相对有一定内在联系的数据，例如，某班有 40 名学生，考试 8 门课程，现要求将有考试成绩保存起来以供处理：显示、求总分、求每让课程的平均分、排名次等。很显然，对这 320 个原始数据用简单变量来存放，并进行相应的处理是不现实的，必须采用一种新的结构，即数组。

在 Quick BASIC 中，为了确定各数据与数组中每一单元的一一对应关系，必须给数组中这些数编号即顺序号（用下标来指出顺序号）。因此，可以说，将一组排列有序的、个数有限的变量作为一个整体，用一个统一的名字来表示，则这些有变量的全体称为数组；或者说，数组是用一个名字代表顺序排列的一组数，顺序号就是下标变量的值。而简单变量是没有序的，无所谓谁先谁后数组中的单元是有排列的顺序的。

有序性和无序性就是下标变量和简单变量之间的重要区别。

8.1.2　数组单元

在同一数组中，构成该数组的成员称为数组单元（或数组元素、下标变量）。数组里面的每一个数据用数组单元名来标识。在 Quick BASIC 中，引用数组中的某一单元，要指出数组名和用括号括起来的数组单元在数组中的位置（顺序号）的下标。因此，数组单元又称“带下标的变量”，简称“下标变量”。例如：

a（3）　　代表 a 数组中顺序号为 3 的那个单元。

x（10）　　代表 x 数组中顺序号为 10 的那个单元。

8.1.3 数组的维数

下标不变量中下标的数称为数组的维数。

如果数组中的所有元素，能按行、列顺序排成一行，也就是说，使用一个下标便可以确定它们各自所处的位置，这样的数组称为一维数组。因此，一个下标的下标变量，构成一维数组。

如果数组中的所有元素，能按行、列顺序排成一个矩阵，换句话说，必须用两个下标才能确定它们各自所处的位置，这样的数组称为二维数组。因此，两个下标的下标变量，构成二维数组。

依次类推，3 个下标的下标变量，构成三维数组。有多少个下标的下标变量，就构成多少维的数组，如四维数组、五维数组等。通常又把二维以上的数组称为多维数组。

一般来讲，数组元素的下标的个数，就是该数组的维数；反之，一个数组的维数一经确定，那么它的元素的下标的个数也就随之确定了。

a（10） 为一维数组

x（2,3） 为二维数组

b（4,5,6） 为三维数组

8.1.4 数组名的命名

为了区分各个不同的数组，必须给每个数组取一个名字。数组名的命名规则与简单变量的命名规则一样，即由 1~40 个字符组成，组成的字符可以是字母、数字或小数点，并且必须以字母开头。例如：a,x,xscj%等

8.1.5 数组类型的说明

Quick BASIC 中数据有各种类型，相应的数组也有各种类型。说明数组类型的方法有两种：

（1）用类型说明符说明数组类型。即数组名的最末一个字符以下的特定字符表示该数组的类型，见表 8-1。

表 8-1 数组类型

符 号	数 组 类 型	例 子
%	整型	x%（8）,nuber%（5,8）
&	长整型	a&（10）,deh&（2,3,5）
!	单精度型	sum!（5）,plusc!（4,6）
#	双精度型	yp#（4）,pl#（10,20）
$	字符型	a$（5）,tele$（6,10）

注意：在这 5 个符号中，只有单精度型的类型说明符“ !”可以省略，例如 sum!（5）和 plusc!（4,6）与 sum（5）和 plusc（4,6）一样都是单精度型。所以，凡是没有类型说明符的，其类型就隐含说明为单精度型。

（2）用 DIM 语句中的关键字 AS 说明数组类型，见表 8-2。

表 8-2 用类型说明词说明数组类型

语 句	定 义 数 组 类 型
INTEGER	整型
LONG	长整型
SINGLE	单精度型
DOUBLE	双精度型
STRING	字符型

例如：

将数组 x 定义为整型数组：

```
DIM x（-9 TO40）AS INTEGER
```

将数组 a 定义为长整型数组：

```
DIM a（20）AS LONG
```

将数组 numb 定义单精度型数组：

DIM numb（4,5）AS SINGLE

将数组 gszi 定义为双精度型数组：

DIM gizi（2,3,4）AS DOUBLE

将数组 tele 定义为字符型数组：

DIM tele（-2 TO 2,-3 TO 4）AS STRING

8.1.6 下标变量的使用说明

组成数组的各个元素一般均为变量，由于这些变量共同使用一个变量名，即所在的数组名，因此必须要有下标才能相互区别，故数组元素又称为下标变量。

在 Quick BASIC 中下标变量的格式为

〈数组名〉（〈一标表〉）

其中〈一标表〉是指一个或者几个下标（代表一维或者几维），各下标之间应该用逗号分隔。

在使用下标变量时有以下说明：

（1）下标变量由数组名后跟用括号起来的下标组成，例如 a（4）,b（2,3）等等。

（2）下标可以是常量、变量或表达式，例如 x（i）,a（2+n,n-1）等

（3）下标值若为非整数，系统将按四舍五入自动取整，其值的范围在-32768 到 32767 之间。例如下标变量 a（2.3），系统将下标按四舍五入自动取整后，即 a（2）进行使用。

（4）下标变量的类型就是数组的类型，所有下标变量都具有一样的类型。

（5）下标变量像简单变量一样，可以被赋值和引用。引用数组元素时的下标值应该在下限与上限之间，否则系统将显示“ Subsript out range ”（下标超出范围）的出错信息。

8.2 定义数组语句 DIM

若要使用下标变量，必须先定义数组。一个数组，包括数组名称、数组维数、数组单元数等因素。在使用数组时，要将以上内容告诉计算机，以便开辟足够的内存单元来存储数据。在 Quick BASIC 中用下面的语句来完成这一工作，称为“ 建立 ”（“ 说明 ”、“ 定义 ”）一个数组。

8.2.1 DIM 语句的格式和功能

格式：

DIM［SHARED］〈数组〉［（〈维数定义〉）］［AS〈类型说明词〉］［，〈数组名〉［（〈维数定义〉）］［AS〈类型说明词〉］］…

功能：建立一维或多维数组。

8.2.2 DIM 语句的维数定义

数组中的〈维数定义〉如下：

（1）下标变量的单元数和维数由数组名后跟用括号起来的下标的下界和上界组成。其格式为：

［〈下标下界 1〉TO］〈下标上界 1〉［，［〈下标下界 2〉TO］〈下标上界 2〉［，…］］

其中下界和上界表示该维的最小和最大下标值，通过关键字 TO 连接起来代表下标的取值范围，且下界和关键字 TO 可以省略，若省略了下界和关键字 TO，下标的取值范围是 0 到上界（若没有 OPTION BASE 语句的作用），即等价于：0 TO 上界。

例如：

DIM a（3TO5）AS LONG　　表示定义了有 3 个数组元素的一维数组 a

DIM a（-1TO10）AS LONG　　表示定义了有 12 个数组元素的一维数组 a

DIM a（10）AS LONG　　表示定义了有 11 个数组元素的一维数组 a

（2）多维数组的各个维下标界值之间，用逗号隔开。例如：

DIM X（-2 TO 2,-1.2 TO 2）　　表示定义了有 20 个数组元素的二维数组 x

DIM X（2，3）　　表示定义了有 12 个数组元素的二维数组 x

DIM X（2，3，4）　　表示定义了有 60 个数组元素的三维数组 x

（3）下标的下界默认为 0。例如：

a（20）表示数组 a 的上界为 20，下界为 0，该数组就有 a（0）,a（1）,…,a（20）,共 21 个数组元素。

b（5,5）表示数组的两个维，每维的下界为 0，上界为 5，该数组就有：

b（0,0）,b（0,1）,…b（0,5）

b（1,0）,b（1,1）,…b（1,5）

…　…

b（5,0）,b（5,1）, …b（5,5）

6 行 6 列共 36 个元素。

8.2.3　DIM 语句的使用说明

（1）在 DIM 语句中 AS〈类型说明词〉表明数据的类型，可以为基本数据类型或用户定义类型。例如：

```
DIM　a（3 TO 5）AS LONG
DIM　x（2）AS STRING,y（4）AS DOUBLE
```

也可以在数组名后加类型说明符实现相同的功能。例如：

```
DIM　a&（3 TO 5）
DIM　x$（2）,y#（4）
```

（2）SHARED 为全程属性选择项。若选择了该属性，模块中的所有过程都能共用该数组，即说明该数组变量是一个全程变量，它不同于 SHARED 语句。

（3）DIM 不但能定义说明数组，分配数组存储空间，而且还能将数组进行初始化，使得数值数组元素初始化为零，字符串元素初始化为空。

（4）DIM 语句是非执行语句，可放在程序的任何部分，但必须放在使用该数组之前，遵守“ 先说明后使用 ”的原则，否则会出现出错信息“ Array already dimensioned ”。一般认为 DIM 语句放在程序的头部较好。

DIM 语句本身不具备再定义功能，即不有直接使用 DIM 语句对已经定义了的数组进行再定义，否则会出现“ 数组已定义 ”的错误。

（5）在程序中引用数组的时候，要特别注意下标的值不能超出说明语句定义的上、下界。

例如：

```
DIM a（3）
FOR i=2 TO 4
READ a（i）
NEXT
```

在 DIM 中定义的存储单元是 a（0）、a（1）、a（2）、a（3），而下面的语句引用数组的存储单元是 a（2）、a（3）、a（4），当读到 a（4）时，下标超出定义的范围，系统统将给出超界错误信息“ Subscript out of range ”（下标出界）。

8.2.4　OPTION BASE 语句

数组中每维的下标上界数不超过 10 的情况下，可以不用 DIM 语句进行说明，系统会自动给每维分配 11 个存储单元。

例如，一维数组 a（10）有 11 个存储单元，二维数组 b（10,10）有 11×11＝121 个存储单元。

在 DIM 语句之前有下界选择语句 OPTION BASE 时，数组的下界就由该语句来确定。

格式：OPTION BASE n

功能：定义所有数组下标的下界。

说明：格式中的 n 是数字 0 或 1，即选择 0 或 1 作数组下标下界。不能随意选用其他任何数。

通常习惯用 1 作下界，则程序如下

```
OPTION BASE 1
DIM a（20）,b（5,5）
```

这样，数组 a 的下界为 1，共有 20 个元素；数组 b 的下界为 1，共有 25 个元素。

别我，如果定义的是多维数组，那么下标下界的确定对每一维都起作用。例如对三维数组 x(2,3,4)用 OPTION BASE 1 定义下标下界，那么该数组的每一维的下标下界均为 1。

```
OPTION BASE 1
DIM x（2,3,4）
```

这样定义后数组 x 的数组元素个数为 24 个。

8.2.5 求数组下标下界和上界的函数

格式：

```
LBOUND（〈数组名〉［，维］）
UBOUND（〈数组名〉［，维］）
```

功能：LBOUND 函数返回数组指定维的下标下界；UBOUND 函数返回数组指定维的下标上界。

被测试的数组必须是用 DIM 语句或者 REDIM 语句定义过的，否则系统将给出“ Array not define ”（数组未定义）的出错信息。

语句中的［，维］为任选项，指的果检测的维，其值为 1 到数组维数的一个整数。当语句中这一选项超出此范围时，系统将给出“ Subscript out of range ”（下标出界）的信息。此选项省略时，默认为第一维。

另外，一个 LBOUND 和 UBOUND 语句只能检测一个数组某一维的下标下界或者上界。

8.2.6 应用示例

[例 8-1] 将下列字符存放到数组中，并以倒序打印出来。字符是

```
a,b,q,r,s,t,w,xy,e,m,n
```

问题分析：把这 12 个字符存放在数组 a$（12）中，首先依次读取，然后利用 FOR 循环，设步长为-1，初值为 12，终值为 1 实现倒序输出。

程序为：

```
'EXAMPLE 1
REM     字符倒序输出
CLS
DIM a$（12）
PRINT "原来字符顺序为：";
FOR x=1 TO 12
READ a$（x）
PRINT a$（x）；"";
NEXT x
PRINT
PRINT "倒序输出结果为：";
FOR x=12 TO 1 STEP -1
PRINT a$（x）；"";
NEXT x
DATA a,b,q,r,s,t,w,x,y,e,m,n
END
```

程序运行结果为：

```
原来字符顺序为：a b q r s t w x y e m n
倒序输出结果为：n m e y x w t s r q b a
```

[例 8-2] 随机产生 10 个二位整数，放入数组 a，从中选出一个最大的和最小的数打印出来。

问题分析：设用 n 和 m 来存放选出的最大数和最小数。由于 m 要存放最小的数，预存放一个较大的数，其

值要大于数组中所有的数，所以赋初值为 100；n 要存放最大的数，预先存放一个较小的数，其值要小于数组中所有的数，所以赋初值为-100。

把随机产生的整数，依次与 n 和 m 比较：

若 a（i）>n，则把 a（i）的值放入 n 中；若 a（i）<m,则把 a（i）的值放入 m 中。

程序如下：

```
'EXAMPLE 2
REM          打印最大最小数
DIM a（10）
m=100:n=-100
PRINT "随机产生的 10 个二位整数是："
FOR i=1 TO 10
a（i）=INT（RND*90）+10
PRINT a（i）；
NEXT I
PRINT
FOR I＝1 TO 10
IF a（i）>n THEN n=a（i）
IF a（i）<m THEN m-a（i）
NEXT i
PRINT "最大数 n="；n,"最小数 m="；m
END
```

程序运行结果为：

```
随机产生的 10 个二位整数是：
26    39    56    90    82    91    20    18    25    48
最大数 n=   91      最小数 m=   18
```

[例 8-3]　编一程序输入 n 个数据，将其排序后按从小到大的顺序输出。

问题分析：利用数组依次输入 n 个数据，逐个进行比较，把较小数排在前面，较大数排在后面。

程序为：

```
'EXAMPLE 3
REM    从小到大排序
INPUT   "请输入 N＝"，n
DIM a（1TO n）                    '定义数组
FOR i=1TO n                       '数组的输入
PRINT "a（";i;"）=";              '显示输入第几个元素
INPUT a（i）                      ’输入
NEXT i
FOR i=1 TO n-1
FOR j=i+1 TO n
IF a（i）THEN SWAP a（i）,a（j）
NEXT j
NEXT I
PRINT   "按从小到大排序后："
FOR i=1 TO n                      '显示已排序数组
PRINT a（i）；
NEXT i
PRINT
```

```
END
```

程序运行结果为：

```
请输入 N＝ 8
a（1）= 60
a（2）= 33
a（3）= 45
a（4）= 25
a（5）= 19
a（6）= 90
a（7）= 4
a（8）= 78
按从小到大排序后：
4   19   25   33   45   60   78   90
```

[例 8-4] **有一个已按升序排序的数组，现在输入一个数，要求按原来的排序规律将它插入数组中。**

问题分析：根据题意，现在已有一个按升序排序的数组，假设该数组有 14 个元素，把这 14 个元素放在 DATA 语句中，利用 READ 语句来读取其中的内容。从键盘输入指定的数据个数 n，但不能超过 14 个，因为 DATA 语句中只有 14 个数。由于要在原数组中插入一个数，所以在用 DIM 语句定义数组单元个数时，定义为 DIMa（n+1），其下标上界为 n+1。

从 DATA 语句中顺序读入 n 个数据，放在 a（1）元素中，读完一个数即输出此数，然后从键盘输入要插入的数 x。将其依次与 a（i）相比，如果 x>a（1），则不作任何处理然后再与 a（2）相比，…，直到遇到某一个 a（i）大于 x 为止。将 a（i）以后的各处元素顺序向后移动一个位置，就完成了插入工作。

另外，还有两种特殊情况需要考虑：

（1）当 x 最小时，即 x<a（1），可以与一般情况同时考虑。

（2）当 x 比所有数都大时，即 x>a（n），则应该直接将 x 的值赋给 a（n+1）。

程序如下：

```
'EXAMPLE 4
INPUT "n=";n
DIM a（n+1）
FOR i=1 TO n
READ a（i）
PRINT a（i）；
NEXT i
PRINT
INPUT"请输入一个数：";x
FOR i=1 TO n
IF x<a（i）THEN
FOR j=n TO i STEP -1
a（j+1）=a（j）
NEXT j
EXIT FOR
END IF
IF x>a（n）THEN a（n+1）=x
NEXT i
PRINT
DATA 2,4,6,8,10,12,14,16,18,20,22,24,26,28
END
```

[例 8-5] 利用一维数组统计一个班学生 0~9、10~19、20~29、…90~99 及 100 各分数段的人数。

问题分析：定义一个有 11 个元素的一维数组 a（0 TO 10）,把 0~9 分的学生人数存入 a（10）中。

程序为:

```
'EXAMPLE 5
INPUT "请输入学生数 N＝"，n
DIM a（0 TO 10）
i=1
DO WHILE i<=n
PRINT "请输入第";i;"名学生的成绩";
INPUT x
p=INT（x/10）
a（p）=a（p）+1
i=i+1
LOOP
p=0
DO WHILE p<=10
IF P<=9 THEN
PRINT p*10;";p*10+9;"分的人数为：";a（p）
ELSE
PRINT "100 分的人数为："；a（10）
END IF
p=p+1
LOOP
END
```

[例 8-6] 有一个 n×m 的矩阵，各元素的值由键盘输入，求全部元素的平均值，并把高于平均值的元素以及它们的行列号打印出来。

问题分析：首先应定义一个 n×m 的二维数组 a（n,m）,用双重循环输入二维数组的各元素值，并将这些值进行累加，由此可以求得平均值 av（累加和/总元素数）。然后按行 i、列 j 逐个元素判断其值 a（i,j）是否大于 av，如果为真，则输出其值 a（i,j）、所在行 i、所在列 j。

程序如下：

```
'EXAMPLE 6
OPTION BASE 1
INPUT "请输入行列数 n,m=";n,m
PRINT "这是一个有";n;"行";m;"列组成的矩阵"
DIM a（n,m）
s=0
PRINT  "请输入各元素的值："
FOR i=1 TO n
FOR j=1 TO m
PRINT "a（";i;",";j;"）=";
INPUT a（i,j）
s=s+a（i,j）
NEXT j
NEXT i
av=s/（n*m）
PRINT "高于平均值的元素"，"行号"，"列号"
```

```
FOR i=1 TO n
FOR j=1 TO m
IF a（i,j）>av THEN PRINTa（i,j）,i,j
NEXT j
NEXT i
END
```

程序运行结果为：

```
请输入行列数 n,m= ? 2,3
这是一个有 2 行 3 列组成的矩阵
请输入元素的值：
a（1,1）= 10
a（1,2）= 20
a（1,3）= 30
a（2,1）= 20
a（2,2）= 10
a（2,3）= 30
高于平均值的元素        行号        列号
30                      1           3
30                      2           3
```

[例 8-7] **将两个 3×3 的矩阵 a 和 b 相加。即将相应位置上的元素相加，放到 c 数组的相应位置上。**

问题分析：本题需要先定义 3 个二维数组，每个二维数组的行和列都为 3，即 a（3,3）、b（3,3）和 c（3,3）。根据题意，数组 a（3,3）和数组 b（3,3）的各元素值是已知的，只需用 READ 语句从 DATA 语句中读出即可。相应位置上的元素相加，放到 c 数组上的相应位置，其实就是将 a（i,j）+b（i,j）的值赋给 c（i,j）。

程序如下：

```
'EXAMPLE 7
OPEION BASE 1
DIM a（3,3）,b（3,3）,c（3,3）
PRINT "数组 a:"
FOR i=1 TO 3
FOR j=1 TO 3
READ a（i,j）
PRINT a（i,j）,
NEXT j
PEINT
NEXT i
PRINT
PRINT "数组 b:"
FOR i=1 TO 3
FOR j=1 TO 3
READ b（i,j）
PRINT b（i,j）,
NEXT j
PRINT
NEXT i
PRINT
PRINT "数组 c:"
```

```
FOR i=1 TO 3
FOR j=1 TO 3
c（i,j）=a（i,j）+b（i,j）,
NEXT j
PRINT
NEXT i
DATA 1,2,3,4,5,6,7,8,9
DATA 2,4,6,8,10,12,14,16,18
END
```

程序运行结果为：

数组 a:

```
1       2       3
4       5       6
7       8       9
```

数组 b:

```
2       4       6
8       10      12
14      16      18
```

数组 c：

```
3       6       9
12      15      18
21      24      27
```

8.3 静态数组和动态数组

在 Quick BASIC **中，数组的应用灵活方便，其主要原因就是在静态数组之外引进了动态数组的概念。我们知道，计算机的内存是有限的，程序在运行的过程中，数据占了很大一部分存储空间，有时会使得一些程序因内存不够而不能正常运行。**Quick BASIC **中的动态数组，通过在程序运行的过程中动态地分配与释放内存，更有效地利用内存空间，使得** Quick BASIC **的功能进一步得到加强和完善。**

8.3.1 静态数组和动态数组

1．静态数组

计算机在程序执行前，系统进行编译的时候，根据数组说明语句开辟的固定存储空间，直到程序执行完毕，在整个过程中不再改变，这种数组就叫静态数组。

在静态数组中，数组一经定义直到程序运行结束，该数组的维数和大小不再改变，因此，它所占用的内存单元也不改变。这是各种高级语言有关数组所共同具有的基本特点。

在 Quick BASIC **语言中规定，在用** DIM **语句定义数组的时候，如果下标的上下界用的是常数，则这样的数组是静态数组。**

2．动态数组

计算机在执行过程中才给数组开辟存储空间，当不需要时，可以用 ERASE **语句删除它，收回分配给它的存储空间，还可以用** REDIM（**或** DIM）**语句再次分配存储空间。这样的数组就叫动态数组。**

Quick BASIC **语言规定，在用** DIM **语句定义数组的时侯，如果下标的上下界用的是变量或者表达式，这样的数组为动态数组。动态数组是** Quick BASIC **语言对数组功能的扩充。**

3．判断静态数组和动态数组的方法

在程序未运行时，动态数组不占用内存，可以将这部分存储空间移作他用。而对于静态数组，编译程序将

为它在 Quick BASIC 数据段据预留空间。

判断静态数组和动态数组的方法主要有：

- 数值常量和 CONST 语句中说明的常量定维的数组是静态数组。
- 用变量作为下标定维的数组是动态数组。

用变量它维的动态数组，在使用 DIM 语句说明前必须先对变量赋值，否则会出错。

静态数组的大小不能超过 64KB，如果超过会发生“Overflow”的错误。动态数组可以超过 64KB，只是在进入 Quick BASIC 环境时必须使用/ah 命令行参数。

8.3.2 数组的释放语句 ERASE

格式：

ERASE〈数组名〉［，〈数组名〉］…

功能：重新初始化静态数组元素，释放动态数组的存储空间。

说明：

（1）该语句对静态数组和动态数组的作用不一样。

对于静态数组不能删除，只能进行初始化，即把数组的数值元素置为 0，或把字符串元素置为空；

对于动态数组就能删除，释放该数组所占用的内存，释放后的内存可以重新定维。

（2）DIM 语句本身不具备再定义的功能，但当动态数组通过 ERASE 语句操作，释放数组空间后，就可以用 DIM 语句对该数组进行再定义。

例如：

```
DIM x（20,20）              '假设 x 是动态数组
…
ERASE x                     '释放 x 数组的存储空间
DIM x（15,30）              '重新进行分配
```

8.3.3 重新定维语句 REDIM

格式：

REDIM［SHARED］〈数组名〉［（〈维数定义〉）］［AS〈类型说明词〉］［，〈数组名〉［（〈维数定义〉）］［AS〈类型说明词〉］］…

功能：重新定义已为动态数组分配的空间。

说明：

（1）使用 REDIM 重新定维时可以改变每维的大小，但不能改变数组的维数。

（2）用 REDIM 语句进行重新定维时，必须首先用 ERASE 语句释放已分配的存储空间。

[例 8-8] 有如下程序，判断其运行后的输出结果。

```
'EXAMPLE 8
n=20
DIM x（n）
ERASE x
REDIM x（30）
x（n+2）=n+2:x（n+5）=n+5
PRINT x（22）；x（25）
END
```

问题分析：在 Quick BASIC 中，可以对动态数组用 REDIM 语句进行重定维操作，还可用 ERASE 语句（释放语句）释放数组所占据的空间。该空间可以用来建立一个新的数组，也可以建立一个同名的数组，维数和大小均可重新指定。

本题中各语句的作用见下：

```
n=20                              '给变量 n 赋值为 20
DIM x（n）                          '定义一个一维动态数组 x
ERASE x                           '用 ERASE 语句释放数组 x 所占据的空间
REDIM x（30）                       '用 REDIM 语句重定义了一维数组 x，其下标上界为 30
x（n+2）=n+2:x（n+5）=n+5            '给 x（22）及 x（25）这两个数组元素分别赋值 22 和 25
PRINT x（22）;x（25）                '输出 x（22）的值 22，x（25）的值 25
END
```

程序运行后输出结果为：

```
22          25
```

8.4 利用数组进行查找数据

程序设计中常常会遇到在一批数据中找出其中某一元素。例如在一个学校的学生姓名中，查找某一学生。查找的方法有两种：一是顺序查找，二是折半查找。

8.4.1 顺序查找

顺序查找（又称线性查找），把要查找的元素与整个数组的元素逐一比较，直到发现有与被查找元素相同的为止。

[例 8-9] **有一批数：**53，76，32，25，81，4，108，64，**找出数据是** 32 **的数。**

```
'EXAMPLE 9
CLS
INPUT"请输入这一批数的数量：";n
DIM num（1 TO n）
INPUT"你找什么数？";find
FOR i=1 TO n                              '输入每个数值
INPUT num（i）
NEXT i
flag=0
FOR i=1 TO n
IF num（i）=find THEN
PRINT find:"已被找到"
flag=1
EXIT FOR
END IF
NEXT i
IF flag=0 THEN PRINT "没有找到";find
END
```

8.4.2 折半查找

折半查找又叫二分查找，使用这种方法查找必须要把数组先排序。

[例 8-10] **将例** 8-9 **中的数采用折半法查找。**

首先将这 8 **个数排列成如下的顺序，例如要查找** 32，**查找步骤见如图** 8-1 **所示：**

折半查找的思路是：

（1**）把整个数组对分，取第一个元素的下标为** low，**最后一个元素的下标为** high,**居中的元素为** mid,**则** mid=INT（(low+high）/2)。

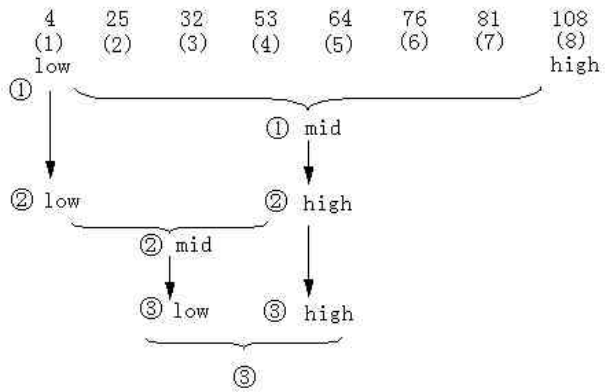


图 8-1 折半查找

将 mid 位置上的元素与被找的元素 find 比较，有 3 种情况：

- 中间元素大于 find，去掉后一半，再从前一半中查找，把被找的区间缩短，这时新的区间最后一个元素的下标应该用 mid 代换 high。
- 中间元素小于 find，去掉前一半，再从后一半中查找，把被找的区间缩短，这时新的区间开始一个元素的下标应该用 mid 代换 low。
- 中间元素等于 find，这正是要查找的对象，这时就算找到了。

（2）如果发生前两种情况之一，查找的过程继续直到没有可查找的元素，或出现第三种情况为止，本例经过 3 次缩短区间，得到要查找的元素。

```
'EXAMPLE 10
DIM num（1 TO 8）
CLS
INPUT"你想找什么数？";find
FOR i=1 TO 8
READ num（i）
NEXT i
low=1
high=8
WHILE ABS（low-high）>1
mid=INT（（low+high）/2）
IF find<num（mid）THEN
high=mid
ELSEIF find>num（mid） THEN
low=mid
ELSE
low=high
END IF
WEND
IF low=high THEN
PRINT "num（";mid;"）是找到的数为";find
ELSE
PRINT find;"没有找到"
END IF
DATA4，25，32，53，64，78，81，108
```

```
END
```

程序说明：

顺序查找，在最好的情况下，只需比较一次，即第一个元素；在最坏情况下，需要比较所有元素，因此平均的比较次数是数组元素的一半。对于大型数组，这种查找可能要花费相当长的时间，但是对于未排序的数组，只有用顺序查找。

二分查找并不查找数组中的各个元素，它比顺序查找要快得多，但它只适合于已排序好了的数组。

8.5 本章小结

1．数组是相同类型数据的有序集合，用一个数组名代表数据的全体，用下标区别同一数组中的各个元素，数组元素又称下标变量，下标变量中下标个数就是该数组的维。

2．常用的数组多为一维数组、二维数组，可以有更高维的数组，在 BASICA **中，最高维不超过** 255**，在** Quick Basic **中，最高维不超过** 60**。**

3．在 BASIC **中数组必须说明后使用，说明数组的维数和每维的大小（也即上下界）使用** DIM **语句。**

在 BASIC **中，维的下界不能为负数，选择** Option BASIC **语句的时候，隐含下界为** 0**。**

在 Quick BASIC **中用关键字** TO**，可以把下界值扩充到**-32768~32767**。**

4．数组可以分为静态数组和动态数组。一般用常量说明维的大小的数组是静态数组，用变量说明维的大小的数组是动态数组。

5．数组经过定维说明，只是在内存中开辟了一片连续的数据存储区。

6．数组的输入输出或处理，和循环结构分不开，通常用单循环作一维数组的处理，用双循环作二维数组的处理。

7.数组的处理,是通过下标变量进行的,因此,下标不能超出数组定义的范围,否则会发生下标出界(Subscript out of range**）的错误。**

8．数组是很重要的一种数据结构，为处理大批量数据提供了方便，对数组的操作有很多有用的算法，如矩阵运算、排序、查找等，应通过实例很好掌握。

习 题

1．任给一个实数矩阵 Am×n（m<100,n<100）,**试找出其中一个最大元素。**

2．设某学校某班共 10 **名学生，为了评定某门课程的奖学金，按规定超过全班平均成绩** 10%**者发给一等奖，超过全班成绩** 5%**者发给二等奖。试设计程序，打印出应获奖学金的学生名单（包括姓名、学号、成绩、奖学金等级）。**

3．求矩阵 A **的转置矩阵** A'**。**

4．假设 5 **个百货公司一月内销售电视机的情况如下表：**

表 8-3 **百货公司一月内销售电视机的情况表**

商店	兰花牌	梅花牌	菊花牌
第一百货公司	50	80	20
第二百货公司	105	92	20
第三百货公司	20	65	32
第四百货公司	80	150	98
第五百货公司	15	36	110

另外，假设各牌号电视机的价格如下：

表 8-4　各牌号电视机的价格

牌号	价格（元）
兰花牌	2500
梅花	3200
菊花牌	2850

试计算各百货公司的营业额。

5．某地区人口普查处理中，需要统计各种年龄下的人口数，以便制定合理的人口发展战略。若已知人类现有最高年龄为 150 岁，试设计程序，求以 1，2，3，…，150 岁为条件下的人口数。

6．在任给 n 个实数（n<2000），试找出其中的众数（即出现次数最多者）、众数的次数、众数频率（即其次数与总次数之比）以及众数首次出现的位置。

7．设某市某宾馆共有 3 层，每层共有 2 排各 5 个客房。试为该宾馆设计程序来统计住宿旅客总人数以及查询某一房间住宿旅客人数。

第九章　函数与子程序

一个较大的程序一般分为若干个程序模块，每一个模块用来实现一个或几个特定的功能，这就是模块化的程序设计思想。由于模块就是能完成程序全部功能或部分功能的独立的源程序代码，也就是子程序或函数。本章重点内容有：

- 模块化程序设计的概念
- 自定义函数
- 独立模块的子程序

9.1　模块化程序设计的概念

在编制程序时，经常遇到这样的情况，有些运算经常重复进行，或者许多人的程序中都可能要进行同类的运算操作。这些重复运算的程序是相同的，只不过每次都以不同的参数进行重复罢了。如果多注重书写执行这一功能的程序段，将使程序变得很长，多占存储空间，繁琐而又容易出错，并且调试起来比较困难。

解决这类问题的有效办法是将上述重复使用的程序，设计成能够完成一定功能的可供其他程序使用（调用）的相对独立的功能模块。它独立出去，即使只执行一次的程序段也可以把它写成独立模块，并把程序应该完成的主要功能都分配给各模块去完成，用主程序把各独立模块联系在一起。

这种设计方法是各种高级语言程序设计中的基本方法，好自顶向下、逐步细化和模块化。其中模块化的具体做法是：将一个大型程序按照其功能分解成若干个相对独立的功能模块，然后再分别进行设计，最后把这些功能模块按照层次关系进行组装。

使用独立模块的优点有：

（1）消除重复的程序行。可以一次性定义一个独立模块并可由其他程序任意调用。

（2）程序容易阅读。分解为一组较小的程序容易阅读和理解。

（3）使程序开发过程简化。独立模块容易设计、编写和调试。

（4）可以在其他程序中重用。可以把具有通用性的独立模块用在其他程序设计项目中。

（5）使 Quick BASIC 语言得到扩充。独立模块可以完成内语句和函数不能直接完成的任务。

独立模块由顺序、选择、循环这 3 种基本结构所组成，但它却有自己的特点，主要体现在主程序与独立模块之间的数据输入、输出，好主程序与各模块之间的数据传递。

由于模块是通过执行一组语句来完成一个特定的操作过程，所以模块又称为“过程”，执行一个过程就是调用一个子程序或函数模块。

Quick BASIC 中功能模块统称为过程，分为函数（FUNCTION）过程和子程序（SUB）过程。

9.2　自定义函数

Quick BASIC 语言提供了两种形式的函数。

1．系统提供的标准函数

用户可以直接调用而不必预先定义它，如 SIN、INT、SQR 函数等就是系统提供的标准函数。标准函数其实就是由 Quick BASIC 系统编成的一个个子程序，用户在使用时只需写出它们的名字（即函数名）和自变量就可以直接引用。例如，要计算某数 x 的平方根，只要写出 SQR（x）即可。

Quick BASIC 中标准函数调用的一般格式为：

格式：函数名［（〈函数参数〉）］

标准函数的使用说明：

（1）函数名是指标准函数的名字，如 SQR、SIN、EXP、FIX、RND 等。

（2）〈函数参数〉就是函数的自变量，又称“形式参数”，可以是一个常量、变量或表达式。如 SIN（3+4）、SQR（4*8-2/5）等，自变量必须用括弧括起来。

（3）一个数学函数的值是一个数值，它可以参加运算，如：

```
2+SQR（8）+EXP（2）/FIX（-3.48*6）
```

但标准函数不能单独作为一个语句，如下面的写法是非法的 Quick BASIC 语句：

```
INT（RND*100+1）
```

它只是一个自变量为一个算术表达式的标准函数调用。

Quick BASIC 标准函数包括数值函数、字符串函数和系统函数等，各有其不同的功能。具体函数名和功能见附录。

2．用户自己定义的函数

Quick BASIC 语言同其他高级语言一样，允许用户根据实际问题的需要以一定的形式定义自己的专用函数，并且在程序中像调用标准函数一样方便地调用它。

Quick BASIC 中的自定义函数又分为两种：

（1）模块内使用的自定义函数—DEF 函数，它只在本模块有效。

（2）作为一个独立模块的外部函数—FUNCTION 函数，它是一个独立的模块，而不是其他模块的一部分。

9.2.1 DEF 函数

DEF 函数又称为模块内使用的自调用函数，它只在本模块内有效。

1．单行 DEF 函数

格式：

```
DEF fn〈函数名〉［(〈参数表〉)］＝〈函数表达式〉
```

功能：定义一个在本模块内有效的单行 DEF 函数。

说明：

（1）〈函数名〉是用户为此函数确定的函数名，它遵循变量名的命名规则。

（2）〈参数表〉用以指定函数表达式中的自变量，它可以是一个，也可以是几个，各参数之间用逗号分隔，每一个参数的格式为：

〈变量名〉［AS〈类型〉］

类型为变量的数据类型。

在调用时应代入实际的参数值。

（3）〈函数表达式〉是所定义函数的具体表达形式，它的类型应该与左边所定义函数的类型一致。

（4）自定义函数由两部分组成，例如定义一个求圆面积的函数 $s=\pi r^2$:

```
DEF fns(r)=3.14*r^2
```

其他 fn 是自变量的标识，不可缺少，即该语句定义的自定义函数名是 fns，而不是 s。在调用自定义函数时应将函数名的两部分及函数自变量的实际值同时写上。如上例中应写 fns（10），不能写 s(10)。

（5）自定义函数可以没有参数，例如：

```
DEF fna$="I am a student."
PRINT fna$
END
```

运行时输出为

```
I am a student.
```

[例 9-1] 求半径为 1，10，0，5，8 时的圆面积。

问题分析：题中半径给定为 1，10，20，5，8，要求分别求出相应的圆面积值。由于计算圆面积的方法都是一样的，我们将其设计成自定义函数 fns。程序如下：

```
'EXAMPLE 1
DEF fns(r)=3.14*r^2
FOR i=1 TO 5
READ r
PRINT "半径为"；r；"时的圆面积为"；fns(r)
NEXT i
DATA 1,10,20,5,8
END
```

程序运行结果为：

```
半径为  1  时的圆面积为  3.14
半径为  10  时的圆面积为  314.00
半径为  20  时的圆面积为  1256.00
半径为  5  时的圆面积为  78.50
半径为  8  时的圆面积为  447.45
```

[例 9-2]　现有两个矩形，已知一个矩形的长和宽分别是 2 和 3，另一个矩形的长和宽分别是 10 和 20，分别求这两种矩形的面积及其和。

程序如下

```
'EXAMPLE 2
DEF fnj(c,k)=c*k
PRINT "长和宽为 2 和 3 时矩形面积是";fnj(2,3)
PRINT "长和宽为 10 和 20 时的矩形面积是";fnj(10,20)
s=fnj(2,3)+fnj(10,20)
PRINT "两矩形的面积和是";s
END
```

程序运行结果为

```
长和宽为 2 和 3 时的矩形面积是      6
长和宽为 10 和 20 时的矩形面积是      200
两矩形的面积和是     206
```

[例 9-3]　已知三角形的三边长为 10，20，30，求该三角形的面积。

程序如下

```
'EXAMPLE 3
DEF fns(a,b,c)=SQR(s*(s-a)*(s-b)*(s-c))
a=20
b=30
c=40
s=(a+b+c)/2
PRINT"三角形的面积为"；fns(a,b,c)
END
```

程序运行结果为

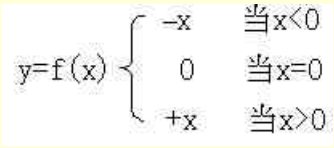
```
三角形的面积为     290.47
```

本例题中在定义函数时，表达式中有 4 个变量，3 个是函数自变量 a,b,和 c,一个是程序变量 s。在调用函数时只需给出变量 a,b 和 c 的。s 的值由程序中的赋值语句进行赋值。另外，还要注意在调用函数 fns 时，参数的个数应和定义函数时参数的个数相同，例如，在语句 DEF fns(a,b,c)=SQR(s*(s-a)*(s-b)*(s-c))中参数个数为 3 个，在调用时不能写成：

```
PRINT   "三角形的面积为"；fns(a,b,s)
```

2. 多行 DEF 函数

简单的函数（用一个表达式表示的函数）是可以用单行 DEF 语句来定义的，因为函数可以在一行内写完。但是有的问题要求定义的函数是难以用一行来定义的（无用一个表达式个来表示），例如：



该函数就无法用单行 DEF 语句来定义。

为此，在 Quick BASIC 中提供了多行 DEF 来定义这类函数。

定义多行 DEF 函数的方法为

格式：

```
DEF  fn〈函数名〉［（〈参数表〉）］
…
…
fn〈函数名〉=〈表达式〉
END DEF
```

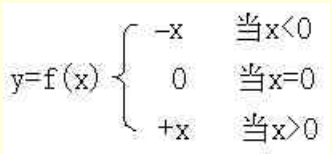
功能：定义一个在本模块内有效的多行 DEF 函数。

说明：

（1）格式中的 DEFfn 和 END DEF 必须配套使用。

（2）〈函数名〉、〈参数表〉的含义与单行 DEF 语句相同。

[例 9-4] 利用多行 DEF 函数，从键盘上输入 x 的值，求出下面分段函数中的 y 值。



程序如下:

```
'EXAMPLE 4
DEF fny(x)
IF x<0 THEN
fny=-x
ELSEIF x=0 THEN
fny=0
ELSE
fny=x
END IF
END DEF
INPUT "X=";x
PRINT "Y=";fny(x)
END
```

程序运行结果为

```
X＝？ -5
Y＝ 5
X＝？ 10
Y＝ 10
X＝？ 0
Y＝0
```

[例 9-5] 利用多行 DEF 函数求 s=1+3+5+7+…+n 的和。

问题分析：题中要求我们求出 1 到 n 之间的奇数和。n 由键盘输入。

程序如下

```
'EXAMPLE 5
DEF fns(n)
s=0
FOR k=1 TO n STEP 2
s=s+k
NEXT k
fns=s
END DEF
INPUT "N=";n
PRINT "s=1+3+5+7+…+n 的和为"；fns(n)
END
```

程序运行结果为：

```
N= ？ 7
s=1+3+5+7+…+n 的和为  16
N＝ ？ 9
s=1+3+5+7+…+n 的和为  25
N＝ ？ 100
s=1+3+5+7+…+n 的和为  2500
```

3．DEF 函数的使用规则

在 Quick BASIC 中 DEF 函数的使用规则如下：

（1）先定义后调用。如果程序中要使用自定义函数，那么必须在调用之前进行定义，否则，将会出现‘ Function not defined ”（函数未定义）的错误信息。因此，习惯上都把 DEF fn 语句放在程序或者程序段的开头。

（2）定义自定义函数的时候，〈函数表达式〉中的自变量只是一个“ 形式参数 ”（或者称“ 虚拟参数 ”），它并不是一个实际存在的变量。因此它使用什么名字对自定义函数本身并不产生影响，也可以和程序中的变量同名，但它们并不代表同一对象。

（3）调用自定义函数的时候，必须以“ 实际参数 ”（一个确定的值）代替“ 形式参数 ”，这个过程叫“ 虚实结合 ”，实际参数必须与形式参数的个数相同。“ 实际参数 ” 简称“ 实参 ”，“ 形式参数 ” 简称“ 形参 ”。

（4）函数可以嵌套定义，也就是定义一个函数时，此函数内可以出另一个已经定义了的函数；函数可以嵌套调用，也就是一次调用函数时，实参可以是另一次调用的结果。

（5）使用多行 DEF 函数的进修，要注意 DEF fn 和 END DEF 必须配套使用；格式中必须有“ fn〈函数名〉＝〈表达式〉” 语句给自定义函数赋值。

9.2.2 FUNCTION 函数

FUMCTION 函数也叫模块化函数或者是函数过程。它不像 DEF 函数是设置在某一个程序模块之中的，而是可以作为独立的模块使用。

1．定义函数过程语句 FUNCTION…END FUNCTION

格式：

```
FUNCTION〈过程名〉［（〈参数表〉）］［STATIC］
［〈语句块 1〉］
〈过程名〉＝〈表达式〉
［〈语句块 2〉］
END FUNCTION
```

功能：定义一个 FUNCTION 函数过程，可以是一个独立模块。

说明：

（1）〈过程名〉即函数过程的名字，并且用指定的数据类型后缀返回它的数据类型，数据类型后缀为%、&、!、#或$。

（2）〈参数表〉中的参数是形式参数，不能用定长字符串变量或定长字符串数组作为形式参数。

（3）STATIC **定义过程中局部变量在内存中的存储方式。如果使用** STATIC **属性，则过程中的局部变量是** STATIC**（静态）的，即在每次调用过程时，各局部变量初始化为** 0 **或空字符串。**

（4）〈表达式〉的值是函数返回的结果。调用 FUNCTION **过程要返回一个值，因此可以像内部函数一样在表达式中使用。**FUNCTION **过程返回一个值，放在格式中的〈表达式〉中，并通过语句〈表达式〉语句，则该过程返回一个默认值，数值函数过程返回** 0**，字符串函数过程返回字符串。因此，为了能使一个** FUNCTION **过程完成所指定的操作，通常要在过程中为过程名赋值。**

（5）〈语句块〉是 Quick BASIC **的程序段，语句块中可以用一个或多个** EXIT FUNCTION **语句从函数中退出。**

2. **调用函数过程**

由于 FUNCTION **过程返回一个值，在调用时完全可以像使用** Quick BASIC **内部函数一样对待，只不过内部函数由软件公司提供，而** FUNCTION **过程由用户自己定义。**

调用 FUNCTION **过程很简单，像使用内部函数一样，把它写在表达式中就可以了。**

[**例** 9-6] **求** 1~6 **的立方数。**

主程序为：

```
'EXAMPLE 6
DECLARE FUNCTION cube!(v!)
FOR x=1 TO 6
PRINT x,cube(x)
NEXT x
END
```

定义计算立方数的 FUNCTION **函数：**

```
FUNCTION cube(v)                                    '计算立方函数
vt=v*v*v
cube=vt
END FUNCTION
```

程序运行后输出结果为

```
1             1
2             8
3            27
4            64
5           125
6           216
```

[**例** 9-7] **定义随机整数函数，产生** 30 **个** 1~100 **之内的随机数。**

主程序为：

```
'EXAMPLE 7
DECLARE FUNCTION randomnum%()
FOR r=1 TO 30
PRINT randomnum%                              '输出产生的随机整数
NEXT r
END
```

定义随机整数函数：

```
FUNCTION randomnum%                           '定义随机整数函数
```

```
RANDOMIZE TIMER
randomnum%=(RND*100+1)\1
END FUNCTION
```

[例 9-8] 任意输入两个数，输出这两个数中的最大数。

主程序为：

```
'EXAMPLE 8
DECLARE FUNCTION max!(x!,y!)
INPUT "请输入两数：";a,b
PRINT "这两个数中的最大数是：";max(a,b)
END
```

定义求两个数中的最大数函数：

```
FUNCTION max(x,y)                    '定义求两数最大数函数
IF x>y THEN
max=x
ELSE
max=y
END IF
END FUNCTION
```

程序运行结果为：

```
请输入两个数：？2，3
这两个数中的最大数是：3
请输入两个数：？60，39
这两个数中的最大数是：60
请输入两个数：？44，80
这两个数中的最大数是：80
```

9.3 子程序

实现模块化的主要手段是自定义函数和子程序。但是无论是单行和多行 DEF 函数，还是 FUNCTION 函数过程，它们的共同点都是返回函数值。但是，有时需要重复的运算只是一个过程，这时就可以采用子程序。

在程序设计中，通常将重复使用的程序，设计成能够完成一定功能的可供其他程序使用（调用）的相对独立的程序段，这种程序段一般称为子程序。它独立存在，但可以被多次调用，调用的程序称为主程序。

Quick BASIC 语言中有两种形式的子程序，即 GOSUB 子程序和 SUB 子程序。其中 GOSUB 语句是沿袭早期的 BASIC 语言版本，而 SUB 则是 Quick BASIC 对早期 BASIC 版本的重要扩展。

9.3.1 GOSUB 子程序

早期的 BASIC 版本只能使用 GOSUB 语句调用和主程序同在一个模块内的子程序，没有提供独立的子程序模块。

GOSUB 是 Goto Subroutine 的缩写，意思是转到“子例行程序”中，执行一个子程序如同“例行公事”一样。整个“子例行程序”简称块内子程序或者子程序。它的主要特征是：子程序与主程序在同一程序模块中，根据调用方式上的不同，Quick BASIC 对块内子程序有 3 种调用方式，对应的则用 3 种语句来实现。

1．GOSUB…RETURN 语句

格式：

```
GOSUB〈子程序入口〉
RETURN
```

功能：调用块内的子程序。

说明：

（1）GOSUB 语句是写在调用程序中的语句：〈子程序入口〉是指子程序第一条语句的行号或者标号。

（2）RETURN 语句应写在子程序中。

（3）GOSUB 语句称为转子语句，其作用将流程转到子程序；RETURN 语句称为返回语句，其作用是使流程和所有子程序中对应 GOSUB 语句的下条语句。

（4）主程序和块内子程序是连续书写的。它们是在同一模块之中，如果一个程序包含多个子程序，那么主程序和所有子程序都应写在同一模块中。子程序的范围是：从 GOSUB 语句指定的行号或者标号开始，到 RETURN 语句结束。

（5）变量在整个程序中有效。这就是说，变量在模块内的主程序和子程序中都是有效的。

[例 9-9] 求 3！+4！+5！的值。

问题分析：由于题中要求计算三个数的阶乘，再累加。而求阶乘的过程都是一样的，这样就把求阶乘的操作设计成子程序，每次分别以不同的值代入求得即可。

程序为：

```
'EXAMPLE 9
s=0
a=3
b=4
c=5
n=a
GOSUB f:
s=s+p
n=b
GOSUB f:
s=s+p
n=c
GOSUB f:
s=s+p
PRINT "3!+4!+5!=";s
END
f:p=1                                    '求阶乘子程序
FOR i=1 TO n
p=p*i
NEXT i
RETURN
```

程序运行结果为

```
3！+4！+5！＝150
```

2．ON GOSUB…RETURN 语句

该语句是根据给定的条件进行判断，从多个子程序中选定其中一个执行。

格式：

ON〈算术表达式〉GOSUB〈子程序表〉

功能：根据〈算术表达式〉的值进行判断，执行〈子程序表〉中的第几个子程序。

说明：

（1）〈子程序表〉是指多个子程序之间用逗号隔开，这里的子程序是指其子程序第一行的行号或标号。

（2）ON GOSUB 语句的用法是根据 ON 后面表达式的值来决定执行哪个子程序。若表达式的值为 1，则执行子程序 1；值为 2，则执行子程序 2，其余类推。招待完指定的子程序语句后，流程返回 ON GOSUB 语句下

面的一个语句继续执行。

（3）如果〈算术表达式〉的值不是整数，则按四舍五入处理；

如果〈算术表达式〉的值为 0 或者大于子程序中的个数，则越过该语句；

如果〈算术表达式〉的值为负，则系统将给出“ Illegal function call ”(非法函数调用)的出错信息。

[例 9-10] 写出下面程序的运行结果。

```
'EXAMPLE 10
a=100
b=50
FOR i=1 TO 2
p=i MOD2+1
ON p GOSUB add,subtract
NEXT i
PRINT
END
add:PRINT a+b;
RETURN
subtract:PRINTa-b;
RETURN
```

程序的运行结果为：

```
150              50
```

本题中的主程序是一个 FOR 循环，共循环两次，子程序 1 和子程序 2 的标号分别为 add 和 subtract，给定的条件是 p=i MOD 2+1,在循环执行过程中，P 依次被赋值为 1 和 2，在执行 ON GOSUB 语句时，分别执行 add t 和 subtract 这两个子程序，先后打印输出 150 和 50。

3．ON KEY（n）GOSUB…RETURN 语句

当用户需要在程序运行的过程中，通过键盘来干预程序的执行，让计算机完成一个特定的操作时，可以使用 ON KEY（n）GOSUB〈子程序入口〉

功能：在程序中指定一个键，设置事件陷阱。

说明：

（1）格式的 n 是一个数值表达式，代表一个键。n 的值设定的键的对应关系见表 9-1 所示。

表 9-1 KDY（n）中 n 与键的对应关系

n 的值	键 名
1~10	功能键 F1~10
11	方向键↑
12	方向键←
13	方向键→
14	方向键↓

（2）ON KEY（n）GOSUB 语句的作用是：在程序中指定一个键，如果在程序执行过程中用户按下该键，程序就会中断原来的操作过程，转而执行事先设置的一个子程序，执行完这个子程序后再返回原来的主程序继续执行。这种功能称为“ 事件捕捉 ”，按一个特定的键就是一个“ 事件 ”。“ 捕捉 ”到此“ 事件 ”就转子程序处理。“ 事件捕捉 ”功能又称“ 陷阱 ”。这样的子程序称“ 事件捕捉子程序 ”或“ 陷阱子程序 ”。

[例 9-11] 编写一个程序，在程序开始运行后，只要按一次 F5 键，屏幕上就显示当时的时间。

程序如下：

```
'EXAMPLE 11
ON KEY（n）GOSUB t
KEY(5) ON
DO
LOOP UNTIL INKEY$<>""
```

```
END
KEY(3)OFF
t:PRINT"时间：";TIME$
RETURN
```

程序中第一行设 n 值为 5，也就是指定键盘上的功能键 F5 为“陷阱键”。在程序运行过程中任何时候按下 F5 键，就转去执行陷阱子程序。该子程序输出当前时间，每按一次 F5 键，显示一次。程序第 4 行的作用的等待键盘上的一个操作，如键盘上无任何输入，循环不断运行，如要程序终止，则可按下 F5 键以外的任何键。

程序运行结果如下（按 3 次 F5 键）：

```
时间：20：20：56
时间：20：21：03
时间：20：21：12
```

按回车键后程序结束。

9.3.2 SUB 子程序

1. 定义子程序过程语句 SUB…END SUB

在 Quick BASIC 中用 SUB 语句定义子程序过程。

格式：

```
SUB〈过程名〉［（〈参数表〉）］［STATIC］
［〈语句块〉］
END SUB
```

功能：定义一个 SUB 子过程。

说明：

（1）〈过程名〉即子程序过程的名字，是一个不超过 40 个字符的字符串，不能有数据类型后缀。一个程序只能有一个惟一的过程名。

（2）〈参数表〉是主程序调用本过程时传送给本过程的一个或多个简单变量名、数组名，各变量名之间用逗号分隔。〈参数表〉指明了传送给过程的变量个数和类型，其格式为：

```
〈变量 1〉［()］［AS〈类型〉］［，〈变量 2〉［()］［AS〈类型〉］］…
```

这里的〈变量〉是一个 Quick BASIC 变量名，如果是数组变量，要在数组名后面加上一对小括号。

〈类型〉是对应〈变量〉的数据类型，可以是 INTEGER、LONG、SINGLE、DOUBLE、STRING 或用户定义数据类型。

〈参数表〉中的参数是形式参数，不能用定长字符串变量或定长字符串数组作为形式参数。不过可以在 CALL 语句中用简单定长字符串变量作为形式参数，在调用 SUB 过程之前，Quick BASIC 把它转换为变长字符串变量。

（3）STATIC 定义过程中局部变量在内存中的存储方式。如果使用 STATIC 属性，则过程中的局部变量是 STATIC（静态）的，即在每次调用过程时，各局部变量的值保持不变；如果省略 STATIC，则局部变量主默认为自动的，即在每次调用过程时，局部变量初始化为 0 或空字符串。

（4）SUB 过程以 SUB 开始，以 END SUB 结束。当程序遇到 END SUB 时，将退出过程并立即返回到调用语句的下面语句。

（5）〈语句块〉是 Quick BASIC 的程序段，语句块中可以用一个或多个 EXIT SIB 语句从过程中退出。

（6）在过程定义中不能使用 SUB…END SUB、FUNCTION…END FUNCTION 、DEF FN…END DEF、COMMON、DECLARE、DIM SHARED、OPTION BASE 和 TYPE…END TYPE 语句。

在 SUB 过程中不能定义 SUB 过程、FUNCTION 过程、DEF FN 函数。即在过程内不能嵌套过程定义或 DEF FN 函数，但是一个过程可以调用另一个过程或 DEF FN 函数。

2. 调用子程序过程语句 CALL

要执行一个过程，必须调用该过程，SUB 过程调用可以作为独立的基本语句。调用 SUB 过程的方式有两种：一种是把过程名放在一个 CALL 语句中；另一种是把过程名本身作为一个语句来使用。

格式：

[CALL]〈过程名〉[(〈变元表〉)]

功能：把控制传送到 SUB **过程。**

说明：

(1)〈过程名〉是需调用的 SUB **过程名。**

(2)〈变元表〉是传送给 SUB **过程的变量或常量，各变元之间用逗号分隔。用数组名后跟空括号指定数组参数。**

(3) 如果省略 CALL **关键字，也要同时去掉包括变元表的小括号，这时要在调用前用** DECLARE **语句说明。如果在** Quick BASIC **环境中保存它，**Quick BASIC **将自动产生** DECLARE **语句。**

下面是两种调用方式的区别：

```
DECLARE SUB proce(first%,second$)              '说明过程语句
…
CALL proce(first%,second$)                     '用 CALL 调用时的语句
…
proce first%,second$                           '省略 CALL 时的调用语句
```

[例 9-12] **用** SUB **子程序计算** 4！+5！+6！**的值。**

问题分析：要计算 s=4!+5!+6!,**先要分别计算出** 4！，5！**和** 6！。**由于** 3 **个求阶乘的运算过程完全相同，因此可以用子程序来计算任意阶乘** m!，**每次调用子程序前给** m **一个值，在子程序中其所求结果放入** total **变量中，返回主程序后** tot **变量接收** total **的值。这样** 3 **次调用子程序便可求得** s。

主程序为：

```
'EXAMPLE 12
DECLARE SUB fact(m!,total!)
CLS
a=4:b=5:c=6
CALL fact(a,tot)
s=tot
CALL fact(b,tot)
s=s+tot
PRINT a;"!+";b;"!+";c;"!=";s
END
```

子程序为：

```
SUB fact(m,total)                    '计算阶乘子程序
total=1
FOR i=1 TO m
total=total*i
NEXT i
END SUB
```

程序运行结果为：

4！+5！+6！=864

9.3.3 说明过程语句 DECLARE

在 Quick BASIC **中，定义的过程要在主程序中用说明过程语句** DECLARE **说明。**

格式：

DECLARE{FUNCTION | SUB}〈过程名〉[(〈参数表〉)]

功能：说明被调用的一个 FUNCTIONA **或** SUB **过程名，并且检查各参数的数据类型。**

说明：

(1)〈过程名〉是程序中的 FUNCTION **或** SUB **过程名。**

（2）〈参数表〉是主程序调用过程时给传送给过程的一个或多个变量，其格式为：

〈变量 1〉［（）］［AS〈类型〉］［，〈变量 2〉［（）］［AS〈类型〉］］…

〈变量〉是一个 Quick BASIC 变量名。

〈类型〉是对应〈变量〉的数据类型，可以是 INTEGER、LONG、SINGLE、DOUBLE、STRING 或用户定义数据类型。即任何允许的数据类型。

（3）DECLARE 语句中的参数表主要是对传送过程的变元进行类型检查。

（4）DECLARE 语句必须和在主程序中第一个可执行语句之前。作为约定 DECLARE 语句出现在程序注释语句之后，常数及变量说明符之前。

（5）在主程序中，如果省略 DECLARE 语句，在 Quick BASIC 编辑环境中，当把程序存盘时，将自动在主程序的开始入加入属于本过程的 DECLARE 语句。

由于 DECLARE 语句会在 Quick BASIC 的编辑环境中自动产生，所以用户不用专门输入。

但是，维护参数表的工作仍要由用户自己完成，如果修改子程序中的参数表，也必须修改 DECLARE 语句中的参数表，使之相互匹配。

9.4 调用过程时的数据传递

调用过程时可以把数传递给过程，也可以把过程中的数据传递回来。在调用过程中，要考虑调用程序和被调用程序之间的数据是如何传递的。通常在编制一个子程序时，要考虑它需要输入哪些量，进行处理后输出哪些量。正确地提供一个子程序的输入数据和正确地引用其输出数据，是使用子程序的关键问题，也就是调用程序和被调用程序之间的数据传递。

在调用一个过程时必须完成形式参数与实际参数的结合，即把实际参数传送给形式参数，然后实际参数执行调用的过程。在 Quick BASIC 中，通常把形式参数叫做参数，把实际参数叫做变元。

9.4.1 参数与变元

变元（实际参数）是在调用 SUB 或 FUNCTION 过程时传送给 SUB 或 FUNCTION 过程的常量、变量或表达式。参数（形式参数）是出现在 SUB、FUNCTION 或 DECLARE 语句中的变量名是接收传送给子程序值的变量。

参数与变元的对应关系为：

实际参数

调用过程：CALL testsub(a%,b!,"Test"， 16.8)

定义过程：SUB testsub(r%,s!, t$, x)STATIC

形式参数

在定义过程中，形式参数为实际参数保留位置，当调用过程时，实际参数被插入形式参数中各变量处，第一个形式参数接收第一个实际参数的值，第二个形式参数接收第二个实际参数的值，…。

实参表和形参表中对应的变量名不必相同，但是变量的个数必须相等，并且对应变量的类型必须相同。

1．形参表

形参表中的各个变量之间用逗号分隔，表中的变量是：

（1）后面跟有左、右圆括号的数组名：若括号内有数字，一般表示数组的维数。

（2）除定长字符串之外的合法变量名。即在形参表中只能用如 x$或 x AS STRING 之类的变长字符串作为形式参数，不能用如 x AS STRING*10 之类的定长字符串作为形式参数。但定长字符串可以作为实际参数传递给过程。

2．实参表

实参表可由常量、表达式、有效的变量名、数组名（后跟左、右括号）组成，变元表中各变元用逗号分隔。例如，下面是具有参数表的一个子程序定义的语句行：

SUBtestsub(i%,aa(),re AS rectype,cc$)

其中：i%是整型参数、aa 是一个单精度数组、re 是 rectype 类型的记录、cc$是一个字符串。

用下面的程序调用过程 testsub，并把 4 个变元传送给相应的参数。

```
TYPE rectype                    '定义用户类型
xm AS STRING*12
dz AS LONG
END TYPE
DIM re AS rectype               '定义一个 rectypt 类型的变量 re
CALL testsub(i%,cj(),re,"Kaifeng")
```

3. 传址调用和传值调用

传递参数的方式有两种。如果 CALL 语句中的变元（实际参数）为变量，就是传址调用；如果 CALL 语句中的变元（实际参数）是常量或表达式，就是传值调用。

9.4.2 传址调用

传址调用或称按地址传递，是指形式参数与实际参数使用相同的内存地址单元。所以，如果形式参数发生变化，实际参数也随着改变。在传址调用时，实际参数必须是变量，绝不能是常量或表达式。

传址调用会把过程的执行结果带回调用的程序，因此，在返回程序时实际参数的内容可能已经发生了变化。

[例 9-13] 阅读下面的程序，分析输出结果，理解传址调用。

主程序为：

```
'EXAMPLE 13
DECLARE SUB prod(x!,y!,z!)
CLS
a=5:b=3
PRINT "主程序调用前的变量值 a,b,c",a,b,c
END
```

子程序为：

```
SUB prod(x!,y!,z!)
PRINT"子程序运算前的变量值 x,y,z",x,y,z
z=x*y
PRINT"子程序中运算后的变量值 x,y,z",x,y,z
END SUB
```

程序的运行结果为

```
主程序调用前的变量值 a,b,c        5    3    0
子程序中运算前的变量值 x,y,z      5    3    0
子程序运算后的变量值 x,y,z        5    3    15
主程序调用后的变量值 a,b,c        5    3    15
```

主程序 CALL 语句中的变量 a、b、c 与子程序中相对应的变量 x、y、z 使用相同的内存地址单元。当执行到 CALL 语句时，就调用子程序 prod()，并把变量 a、b、c 的值传给变量 x、y、z。在子程序 prod()中如果变量的 z 值发生变化，则返回主程序中，变量 c 的值随之改变。

9.4.3 传值调用

传值调用是把实际参数的值传递给对应的形式参数。它是把需要传送的实际参数的值复制到一个临时内存单元中，然后把该临时单元的地址传送给子程序。由于子程序没有访问实际参数的原始地址，因而不会改主原来参数的值，所有的变化都是在变量的副本上进行的。因此，在过程中无论形式参数如何改变，主程序中的实际参数不会受到影响。

当要求用变量按值传送时，可以先把变量变成一个表达式，例如把变量用括号括起来，如(a%)是一个表达式。

传值调用子程序的工作过程：

（1）计算出实际参数的值。

（2）子程序内使用的变量被定义为局部变量，在退出该子程序时即消失。与形式参数相对应的各变量按特殊的局部变量来处理。

（3）实际参数的值复制给子程序中对应的形式参数。

（4）继续执行子程序体中的语句。

（5）退出子程序，子程序中所有局部变量消失。返回到主程序中执行子程序调用语句的下一语句。

[例 9-14] 阅读下面的程序，分析运行结果，理解传值调用。

主程序为：

```
EXAMPLE 14
DECLARE SUB prod(a!,b!,c!)
CLS
a=5:b=3
PRINT"主程序调用前的变量值 a,b,c:",a,b,c
CALL prod((a),(b),(c))
PRINT "主程序调用后的变量值 a,b,c",a,b,c
END
```

子程序为：

```
SUB prod(a,b,c)
PRINT "子程序中运算前的变量值 a,b,c:",a,b,c
a=6:b=8
c=a*b
PRINT "子程序中运算后的变量值 a,b,c:",a,b,c
END SUB
```

程序的运行结果为：

```
主程序调用前的变量值 a,b,c:        5    3    0
子程序中运算前的变量值 a,b,c:      5    3    0
子程序中运算后的变量值 a,b,c:      6    8    48
主程序调用后的变量值 a,b,c:        5    3    0
```

在子程序中即使与主程序中使用的变量名相同，但在内存中仍表示不同的内存单元地址。即执行子程序时使其变量内容发生变化，主程序中的变量内容并举随之改变。

在过程中以传值调用为主，如果希望把过程中的变量值返回主程序，才使用传址调用。

用传值调用这种传递参数的方法只能传递计算值，如数值、字符串、数值数组的值。

9.5 过程的嵌套和递归调用

Quick BASIC 的过程定义都是互相平行和孤立的，也就是说在定义过程时，一个过程内不能包含另一个过程。Quick BASIC 虽然不能嵌套定义过程，但可以嵌套调用过程，也就是主程序可以调用子过程，在子过程中还可以调用另外的子过程，这种程序结构称为过程的嵌套。

9.5.1 过程的嵌套

[例 9-15] 求组合数 $C^m_n=\frac{N!}{M!(N-M)!}$ 的值，设 N＝10，M＝6。

在程序中我们把求阶乘与求组合数公式分别定义为函数。

主程序为:

```
'EXAMPLE 15
DECLARE FUNCTIONcomb!(n!,m!)
DECLARE FUNCTION fact!(x!)
CLS
INPUT"n,m=";n,m
PRINT"C(";n;",";m;")=";comb(n,m)
END
```

自定义函数为（求组合数）

```
FUNCTION comb(n,m)
comb=fact(n)/(fact(m)*fact(n-m))
END FUNCTION
```

自定义函数为（求阶乘）

```
FUNCTION fact(x)
==1
FOR i=1 TO x
==p*i
NEXT i
fact=p
END FUNCTION
```

程序的运行结果为:

C（10，6）＝210

在 FUNCTION **子过程** comb **中调用了** FUNCTION **子过程** fact，**从而形成过程的嵌套。**

9.5.2 过程的递归调用

递归调用是指一个过程直接或间接调用自己本身即自己调用自己。在递归调用中，一个过程执行的某一步要用到它自身的上一步（或上几步）的结果。

Quick BASIC **过程具有递归调用功能，递归调用在处理阶乘运算、级数运算、幂指数运算等方面特别有效。**

[**例** 9-16] **计算** n!

根据数学知识，负数没有阶乘，0 **的阶乘为** 1，**正整数** n **的阶乘为：**

n×(n-1)×(n-2)×…×2×1

则可以用下式表示：

$$fact=\begin{cases}1 & n=0\\ n(n-1)! & n>0\end{cases}$$

利用上式，把求 n!**转换为求** n×(n-1)!。

上面的公式说明了每一循环的结果都有赖于上一循环的结果递归总有一个“ 结束条件 ”，如 n!**的结束条件为** n=0。

在 Quick BASIC **中用递归过程实现这种运算的程序如下。**

主程序

```
'EXAMPLE 16
DECLARE FUNCTION factorial#(n%)
CLS
DO
INPUT"请输入一个 0~20 之间的整数（-1 结束）：",num%
IF num%>=0 AND num%<=20 THEN
```

```
PRINT num%,factorial#(num%)
END IF
LOOP WHILE num%>=0
END
```

自定义函数为：

```
FUNCTION factorial#(n%)STATIC
IF n%>0 THEN
factorial#=1
END IF
END FUNCTION
```

当 n%>0 **时，在过程** factorial#**中调** factorial#**过程，参数为** n%-1,**这种操作一直持续到** n%=1 **为止。**

例如，当 n%=5 **时，求** factorial#(5)**的值变为求** 5×factorial#(4);**求** factorial#(4)**的值又变为求** 4×factorial#(3),…, **当** n%=0 **时，**factorial#**的值为** 1，**递归结束，其结果为** 5×4×3×2×1。**如果把第一次调用过程** factorial#**叫做** 0 **级调用，以后每调用一次级别增加** 1，**过程参数** n%**减** 1，**则递归调用的过程如下：**

递归级别	执行操作
0	factorial#(5)
1	factorial#(4)
2	factorial#(3)
3	factorial#(2)
4	factorial#(1)
4	返回 1 factorial#(1)
3	返回 2 factorial#(2)
2	返回 6 factorial#(3)
1	返回 24 factorial#(4)
0	返回 120 factorial#(5)

9.6 变量的属性和作用域

当所编写的程序越来越长时，程序中所使用的变量也会越来越多。为了帮助用户管理大量的变量，Quick BASIC **规定了一些特殊的规则处理过程中的变量。**Quick BASIC **将变量分为全程变量、局部变量和共享变量** 3 **类，这** 3 **类变量的作用域也各不相同。**

9.6.1 局部变量

在 Quick BASIC **中使用的变量，凡未经特殊说明的变量均属于局部变量（**Local Verable**），局部变量在主程序、子程序或函数中建立，但是只能在建立的过程内有效，即便是在主程序中建立的变量，也不能在子过程中使用。**

局部变量不用任何语句说明。它的作用域仅限于它自己所在的过程，使用局部变量的程序比仅使用全程变量的程序更具有通用性。

[**例** 9-17] **未经说明的局部变量示例。**

主程序为:

```
'EXAMPLE 17
DECLARE SUB prod()
DECLARE SUB sum()
CLS
a=5:b=3
```

```
PRINT TAB(16);"a";TAB(30);"b";TAB(42);"c=a*6"
PRINT "调用 prod 前"，a,b,c
CALL prod
PRINT"调用 prod 后"，a,b,c
PRINT
PRINT"调用 sum 前"，a,b,c
CALL sum
PRINT"调用 sum 后",a,b,c
END
```

子程序为:

```
SUB prod
c=a*b
PRINT"prod 子程序",a,b,c
END SUB
```

子程序为：

```
SUB sum
c=a+b
PRINT "sum 子程序",a,b,c
END SUB
```

程序的运行结果为:

```
                        a   b c=a*b
调用 prod 前            5 3     0
prod 子程序             0 0     0
调用 prod 后            5 3     0

调用 sum 前             5 3     0
sum 子程序              0 0     0
调用 sum 后             5 3     0
```

从上面程序的运行结果可以看出，主程序中的变量没有带到子程序中。

9.6.2 全程变量

全程变量（Global Variable）是指在所有程序（包括主程序和过程）中都可以使用的内存变量。全程变量只能在主程序中定义，在主程序中用 DIM、REDIM 或 COMMON 语句中的 SHARED 短语定义全程变量属性；全程变量就像在一个程序中定义的变量一样，可以任意改变和调用，当过程执行完后，其值仍然保留，并带回主程序。

把变量定义为全程变量虽然很方便，但这样会增加变量在程序中被无意修改的机会，因此许多程序员不赞成在程序中使用全程变量。

另外，在主程序中用 CONST 语句定义的符号常量也是全程的，即在各过程中可以引用该符号常量。过程中用 CONST 定义的符号常量都是局部的，只能在该一个过程内引用。

在主程序中，可用 DIM SHARED 或 REDIM SHARED 语句把某些变量或数组定义为全程变量，使该变量在主程序及各过程中到处可以使用和修改。用 DIM SHARED 或 REDIM SHARED 语句说明变量后，要在后面程序或过程中对变量用赋值语句赋值。

[**例** 9-18] **在主程序中用 DIM SHARED 语句来定义全程变量。**

主程序为：

```
'EXAMPLE 18
DECLARE SUB prod()
```

```
DECLARE SUB sum()
CLS
DIM SHARED a,b,c
a=5:b=3
PRINT TAB(16);"a";TAB(30);"b";TAB(42);"c=a*b"
PRINT"调用 prod 前",a,b,c
CALL prod
PRINT"调用 prod 后"，a,b,c
PRINT"调用 sum 前"，a,b,c
CALL sum
PRINT"调用 sum 后"，a,b,c
END
```

子程序为：

```
SUB prod
c=a*b
PRINT "prod 子程序",a,b,c
END SUB
```

子程序为：

```
SUB sum
c=a+b
PRINT"sum 子程序",a,b,c
END SUB
```

程序的运行结果为：

```
                          a       b       c=a*b
调用 prod 前                      5       3       0
prod 子程序                       5       3       15
调用 prod 后                      5       3       15

调用 sum 前               5       3       15
sum 子程序                        5       3       8
调用 sum 后                       5       3       8
```

从程序的运行结果可以看出，在主程序中用 DIM SHARED **语句定义的全程变量** a、b、c**，把值带到了子程序中；返回主程序时，把修改后的值又带回了主程序。**

9.6.3 共享变量

在 Quick BASIC **中还有一种特殊的使用变量的方式，就是在主程序中不用** DIM、REDIM **或** COMMON **语句中的** SHARED **短语定义全程变量而在子程序中用** SHARED **语句来定义调用主程序中的同名变量，这种变量称共享变量（**Sharing Variable**）。用** SHARED **定义的变量可以在子过程中引用主程序中的变量，子过程中修改后的变量值也将带回主程序。**

格式：

SHARED**〈变量** 1**〉［（）］［**AS**〈类型〉］［，〈变量** 2**〉［（）］［**AS**〈类型〉］］…**

功能：定义过程中可以访问主程序中的变量。

说明：

（1**）〈变量〉是主程序中的变量名，也就是本子过程与主程序共用的变量名，可以是简单变量或数组变量。**

（2**）〈类型〉用来定义对应〈变量〉的数据类型（**INTEGER、LONG、SINGLE、DOUBLE、STRING **或用户定义类型）。**

（3）SHARED 语句只能在 SUB 或 FUNCTION 过程使用。SHARED 后面列出的变量必须是主程序中存在的变量。

[例 9-19] 在子程序中用 SHARED 语句来定义共享变量。

主程序为：

```
'EXAMPLE 19
DECLARE SUB prod()
DECLARE SUB sum()
CLS
a=5:b=3
PRINT TAB(16);"a";TAB(30);"b";TAB(42);"c=a*b"
PRINT"调用 prod 前"，a,b,c
CALL prod
PRINT"调用 prod 后"，a,b,c
PRINT
PRINT"调用 sum 前",a,b,c
CALL sum
PRINT"调用 sum 后"，a,b,c
END
```

子程序为：

```
SUB prod
SHARED a,b,c
c=a*b
PRINT"prod 子程序",a,b,c
END SUB
```

子程序为：

```
SUB sum
c=a+b
PRINT"sum 子程序",a,b,c
END SUB
```

程序的运行结果为：

```
                    a     b        c=a*b
调用 prod 前        5     3        0
prod 子程序         5     3        15
调用 prod 后        5     3        15
调用 sum 前         5        3        15
sum 子程序      0        0        0
调用 sum 后     5        3        15
```

从程序的运行结果可以看出，在子程序 prod 中定义了共享主程序中的 a、b、c 变量，因此在子程序 prod 中可以引用和修改，返回主程序时也把修改后的变量值带回了主程序。子程序 sum 中的 a、b、c 没有特别说明，都是局部变量，也就无法引用主程序中的变量。

9.6.4 变量作用域规则小结

Quick BASIC 的变量作用域规则：

（1）主程序中用 DIM、REDIM 或 COMMON 语句中的 SHARED 短语说明的变量是全程变量，任何 SUB 或 FUNCTION 都能调用这些变量。

（2）主程序中用 CONST 语句说明的符号常量全程符号常量。在 SUB 或 FUNCTION 中说明的符号常量是

局部符号常量。

（3）如果一个变量未经任何说明，则这个变量是局部变量 ，它只能在建立这个变量的主程序或过程内使用。

（4）如果在一个过程内用 STATIC 语句说明一个全程变量名或者在一个过程内把全程变量当作形式参数，则这个全程变量名在这外过程中就作为局部变量。

（5）SHARED 语句使主程序和子过程中同名变量成为共享变量。

9.7 静态变量与动态变量

比如子程序内部有一个变量，当程序运行进入该程序时，要分配给该变量一定的内存单元，一旦程序退出该过程，变量占有的内存单元是释放还是不释放？根据变量在程序运行期间的生命周期，Quick BASIC 把变量分为静态变量（Static）和动态变量（Dynamic）。静态变量不释放内存单元，动态变量释放内存单元，因此有时就需要某些局部变量是静态变量，而其他变量仍然为动态变量。

所谓静态存储方式是指在程序运行期间分配固定的存储空间的方式。而动态存储方式则是在程序运行期间根据需要进行动态的分配存储空间的方式。

静态变量是指程序运行进入该变量所在的子程序，修改变量的值后，退出该子程序其值仍被保留，即变量所占内存单元没有释放。当以后再次进入该子程序，原来变量的值可以继续调用。

动态（自动）变量是指程序运行进入变量所在的子程序，才分配该变量的内存单元，经过处理退出该过程后，该变量占用的内存单元 自动释放，其内存能被其他变量占用。动态变量多用于动态数组。

9.7.1 STATIC 语句

格式：

STATIC〈变量 1〉［（）］［AS〈类型〉］［，〈变量 2〉［（）］［AS〈类型〉］］…

功能：在函数或过程中定义局部变量，并在两次调用之间保留变量的值。

说明：

（1）〈变量〉是过程中要定义的局部变量，可以是简单变量或数组变量。

（2）〈类型〉用来定义对应〈变量〉的数据类型（INTEGER、LONG、SINGLE、DOUBLE、STRING 或用户定义类型）。

（3）STATIC 语句只能在 SUB 或 FUNCTION 过程中使用。

（4）STATIC 语句后面列出的变量可以与主程序中用 DIM、REDIM 或 COMMON 语句说明的全程变量同名。但是，用 STATIC 说明的变量优先级高于用 DIM、REDIMK 或 COMMON 语句说明的变量，所以，STATIC 语句后列出的变量是局部变量，与主程序中定义的同名全程变量无关。

[例 9-20] 下面程序在主程序中用 DIM 定义了全程变量 r、n，在子程序中定义了同名的局部变量。

```
'EXAMPLE 20
DECLARE SUB test()
DIM SHARED r,n
CLS
r=60:n=80:rep=0:num=0
PRINT"r","n","rep","num"
PRINT r,n,rep,num
FOR i=1 TO 10
CALL test
NEXT i
PRINT r,n,rep,num
END
```

子程序为：

```
SUB test STATIC
SHARED tep,num
STATIC r,n
r=r+1:n=n+2:rep=r:num=n
```

程序的运行结果为:

```
r       n       rep       num
60      80        0         0
60      80       10        20
```

从程序的运行结果可看出，主程序中定义的全程变量 r、n **被子程序中定义的局部变量屏蔽起来，在子程序中起作用的是子程序中的** r、n**，它与主程序中的** r、n **不是相同的变量。**

9.7.2 动态变量与静态变量

如果在 SUB **或** FUNCTION **语句中不使用** STATIC **关键字，则为动态变量，即每次调用过程时，都行到新的局部变量。**

如在 SUB **或** FUNCTION **语句中使用** STATIC **关键字，则这个过程中所有变量都是静态变量，即在过程调用期间变量的值保持不变。**

当在子程序中使用 SHARED **语句定义共享变量时，可以使** STATIC **属性无效，即无论在过程定义行（**SUB **或** FUNCTION**）中使用了** STATIC **属性还是省略** STATIC **属性，只要这个与主程序同名的变量出现在** SHARED **语句中，这个变量就可以返回到主程序，因而不是局部变量。**

在 STATIC **语句说明的变量与在过程行中用** STATIC **关键字说明的变量不同。用** STATIC **语句说明变量绝对保证是局部变量，因为** STATIC **语句的优先级高于** SHARED **语句，它可以使** SHARED **语句的作用无效。**

当数组作为局部变量放在 STATIC **语句中时，在使用之前要标出它的维数。例如：**

```
SUB supp
STATIC array()AS INTEGER
DIM array(-5 TO 8)AS INTEGER
…
END SUB
```

[**例** 9-21] **下面程序说明了** STATIC **语句的作用。**

主程序为：

```
'EXAMPLE 21
DECLARE SUB testsub()
CLS
FOR i%=1 TO 5
testsub
NEXT i%
END
```

子程序为:

```
SUB testsub
STATIC y
x=x+1:y=y+1
PRINT"x=";x,"y=";y
END SUB
```

程序的运行结果为:

```
x=1       y=1
x=1       y=2
```

```
x=1       y=3
x=1       y=4
x=1       y=5
```

程序中 x 和 y 都是过程 testsub 中的局部变量 ，y 被说明为 STATIC 变量，每次调用保持上一次的值，y 的值会变化；x 未被说明，它是动态变量，每次调用都被重新初始化为 0，它的值总是不变。

[例 9-22] 分析程序的运行结果。下面子程序中的 x$、y$都是局部变量，但 x$是动态变量，y$是静态变量。

主程序为:

```
'EXAMPLE 22
DECLARE SUB star()
CLS
FOR i%=1 TO 5
CALL star
NEXT i%
```

子程序为:

```
SUB star
STATIC y$
x$=x$="#":y$=y$="*"
PRINT x$,y$
END SUB
```

程序运行结果为：

```
#          *
#          **
#          ***
#          ****
#          *****
```

9.8 本章小结

本章共讲了如下的语句：

DEF FN 自定义函数、GOSUB 子程序、FNCTION 子程序、SUB 子程序。

下面将它们的功能作一下简单的比较：

1．DEF FN 与 FUNCTIION 的比较

DEF FN 在 BASIC 中只能定义一个单行函数关系。只能在本程序单位中使用。

FUNCTION 子程序可以实现单行函数所不能实现的复杂函数关系,函数内部的的有变量都隐含为局部变量，变量的传送方式灵活多样。可以实现递归调用，还可被其他程序模块调用，资源可以共享。

2．GOSUB 和 SUB 子程序的比较

GOSUB 子程序中的变量是全局变量，只能在同一个程序中使用。适合小型的程序开发。

SUB 子程序中的变量都是局部变量，数据传送更方便，可以实现双向传送。既可“ 按地址传送 ”，又可以“ 按值传送 ”。很适宜大型的结构化程序的设计。

3．FUNCTION 子程序和 SUB 子程序的比较

（1）共同点是：过程中变量都是局部变量，数据能双向传送。能将通用的过程全并起来组成一个 Quick 库，供所有程序模块调用，实现资源共享。

（2）不同点：

FUNCTION 主要通过函数名返回数据，函数名被赋值后，不能再参加运算。它的数据虽然也能双向传送，

但建议初学者尽量不要改变虚参表中各虚参的值，以免引起一些无法预料的问题。

SUB 子程序的数据传送主要是通过虚实结合方式，数据传送量多，比 FUNCTION 子程序更灵活。它能取代 FUNCTION 的一切功能。

总之，每个过程只完成一个单纯的任务，不要把过多的任务放到一个过程中，当然每一个任务还可以有若干个了任务，因而形成了过程的多层调用，即过程的嵌套。不要只习惯于写一个程序去包罗万象，而要善于去调用过程，利用过程的嵌套调用。当然，对比较简单的程序，有一级调用就够了。但在大型的程序中往往采用多层调用的短的过程集合，通过逐步地实践，这一程序设计的技术是不难掌握的。

习 题

1．有 5 个人坐在一起，问第 5 个人多少岁？他说比第 4 个人大 2 岁。问第 4 个人岁数，他说比第 3 个人大 2 岁。问第 3 个人，又说比第 2 个人大 2 岁。问第 2 个人，说比第 1 个人大 2 岁。最后问第 1 个人，他说是 10 岁。请问第 5 个人有多大岁数。

2．用递归法求两个整数 m 和 n 的最大公约数。

3．利用 RND 函数，产生 10 个随机存放在数组 x1()中，再按从大到小的顺序显示出来。

4．编写一个函数，求两个整数的最在公约数。在主程序中给出 3 个整数 a、b、c，通过调用函数求 3 个整数的最大公约数。

5．求两个数 a 和 b 的最大公约数和最小公倍数。要求：最大公约数由一个函数求出，然后求出最小公倍数，最小公倍数和 a、b 均用全局变量来存放。

6．定义一个求圆面积的用户定义函数，计算 8 个面积之和，并在函数中累计调用函数的次数。

7．制用户定义函数，输入一个 0~6 数字，显示汉英对照星期的函数。

8．利用自定义函数，计算某一个字符在字符串中出现的次数。

第十章　字符处理

前面主要介绍了计算机在数值计算中的应用，只涉及了一些简单的字符处理。事实上计算机不仅在数值计算上应用广泛，而且在非数值计算中应用范围正在不断扩大，尤其是用于事务管理领域。如图书检索，人事管理，教务管理，经济信息管理等。有关资料表明，计算机在非数值计算中的应用，要远远超过计算机在数值计算中的应用。这就要求计算机具有相当强的文字处理能力。如果一台计算机没有文字处理功能，充其量只能算是一个高级的计算器。Quick BASIC 的一个非常重要的功能就是可以进行字符处理，而且它的字符处理功能十分强大。它提供了字符串变量，字符串数组，字符串语句以及丰富的字符串函数，使用起来直当灵活方便。本章主要介绍 Quick BASIC 对字符串的各种处理功能，涉及的主要内容有：

- 字符串变量
- 字符串变量和数组字符
- 字符串变量的赋值
- 字符串表达式及字符串的比较
- 取子字符串
- 字符串的生成
- 字符串与数值的相互转换
- 自选输出格式

10.1　字符串变量

10.1.1　字符

字符是指单个字母、数字或其他特殊符号。本书附录中的 ASCII 码表中，列出了 IBM-PC 系列微机中的字符集。

10.1.2　字符串

所谓“字符串”就是系统允许使用的若干个字符构成的序列，也称为字符串常量。一个字符串一般是用双引号括起来的一串字符。例如：

```
"abcdefg"        "1234567"        "This is a book"
```

字符串常量与数值常量一样，在程序执行过程中其值是不变的。

Quick BASIC 的字符串中可以使用的字符主要是在键盘上出现的字符，包括：26 个英文字母（大写字母 A~Z、小写字母 a~z），数字 0~9，标点符号，空格，数学符号，专用符号（@、￥、~、＆等）。除了键盘上出现的字符外，还可以包括用 ASCII 码形式给出的图形字符或控制字符。

10.1.3　字符串长度

字符串中的每一个字符在内存中占一个字节。一个字符串占据一段连续的内存单元。例如字符串“ABCDEF”有 7 个字符，因此它在内存中占 7 个单元。

字符串的长度是指字符串常量中的字符的个数。例如：

```
"This is a book"                    该字符串的长度为 14 个字节
```

引号不是字符串的值，它只是字符串常量的起、止界限，而且在引号中不能再套引号。

例如：

"I say:"This is a book""　　这是 Quick BASIC 中不允许的

一个汉字占两个英文字符的宽度。Quick BASIC 规定一个字符串最多可以容纳 32767 个字符。

10.1.4　求字符串长度函数 LEN

Quick BASIC 提供的 LEN 函数可以方便、准确地测出字符串的长度。

格式：

LEN（〈字符串表达式〉）

功能：求字符串表达式中字符的个数，即字符串的长度。

说明：〈字符串表达式〉既可以是字符串常量，也可以是字符串变量。

例如：

```
PRINT LEN（"ABCDEFG"）                '输出：  7
PRINT LEN（“ 10*20＝ ”）              '输出：  6
PRINT LEN（""）                       '输出：  1（空格也为字符）
```

10.1.5　字符串常量的定义

字符串常量是指在程序运行过程中始终保持不变的字符串。Quick BASIC 允许使用两种形式的字符串常量。

1．显示的字符串常量

显示的字符串常量就是用双引号括起来的一串字符。如："I am a student."

在使用字符串时应注意以下几点：

（1）空格或者间隙也是一个字符，应计在字符串长度之内。

（2）字符串中，大小写字母是有区别的。如"ABCD"与"abcd"被看成是 2 个不同的字符串。

（3）无任何字符的串叫“ 空串 ”，用""来表示，空串的长度为零。

（4）应将字符串与数值严格区分开来。例如：12345 是一个数值，而"12345"则是一个字符串，其中每个都是独立的字符。字符串不能用作算术运算。

2．符号字符串常量

用一个符号名代表一个字符串常量：

```
CONST xm="Wang Hong"
```

在本程序块中，xm 与"Wang Hong"等价，如果有：

```
PRINT xm
```

则输出 Wang Hong。

为了阅读程序方便，可以在符号常量名后加“ $ ”，以表明是字符串常量。如：

```
CONST xm$="Wang Hong"
```

注意把符号字符串常量与字符串变量相区别。字符串常量的值是不能改变的，例如下面的程序段是非法的：

```
CONST xm$="Wang Hong"                 '定义了一个符号字符串变量 xm$="Wang Hong"
xm$="Li Qiang"                        '又要用赋值语句给字符串变量 xm$赋值
```

10.2　字符串变量和数组

10.2.1　字符串变量的定义

存放数值的变量称为数值变量，存放字符串的变量就称为字符串变量。数值型变量的值是一个数值，而字符串变量的值则是一个字符串，并且，程序执行的过程中，这个值是可以改变的。

Quick BASIC 规定了两类字符串变量：变长字符串变量和定长字符串变量。

1．变长字符串变量

在程序执行过程中，变长字符串变量长度可在 0~32767 范围内增加或减短，这种变量使用起来比较方便灵

活。

定义变长字符串变量类型的方法有 3 种：

（1）用变量名加上类型申明符

格式：变量名$

变量名的命名规则与数值变量相同。例如：al$,b$,erx$,dap$等都是合法的变量名。例如：

```
jtdz$"kaifeng"
```

其中 jtdz$就是一个字符串变量，现在已将字符串"kaifeng"赋给了变量 jidz$，以后需要输出字符串"kaifeng"时，就和使用数值变量一样，直接在 PRINT 语句中输出变量 jtdz$即可。

例如：

```
PRINT jtdz$
```

输出结果为：

```
kaifeng
```

（2）用 DEFSTR 类型说明语句

用类型说明语句（DEFSTR 语句）来定义以某个字母开头的变量为字符串变量。

格式：DEFSTR〈字母表〉

其中，字母表是单个字母或一段连续的字母范围。

例如：

```
DEFSTR a,m-p
ma="ABCD"
PRINT ma
```

上例中，DEFSTR 语句是说明以 a 和 m,n,o,p 开头的所有变量为字符型变量，ma 在的说明的范围内，所以是字符串变量。

（3）在 DIM 语句中使用 AS STRING

格式：DIM〈变量名〉AS STRING

例如：

```
DIM xm AS STRING
xm="Li Qiang"
PRINT xm
```

应注意，用 DIM…AS STRING 语句定义字符串变量时，变量名不应包含类型说明符，例如下面的用法是非法的：

```
DIM xm$ AS STRING                    '将 xm 写成了 xm$
```

2. 定长字符串变量

变长字符串虽然灵活方便，但人们有时希望在某些字符串取固定的长度，尤其是在打印输出时使上下对齐。

定长字符串是指它在程序执行过程中，始终保持其长度不变的字符串。定长字符串变量的长度要用 DIM 语句进行说明。

格式：DIM 〈变量名〉 AS STRING*n

例如：

```
DIM aa AS STRING*8
```

在定义了 aa 为定长字符串变量之后，可以用赋值语句为其赋值。上面的语句说明了 aa 的字符长度为 8，如果赋的值长度超过了 8，则多余的字符被截去，如果少于 8，则在其后补足空格。例如：

```
aa$="ABCDEF12345"            '其值为："ABCDEF12"
aa#="ABC"                    '其值为："ABC      "（后面补 5 个空格）
```

串变量在未赋值之前的初值为空字符串，对于可变长字符串，其长度为 0；而对定长字符串，其他长度为 n，即使用空格为其赋值，如 aa$=""其长度不变，也为 8。

使用定长字符串，可以实现定格输出。

[例 10-1] 阅读下面的程序。

```
'EXAMPLE 1
CLS
DIM a AS STRING *6
FOR i=1 TO 4
READ a
PRINT i,a
NEXT i
DATA ABCDEFGHIJKLMN,ABC,XYZOP ,&&&&&&&&&
END
```

运行结果为：

```
1       ABCDEF
2       ABC
3       XYZOP
4       & & & & & &
```

从上例的输出结果可以看出，字符串是以左边对齐，右边补空格的形式输出的。

10.2.2 字符串数组

字符串数组的概念与前介绍的数组相似，即字符串数组是一组有序的字符串变量的集合。字符串数组中的元素称为“串下标变量”。根据字符串数组的维数，同样可分为：一维、二维、三维等。例如：

```
a$（28）,b$（2,3）,c$（10,20,30）
```

其中 a$,b$,c$均为字符串数组名，其下标书写规则同数组下标变量。在同一程序中，字符串数组名与字符串变量名可以相同。当程序中使用字符串数组时，同样使用 DIM 语句进行说明。一个 DIM 语句可同时对字符串数组和数值数组进行说明。例如：

```
DIM a（8）,b（2,3）,a$（5）,cb$（4,5）
```

[例 10-2] 有一班级学生，要求按学生姓名的汉语拼音顺序输出。

问题分析：建立数组来存放学生姓名，用字符串比较的方法来实现顺序输出。程序为：

```
'EXAMPLE 2
CLS
DIM a$（6）                    '定义字符串数组
FOR i=1 TO 6                   '各数组元素读入数据
READ a$（i）
NEXT i
FOR i=1 TO 5                   '排序
p=i
FOR j=i+1 TO 6
IF a$（j）<a$（p）THEN p=j
NEXT j
IF p<>i THEN SWAT a$（p）,a$（i）      '交换数组元素的值
NEXT i
FOR i=1 TO 6                           '排序后输出
PRINT a$（i）                          '输出
NEXT i
PRINT
DATA   "AHANG SAN","LIU FENG","WANG AHI"
DATA   "SONG HONG","YANG PING","KANG XIANG"
END
```

程序运行结果为：

```
KANG XIANG
LIU FENG
SONG HONG
WANG HAI
YANG PING
ZHANG SAN
```

10.3 字符串变量的赋值

我们已经知道要把一个数值送给一个数值型变量，可以使用 LET、INPUT、READ…DATA **语句来实现。同理也可以使用这** 3 **种赋值语句字符型变量赋值，另外还可以用行输入语句** LINE INPUT **语句来为串变量赋值。**

10.3.1 用 LET 语句赋值

与数值变量相同，但要注意字符串常量必须用双引号括起来。

例如，下面的程序用 LET **语句对字符型变量赋值。**

```
CLS
LET b$="BeiJing"
LET c$="ShangHai"
PRINT b$,c$
END
```

运行结果为：

```
BeiJing         ShangHai
```

10.3.2 用 INPUT 语句赋值

INPUT **语句也可以用来从键盘输入字符串数据，其用法与从键盘输入数值数据基本相同。但要注意：**

（1）在键盘上输入字符串变量的值时，字符串可用引号也可不用引号括起来。但在下列情况下必须用引号括起来：

- **字符串中有首尾空格时，要用引号括起来。否则首尾空格被删去。**
- **字符串中带有逗号时，要用引号括起来。因逗号是两个数的分隔符，否则一个字符串将在逗号处分开。**

（2）在一个 INPUT **语句中可以既有数值型变量又有字符串变量，当从键盘回答时，应键入相应的数字和字符串。否则把一个字符串赋给一个数值型变量时将会出错。**

（3）建议初学者在用 INPUT **语句给变量赋值时，最好用引号将字符串括起来，以避免不必要的出错。**

[**例** 10-3] **阅读下面的程序。**

```
'EXAMPLE 3
CLS
INPUT"姓名：";xm$
INPUT"学号：";xh$
PRINT xm$,xh$
```

运行结果为：

```
姓名：？ Wang  Hong
学号：？ 2000102
Wang Hong                    2000102
```

10.3.3 用 READ…DATA 语句赋值

READ…DATA 语句除了可以用来读入数值变量外，还可以读字符串。其用法与数值型数据基本相同。但要注意：

（1）在一个 READ 语句中既可以出现数值型变量，也可以出现字符串变量，同样在 DATA 语句中也可以出现这两类数据。

（2）如果 READ 语句中既有数值型变量，又有字符型变量，那么 DATA 语句中相应的数据类型，必须与 READ 语句中相应的数据类型一致，否则将会出现数据类型不匹配的错误。

（3）DATA 中的字符串可用引号也可不用引号括起来。但在下列情况下必须用引号括起来：

- 字符串中有首尾空格时，要用引号括起来,否则首尾空格被删去。
- 字符串中带有逗号时，要用引号括起来。因逗号是两个数的分隔符，否个字符串将在逗号处分开。

[例 10-4] 编制程序，输出学生姓名、学号、年龄、成绩，假设总共有 6 个学生。

程序为:

```
'EXAMPLE 4
CLS
PRINT"姓名","学号"，"年龄","总分"
FOR i=1 TO 6
READ    na$,no$,age,totaol
PRINT na$,no$,age,totaol
NEXT i
DATA "丁红莉","101",15,578
DATA "李小翔","102",16,532
DATA "刘大勇","103",15,512
DATA "杜姗姗","104",17,590
DATA "孙　红","105",16,580
DATA "张小华","106",16,587
END
```

运行结果为

姓名	学号	年龄	总分
丁红莉	101	15	578
李小翔	102	16	532
刘大勇	103	15	512
杜姗姗	104	17	590
孙　红	105	16	580
张小华	106	16	587

10.3.4 用 LINE INPUT 语句赋值

格式：

LINE INPUT［;］ ["〈提示字符〉";］〈串变量〉

功能：从键盘上读入一整行字符串（从提示字符之后到按回车键之前的所有字符），赋给字符串变量。

说明：

（1）如果 LINE INPUT 后紧跟一个分号，那么按回车键结束输入时，光标仍保留在输入的同一行上。

（2）LINE INPUT 语句称为行输入语句，执行此行语句时，提示字符的后面不显示问号“？”，这一点是与 INPUT 语句不同的。

（3）LINE INPUT 语句对输入的字符没有限制，包括引号、逗号、空格等，这一点弥补了 INPUT 语句的不足。

（4）该语句一行最多可以连续输入 32767 个字符。

（5）输入字符串的中途，按〈Ctrl〉+〈Break〉键能停止行输入语句的操作，这时，返回到命令状态并显示 OK。如果需要，可以输入 CONT 命令，再重新执行 LINE INPUT 语句。

[例 10-5] 观察下面的程序及运行结果。

程序为：

```
'EXAMPLE 5
CLS
PRINT"**********************"
LINE INPUT"A$=";a$
PRINT"**********************"
PRINT a$
PRINT"**********************"
END
```

运行结果为：

```
**********************
A$= She said:"Good Moring!"
**********************
She said:"Good Moring!"
**********************
```

10.4 字符串表达式及字符串的比较

10.4.1 字符串表达式

字符串运算符只有一个，就是“+”，它把两个或多个字符串型的常量、变量、函数依次首尾连接起来，组成一个新的字符串。

其一般格式为：

```
x1$+x2$+x3$+…
```

其中，“+” 连接运算符，x1$,x2$,x3$,…,为字符型常量、变量或字符型函数。

例如：

```
"THIS"+"IS"+"A"+"BOOK"
```

将得到一个新的字符串：

```
THIS IS A BOOK
```

用“+”将字符串数据连接起来的式子，称为字符串表达式。连接后的字符串长度不应超过 32767 个字符。

“+”两侧可以是字符串常量、字符串变量或值为字符串的函数。

同数值表达式一样，可以将一个字符串表达式的值赋给一个字符串变量或字符串数组元素。例如：

```
ss$="THIS"+"IS"+"A"+"BOOK"
```

则 ss$的值为

```
THIS IS A BOOK
```

10.4.2 字符关系表达式

在非数值计算中，经常会遇到字符串的排序或检索问题。例如要求检索学生学号，并按学号排序，这就产生了字符串的比较问题。

用关系运算符可以写出各种不同的关系表达式。例如：

```
x$="Quick BASIC"
"THE"<"THESE"
```

```
"NAME"+"LI HAO">"LIU BING"
```

关系表达式只有“真”、“假”两个值。如执行第一句时，若 x$中存放的也是 Quick BASIC，则表达式的值为“真”，否则为“假”（式中的“＝”是关系运算符，而不是赋值语句）。

10.4.3 两个字符串大小的比较

在计算机内部，所有字符都是以 ASCII 码表示的。比较两个字符的大小，就是比较它们 ASCII 码值大小，例如字符 A 的 ASCII 码值为 65，B 为 66，所以关系表达式"B">"A"的值为“真”。

当两字符串的长度不等时，其比较的方法是将两个字符串从左至右逐个字符进行比较，直到出现不同的字符为止，其字符串的大小就由这两个不同的字符在 ASCII 码中的位置而定。

字符串比较有如下简单的规律：

（1）同规格的字母比较，按字母的顺序排，后面的大。

（2）大小写字母比较，小写字母大于该大写字母。

（3）数字字符比较，按数值的大小排，数值大的大。

（4）字母与数字比较，数字（0~9）的代码均小于字母（A~Z）的代码。

（5）空格比数字、标点、字母等常见符号都小。

记住这以上几点就可以较快得出比较的结论。

[例 10-6] 阅读下面的程序。

```
'EXAMPLE 6
CLS
INPUT"ARE YOU A TEACHER?";x$
IF x$="YES"THEN
PRINT"YOU ARE A TEACHER!"
ELSE
PRINT"YOU ARE NOT A TEACHER!"
END IF
END
```

在执行该程序时，如果输入“YES”，则程序自动输出“YOU ARE A TEACHER！”，如果输入“yes”，则程序就按非“YES”对待，程序将自动输出“YOU ARE NOT A TEACHER！”，这是因为大小写字母是不同的字符，另外空格也参与比较，“YES”与“ YES”是不相等的两个字符串。

[例 10-7] 有若干个城市的名字，找出按英字母顺序排列在最后面的城市名字。

程序为：

```
'EXAMPLE 7
CLS
READ a$                          '读入第一个名字
FOR i=1 TO 7
READ b$                          '读数
IF a$<b$ THEN a$=b$              '比较较大字符串数据
NEXT i
PRITN   "This last city is:";a$   '输出结果
END
DATA "BEI JING","SHANG HAI"，"CHANG CHUN","WU HAN"
DATA"XI AN","LUO YANG","ZHENG ZHOU","ZHOU KOU"
```

程序运行结果为：

```
This last city is : ZHOU KOU
```

10.4.4 字符串的检索

在用计算机进行文字处理时，常利用 INSTR 函数来从一段文章中查找某一个关键字。其查找过程无非是利用字符串的比较来进行的。

格式：INSTR（n,y$,x$）

功能：从 y$中但找子串 x$的位置。

说明：

（1）n 为一数值表达式，其值表示查找的起始位置，1≤n≤255.省略 n 时，默认为 1。

（2）x$,y$为字符串表达式。

（3）如果 y$中含有 x$,则输出 x$在 y$中的位置。

如果 n>LEN（y$）或者找不到 x$，则 INSTR 函数的值为 0。

如果从 n 开始，x$在 y$中不止出现一次，则只指出第一次出现的位置。

[例 10-8] 阅读下面的程序。

```
'EXAMPLE 8
a$="BASIC PROGRAMMING"
PRINT INSTR（a$,"BASIC"）,
PRINT INSTR（a$,"A"）,
PRINT INSTR（4,a$,"A"）,
PRINT INSTR（a$,"Y"）
END
```

运行结果为：

```
1       2       12       0
```

[例 10-9] 编制一个查找某学生姓名的程序。

问题分析：

（1）将全部学生的学号和名字放在 DATA 语句中。

（2）输入要找的学生名字，与注册的学生名一一进行比较，如找到了（两名字相等）就输出结果，并显示是否还要找？如果找不到，就显示“找不到”等信息。

设置变量：

n$,c	为在册学生名及学号
x$	为要找的学生名
d$	回答是否还要继续找
f	判断标志
f=0	表示没有找到
f=1	表示已找到
n	在册学生数
I	计数器

程序为：

```
'EXAMPLE 9
f=0
i=1
n=6                              '假设学生数为 6
DO WHILE f=0
PRINT TAB（20）;"输入学生名：";
INPUT x$
DO
READ n$,c                        '每次读一个学生名及学号
IF x$=n$ THEN f=1                'f=1 表示已找到
```

```
i=i+1
LOOP UNTIL i>n OR f=1
IF f=0 THEN
PRINT TAB（20）;"没有找到！"
ELSE
PRINT"姓名："x$;"已找到"，"学号为"；c
END IF
PRINT"继续找吗（Y/N）？"；
INPUT d$
IF d$="Y"ORd$="y"THEN
f=0
RESTORE                    '要继续找，则重新设 f=0 并恢复数据区
ELSE
f=1
END IF
i=1
LOOP
DATA Li Fang,1,Yang Hua,2,Liu Ying,3,Du Hua,4,Sun Hong,5,Wang Xiw,6
END
```

程序运行结果为：

```
输入学生名：Wang Hong
没有找到！
继续找吗（Y/N）？Y
输入学生名： LiFang
姓名：Li Fang    已找到      学号为 1
继续找吗（Y/N）？N              （结束程序）
```

10.5 取子字符串

子字符串是从一个字符串中抽出一组相连的字符组成的中一个新的字符串，简称子串，又称字符串的分割。实际上，子串是字符串的一部发。例如 a$的值为“ ABCDEFG ”则“ DEF ”就是 a$的一个子串，当然一个字符串可以有多个子串，如“ ABC ”,“ A ”,“ EFG ”等等都是 a$的子串。

取子串的方法，可以利用 LEFT$、RIGHT$、MID$等函数从一个字符串的左边、右边或中间取一部分字符出来。

10.5.1 LEFT$函数

格式：

LEFT$（x$,n）

功能：从 x$的左边第一个字符起，取出 n 个字符组成一个子字符串。

说明：

（1）0≤n≤255。

（2）如果 n≥x$的长度时，则得到整个字符串；如果 n=0，则得到空字符串（即子串长度为零）。

[例 10-10] 阅读下面的程序。

```
'EXAMPLE 10
a$="KAIFENG.HENAN.CHINA"
```

```
PRINT LEFT$（a$,8）
END
```

运行结果为

```
KAIFENG.
```

[例 10-11] **有一批城市名字是用汉语拼音拼写的，希望打印出字母以“** ZH **”开头的城市名。**

程序为：

```
'EXAMPLE 11
CLS
FOR i=1 TO 8
READ a$
IF LEFT$（a$,2）="ZH"THEN PRINT a$
NEXT i
DATA "ZHU HAI"，"XI AN","ZHENG ZHOU","ZHOU KOU"
DATA"LUO HE","SAN MEN XIA","ZHU MA DIAN","ZHEN JIANG"
END
```

运行结果为：

```
ZHU HAI
ZHENG ZHOU
ZHOU KOU
ZHU MA DLAN
ZHEN JIANG
```

10.5.2 RIGHT$函数

格式：

```
RIGHT$（x$,n）
```

功能：从 x$**的右边取出** n **个字符串组成一个子字符串。**

说明：同 LEFT$**函数一样。**

[例 10-12]**某校学生学号的最后两位是表示学生性别 ，后面两位是“** 10 **”则认为是女同学，其他的为男同学。要求输入学生学号，并统计出男、女同学的人数。**

程序为：

```
'EXAMPLE 12
boy=0                                    '男生人数计数器赋初值
girl=0                                   '女生人数计数器赋初值
INPUT "输入需统计的学生总数："，n
FOR i=1 TO n                             '输入 n 个学生学号
INPUT "请输入学生学号："a$
IF RIGHT$（a$,2）="10"THEN
girl=girl+1
ELSE
boy=boy+1
END IF
NEXT i
PRINT "boy=";boy
PRINT "girl=";girl
END
```

程序运行结果为：

```
输入需统计的学生总数：  6
请输入学生学号：  9910110
请输入学生学号：  9934044
请输入学生学号：  9910230
请输入学生学号：  10010110
请输入学生学号：  45800610
请输入学生学号：  902218810
boy= 2
girl= 4
```

10.5.3 MID$函数

格式：

MID$（x$,m[,n]）

功能：从 x$中第 m 个字符开始，取 n 个字符组成一个子字符串。

说明：

（1）1≤m≤255,0≤n≤255。

（2）当 n=0 或 m 大于字符串的长度时，得到空字符串。

（3）当 n 省略或大于要取的字符串长度时，则得到从 m 开始的怕有字符。

[例 10-13] 将 26 个英文字母按逆序打印出来。

```
'EXAMPLE 13
CLS
a$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
PRINT"逆序打印 26 个英文字母为："
FOR i=26 TO 1 STEP -1
PRINT MID$（a$,i,1）;
NEXT i
PRINT
END
```

程序运行结果为：

```
逆序打印 26 上英文字母为：
ZYXWVUTSRQPONMLKJIHGFEDCBA
```

10.5.4 删除字符串的首尾空格

在用定长字符串时，经常要删除其首尾空格。可利用下面的函数实现。

格式：

LTRIM$（x$）

RTRIM$（x$）

功能：LTRIM$（x$）**函数能删除** x$**开头的空格；**RTRIM$（x$）**函数能删除** x$**结尾的空格。**

[例 10-14] **阅读下面的程序。**

```
'EXAMPLE 14
CLS
DEFSTR x                                    'x 为可变长字符串变量
DIM x AS STRING*8，y AS STRING*8   'x,y 为定长串变量
x="Quick BASIC,"
y="      good"
PRINT x,"长度＝"；LEN（x）
PRINT y,"长度＝"；LEN（y）
x=RTRIM$（x）                                  '消右边空格
```

```
y=LTRIM$（y）                    '消左边空格
PRINT x,"长度＝";LEN（x）
PRINT y,"长度＝";LEN（y）
END
```

运行结果为：

```
Quick BASIC,    长度＝8    （x 的右边补 1 个空格）
good            长度＝8
Quick BASIC,    长度＝8    （x 的右边补 1 个空格）
good            长度＝8
```

10.6 字符串的生成

前面介绍了用赋值的方法可以生成一个字符串，现在我们介绍使用 STRING$,SPACE$**函数来生成一串由某个字符重复组成的字符串。**

10.6.1 STRING$函数

格式：

STRING$（n,m **或** x$）

功能：得到一个由 n **个** ASCII **码** m **个对应的字符组成的字符串，或得到由** n **个** x$**中第一个字符所组成的字符串。**

说明：

（1**）**0≤n≤255,0≤m≤255。m **为数值时，则代表该字符的** ASCII **码。**

（2**）这个函数常用于制图或打印表格，例如打印一行“** - **”或“** * **”等。**

[**例** 10-15] **阅读下面程序。**

```
'EXAMPLE 15
PRINT STRING$（10,"*"）
PRINT STRING$（7,66）
PRINT STRING$（10,"-"）
PRINT STRING$（12,"bcd"）
END
```

程序运行后输出：

```
**********
BBBBBBB
----------
bbbbbbbbbbbb
```

10.6.2 SPACE$函数

格式：

SPACE$（n）

功能：产生 n **个空格组成的空格字符串。**

说明：

（1**）**0≤n≤255。

（2**）该函数常用于数据对齐、规定格式等输出。**

10.6.3 字符串中大小写字母之间的转换

在程序中经常需要将某些大写字母改写成小写字母，或把小写字母改写成大写字母。在 Quick BASIC 中可以利用 LCASE$、UCASE$函数进行变换。

格式：

```
LCASE$（x$）
UCASE$（x$）
```

功能：LCASE$（x$）函数把 x$中的大写字母改写成小写字母；UCASE$（x$）函数把 x$中的小写字母改写成大写字母。

例如：

```
PRINT LCASE$（"ABCdefghijk"）
PRINT UCASE$（"ABCdefghijk"）
```

输出：

```
abcdefghijk
ABCDEFGHIJK
```

这两个函数可以使键盘输入标准化，简化两个字符间的比较。例如：

```
INPUT"输入（yes/no）";a$
IF LEFT$（a$,1）="Y"OR LEFT$（a$,1）="y"THEN n=n+1
```

可以利用 UCASE$函数简写为：

```
IF UCASE$（LEFT$（a$,1））="Y"THEN n=n+1
```

10.6.4 改变字符串中的字符语句 MID$

字符串中的字符一般是没有改变的，若要改变，需重新设置一个串变量。而 Quick BASIC 提供的 MID$语句（不要与 MID$函数混淆）能够通过置换的办法来改变字符串中的字符。

格式：

```
MID$（x$,m,n）=y$
```

功能：在 x$中从位置 m 开始被 y$中的前 n 个字符置换，从而改变 x$的值。

说明：

（1）1≤m≤255,0≤n≤255。

（2）选择项 n 表示改变 x$中字符的个数，如果省略 n 或 n>LEN（y$）,则用 y$的全部字符去置换。

（3）执行此语句后，x$的长度是不能改变 。当需要置换的位置不够时，多余的 y$中的字符被截去。

[例 10-16] 阅读下面的程序。

```
'EXAMPLE 16
a$="ABCDEFG"
b$=a$:c$=a$:d$=a$
MID$（b$,4,3）="XXX"
MID$（d$,4,3）="XX"
MID$（d$,4,3）="XXXXXX"
PRINT a$;b$;c$:d$
END
```

运行结果为：

```
ABCDEFG  ABCXXXG  ABCXXFG  ABCXXXG
```

[例 10-17] 编制程序，将"Your are a student"中的小写字母全部改成大写字母。

问题分析：

（1）从 ASCII 码表中查得，小写字母在 97~122 之间，与大写字母的 ASCII 码值相差 32。从标题中逐个取出字符，用 ASC 函数将它转换成 ASCII 码，如果此值在 97~122 之间，说明它为小写字母。

（2）将小写字母的 ASCII 码值减去 32 后，利用 CHR$函数和 MID$语句转换成大写字母，去置换相应的小

写字母，即 MID$（x$,1,1）=CHR$（x-32）。

程序为：

```
'EXAMPLE 17
CLS
a$="You are a student"
cd=LEN（a$）
FOR i=1 TO cd
dgzf=ASC（MID$（a$,i,1））                    '取一个字符转换成 ASCII 码
IF dgzf>96 AND dgzf<123 THEN                 '判断是否为小写字母
MID$（a$,i,1）=CHR$（dgzf-32）                '用大写字母置换小写字母
END
NEXT i
PRINT a$
END
```

运行结果为：

```
YOU ARE A STUDENT
```

10.7 字符串与数值的相互转换

Quick BASIC 不允许将数值赋值给一个字符串变量，也不允许将字符串变量赋给数值变量，否则将会产生数据类型不匹配的错误。现在我们介绍 ASCII 码与字符的相互转换以及数值与字符的相互转换。

10.7.1 ASCII 码与字符的相互转换

1.ASC 函数

格式：

```
ASC（x$）
```

功能：将 x$的第一个字符转换成它所对应的 ASCII 码，并以十进制形式表示。

说明：如果 x$是空串，则输出“非法函数调用”的错误信息。

例如：

```
PRINT ASC（"A"）                    '输出：65
PRINT ASC（"J"）                    '输出：74
PRINT ASC（"BGFJ"）                 '输出：66
```

2.CHR$函数

格式：

```
CHR$（n）
```

功能：将 n 转换成 ASCII 码对应的字符。其中 n 为 ASCII 代码。

说明：

（1）0≤n≤255.

（2）字符代码 0 至 127 在 ASCII 码中是标准的，并且所有的机器上都一样。代码在 128 至 255 在不同的机器上有所不同，用时请查看使用说明书。

例如：

```
PRINT CHR$（65）                    '输出：A
PRINT CHR$（65+32）                 '输出：a
PRINT CHR$（32）                    '输出：（空格）
```

（3）Quick BASIC 规定，字符串中不能带有引号（"）,然而用 CHR$函数可以在字符串中插入引号。

[例 10-18] 阅读下面的程序。

```
'EXAMPLE 18
CLS
a$="Oh,This is my book."
b$="she said."
c$=DHR$（34）+a$+DHR$（34）+b$
PRINT c$
END
```

运行结果为：

```
"Oh,This is my book."she said.
```

10.7.2 数值与字符串的相互转换

1．STR$函数

格式：

```
STR$（n）
```

功能：将算术表达式 n 的值转换成字符串。

说明：

（1）转换之后，括号内的数值已变为字符串，所以不能再进行算术运算了。

（2）如果 n 为正数，那么由 STR$转换成的字符串前保留一个引导空格。

[例 10-19] 阅读下面的程序。

```
'EXAMPLE 19
CLS
a=-100
b=200
a$=STR$（a）
b$=STR$（b）
PRINT a,a$
PRINT b,b$
PRINT a+b,a$+b$
END
```

运行结果为：

```
-100          -100
200           200
100           -100 200
```

2．VAL 函数

格式：

```
VAL（x$）
```

功能：将 x$转换成数值。VAL 是 STR$的反函数。

说明：

（1）VAL 函数自动忽略 x$中的空格或一直遇到非数字为止。

（2）如果 x$中的第 1 个字符不是数字，那么 VAL（x$）的值为零。

例如：

```
PRINT VAL（"-2"）              '输出：-2
PRINT VAL（"8.7654"）          '输出：8.7654
PRINT VAL（"1.23E5"）          '输出：123000
PRINT VAL（"987FF"）           '输出：987
```

```
PRINT VAL（"8+3"）          '输出：8
```

10.7.3 数制与数制之间转换

1．HEX$函数

格式：

HEX$（算术表达式）

功能：把十进制数转换为十六进制数的字符串。

说明：

（1）算术表达式为十进制的算术表达式。

（2）在进行转换之前，先把表达式的值四舍五入成为整数，然后给出该整数转换后的字符串。

[例 10-20] **由键盘输入任意一个十进制数，把它转换为十六进制数。**

程序为：

```
'EXAMPLE 20
CLS
INPUT"请输入一个十进制数 X＝";x
a$=HEX$（x）
PRINT"十进制数";x;"其转换为十六进制数";a$
END
```

程序运行结果为

```
请输入一个十进制数 X＝？ 79
十进制数 79 其转换为十六进制数  4F
```

2．OCT$函数

格式：

OCT$（算术表达式）

功能：把十进制数转换为八进制数的字符串。

说明：

（1）算术表达式为十进制的算术表达式。

（2）在进行转换之前，先把表达式的值四舍五入成为整数，然后给出该整数转换后的字符串。

[例 7-21] **由键盘输入任意一个十进制数，把它转换为八进制数。**

程序为：

```
'EXAMPLE 21
CLS
INPUT"请输入一个十进制数 X＝";x
b$=OCT$（x）
PRINT"十进制数";x;"其转换为八进制数";b$
END
```

程序运行结果为：

```
请输入一个十进制数 X=? 79
十进制数 79 其转换为进制数 117
```

10.8 自选输出格式

10.8.1 屏幕定位语句 LOCATE

格式：

LOCATE［〈行号〉］［，〈列号〉］

功能：把光标移动到指定位置。

说明：

（1）〈行号〉为屏幕上的行号，是一个结果为整型的算术表达式，取值范围在 1~25 之间。若不选该项，默认为当前行号。

（2）〈列号〉为屏幕上的列号，是一个结果为整型的算术表达式，取值范围在 1~80 之间。若不选该项，默认为当前列号。

例如下面的程序段：

```
CLS                          ’清除屏幕
PRINT "您好！"               '打印字符串：您好！
LOCATE 5.20                  '把光标定在第 5 行第 20 列
PRINT "欢迎您！"             '打印字符串：欢迎您！
LOCATE 10，40                '把光标定在第 10 行第 40 列
PRINT "再见！"               '打印字符串：再见！
END
```

10.8.2 屏幕格式输出语句 PRINT USING

1．PRINT USING 语句的格式及功能

格式：

```
PRINT USING"〈格式字符串〉"；〈表达式表〉 [{，丨；}]
```

功能：在屏幕上按照指定的格式显示字符或数字。〈格式字符串〉用来指定显示的格式。

在 PRINT USING 语句中各输入各项之间的间隔和显示格式仅取决于格式字符串。在〈表达式表〉后面的逗号和分号的作用与 PRINT 语句中的作用相同。

2．PRINT USING 语句用于输出字符串

在 PRINT USING 语句中用于输出字符串的格式符有：!、\n 个空格\、 &共 3 种。

（1）“ !”格式符。表示仅显示字符串中的第一个字符 。例如：

```
PRINT USING "!";"GOOD"
```

运行结果为：

```
G
```

（2）“ \n 个空格\ ”格式符。表示显示字符串中的 2+N 个字符，N 是两个反斜号间的空格数。如果反斜号之间没有空格，则仅显示两个字符；如果有一个空格，则显示 3 个字符等等。如果要显示的字符串比指定的显示域长，则右边多出的字符不显示；如要显示的字符比指定的显示域短，则显示时向左对齐，右边用空格填满。例如：

```
PRINT USING "\      \";"AB"
PRINT USING "\      \";"CDEFG"
```

显示结果为：

```
AB
CDEF
```

（3）“ & ”格式符。表示指定可变长度的字符串显示格式。当用 & 指定显示格式时，显示出的字符串与输入的一样。例如：

```
a$="AB"
b$="CDEFG"
PRINT USING "&";a$;b$
```

显示结果为：

```
ABCDEFG
```

3．PRINT USING 语句用于输出数值

PRINT USING 语句用于输出数值的格式符有：#、+、-、、$$、**$、__等。**

（1）“ # ”格式符。表示一个数字位置。如果要显示的数字位数少于#号的个数，则在显示时，数字向右对

齐，前边用空格补足。少几位，补几个空格。例如：

```
PRINT USING"####";18
PRINT USING"####";-75
```

显示结果为：

```
18
-75
```

如果需指定小数点的位置时，可以用“ . ”来确定。格式字符串中小数点右边#号的个数指定了显示到小数后的位数。被显示数字的小数部分位数不足的，在右边的以 0 补足；超过的四舍五入。例如：

```
PRINT USING"###.##";3.2
PRINT USING"###.##";23
PRINT USING"###.##";9.876
```

显示结果为：

```
3.20
23.00
9.88
```

（2）“ + ”格式符。表示将数字的正负号与数字一同显示。如果加号在格式字符串中的最左边，在显示数字时，数字的符号在数字的前边；如果加号在格式字符串的最右边，则在显示数字时，数字的符号显示在数字的后边。例如：

```
PRINT USING"+##.##";21.4
PRINT USING"+##.##";-98.932
```

显示结果为：

```
+21.3
-98.93
```

又如：

```
PRINT USING"##.##+";21.4
PRINT USING"##.##+";-98.932
```

显示结果为：

```
21.40+
98.93-
```

（3）“ - ”格式符。表示如果减号在格式字符串的末尾，则显示的负数的符号在右边。例如：

```
PRINT USING"##.##-";-45.98
PRINT USING"##.##-";3.2221
```

显示结果为：

```
45.98-
3.22
```

（4）“ ** ”格式符。表示显示数字时，如果数字的前面有空格，则用*填充。每一个*号代表一个数字位置。例如：

```
PRINT USING"**#.#";123.4
PRINT USING"**#.#";5.4
PRINT USING"**#.#";-2
```

显示结果为：

```
123.4
**5.4
*-2.0
```

（5）“ $$ ”格式符。表示显示数字时，使一个$号显示在数字的左边。每个$号也代表一个数字位置。例如：

```
PRINT USING"$$#.#";3.5
```

```
PRINT USING"$$#.#";23.1
```

显示结果为：

```
$3.5
$23.1
```

格式符“ **$ ”结合了**和$$的功能。在数字的左边显示$，并在$的左边空格（如果有的话）填充*号。格式字符串中每个*或$均代表一个数字的位置。例如：

```
PRINT USING"**$#.#";3.46
PRINT USING"**$$#.#";34.5
```

显示结果为：

```
**$3.5
*$34.5
```

（6）“ , ”格式符。如果在格式字符串中，逗号出现在小数点的左边，则被显示的数字从小数点向左每三位数字的左边有一个逗号。如果逗号出现在格式字符串的末尾，则作为字符显示出来。如果在指定了指数（ ^ ^ ^ ^ 或 ^ ^ ^ ^ ^ ）的格式下使用，则逗号不起作用。例如：

```
PRINT USING"#######,.##";1234567.89
```

显示结果为：

```
%1，234，567.89
```

（7）“ ^ ^ ^ ^ ”或“ ^ ^ ^ ^ ^ ”格式符。用来指定指数格式。 ^ ^ ^ ^ 指定以 E+××的格式来显示数字， ^ ^ ^ ^ ^ 则指定以 E+×××的格式来显示数字。小数点可以指定在任何位置。除了指定前导的“ + ”或尾随的“ + ”或“ - ”外，小数点左边有一个字符位置用来显示空格或减号。

例如：

```
PRINT USING"+#.##^^^^";1234
PRINT USING"+#.##^^^^";-567
PRINT USING".##^^^^^";1234
```

显示结果为：

```
+123E+03
-5.67E+02
.12E+004
```

（8）“ - ”格式符。表示将下划线后面的那个字符作为文字显示。例如：

```
PRINT USING"-!##.##___";12.34
```

显示结果为：

```
！12.34__
```

10.9 本章小结

前几章的数据是数值量，本章着重介绍了字符量的概念和它的处理方法，这两种数据有很多相似之处，表 10-1 列出它们之间的对比情况。

表 10-1 数值与字符型量的比较

项 目	数 值 量	字 符 量
常数	数值常数：12，－12.3	字符型常数："ABCD","1234"
变量	数值变量：A，NA，B（6）	字符变量：A$,NAME$,B$(6)
数值	两种数据的数组概念相同，使用时要用 DIM 语句说明。如： DIM A（30），B（40，60），A$(50),NAME$(20),AD$(100)	
函数	算术、随机、取整、符号、 自定义等函数	取子串、数字与字符串之间的转换、置日期时间、大小写字母转换、删除首尾空格等函数
输入	可用 LET，INPUT，READ/DATAO 为变量赋值 变量的初值为 0	除这三种之外，还可用 LINE INPUT 为变量赋值 变量初值为“ 空字符串 ”
输出	使用 PRINT 语句输出。如：PRINT 1，2，A，B4	使用 PRINT 的字符串原样照印功能输出。 如：PRINT “ ABCD ”; NAME$;“ 123 ”

本章语句和函数摘要见表 10-2。

表 10-2 字符串处理用的语句和函数

功 能	语句或函数	作 用
取子字符串	LEFT$ RIGHT$ MID$ 函 数 LIRIM$ RTRIM$	从左边第一个字符开始取 N 个字符 从右边第一个字符开始取 N 个字符 从中间某位置开始取 N 个字符 删去左边空格后，得到的字符串 删去右边空格后，得到的字符串
生成字符串	STRING$ SPACE$	生成 N 个字符所组成的串 生成 N 个空格所组成的串
检索	INSTR	在某个中中查找另一个串
更改字符串	MID$ 语 句 LSET RSET	用一个串置换另一串的某部分 将字符串赋给固定长字段的左边 将字符串赋给固定长字段的右边
数、串的相互转换	RTS$ VAL	把数值转换成字符串 把字符串转换与数值
ASCII、字符互换	ASC CHR$	把字符转换成 ASCII 值 将 ASCII 值转换成字符
测串的长度	LEN	测出串中的字符个数
设 置 时间、日期	DATE$ TIME$	设置与读取系统的日期 设置与读取系统的时间
大小写字母转换	LCASE$ UCASE$	将大写字母转换成小写 将小写字母转换成大写
键盘输入字符	INKEY$	从键盘读入一个字符，不回车，不显示

本章的内容在计算机应用中相当广泛，将别是字符处理函数可对字符串进行多方面的处理，这对非数值计算和领域是很有用的。在实际应用中，根据问题的要求综合使用这些函数，会进一步体会到它们的方便之处。

通过本章的两个应用例题，复习了如下几个字符串函数：如 MID$、STRING$、VAL、INKEY$、CHR$、ASC、UCASE$等函数，同时还复习了前几章的内容。读者可以从中学会如何使用菜单技术和模块化程序设计技术。在此基础上，读者再结合自己的专业作些变化和扩充，就可以开发出很实用的软件来。

习 题

1．下面程序读取 5 个学生的数据，分别为学号、姓名、语言、英语、数学，并计算每个学生的平均成绩。

2．将一个包含 n 位学生数据的记录数组，按平均分数从大到小排序，然后显示排序结果。

3．假设某单位有 10 名职工，试编制程序，将他们的工作编号、姓名、性别、工资输入到一个文件名为“zgqk.dat”的文件。

4．求学生记录最高分，并输出与最高分相差在 10 分以内的学生记录。

5．已知磁盘上存放着某单位全年每次报销的经费（假定为整数），试编写一个程序，从磁盘上读入每次报销的经费，计算其总和，并将结果存入另一新建文件中。

6．假定磁盘上有一个学生成绩文件，存放着 100 个学生的情况，包括学号、姓名、性别和 5 门课程的成绩，试编写一个程序，建立以下 2 个文件：

（1）女生情况的文件；

（2）按 5 门课程平均成绩的高低排列的学生情况的文件（须增加平均成绩一栏）。

7．用随机文件建立一个人事档案管理文件。文件包括编号、姓名、性别、年龄、工作单位、职称等字段。输入若干男女职工档案，并使其完成下述功能：

（1）计算男女职工各多少人。

（2）列出全部师档案。

（3）根据编号查找某人档案。

第十一章　屏幕控制和绘图

图形是一种能迅速且精确地传递信息的有效方式，很容易被人们理解。利用计算机来绘制各种图形、图表已成为各个领域的重要手段。例如，可以用曲线、直方图、饼状图来输出各种信息，以增加形象性，使人一目了然。对于许多应用程序而言，以图形方式描述的结果或输出可能更有意义，有时甚至“一幅图”可以顶“千句话”。

Quick BASIC 提供了彩色绘图的功能，我们可以利用绘图语句及颜色选择，在屏幕上显示各种美丽的图形。本章的主要内容有：

- 屏幕坐标系
- 屏幕方式及颜色的设置
- 基本绘图语句
- 填涂颜色语句 PAINT
- 图形的窗口操作

本章的难点在于对各种绘图语句在综合运用时的合理安排。

11.1　屏幕坐标系

微机提供了两种屏幕显示模式，即文本显示模式和图形显示模式，简称文本方式和图形方式。

11.1.1　文本方式与字符坐标系

在文本显示模式下，显示的最小单位是字符。在这种显示模式下，只能用字符和线条组成图形。因而，这种模式主要用于显示文本（程序的数值计算或字符处理的结果）。此时，整个屏幕可显示 25（行）×40（列）或 25（行）×80（列）个字符，可同时使用 16 种颜色。

文本方式下屏幕显示位置用字符坐标系来描述。这种坐标系以屏幕的左上角为坐标原点，水平方向为 X 轴，垂直方向为 Y 轴，文本方式分为 40 列文本和 80 列文本。如图 11-1 所示。

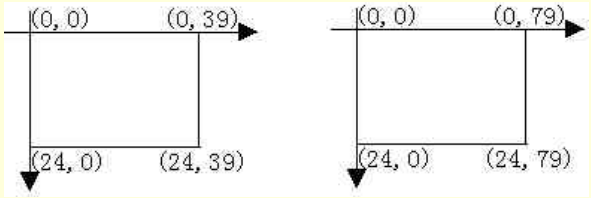


图 11-1　40 列和 80 列字符坐标系

在字符坐标系中，确定屏幕上某字符的位置用屏幕定点语句 LOCATE 来实现。

格式：

LOCATE［〈行号〉］［，〈列号〉］［，［〈变量 1〉］［，［〈变量 2〉］［，〈变量 3〉］］］

功能：把光标移动到指定位置。

说明：

（1）〈行号〉与〈列号〉为屏幕上的行号和列号，是结果为整型的算术表达式。若省略，默认为当前行号和当前列号。

（2）〈变量 1〉说明光标是否可见。当值为 1 时，表示光标打开；取值为 0 时，表示光标关闭。若省略，默认为原值。

（3）〈变量 2〉为屏幕上光标起始扫描行，是一结果为整型的算术表达式。省略时，默认为原值。

（4）〈变量 3〉为屏幕上光标停止扫描行，是一结果为整型的算术表达式。省略时，默认为原值。

[例 11-1] 阅读、分析下面的程序。

```
'EXAMPLE 1
CLS                     '清除屏幕
PRINT "Hello!"          '打印字符串：Hello!
LOCATE 10,15            '把光标定在第 10 行第 15 列
PRINT "Welcome!"        '打印字符串：Welcome!
LOCATE 15,20,0          '把光标定在第 15 行第 20 列
PRINT "Goodbye!"        '打印字符串：Goodbye!
END
```

11.1.2 图形方式与点坐标系

在图形显示模式下，屏幕上每一个可以控制的单元叫做像素，它是图形的基本元素，一般称为“点”。在这种显示模式下，图形是由一行行整齐排列的像素构成的光栅图形，光栅图形的一个像素对应屏幕的一个光点。Quick BASIC 的绘图语句在屏幕上绘图或涂色就是点亮或关闭这些带有各种颜色的像素点。屏幕上像素的密度，决定了图像的质量，密度越高，图形就越清晰。

通常把屏幕上像素的个数叫做分辨率。例如，如果屏幕的水平方向有 320 个像素，垂直方向有 200 个像素，则该屏幕的分辨率为 320×200。

在图形显示模式下，又有中分辨率和高分辨率之分：

1．中分辨率图形模式

可用于显示图形和字符，其分辨率为 320×200，共有 64000 个像素点。即水平方向像素个数为 320，坐标编号 0~319，垂直方向像素个数为 200，坐标编号为 0~199，显示字符时的行列数为 25 行×40 列。

2．高分辨率图形模式

可用于显示精度较高的图形和字符，其分辨率为 640×350 或 640×200，共有 224000 个或 128000 个像素点，显示字符时的行列数为 25 行×80 列。因此，分辨率为 640×350 的图像要比分辨率为 320×200 的图像清晰。

另外，计算机配置的图形适配器不同，屏幕分辨率也不同；适配器上的显示缓冲存储区大小不同，也有不同的显示页数和颜色数目 。

图形方式中，每个像素在屏幕上显示位置用点坐标系来描述。即用坐标（x,y）来表示，其中 0≤x≤319（或 639），0≤y≤199。

绘图语句中的坐标数据可以用两种形式给出，一种是绝对坐标，另一种是相对坐标。

绝对坐标的参考点是坐标系的原点（0，0），x、y 只能取规定范围内的正整数。

相对坐标是相对于“最后参考点”的坐标，通常把最近一次绘图的结束点称为最后参考点。在相对坐标中，x、y 是相对于最后参考点 x 方向和 y 方向上的位移量，x 和 y 可以是正整数，也可以是负整数。Quick BASIC 中的相对坐标通过绘图语句中的选择项“STEP”来标识。

字符坐标系则精确地指出了屏幕上一个像素的位置。此外，字符坐标系以行、列的形式给出字符位置，而点坐标系中像素的位置是以列、行的形式给出的。LOCATE 语句对屏幕上的图形输出位置没有影响。

11.2 屏幕方式及颜色的设置

为了能根据需要在屏幕上画出图形，首先介绍设置屏幕方式及设置图形颜色的方法。

11.2.1 设置屏幕方式语句 SCREEN

格式：

SCREEN［〈屏幕方式〉］［，［〈颜色开关〉］］［，［〈非当前页号〉］］［，［〈当前页号〉］］

功能：设置屏幕显示方式。

说明：

（1）〈屏幕方式〉表示显示方式的算术表达式。方式 0 为文本方式，其他都为图形方式。根据计算机配制的适配器的不同，有不同的显示方式。SCREEN 语句共有 10 种方式，编号为 0~2、7~13。对 VGA 适配器，常见的屏幕方式为 0，7，8，9，12，13。不同的显示方式，屏幕有不同的分辨率、不同颜色属性以及不同的字符大小。表 11-1 列举了几种常见屏幕显示方式。

表 11-1　常见屏幕显示方式

方式编号	分辨率（列数×行数）	颜色种类	显示器与适配器
0	文本方式	16	
1	320*200	8	CGA，EGA，VGA
2	640*200	2	CGA，EGA，VGA
7	320*200	16	EGA，VGA
8	640*200	16	EGA，VGA
9	640*350	16	EGA，VGA
10	640*350	2	EGA，VGA
11	640*480	2	VGA
12	640*480	16	VGA
13	320*200	256	VGA

（2）〈颜色开关〉一个算术表达式，其取值范围在 0~255，是决定是否显示颜色的开关。表 11-2 列出了各方式下，颜色开关为不同值时，是显示黑白图形还是显示彩色图形。

（3）〈非当前页号〉是一个算术表达式，表示图形数据要写入的显示缓冲区页号，不同的屏幕方式有不同的面数值，省略时默认为 0 页。

（4）〈当前页号〉为一个算术表达式，是指能够显示的缓冲区页号，不同的屏幕方式有不同的页数值，省略时默认为当前页。当图形数据写入的缓冲区页号未指定为当前页时，该图形看不见。

（5）屏幕的省略方式为 0，即文本方式。在显示图形之前，要先用 SCREEN 语句设置相应的屏幕方式。

表 11-2　颜色的显示开关

方式编号	颜色开关为 0 时	颜色开关为非 0 数时
0	黑白	彩色
1	彩色	黑白
其他	黑白	黑白

[例 11-2]　**下面各 SCREEN 语句分别设置了不同的屏幕方式。**

```
'EXAMPLE 2
SCREEN0，1          '设置屏幕为文本方式彩色显示
SCREEN1，1          '设置屏幕为图形方式 1，单色显示
SCREEN7,,1,0        '设置屏幕为图形方式 7，数据写入缓冲区非当前页 1 页
                    '并设置 0 页为显示的当前页
SCREEN 12           '设置屏幕为图形方式 12
```

11.2.2　屏幕颜色设置语句 COLOR

我们在设计屏幕图形时，不仅希望图形具有美丽的外观开头，还希望它具有让人赏心悦目的色彩。Quick BASIC 中提供的 COLOR 语句可以设置字符显示的前景色和背景色，以及各种图形的颜色。

格式 1：

COLOR［〈前景色〉］［，［〈背景色〉］［，［〈边框色〉］］］

格式 2：

COLOR［〈背景色〉］［，［〈调色板〉］］

格式 3：

COLOR［〈前景色〉］［，［〈背景色〉］］

格式 4：

COLOR［〈前景色〉］

功能：设置屏幕显示的颜色。

说明：

（1）该语句可以设置显示器的前景颜色和背景颜色，但是受到屏幕方式的制约。在 ACREEN7~10 **及** 12 **和** 13 **中，〈前景色〉不是颜色号而是颜色的属性号，**COLOR **语句不能决定颜色的范围，它是由适配器/显示器的组合以及用** SCREEN **语句设定的屏幕方式共同决定。**

（2）格式 1 **只能在文本方式下使用，用该语句可以设置前景、背景和边框的颜色。**

〈前景色〉是一个算术表达式，取值范围为 0~31**，表示字符的颜色号。〈前景色〉共** 16 **种编号为** 0~15**，其定义如表** 11-3**。**

表 11-3　**颜色代码及其所表示的颜色**

代 码	颜 色	代 码	颜　色
0	黑	8	灰（或黑）
1	兰	9	亮兰
2	绿	10	亮绿
3	青	11	亮青
4	红	12	亮红
5	洋红	13	亮洋红
6	棕（或暗黄）	14	亮黄
7	白（或灰）	15	亮白（或亮灰）

这 16 **种颜色中，编号** 8~15 **实际上是在前** 0~7 **的基础上加强了亮度。即把** 0~7 **的颜色分别加上** 8 **就可以得到相应的高亮度色，但是没有高亮度的黑色。如果把这** 16 **种编号再加上** 16 **后，不仅能给字符设置一定的颜色，而且可以使字符闪烁。例如，**2 **表示绿色，**10**（即** 2+8**）表示亮绿，**26**（即** 2+8+16**）表示闪烁的亮绿色。**

〈背景色〉是一个算术表达式，取值范围为 0~17**，表示背景的颜色号，要注意背景的颜色不能闪烁。**

〈边框色〉是一个整数表达式，取值范围为 0~15**，表示边框的颜色号。边框颜色也不能闪烁。**

（3）在 COLOR **语句中，〈前景色〉和可以取相同的颜色，此时输入的字符在屏幕上是看不见的。利用这一点，在设置密码时可以达到保密的作用。相反地，若要使字符可见，则必须使前景色和背景色取不同的值。**

（4）格式 2 **用于** SCREEN 1**，即中分辨率图形方式下使用。它可以设置背景色，但不可以设置前景色，可选参数调色板，决定当前颜色代码的属性值。**

〈背景色〉取值范围在 0~15**，表示背景颜色的编号，其含义与文本方式下（即** SCREEN 0**）相同。**

〈调色板〉是算术表达式，其取值范围为 0~255**，当取值为偶数时，选择** 0 **号调色板；当取值奇数时，选择** 1 **号调色板，两个调色板中** 4 **个不同的前景色号如表** 11-4 **所示。**

表 11-4　**在方式** 1 **下前景色号所表示的含义**

前景色号	0 号调色板	1 号调色板
0	当前背景色	当前背景色
1	绿	青
2	红	洋红
3	棕（黄）	白

如果〈背景色〉和〈调色板〉省略，则使用以前设定的值。当用 SCREEN 1 **进入中分辨率图形方式时，背景色为黑色，调色板为** 1 **号。**

（5）格式 3 **用于** SCREEN 7~10**。可设置前景色和背景色，不以设置边框颜色，图形的〈背景色〉的颜色号须在当前屏幕方式下有效颜色号范围内，〈前景色〉是省略的画线颜色。**

（6）格式 4 **用于** SCREEN 12**、**13**，在这两种方式下，不能指定背景色。〈前景色〉是前景颜色的属性，该属性必须在相应屏幕方式的正确范围内。**

[**例** 11-3]　COLOR **语句的应用示例。**

```
SCREEN 0：COLOR 1，2，3          '前景色 1，背景色 2，边框色 3
SCREEN 1：COLOR 2，0             '背景色 2，偶混合号
SCREEN 1：COLOR 2，1             '背景色 2，奇混合号
SCREEN 9：COLOR 1，2             '前景色 1，背景色 2
```

11.3 基本绘图语句

11.3.1 画点语句 PSET 及 PRESET

格式 1：

PSET［STEP］（x,y） ［，〈颜色〉］

格式 2：

PRESET［STEP］（x,y） ［，〈颜色〉］

功能：在屏幕上指定的坐标位置画一个点。

说明：

（1）格式中（x,y）为坐标点，即要画点的坐标。x、y 均为算术表达式，其值取整数，当 x、y 的值超出坐标系的限制时，则无任何显示，也不出错。

（2）当选择 STEP 选择项时，坐标点（x,y）就是相对坐标点，也就是逻辑坐标系中的坐标点，此点把当前坐标点作为逻辑坐标系的原点。例如：

设当前坐标点为（2,3），执行下面的语句：

```
PSET STEP（10,20）
```

执行后就是原坐标系（物理坐标系）的点（12,23）。

（3）〈颜色〉是指要画点的颜色属性，它是由 COLOR 语句定义的当前配色器中的颜色。

（4）格式 1 中，当省略〈颜色〉时，使用前景颜色绘点，PSET 语句执行一次只能在屏幕上点亮一点，而屏幕上的任何图形都是由一个个的离散点组成的。所以，使用 PSET 语句同样可以绘制出任何图形和图像。

（5）格式 2 中，当省略〈颜色〉时，使用背景颜色绘点。这一点是 PRESET 语句与 PSET 语句的唯一区别，这两个格式的其他参数含义及功能相同。

（6）这两个语句只能用于图形方式。

[例 11-4] 在屏幕上画出“ 满天星 ”。

```
'EXAMPLE 4
CLS
SCREEN 1
DO
x=INT（RND*4）
i=INT（RND*320）
j=INT（RND*180）
PSET（i,j）,x
FOR n=1 TO 100:NEXT n
LOOP WHILE INKEY$=""
END
```

[例 11-5] 在屏幕上从坐标点（50，100）到点（200，100）画一条横线，然后按相反方向将该横线擦掉。

```
'EXAMPLE 5
SCREEN 1
CLS
FOR i=50 TO 200
PSET（i,100）
NEXT I
FOR i=200 TO 50 STEPa-1
PRESET（i,100）
NEXT i
END
```

11.3.2 画直线和矩形框语句 LINE

格式：

LINE［［STEP］（x1,y1）］-［STEP］（x2,y2）［，［〈颜色〉］［，B［F］］［，［〈线型〉］］］

功能：在屏幕上指定的位置画一直线段或矩形。

说明：

（1）（x1,y1）为算术表达式，为所画线段的起始坐标。省略时为上一次绘图操作的坐标（当前坐标）。

（2）（x2,y2）为算术表达式，为所画线段的终点坐标。该项不可省略。坐标的数值如果超过当前视区，语句不出错，只是看不到超出区域的图形。

（3）STEP 为可选项。若选择 STEP，则是指后面所跟的坐标是相对坐标。例如：

假设最后的参考点为（5,5）,则：

```
LINE-STEP（25,30）
```

画出一条从（5,5）到（30,35）的直线。最后参考点可以通过 CLS 和 SCREEN 语句初始化屏幕来建立，也可以用 PEST、PRESET 及后面介绍的 CIRCLE 和 DRAW 语句来建立。

STEP 在使用中有一些变化，设最后的参考点为（5,5），则：

```
LINE-（40，40）                    '从（5,5）到（40,40）画线
LINE－STEP（40,40）                '从（5,5）到（45,45）画线
LINE（8,8）-STEP（40,40）              '从（8,8）到（48,48）画线
LINE STEP（10,10）－（70,70）      '从（15,15）到（70,70）画线
LINE STEP（10,10）-STEP（70,70）   '从（15,15）到（85,85）画线
```

（4）〈颜色〉为颜色属性可选项，指画线或画矩形所使用的颜色。省略时为用 COLOR 语句指定的前景色，如果颜色属性的值超过当前屏幕颜色范围，语句不出错，按最大颜色属性值。

（5）若选择 B 表示以（x1,y1）和（x2,y2）为对角线顶点画一矩形，即矩形左上角和右下角坐标分别为（x1,y1）和（x2,y2）。如果只有参量 B，则只画矩形边框。

（6）若选择 BF，则不仅画出矩形边框，而且要用指定的颜色填充该矩形。填充颜色与边框颜色相同。

（7）〈线型〉为一整型变量，指用来填充屏幕像素的 16 位二进制数格式。画线时，LINE 从〈线型〉中从左向右读取各位。如果所读的位是零，则不画任何点。如果所读的位是 1，则画一个点。所以，选择〈线型〉参量可以画特殊类型的线，如虚线、点线和点划线等。该参量的取值范围是 0~＆HFFFF，由 16 位二进制数组成。例如，当该参量为＆HFFFF（16 位全为 1）时，表示 16 个点都显示；当它为＆HAAAA（1010101010101010）时，每隔一点显示一点，画出虚线。当该参量省略时画实线。如选 BF 项，则此项不起作用，即〈线型〉对已填色的框无效。

[例 11-6] 在屏幕上用各种颜色打印随机线。

```
'EXAMPLE 6
CLS
SCREEN 12
DO
sh%=INT（RND（1）*16）+1
x1%=INT（RND（1）*320）
y1%=INT（RND（1）*200）
x2%=INT（RND（1）*320）
2%=INT（RND（1）*200）
LINE（x1%,y1%）-（x2%,y2%）,sh%
LOOP WHILE INKEY$=""
END
```

[例 11-7] 使用 LINE 语句画出“凸”字。

```
'EXAMPLE 7
SCREEN 1
```

```
CLS
READ a,b
FOR i=1 TO 8
READ a,b
LINE-（a,b）
NEXT
DATA 130,50,170,50,170,100,200,100,200,150,100,150,100
DATA 100,130,100,130,50
END
```

[例 11-8] **画出两个方框与方块。**

```
'EXAMPLE 8
SCREEN 12
CLS
LINE（110,60）-（210,120）,1,BF
LINE（130,80）-（190,100）,0,B
LINE（260,60）-（360,120）,1,BF
LINE（10,10）-（500,200）,1,B
END
```

[例 11-9] **以（150,100）与（200,150）为对角线的两个端点，画出由内向外嵌套的 5 个方框。**

```
'EXAMPLE 9
SCREEN 12
CLS
FOR i=0 TO 80 STEP 20
LINE（150-i,100-i）-（200+i,150+i）,,B
NEXT i
END
```

[例 11-10] **画一条虚线。**

```
'EXAMPLE 10
SCREEN 12
CLS
LINE（50,50）-（360,200）,,,&HFOFO
END
```

11.3.3 画圆、椭圆和画弧语句 CIRCLE

格式：

CIRCLE［STEP］（x,y）,〈半径〉［，［〈颜色〉］］［，［〈起角〉］］［，［〈终角〉］］［，［〈比率〉］］

功能：按照指定的圆心和半径画一个圆或一个椭圆或一段弧线。

说明：

（1）（x,y）表示圆、椭圆或弧的中心坐标。

（2）STEP 表示（x,y）为相对坐标。或省略，则为绝对坐标。

（3）〈半径〉为算术表达式，指出所画圆、椭圆（主轴方向）或弧的半径。

（4）〈颜色〉为算术表达式，表示所画圆、椭圆或弧的起始角度。省略时默认为 0。

（5）〈起角〉为弧度值，表示所画圆、椭圆或弧的起始角度。省略时默认为 0。

（6）〈终角〉为弧度值，表示所画圆、椭圆或弧的终止角度。省略时默认为 2π。

CIRCLE **语句参数中的起角和终角的取值范围为**-2π~2π**。弧度值的增大方向为水平线逆时针方向。屏幕上画弧时，根据起角和终角的不同，可以屏幕上画出任意开口方向的弧。当起角或终角的弧度值都是正值时，则**

只画出圆弧；若两者都是负值或两者之一为负值时，则不仅画出弧，还会从圆心到负值的点画一条直线。

（7）〈比率〉表示纵横比，即 Y 轴半径与 X 轴半径之比。省略时，在当前方式下面一个圆。否则将按纵横比画一个椭圆。

比率值的计算公式为：4*（Y 像素/X 像素）*3

这里 Y 像素和 X 像素是屏幕的分辨率。如果纵横比小于 1，则以 X 轴半径作为主轴半径；如果纵横比大于 1，则以 Y 轴半径作为主轴半径。例如：

```
CIRCLE（50,100）,,,,3/10
```

所画的椭圆长轴为 X 轴。

```
CIRCLE（150,100）,,,,3
```

所画的椭圆长轴为 Y 轴。

（8）如果要画一条角度为 0 的半径（即向右的水平线段），不能使用-0 表示，而必须用一个很小的非 0 值来代替：

```
CIRCLE（150,100）,60,,-0.00001,-1.55
```

（9）CIRCLE 语句中有些参量可以省略，但必须保留逗号。如果省略的是最后的一个或几个参量，则逗号可以不写。

[例 11-11] 在屏幕上用红色画一个圆，用黄色画一个椭圆，用绿色画一段弧构成一个扇形，用蓝色画一个椭圆。

```
'EXAMPLE 11
SCREEN 12
CLS
CIRCLE（70,70）,50,12                    '画红色圆
CIRCLE（100,190）,80,14,,,1/2            '画黄色椭圆，垂直方向半径小
CIRCLE（280,70）,60,10,0,3.14/2          '画绿色 0~90 度的弧
LINE（280,70）-（280,10）,10               '构成扇形
LINE（280,70）-（340,70）,10
CIRCLE（300,200）,80,9,,,2/1             '画蓝色椭圆，水平方向半径小
END
```

[例 11-12] 阅读并运行下面的程序。

```
'EXAMPLE 12
CONST pi=3.1416
SCREEN 12
CIRCLE（220，250），200，，-pi,-pi/2
CIRCLE STEP（-100,-100）,100
CIRCLE STEP（0,0）100,,,,5/25
LOCATE 25,1
PRINT"按任意键返回"
DO：LOOP WHILE INKEY$=""
END
```

上例中程序先画一个左上部缺四分之一的圆，然后用相对坐标在缺口内画第二个圆，最后用小于 1 的纵横比在第二个圆中画一个小椭圆。该程序最后用了一个空 DO 循环，来保持图形的完整性，只要不按键，图形就一直保留在屏幕上，这种技巧可以用在任何图形程序中。

[例 11-13] 画出各种不同的圆弧。

```
'EXAMPLE 13
SCREEN 12
CLS
pi=3.14
```

```
CIRCLE（100,100）,30,,0,pi,20/27
CIRCLE（200,100）,30,,0,pi*2,20/27
CIRCLE（300,100）,30,,-pi*2,-pi/2,20/27
CIRCLE（400,100）,30,,-pi,-pi*3/2,20/27
CIRCLE（100,200）,30,,-pi*2,pi,20/27
CIRCLE（200,200）,30,,0,pi/2,20/27
CIRCLE（300,200）,30,,-pi/4,-pi*7/4,20/27
CIRCLE（400,200）,30,,-pi5/4,-pi*3/4,20/27
END
```

11.3.4 连续画线语句 DRAW

1．DRAW 语句的格式和功能

格式：

DRAW〈字符串〉

功能：根据画图命令连续画出各种直线。

说明：〈字符串〉是各种画图命令，具体用法见下面的说明。

2．DRAW 语句中的画图命令

每个画图命令有两部分组成，一个命令字母以及一个数值 n。各命令之间要用分号或空格隔开。主要的画图命令有：

（1）位置移动命令

连续画线语句 DRAW 中的位置移动命令见表 11-5。

表 11-5 DRAW 语句中的位置移动命令

位置移动命令	含 义
B	只移动，但不画线
N	移动结束后返回到原来的位置
U[n]	向上移动 n 个单位
D[n]	向下移动 n 个单位
L[n]	向左移动 n 个单位
R[n]	向右移动 n 个单位
E[n]	沿对角线向右上方移动 n 个单位
F[n]	沿着对角线向右下方移动 n 个单位
G[n]	沿对角线向左下方移动 n 个单位
H[n]	沿对角线向左上方移动 n 个单位
Mx,y	如果 x 或 y 前无符号，表示绝对移动，从当前位置向点（x,y）画直线；如果 x 或 y 前有正号或负号，则表示以当前位置为起点，按相对坐标画直线。

例如：

```
DRAW"R100 U100 L100 D100"          '画一个方框
DRAW"R25;BR25;R25;BR10;R25"        '在同一水平位置上画三条直线
```

n 省略时，默认为 1 个单位。

（2）角度命令

连续画线语句 DRAW 中的角度命令见表 11-6。

表 11-6 DRAW 语句中的角度命令

角 度 命 令	含 义
An	设置旋转角度 n。N 的取值为 0、1、2 和 3，0 表示不转，1 表示逆时针转 90 度，2 为逆时针转 180 度，3 为逆时针转 270 度。在标准比例因子 4/3 的显示屏幕上，转 90 度或 270 度的图形与转 0 度或 180 度显示的图形尺寸相同
TAn	旋转 n 角度，n 在-360~360 度之间。如果 n 是正数，则逆时针旋转；如果是负数，则顺时针旋转

（3）颜色命令

连续画线语句 DRAW **中的角度命令见表** 11-7。

表 11-7　DRAW **语句中的颜色命令**

颜 色 命 令	**含　　义**
Cn	**把颜色号设置为** n
Pj,k	j **为填充色，即图形内部的颜色；**k **为边界色，是图形边界的颜色。注意：**Cn **中设置的颜色号** n **必须与边界色** k **相同，否则** k **色将充满整个屏幕**

（4）其他命令

连续画线语句 DRAW **中的其他命令见表** 11-8。

表 11-8　DRAW **语句中的其他命令**

命　令	**命 令 格 式**	**含　　义**
比例因子命令	Sn	**设置比例因子** n。n **的取值范围为** 1~255。**用比例因子乘以由** U、D、L、R、E、F、G、H **和** M **命令给出的距离，就是实际移动的距离**
字符串命令	"X"+VARPTR$（**字符串变量**）	**用** X **之后的字符串变量的内容取代该组命令来绘图**
执行逻辑坐标命令	V	**用逻辑坐标而不是物理坐标执行后续的所有** ERAW **绘图命令**

[**例** 11-14]　**请用** DRAW **语句画一座房子。**

```
'EXAMPLE 14
SCREEN 1
CLS
DRAW"BM100,100;R125;H43;L40;G43"
DRAW"BM150,56;U20;R5;D20"
DRAW"BM120,100;D50;R80;U50"
DRAW"BM150,150;U20;R20;D20"
DRAW"BM125,105;R10;D10;L10;U10"
DRAW"BM130,110;NU5;ND5;NL5;R5"
DO:LOOP WHILE INKEY$=""
END
```

[**例** 11-15]　**请用** DRAW **语句绘出火箭图形。**

```
'EXAMPLE 15
SCREEN 1
CLS
DRAW"BM100,150M-30,+40 U40 M+30,-40 ND40 U80 M+20,-30"
DRAW"M+20,+30 D120 NL40 M+30,+40 U40 M-30,-40"
END
```

[**例** 11-16　]**绘出分别朝** 4 **个方向的四面旗子。**

```
'EXAMPLE 16
SCREEN 10
CLS
FOR n=0 TO 3
DRAW"BM160,150"
DRAW"A="+VARPTR$（n）
DRAW"BU50 L20 U20 R20 D50 BU50 G20"          '画旗子
NEXT n
END
```

[**例** 11-17]　**绘制放射线。**

```
'EXAMPLE 17
SCREEN 1
CLS
FOR x=0 TO 360 STEP 20
DRAW"TA="+VARPTR$（x）                    '把直线按逆时针方向旋转
DRAW"NR60"                                '从屏幕中心点向右画一条 60 点的直线
NEXT x
END
```

[例 11-18] 画出嵌套的 5 个从小到大的菱形图案。

```
'EXAMPLE 18
SCREEN 1
CLS
FOR i=1 TO 20 STEP 4
DRAW "BM200,100"
DRAW"S="+VARPTR$（i）                    '图形放大比例
DRAW"BM+20,+0"
DRAW"H20 G20 F20 E20"                    '画菱形
NEXT i
END
```

11.4 填涂颜色语句 PAINT

在画图过程中，有的时候需要将某一区域用某种颜色填充起来。对于矩形块，我们可以用 LINE 语句加上选择项［BF］来解决，而对于其他形状的封闭图形，就要用下面介绍的 PAINT 语句来完成。

格式：

PAINT［STET］（x,y）［，［〈涂色〉］］［，［〈边框色〉］］

功能：用指定的颜色对屏幕的指定区域填充。

说明：

（1）若选用 STEP 时，坐标（x,y）为相对坐标。

（2）（x,y）是边界线内的任意一点的坐标，它可以是绝对坐标也可以是相对坐标。例如：

```
SCREEN 1
CLS
CIRCLE（190,150）,50,1                   '用青色画一个圆
PAINT（190,150）,2,1                     '在圆内涂洋红色
```

该点坐标不能在图形线上。如果该点在图形内部，则在图形内部涂色；如果该点在图形外部，则涂的是图形的背景色；如果该点在图形的边界线上，则不涂色。

（3）〈涂色〉是一个算术表达式，其值必须是一个有效的颜色属性值，用相应的颜色涂指定的区域；若省略则使用前景颜色。

（4）〈边框色〉是一个算术表达式，用于标识图形边框颜色的属性。若省略，则使用〈涂色〉。

（5）在给图形涂色时，应注意：当需要对图形内部涂色时，图形必须是全封闭的。否则被涂的颜色将“泄漏”到屏幕的其他地方。例如：

试着画一个方框，然后在它的里面画一个有缺口的椭圆，在椭圆内着色，看看会出现什么现象。

程序为:

```
SCREEN 12
CONST pi=3.14
```

```
LINE（200,10）-（400,200）,,B
CIRCLE（300,100）,80,,0,1.9*pi,3
PAINT（300,100）
```

可以看到，虽然从椭圆的内部开始涂色，但所涂的颜色从缺口泄出并涂满方框。如果没有方框，则会填满整个屏幕或窗口。

（6）如果用与被涂对象的边线不同的颜色涂色，就应该选择〈边框色〉。〈边框色〉必须与图形的边线色一样。否则涂色将超出指定的范围。例如：

```
SCREEN 1
COLOR，0
LINE（15,30）-（315,30）,1
LINE-（165,180）,1
LINE-（15,30）,1
PAINT（160,100）,2
```

上面的程序画了一个边框是绿色（调色板 0，属性 1）的三角形，然后企图用红色填充三角形的内部，但是由于没有使用〈边框色〉参量，结果将整个屏幕涂成了红色。为了产生我们所需要的结果，可以对 PAINT 语句作如下修改;

```
PAINT（160,100）,2,1
```

[例 11-19] 在屏幕上画出彩色气泡。

```
'EXAMPLE 19
DO WHILE INKEY$<>CHR$（27）
SCREEN 1,0
CLS
DEFINT a-z
a=RND*310:b=RND*200:c=RND*30
CIRCLE（a,b）,c,2
d=RND*4
PAINT（a,b）,d,2
FOR n=1 TO 100:NEXT
LOOP
END
```

[例 11-20] 画出一个打靶板。要求其内圆是黑色的，3 个圆环分别是蓝色、绿色和青色。

```
'EXAMPLE 20
SCREEN 7
CLS
FOR i=1 TO 4
CIRCLE（150,100）,10*i,3
NEXT i
FOR i=1 TO 3
PAINT（165+10*（i-1）,100）,i,3
NEXT i
END
```

11.5 图形的窗口操作

前面在介绍绘图语句时，使用的是系统设置好的坐标系，我们称之为物理坐标。其实坐标系是无限的，我们在使用时只能显示坐标系中的一个矩形部分，该矩形称为图形窗口。除了使用系统设置的物理坐标外，Quick

BASIC 还允许用户设计自己的坐标系，在屏幕上开设自己所需要的图形窗口，并使用自己定义的坐标进行绘图，而不用管原来的坐标。当用户给出图形上某点的坐标时，Quick BASIC 会自动转换为物理坐标上的点。

11.5.1 窗口语句 WINDOW

格式 1：

WINDOW（x1,y1）-（x2,y2）

格式 2：

WINDOW SCREEN（x1,y1）-（x2,y2）

功能：定义用户自己的坐标系。

说明：

（1）格式 1 定义一个标准的坐标系，即 y 轴向上。如图 11-3 所示。

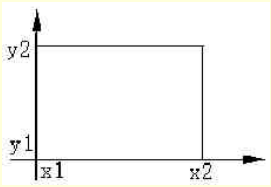


图 11-3 标准坐标系

x 的坐标范围从 x1 到 x2，y 的坐标范围从 y1 到 y2。

（2）格式 2 定义一个非标准的坐标系，即 y 轴向下。如图 11-4 所示。

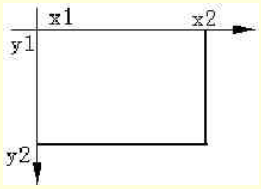


图 11-4 非标准坐标系

x 的坐标范围从 x1 到 x2,y 的坐标范围从 y1 到 y2。

（3）在程序中使用了 WINDOW 语句后，即定义了一个新的坐标系，我们称之为逻辑坐标。程序中的绘图语句如 PSET、PRESET、LINE、CIRCLE、PAINT 等都将使用该逻辑坐标，直到出现下一个 WINDOW 语句为止。

（4）如果 WINDOW 后无参数，返回到屏幕坐标。通过改变坐标系，可以改变窗口的大小，这样就可以实现对图形的放大、缩小。

[例 11-21] 设置一个标准的坐标系，并画出一个长方形和一个圆形。

```
CLS
SCREEN 13
WINDOW（-2,-2）-（6,8）
LINE（0,0）-（1,1）,6,BF
CIRCLE（3,3）,1,12
END
```

[例 11-22] 修改例 11-21 设置一个非标准的坐标系。

```
CLS
SCREEN 13
WINDOW SCREEN（-2,-2）-（6,8）
LINE（0,0）-（1,1）,6,BF
CIRCLE（3,3）,1,12
END
```

11.5.2 视窗语句 VIEW

Quick BASIC 提供了一种功能，可以在屏幕上定义一个矩形区域，使图形只能在这个区域内显示，该矩形区域称为视窗。

格式：

VIEW［SCREEN］（x1,y1）-（x2,y2）,〈视窗底色〉，〈边界色〉

功能：定义一个视窗。

说明：

（1）（x1,y1）和（x2,y2）是定义视窗的左上角和右下角坐标。

（2）〈视窗底色〉是所定义的视窗的背景色。〈边界色〉是所定义的视窗边界的颜色。

（3）如果选用了 SCREEN，则表示程序中图形语句的坐标仍然使用系统提供的物理坐标。否则，使用 WINDOW 定义的坐标。

[例 11-23] 运行程序。

```
SCREEN 1,0
COLOR 1,1
CIRCLE（90,90）,15
WINDOW SCREEN（0,0）-（300,180）
VIEW（200,1）-（290,100）,,2
CIRCLE（80,100）,20
END
```

11.6 本章小结

本章，我们已经对 BASIC 图形处理中最基本的语句作了简要介绍。事实上，Quick BASIC 语言具有较全面的图形处理功能和丰富的作图语句。除了这些最基本的功能之外，其他功能，如图形填充技术，交互式图形显示技术、三维图形显示技术等，由于篇幅所限，不能予以全面介绍。因此，本章中所讲到的每一条画图语句，都应作为重点掌握。表 10-4 列出了本章介绍的基本画图语句。

BASIC 中最基本的画图语句如表 10-4 所示:

表 10-4 BASIC 中最基本的画图语句

语 句	形 式	作 用
SCREEN	SCREEN＜模式代码＞，＜彩色开关＞	指明屏幕方式，确定屏幕特征
WIDTH	WIDTH＜宽度值＞	指定每行字符数
LOCATE	LOCATE＜行坐标＞，＜列坐标＞， ＜光标＞，＜开始＞，＜结束＞	实现屏幕上光标的定位
CLS	CLS	屏幕清除
PSET PRESET	PSET［STEP］（X，Y）［，＜彩色码＞］ PRESET［STEP］（X，Y）［,＜彩色码＞］	画点
LINE	LINE［［STEP］（X1，Y1）］-［STEP］（X2，Y2）［，＜彩色码＞［，B［F］］］	画线、框，框着色
CIRCLE	CIRLE［STEP］（XC，YC），＜半径＞，＜彩色码＞，＜起始角＞，＜结束角＞，＜纵横比＞	画圆、椭圆、弧
DRAW	DRAW＜字符串＞	画简单图形
COLOR	COLOR＜背景色号＞,＜配色器表达式＞	设置屏幕的颜色
PAINT	PAINT［STEP］（X，Y）＜区内着色＞，＜边界着色＞	图形的着色
WINDOW	WINDOW［［SCREEN］（X1,Y1）-（X2,Y2）］	
VIEW	VIEW［［SCREEN］（X1,Y1）-（X2，Y2），＜窗口底色＞, ＜边界颜色＞］	

习 题

1．用绘图语句给出下列几何图形：

（1）三角形　　（2）圆形　　（3）扇形　　（4）圆内接多边形

2．在屏幕上画一幅“众星捧月”的夜景。

3．画 5 个同心圆，并在里面涂上不同的颜色。

4．用 PSET 和 PRESET 语句画出动态的立体圆。

5．分别用图表表示某班学生成绩在 60 分以下的人数，60~70 分的人数，70~80 分的人数，80 分以上的人数。

6．先在屏幕上画出一条帆船，如果要逆时针旋转 45°，如何修改；如果要画出 3 条大小不同的帆船，又如何修改。

7．绘制时钟图。

第十二章 文 件

文件是建立在外部介质上的记录的集合，Quick BASIC 具有较强的文件处理功能，他除了可以处理顺序文件、随机文件之外，还可以处理二进制文件。本章主要介绍：

- 文件的概念和分类
- 用户定义类型
- 顺序文件
- 随机文件
- 文件与目录维护语句

本章的难点是在对顺序文件、随机文件的理解和操作。

12.1 文件的概念

计算机的用途之一就是处理庞大且繁杂的数据，对于大量的数据，用前面章节介绍的任何方法把它们调入内存都是无法实现的。因此只能把它们存储到计算机的磁盘上从而形成文件，需要处理时再把它们装入计算机。

文件（File）是同一类数据存储在外部介质上数据的集合。操作系统是以文件 为单位对数据进行管理的，也就是说，如果要找磁盘上的数据，必须先按文件名找到所指定的文件然后再从该文件中读取数据。要向磁盘上存储数据时，也必须以文件名为单位先建立一个文件，才能向它输出数据。

12.1.1 文件的分类

1. 按存储介质分类

按输入输出介质，文件可以分为设备文件和磁盘文件两大类。

由于计算机可以通过标准输入设备（键盘、鼠标）输入数据，也可 通过标准输出设备（显示器、打印机）输出数据，这些标准输入输出设备具有文件的性质，所以把它们称为设备文件。

在程序运行时，需要把一些程序运行的中间数据或最终结果输出到磁盘上存放起来，以后需要再从磁盘上输入到计算机内存。在外存储器上形成的文件称为磁盘文件。

2. 按文件的内容分类

按照存放的内容不同，文件可分为程序文件（Program File）和数据文件（Data File）两大类。

一个文件如果存放的是程序，则称程序文件，像我们把 Quick BASIC 程序保存到磁盘上，就是一个程序文件。

一个文件内如果存放的是由程序所产生的数据，就称数据文件。数据文件按其存取方式的不同还可以分为顺序文件和随机文件两种。

3. 按存储信息的形式分类

按存储信息的形式可以分为 ASCII 码文件和二进制文件。ASCII 码文件也叫文本文件，除数值型数据外都是以对应的 ASCII 码形式存放的，一个字符占一个字节（一个汉字占两个字节）。因此，这种文件既便于对字符进行逐个处理，也便于打印输出。其缺点是：处理起来比较麻烦，占据的空间也较多。Quick BASIC 的源程序都是以 ASCII 码形式存放的，字符串数据也是用 ASCII 码形式存放的。只有数值型数据既可以用 ASCII 码形式存放，也可以用二进制形式存放，ASCII 码文件可以用 DOS 命令中的 TYPE 命令显示或打印，二进制文件不能。

二进制形式文件是以机内存储数据的形式存储的。其优点是节省存储空间和节省转换时间，相对来说效率较高，故一般中间结果数据暂时保存在外存中，以后又需要输入到内存的，常用二进制文件保存。

12.1.2 程序文件与数据文件的区别

程序文件与数据文件除了存储介质、基础操作可相同外，在许多方式都是不同的：

（1）对象不同。程序文件所描述对象是算法；数据文件所描述的对象是程序要处理的数据。

（2）存取方式不同。程序文件是以整个程序为单位进行存取操作（既从内存保存到外存，或从外存调入内存）；数据文件是以记录为单位进行存取操作。

（3）主要操作方式不同。程序文件 主要操作有运行、连接、覆盖等数据文件的主要操作则是查找、修改、排序等。

（4）根本用途不同。程序文件的根本用途是保存程序，以备再用；数据文件的根本用途保留数据，以备共用。

（5）相互地位不同。在计算机应用系统中，程序文件处于主导地位；而数据文件则处于辅助地位。

在 Quick BASIC 中，如果要存取程序文件，只要利用 Quick BASIC 的操作命令就可以了，程序文件操作比较简单。但是，数据文件的建立一存取都需要通过程序来操作，操作较为复杂。因此，本章主要介绍如何利用 Quick BASIC 程序来建立与存取磁盘数据文件。

12.1.3 文件属性

文件属性包括：文件名、文件号和文件指针。

1. 文件名

给文件起一个名字就是文件名。不过，这里指的文件名一般不包括盘符和路径。

格式：〈文件名〉 [.〈扩展名〉]

〈文件名〉及〈扩展名〉的命名规则与 DOS 中文件的命令规则相同。在 Quick BASIC 中，程序文件的扩展名默认为.BAS，数据文件的扩展名默认为.DAT。

2. 文件号

对磁盘数据文件进行读写操作时，必须首先开辟一个内存缓冲区，以建立必要的通道，使一个独立的文件号与文件名相对应而联系起来。实际上文件号也就规字了对该文件进行操作的通道，所以文件号也就是通道号，文件号的取值范围为 1~255 的整数。

3. 文件指针

在对文件进行操作时，必须首先确定将要进行操作的位置，并且随操作的需要移动到相应的位置。文件系统采用“文件指针”来指示被操作的位置，因此，文件指针是文件操作的依据。

当文件提针指向某一记录时，该记录称为当前记录，文件指针总是指向将被读写的数据的后面。

4. 文件缓冲区及读写操作

从内存的角度来说，前面所说到的通道就是一个文件缓冲区。为了减少直接读写磁盘的次数，节省操作时间，需要建立缓冲区。

对文件进行读/写操作的具体含义如下：

“写文件”表示从主机向磁盘传送数据。进行写操作时，先将数据送入缓冲区中暂存，等到把缓冲区放满之后一起传送到磁盘，然后再开始下一批数据的存放、传送；

“读文件”将磁盘文件中的数据调入内存。进行“读文件”操作时，先从磁盘读入数据，一次读入若干数据将缓冲区放满，然后从缓冲区分几次送到内存数据区中。

文件操作中，每次打开一个文件就同时建立了一个对应的缓冲区，并确定 缓冲区号（通道号）。因而实现了文件叼、通道号、缓冲区号三者的统一。

12.2 顺序文件

12.2.1 数据文件的基本概念

就像数组是下标变量的有序集合那样，数据文件是外存记录的有序集合，即存储在外存介质上描述同类对

象的记录所构成的有限序列。也就是说数据文件是记录结构的继续和发展。

数据文件是一组性质相同且互有关联的数据的集合体，如学生各科成绩数据文件（存放学生成绩数据）、人事数据文件（存放职工的人事数据）、库存数据文件（存放货物的库存数据）等等。

文件通常有固定的结构，它由许多记录组成，每条记录由固定个数的数据项或字段组成。假设某校的学生成绩如表 12-1 所示。

表 12-1　学生成绩表

学生	姓名	性别	出生日期	语言	数学	总分
2000310	王小红	男	65.02.22	85	95	180
2000103	周玉	女	60.05.18	73.5	87	160.5
2000011	孙友朋	男	65.09.21	82	79	161
2000016	张小伦	女	70.11.09	60	86	146
2000021	张敬画	女	73.10.27	79	85.5	164.5
2000011	赵艳芳	女	63.08.08	92	100	192
2000002	林奔驰	男	60.12.26	65	80	145
2000003	温琳琳	女	66.01.16	70	81	151

“学生成绩表”中的表名对应磁盘中的数据文件；表中第 1 行中的 7 个栏目的集合就是记录类型，在它下面填入每个学生的 7 个具体属性值的集合就是记录值（简称记录），该表中共计有 8 个记录值，每位学生有一个记录。

表中的每个栏目称为字段（或称为数据项），字段的名称，如学号、姓名、性别、出生日期等称为字段名（或称数据项名）。如果该校有 2000 名学生，则该校学生成绩数据文件中就应该有 2000 个记录。

数据文件中有一个记录指针，指针指向的位置就是当前读、写的位置。

对数据文件的操作包括建立、查看、复制、排序、查找、修改、删除等。

12.2.2　记录

在 Quick BASIC 语言中，提供了整型、长整型、单精度型、双精度型、可变长度字符串、固定长度字符串等系统标准数据类型，我们可以用系统提供的这些标准类型定义变量，它们都是计算机处理的基本项组成的组合项，如表 12-1 中，学生对象由学号、姓名、性别、出生日期、语言、数学、总分等基本项组成组合项。用这些组合项来描述相应对象的若干属性，这些描述相同对象的组合项的集体形成了记录。它通常由描述某实体对象各属性数据的若干字段（即数据项）构成。

在 Quick BASIC 中提供了定义记录结构的语句，Quick BASIC 称这种语句为用户定义数据类型（User defined data type）。

记录一数组一样是由多个数据项目组成，但是，记录与数组最大的不同就是数组中的每一元素必须具有相同的数据类型，而记录中的每一个数据项目却可以具有不同的数据类型。

记录结构对处理随相文件非常方便。

12.2.3　用户类型定义语句 TYPE

格式：

```
TYPE〈用户类型名〉
〈字段名〉AS〈类型名〉
[〈字段名〉AS〈类型名〉]
…
END TYPE
```

功能：定义一个包含多个元素的数据类型。

说明：

（1）〈用户类型名〉是用户定义的数据类型名，而不是变量。其命名规则与变量名的命名规则相同。

（2）〈字段名〉是用户定义数据类型中的一个数据项名。不能用数组名作为字段名。

（3）〈类型名〉是所定义元素的类型（INTEGER、LONG、SINGLE、DOUBLE、STRING 定长或用户定义数据类型）。

[例 12-1] 建立一个学生成绩处理程序，由于每一位学生都需要学号、姓名、学分和平均成绩等数据项（字段），用 TYPE…END TYPE 语句来定义一个名称为 studentrec 的记录。

```
TYPE studentec
stunum AS STRING*6          '学号元素为 6 个字符的定长字符串
names AS STRING*8           '姓名元素为 8 个字符的定长字符串
credit AS INTEGER           '学分元素为整型
avg AS SINGGLE              '平均成绩为单精度型
END TYPE
```

12.2.4 定义和使用记录变量

1. 定义记录变量

用户数据类型定义后，可以用 DIM、REDIM、COMMON、STATIC 或 SHARED 建立一个具有这种数据类型的变量。

例如：定义一个具有 studentrec 类型的变量 stu。

```
DIM stu AS studentrec
```

记录也可以和为数组元素的数据类型。

例如：定义一个拥有 100 个记录元素的数组 student。

```
DIM student（1 TO 100）AS studentrec
```

2. 存取记录变量的方法

如果要存取记录变量中的某个字段的数据，其格式如下：

〈记录变量名〉.〈字段名〉

例如，要存取记录变量 stu 中 names 这个字段的数据，要写为：

```
stu.names
```

[例 12-2] 把数据值分别赋给 stu 变量中的各个字段。

```
REM 定义学生记录
TYPE studentrec
stunum AS STRING*6          '学号元素为 6 个字符的定长字符串
name ASSTRING*8             '姓名元素为 8 个字符的定长字符串
credit AS INTEGER           '学分元素为整型
avg AS SINGLE               '平均成绩为单精度型
END TYPE
REM 定义记录量
DIM stu AS studentrec       '定义一个具有 studentrec 类型的变量
stu '赋值给记录变量
stu.stunum="990101"
stu.names="Li Hong"
stu.credit=80
stu.avg=98
CLS
PRINT stu.stunum
PRINT stu.names
PRINT stu.credit
PRINT stu.avg
END
```

12.2.5 记录数组

1. 定义记录数组

如果一个数组中元素的数据类型是记录类型，则称为记录数组（Array of records）。

[例 12-3] 假设某班有 50 位学生，每位学生一个记录，定义一个包含 50 个元素的记录数组。

```
TYPE studentrec
stunum AS STRING*6
namts AS STRING*8
credit AS INTEGER
avg AS SINGLE
END TYPE
DIM student（1 TO 50）AS studentrec          '定义记录数组
```

DIM 语句定义 student 是一个有 50 个元素的记录数组，它的每一个元素的数据类型都是记录 studentrec。

2. 存取记录元素

存取记录元素的写法为：

〈记录数组元素〉.〈字段名〉

例如，要存取第 1、第 28 位学生的平均分数，其写法是：

```
student（1）.avg
student（28）.avg
```

[例 12-4] 给例 12-3 中定义的记录数组 student 中的第 32 位学生赋值。

```
TYPE studentrec                    '定义记录类型
stunum AS STRING*6
name AS STRING*8
credit AS INTEGER
avg AS SINGLE
END TYPE
DIM student（1 TO 50）AS studentrec     '定义记录数组
student（32）.stunum="2000101"          '给记录数组赋值
student（32）.names="Liu ping"
student（32）.credit=65
student（32）.avg=88
CLS
PRINT student（32）.stunum          '输出结果
PRINT student（32）.names
PRINT student（32）.credit
PRINT student（32）.avg
END
```

12.2.6 嵌套记录

如果一个记录是某一个记录的一个字段，则称为嵌套记录（Nested record）。

[例 12-5] 定义表 12-2 所示的学生成绩记录。

表 12-2 学生成绩表

学号	姓名	成绩		
		语文	数学	平均分

由于表 12-2 为三维表格，要实现这种种形式记录的定义，可将该表转换为表 12-3、表 12-4 两个表格。

表 12-3　成绩表

成绩		
语文	数学	平均分

表 12-4　学生成绩表

学号	姓名	成绩

先定义表 12-3，在定义表 12-4 时将表 12-3 的定义嵌套进去。

```
CLS
TYPE cj_rec                    '定义成绩记录（表 12-3）
yw AS SINGLE
sx AS SINGLE
pjf AS SINGLE
END TYPE
TYPE xscj_rec                  '定义学生成绩记录（表 12-4）
xh AS STRING*6
xm AS STRING*10
cj AS cj_rec                   '此字段为记录类型，嵌套记录
END TYPE
'定义记录变量
DIM cjbl AS cj_rec
DIM xsbl AS xscj_rec
'给 cjbl 记录变量的各个字段赋值
cjbl.yw=86
cjbl.sx=93
cjbl.pjf=（cjbl.yw+cjbl.sx）/2
'给 xsbl 记录变量的字段赋值
xsbl.xm="Wang mazi"
'给 cj 嵌套记录变量赋值
xsbl.cj=cjbl
'打印嵌套记录（按表 12-2 的记录形式）
PRINT xsbl.xh;xsbl.xm;xsbl.cj.yw;xsbl.cj;sx;xsbl.cj.pjf
END
```

程序的运行结果为:

```
990601 Wang mazi      86      93      89.5
```

例 12-5 只是为演示嵌套的一个例子，在实际设计程序时，一般直接把表 12-2 转换为二维表格（如表 12-5 所示），而不用嵌套的形式。

表 12-5　学生成绩表

学号	姓名	语文	数学	平均分

12.3 顺序文件

12.3.1 顺序文件的存放格式和特点

1．顺序文件的操作特点

顺序存取文件简称顺序文件，是把每一个记录按照存入时的先后顺序存储在文件内，以后要从文件中读取数据时，也要从第一个记录开始逐个依次读取。例如，如果要读取文件中的第 100 个记录，就得先读过它前面的 99 个记录，才能读到第 100 个记录。这是因为顺序文件的读写操作只能对当前记录，即文件指针所指示的记录，而每当读写完一个记录，则文件指针自动指向一个记录。

如果要在顺序文件中加入新的记录，只能加在原有文件的最后，不能加在文件的开始或中间。

因而，顺序文件读写操作的特点是：文件中各记录写入的顺序、存放的顺序、读出的顺序三者一致，记录的逻辑顺序与物理顺序是一致的。对于顺序文件操作，一次打开后只能进行一种操作，要么进行读操作、要么进行写操作，而不能在一次打开文件后即读又写。

顺序文件的这种读写特性，使顺序文件的操作（如查看、查找、修改等）过程，常常难以避免对不需要记录的额外读写操作，因而往往会降低计算机的执行效率。因此一般来说，顺序文件只适合于处理的数据变动频繁、顺序性强、记录的处理量较在的数据文件组织，而不适合处理需要快速直接读写操作的数据组织。例如，学生成绩、职工工资管理一般设计为顺序文件。而电话号码查号系统则不应采用顺序文件的形式。

2．顺序文件的存放特点

顺序文件中的数据是按字符顺序进行存放的，以 ASCII 方式来存储数据，可以用他字处理软件查看和修改。

顺序文件的处理效率较低，但每一个记录的长度可不相同，而且由于各个记录之间仅以 CHR（13）（回车符）的一个字符一分隔，故存储空间较经济。

顺序文件由记录组成，记录以顺车符和换行符结尾。每个记录由一个或多个字段组成。在顺序文件中，每个记录可以具有不同的长度，不同记录中的字段长度也可以不同。

当把一个字段存入变量时，存储字段的变量类型决定了该字段的开头和结尾。当把字段存入字符串变量时，下列符号标识该字符串结尾：

（1）双引号（"）:当字符串以双引号开头时。

（2）逗号（,）：当字符串不以双引号开头时。

（3）回车和控制符：当字段位于记录的结束处时。

如果把一个字段写入一个数值变量，可以用逗号、一个或多个空格、回车控制符标识该字段的结尾。

12.3.2 建立和打开顺序文件语句 OPEN

格式：

OPEN〈文件〉FOR〈方式〉AS［#］〈文件号〉

功能：打开一上文件或设备。OPEN 具有双重功能，建立新文件和打开现有文件。

说明：

（1）〈文件〉是要打开的文件或设备名。文件名是合法的 DOS 文件名，可以包含驱动器和路径。

（2）〈文件号〉是一个整型表达式，其值范围从 1 到 255。执行 OPEN 语句时，每一个文件被赋予一个文件号。赋予文件的编号在关闭文件前一直有效。

设备名是文件所存放的名称，Quick BASIC 中常用的设备名见表 12-6。

Quick BASIC 默认当前驱动器为默认设备名。当设备名 A：、B：、C：、…、Z：时，要带有文件名，表示一个磁盘文件。例如：

```
A:filnamel.BAS
C:filname2.DAT
```

当设备名为 DYBD：、SCRN：、 LPT1：、LPT2：、COM1：COM2：时，只写设备名。这些设备中有的只用于输入，如 KYBD：；有的只用于输出，如 CONS：、SCRN：、LPT1：和 LPT2:；有的既可用于输入，也可用于

输出，如 COM1：、COM2。

（3）〈方式〉表示文件的使用方式，对于顺序文件，可以使用的操作方式见表 12-7。

这里的输入和输出都是对程序而言的。程序中的数据保存到文件称输出；程序读取文件中的数据称输入。

[例 12-6] 打开名为 student.txt 的顺序文件，它将保存来自程序的输出。如果 student.txt 文件不存在，则建立；如果它存在，则先删除它，再建立。

```
PREN"student.txt"FOR OUTPUT AS#1
```

[例 12-7] 打开 student.txt 现有文件，并使它与文件号 1 联系。程序将它的输出存储在 student.txt 现有记录的尾部。

```
OPEN"student.txt"FOR APPEND AS#1
```

由于 APPEND 方式兼有建立文件的功能，因此开始建立文件时就可使用 APPEND 方式。

[例 12-8] 打开 student.txt 现有文件，并使它与文件号 1 联系。可以利用 student.txt 的内容作为程序的输入。

```
OPEN"student.txt"FOR INPUT AS#1
```

[例 12-9] 在提示中由用户输入一个建立的数据文件名。

```
INPUT"Enter a file name:",filename$
OPEN filename$ FOR OUTPUT AS #1
```

表 12-6 Quick BASIC 中常用的设备名

Quick BASIC 中常用的设备名	含　义
A：B:	软盘驱动器
C：D:…Z:	硬盘驱动器
KYBD：	键盘
CONS：	屏幕
SCRN：	屏幕
LPT1：	并口打印机 1
LPT2：	并口打印机 2
COM1：	串地通信口 1
COM2：	串行通信口 2

表 12-7 顺序文件的操作方式

顺序文件的操作方式	含　义
OUTPUT	建立和打开程序输出的文件。如果该文件已存在，则被覆盖
APPEND	打开一个现有的文件，程序的输出将追加到该文件的尾部，文件原有内容保留
INPUT	打开一个仅由程序读取的文件。程序不能修改文件，但可使用文件输入给程序的内容

12.3.3 关闭文件语句 CLOSE

当你处理文件结束以后，部分数据可能仍然保留在缓冲区，并没有写入磁盘，如果出现断电、死相等问题时就会丢失数据。因此一定要用 CLOSE 语句关闭该文件，这就就能保证缓冲区中的数据写入磁盘。文件关闭后与它相联系的文件号释放并可赋给另一个文件。

格式：

```
CLOSE［[#]〈文件号〉［，[#]〈文件号〉］…]
```

功能：关闭一个或多个打开的文件或设备。

说明：

（1）〈文件号〉是与需要关闭的文件或设备相联系的编号。

（2）如果省略所有可选项，则关闭所有打开的文件和设备。

另外，如果满足下列条件之一，也将写入磁盘：一是缓冲区已满；二是缓冲区未满，但执行下一个输出语句，如 PRINT#。

[例 12-10] 关闭与文件号 1 相联系的文件。

```
CLOSE#1
```

关闭所有打开的文件和设备。

CLOSE

虽然 Quick BASIC 在程序结束时会自动关闭所有打开的文件，还是应该养成在文件处理结束后及时关闭的习惯。否则，如出现断电、死相等问题时就会丢失数据。

文件关闭后，如想再次使用它，则必须重新打开它。

12.3.4 把数据存储到文件中

当以 OUTPUT 或 APPEND 方式打开文件以后，程序就可以输出数据到文件中。以下 3 个语句可用于把数据存入打开的顺序文件。

PRINT# 把无格式的数据存入文件。

PRINT#USING 把格式数据存入文件。

WRITE# 把组织成字段的数据存入文件。

1. 建立顺序文件的步骤

要在磁盘上建立顺序文件，其步骤为：

（1）用 OPEN 语句打开文件（指定文件名、文件编号，并指出为输出文件）。

（2）利用 PRINT#、PRINT#USING 或 WRITE#语句将数据写入文件中。

（3）文件建立后，必须用 CLOSE 语句把文件关闭，否则数据可能会遗漏。

下面详细介绍存储数据到文件所用的语句。

2. 无格式存入语句 PRINT#

PRINT#语句在功能上类似于 PRINT 语句，但它是把数据输出到文件而不是屏幕。

格式：

PRINT#〈文件号〉，［〈表达式表〉］［{；|，}］

功能：把数据写入指定的顺序文件。

说明：

（1）〈文件号〉是一个打开文件的编号。

（2）〈表达式表〉是输出到文件的一个或多个数值或字符串表达式。输出之前先计算各表达式的值，然后把计算结果输出到顺序文件。如果省略〈表达式表〉，则在文件中写入一空行。

（3）“ ；”或“ ，”决定下一个输出语句的开始位置：

；接着最后一个输出位置输出。若分隔的是两个字符串，在读取时将不能恢复。

，输出到下一个打印区，每个打印区有 14 字符宽。

[例 12-11] 建立 studl.txt 文件，并将 3 个无格式的学号，姓名，年龄存入该文件。

```
OPEN "stud.txt"FOR OUTPUT AS#1          '在当前磁盘和目录下建立文件
CLS
INPUT"输入学号："xh$                    '输入：2000101
INPUT "姓名："xm$                       '输入：王红
INPUT "年龄："，nl                      '输入：16
PRINT#1，xh$,xm$,nl                     '写入 1 号文件
PRINT#1，"2000102","李小强"，17         '写入 1 号文件的尾部
PRINT#1，"2000103","张大勇"，16         '写入 1 号文件的尾部
CLOSE#1                                 '关闭 1 号文件
END
```

程序运行完成后，当前目录中已建立一个文件名为 stud1.txt 的顺序文件，如果要查看该文件的内容，可回到 DOS 状态，用 TYPE 命令显示其内容。

```
TYPE    stud1.txt
2000101     王  红      16
2000102     李小强      17
2000103     张大勇      16
```

从显示的数据文件内容可以看出，PRINT#语句与 PRINT 语句相似，只不过 PRINT#输出到磁盘文件，PRINT 输出到屏幕。在存储时，PRINT#也把空格存到了文件中，这会浪费许多磁盘空间。

3. 格式存入语句 PRINT#USING

PRINT#USING 语句和 PRINT USING 语句遵守同样的规则并使用相同的格式符，但它是把数据输出到文件而不是屏幕。PRINT#USING 在需要把大量的表格数据送入文件时很有用。

格式：

PRINT#〈文件号〉，USING〈格式字符串〉；［〈表达式表〉］［{；|，}］

功能：把格式数据写入指定的顺序文件。

说明：

（1）〈文件号〉是一个打开文件的编号。

（2）〈表达式表〉是输出到文件的一个或多个数值或字符串表达式。输出之前先计算各表达式的值，然后把计算结果输出到顺序文件。

（3）“；”或“，”决定下一个输出语句的开始位置：

；接着最后一个输出位置输出。

，输出到下一个打印区，每个打印区有 14 个字符宽。

（4）〈格式字符串〉请看第 7 章有关 PRINT USING 语句的说明。

[例 12-12] 把 3 行格式数据送入 stud2.txt 文件。本程序中的格式字符串是字符串变量 tmp$,它使输入数据垂直对齐。在本程序中用 APPEND 代替 OUTPUT，如果文件不存在，则建立；如果文件存在则把数据加入原有文件的尾部。

```
OPEN "c:\qb\stud2.txt"FOR APPEND AS#1              '在 C：盘 QB 目录下建立文件
CLS
tmp$="学号：\        \学分：###   平均分数：###.#"
INPUT"学号：",xh$                                  '输入：2000101
INPUT"学分：",xf%                                  '输入：80
INPUT"平均分数：",pj!                              '输入：96.8
PRINT#1，USING tmp$;xh$;xf%;pj！
PRINT#1，USING tmp$;"2000102";60;89
PRINT#1,USING tmp$;"2000103";50;76.2
CLOSE#1
END
```

在 DOS 下用 TYPE 命令显示的程序运行结果：

```
TYPE stud1.txt
学号：2000101   学分：80   平均分数：96.8
学号：2000102   学分：60   平均分数：89.0
学号：2000103   学分：50   平均分数：76.2
```

4. 记录写入语句 WRITE #

WRITE #语句主要用于建立供其他计算机语言程序读取的数据文件。

格式：

WRITE #〈文件号〉，〈表达式表〉

功能：把数据写入指定的顺序文件。

说明：

（1）〈文件号〉是一个打开文件的编号。

（2）〈表达式表〉是输出到文件的一个或多个数值或字符串表达式。输出之前先计算各表达式的值，然后把计算结果输出到顺序文件。

（3）WRITE #语句写入顺序文件的各项数据之前用逗号分隔，逗号分隔的每个数据项称为字段，由字段组成的每一行称为一条记录。数值数据的前后不留空格。

字符串数据用双引号（"）包围。如果这一字符包含空格或逗号，则其他计算机语言将会把空格和逗号作为字符串的一部分。

（4）每条记录用回车符和换行符（CHR$（13）和 CHR$（10））作为结束。也就是说，输出到文件的每条记录 都附加了回车和换行符，在原来长度的基础上多 2 个字节。例如：

"	9	6	0	0	0	1	"	,	"	W	a	N	g	H	o	n	g	"	,	2	1	CHR$（13）	CHR$（10）

在把全部记录写入文件后，将在全部记录的最后附加一个文件结束符号 CHR$（26），以标识文件的结束（End Of File）。例如：

记录 1	回车符	换行符	记录 2	回车符	换行符	…	记录 n	回车符	换行符	结束符

顺序文件的特点就是从数据文件中读取记录时，以 CHR$（13）和 CHR$（10）标识一个记录的结束，以 CHR$（26）标识一个数据文件的结束。一个记录的长度不能超过 255 个字符。

[例 12-13] 下面程序输入学号、语文、英语、数学成绩，并计算平均在绩，然后用 WRITE #写入顺序文件 stud3.txt 中。

```
OPEN "stud3.txt"FOR OUTPUT AS # 1
CLS
xh$=""
DO WHILE xh$<>"0"                    直到输入 0
INPUT"学号："，xh$
IF xh$<>"0"THEN                      如果输入 0，则不写入文件
INPUT"语文：",yw%
INPUT"英语：",yy%
INPUT"数学："，sx%
pj=（yw%+yy%+sx%）/3
WRITE # 1,xh$,yw%,yy%,sx%,pj
END IF
PRINT
LOOP
CLOSE # 1
END
```

在 DOS 状态下用 TYPE 显示 stud3.txt 文件的内容。：

```
TYPE STUD3.TXT
```

显示的结果与以下类似：

```
"2000101",89,95,100,94,66666
"2000102",91,86,95,90,66666
"2000103",78,86,82,82
```

从显示的顺序文件内容可看出，每一条记录占一行，记录的各字段之间用逗号分隔，字符串用双引号括起来。

12.3.5 读取顺序文件中的数据

上一节我们已经建立了 3 个文件，并用 DOS 的 TYPE 命令显示了它们的内容。现在我们将介绍用 Quick BASIC 的程序读取数据文件。要读取顺序文件中数据，用 OPEN 语句打开文件时要使用 INPUT 方式。

1．读取顺序文件中数据的步骤

在从数据文件中读取时，要用到两个语句和几个函数：

```
INPUT#          从文件读取一个或多个字段
EOD（）           记录指针确定是否已到达文件结束标志
LINE INPUT#   从文件读取一个记录（一整行数据）
```

INPUT$（） 从指定的文件中返回指定数目的字符串

LOF（） 从指定的文件中返回已经写入或读出的字节数

PREEFILE 返回当前未被使用的最小文件号

要读取顺序文件中的数据，其步骤为：

（1）利用 OPEN 语句打开一个输入文件（INPUT file）。

（2）利用 INPUT #或 LINE INPUT #语句，从文件中读取数据。

（3）利用 EOF（）函数测试文件中的数据是否读完。

（4）读完数据后，利用 CLOSE 语句把文件关闭。

2．数据项输入语句 INPUT #

INPUT #语句是从顺序文件中读取数据的主要工具。INPUT #语句人顺序文件得到输入数据的方式与 INPUT 语句从键盘得到输入数据的方式相当一致。这两个语句将一个或多个数据项赋给与其类型相匹配的变量。

格式：

INPUT #〈文件号〉，〈变量表〉

功能：从指定的文件中读取输入数据。

说明：

（1）〈文件号〉是一个打开文件的编号。

（2）〈变量表〉是从文件中读取的一个或多个用逗号分隔的变量名，从数据文件中读出的数据赋给这些变量。变量表中的变量名必须与顺序文件中的数据项具有相同的类型。

（3）顺序文件中的数据项可以是用 WRITE #语句建立的数据字段，也可以是用 PRINT #语句、PRINT #USING 语句或任何可以建立数据文件的程序。

（4）在用 INPUT #语句把读出的数据赋给对应变量时：

如果是数值变量，将忽略前导空格、回车换行控制符，把遇到的第一个非空格、回车和换行符作为数值的开始，遇到空格、加车控制符则认为数值结束；

如果是字符串数据，同样忽略开头的空格、回车控制符。如果需要把开头带有空格的字符串赋给变量，则必须把字符串放在引号中。

[例 12-14] 按用户输入的文件名建立顺序文件，然后显示顺序文件中的内容。

```
INPUT"Enter Filename:";n$                    '输入文件名
OPEN n$ FOR OUTPUT AS # 1                    '建立文件
PRINT #1。"This is saved to the file." '写入 1 行内容
CLOSE
OPEN n$ FOR INPUT AS # 1
INPUT # 1,a$
PRINT"Read from file:";a$
CLOSE
END
```

[例 12-15] 下面程序先建立一个数据文件，并用 WRITE #语句将 5 个人的姓名和年龄输出给该文件。然后程序将该文件关闭，再次将它打开用于输入，并把这 5 个记录读回到程序中，并显示到屏幕上。

```
CLS
OPEN"TEST.TXT"FOR OUTPUT AS # 1
PRINT"输入 5 个姓名，年龄"
FOR i%=1 TO 5
INPUT"输入姓名，年龄：";Name$,Age%
WRITE # 1,Name$,Age%
NEXT%
CLOSE # 1
```

```
OPEN"TEST.TXT"FOR INPUT AS # 1
PRINT
PRINT"你输入的姓名，年龄如下："
PRINT
FOR i%=1 TO 5
INPUT # 1,Name$,Age%
PRINT Name$,Age%
NEXT i%
CLOSE # 1
END
```

[例 12-16] 上机运行程序。

```
OPEN"TEST.DAT"FOR OUTPUT AS # 1
PRINT # 1,USING"##.###";12.12345
CLOSE
OPEN"TEST.DAT"FOR INPUT AS # 1
INPUT # 1,a$
PRINT a$
END
```

3. 文件结束函数 E0F（）

在读取数据文件的数据记录时，如果不能确切知道该文件中有几个记录，到达文件结束标志以后仍继续读取时，就会出现文件结束错误信息“ Input past end of file ”。为了预防这一错误的发生，必须检测文件结束标志，可使用 EOF（）函数，当 EOF（）函数的值为真时，遇到文件结束标志，表示全部记录已经读完，应停止继续读取操作。

格式：

EOF（〈文件号〉）

功能：测试一个文件的结束。

说明：

（1）如果记录指针到达文件的结束位置（以 CHR$（26）标识），EOF（）返回真（-1），否则返回假（0）。

（2）〈文件号〉是一个打开文件的编号。

[例 12-17] 下面程序建立一个 LIST 文件，然后向该文件中输入姓名（Name$）和年龄（Age%），并询问是否继续输入，回答 Y 继续，输入其他结束输入。接着打开刚才写入的文件，从该文件中读取数据，并显示所有数据。

```
CLS
OPEN"LIST"FOR OUTPUT AS # 1
DO
INPUT"    NAME:"Name$
INPUT"     AGE:",Age%
WRITE # 1,Name$,Age%
INPUT"Add another entry";R$
LOOP WHILE UCASE$（R$）="Y"
CLOSE # 1
'Print the file to the screen.
OPEN "LIST"FOR INPUT AS # 1                    '按 INPUT 方式打开文件
CLS
PRINT"Entries in file:":PRINT
DO WHILE NOT EOF（1）                          '如果记录指针没有达到文件尾则继续
```

```
INPUT # 1，Rec1$,Rec2%                    '从文件中读取数据项
PRINT Rec1$,Rec2                          '显示到屏幕上
LOOP
CLOSE # 1
KILL"LIST"                                '删除 LIST 文件
END
```

[例 12-18] 统计学生文件中的平均成绩。score 是保存在文件中的成绩。

```
OPEN"STUDENT.DAT"FOR INPUT AS # 1
DO WHILE NOT EOF（1）
count=count+1
INPUT # 1,score
total=total+score
PRINT count,score
LOOP
PRINT
PRINT"Total students:";"Average score:"total/count
CLOSE
END
```

[例 12-19] 查询记录。在顺序文件查找某些记录，必须从文件头开始逐个读入每条记录，查到需要的记录时显示出来。

```
INPUT"Enter file to Search:",file$
INPUT"Enter name to search for:",s$
OPEN file$ FOR INPUT AS #1
DO UNTIL（s$=name$）OR EOF（1）
INPUT #1,name$,addr$,tel$
LOOP
IF s$=name$ THEN
PRINT"Name:";name$
PRINT"Address:";addr$
PRINT"Tel:";tel$
ELSE
PRINT"Name not found"
END IF
CLOSE #1
END
```

4. 行输入语句 LINE INPUT #

INPUT #是从文件中得到一个数据项（字段）的数据，如果要得到某一行（记录）所有数据，则要逐个列出，比较麻烦。LINE INPUT #语句是用来从顺序文件中读取一整行（最多 32767 个字符）的数据，并赋值给它后面的字符串变量 中。

格式：

```
LINE INPUT # 〈文件号〉，〈字符串变量〉
```

功能：从指定的文件中读取一整行数据，并把它赋给指定的字符串变量。

说明：

（1）〈文件号〉是一个打开文件的编号。

（2）从文件中读取数据直到遇到回车控制符为止的一行数据，按一个字符串赋值给〈字符串变量〉。

（3）一行最多为 32767 个字符。

[例 12-20] **自动产生由 3 个字段组成的 10 个记录，然后显示所有生成的数据。**

```
CLS
OPEN"TEST.DAT"FOR OUTPUT AS #1
FOR i%=1 TO 10
WRITE #1,i%,2*i%,5*i%
NEXT i%
CLOSE #1
OPEN "TEST.DAT"FOR INPUT AS #1
DO
LINE INPUT #1,a$
PRINT a$
LOOP UNTIL（EOF（1））
CLOSE #1
END
```

程序的运行结果为:

```
1，2，5
2，4，10
3，6，15
4，8，20
5，10，25
6，12，30
7，14，35
8，16，40
9，18，45
10，20，50
```

[例 12-21] **复制一个文本文件。**

```
INPUT"Enter Source File:",sourcename$
IF sourcename$=""THEN END
INPUT"Enter Desination File:,destname$
IF destname$=""THEN END
OPEN sourcename$ FOR INPUT AS #1
OPEN destname$ FOR OUTPUT AS #2
DO UNTIL EOF（1）
LINE INPUT #1,linebuffer$
PRINT #2,linebuffer$
LOOP
END
```

在程序中同时打开两个文件，一个作为输入文件，一个作为输出文件。程序从输入文件中每读取一行数据，就把它写到输入文件中，从而实现了文件的复制。

5. INPUT$（）函数

格式：

```
INPUT（〈n〉,#〈文件号〉）
```

功能：从指定的文件中返回指定数目的字符串。

说明：

（1）〈n〉是返回的字符数目（字节）。

（2）〈文件号〉是一个打开文件的编号。

[例 12-22] 从 TEST.DAT 文件中读取 3 个字符。

```
OPEN"TEST.DAT"FOR OUTPUT AS #1
PRINT #1,"The text"
CLOSE
OPEN"TEST.DAT"FOR INPUT AS #1
PRINT INPUT$（3,1）                    打印前 3 个字符
CLOSE
END
```

6．LOF（）函数

格式：

LOF（〈文件号〉）

功能：从指定的文件中返回已经写入文件的长度或读出的字节数。

说明：〈文件号〉是一个打开文件的编号。

[例 12-23] 测试 TEST.DAT 文件中的字节数。

```
INPUT"Enter filename:";f$
OPEN f$ FOR BINARY AS #1
PRINT "File length=";LOF（1）
CLOSE
```

7．FREEFILE 函数

格式：

FREEFILE

功能：返回当前未被使用的最小文件号。

说明：本函数没有参数和括号。

[例 12-24] 测试当前未被使用的小文件号。

```
OPEN"TEST.DAT"FOR OUTPUT AS #1
PRINT"Next file number:";FREEFILE
CLOSE
```

12.4 随机文件

12.4.1 随机文件的和格式及特点

随机存取文件简称随机文件，又称直接存取文件（Direct Access File），随机文件有以下特点：

（1）在缓冲区中，随机文件中的数据是以字符串的形式存放的，当写入磁盘文件时，由文件系统把它们转换为压缩的二进制码，以节省存储空间。因此，在把数值数据存入缓冲区之前必须先转换为字符串形式；而从 读取数据时，必须先把压缩二进制码转换成字符串形式，然后再把它们转换成数值数据。

（2）随机文件中每一个记录的长度都固定，整个文件结构像一个表格。每一第记录都有一个记录号（Record Number）。因此，存取数据时，要指明记录号 n，通过“（n-1）×记录长度”计算出该记录与文件首记录的相对地址，就可以直接从指定位置存取数据。因此，打开文件时必须 在 OPEN 语句中指定记录的长度。

（3）每个记录划分为若干个字段，每个字段的长度也固定，等于相应变量的长度。

（4）各变量（数据项）要按一定格式置入相应的字段。

（5）打开随机文件后，既可读了可写。

用 Quick BASIC 中的用户定义数据类型，可以建立含有字符串数据和数值数据的复合数据类型，既记录类型。然后用 GET 语句和 PUT 语句中的变量读取或写入记录。使用用户定义数据类型，可以简化对随机文件的操作。

12.4.2 建立和打开随机文件 OPEN

格式：

OPEN〈文件〉［FOR RANDOM］AS［#］〈文件号〉LEN＝〈记录长度〉

功能：打开一个随机文件。

说明：

（1）〈文件〉是要打开的文件或设备名。

（2）〈文件号〉是文件的编号。

（3）〈记录长度〉是一个整型表达式，为随机存取文件设置记录长度。

（4）Quick BASIC 默认为随机文件，即省略或加上 FOR RANDOM 都是随机文件。

12.4.3 定义随机文件缓冲区中的字段 FIELD

用 FIELD 语句把输入、输出记录的缓冲区空间划分为若干个字段，并定义每个字段的长度，即用 FIELD 语句定义记录的结构。

格式：

FIELD［#］〈文件号〉，〈字段宽度 1〉AS〈字段名 1〉［，〈字段宽度 2〉AS〈字段名 2〉］…

功能：定义随机文件缓冲区中各字段所占的长度。

（1）〈文件号〉是文件的编号。

（2）〈字段宽度〉是一个字段中字符的总数目，以字节为单位。

（3）〈字段名〉是一个字符串变量，每一个字段用一个字段名，但不能与程序中其他字符串变量同名。

（4）FIELD 语句为各字段分配的字节数总和不能超过 OPEN 语句中指定的长度。

随机文件各记录之间没有分隔符（顺序文件用回车换行符作为分隔），由于记录是等长的，因此可以计算出个记录的起始位置。

从 FILED 语句可以看到，在随机文件的记录中，各字段只能存放字符型数据。如果想存入数值型数据，则要先把它转换成字符型数据。读取记录中的数据时，再将字符型数据转换成数值型数据。

示例：

```
OPEN"FILEDAT.DAT"FOR RANDOM AS #1 LEN=80
FIELD #1,30 AS name$,50 AS address$
```

12.4.4 把数据存储到随机文件中

把数据写入随机文件通常需要以下几步：

（1）用随机存取方式打开文件，并指定记录的长度，记录长度由 LEN（）函数计算。

（2）使用用户定义数据类型或 FIELD 语句定义随机文件缓冲区中各个字段的长度。

（3）用 PUR #语句把文件缓冲区中的数据写到磁盘文件中。

1．把数值型转换为字符型数据函数

格式：

MKI$（〈整型表达式〉）

MKI$（〈长整型表达式〉）

MKS$（〈单精度型表达式〉）

MKD$（〈双精度型表达式〉）

功能：把相应的数值型表达式转换为字符串，以便能用 FIELD 语句存储字符型变量。

说明：

MKI$函数返回一个 2 字节字符串；

MKL$函数返回一个 4 字节字符串；

MKS$函数返回一个 4 字节字符串；

MKD$函数返回一个 8 字节字符串。

2．给缓冲区中的字段赋值语句

格式：

LSET〈字段名〉＝〈字符串表达式〉

RSET〈字段名〉＝〈字符串表达式〉

LSET〈记录变量 1〉＝〈记录变量 2〉

功能：LSET 和 RSET 把字符串表达式的值赋值给随机文件缓冲区中的字段，以便用 PUT 语句写入磁盘。LSET 把一个记录型变量的目录复到另一个记录型变量。

说明：

（1）〈字段名〉是在 FIELD 语句中定义的随机文件的字段名。

（2）LSET 语句在将一个字符串赋给缓冲区中的一个字段时，字行串向左对齐。RSET 语句则向右对齐，多余的字节以空格填充。

（3）〈记录变量〉是任意用户定义数据类型的记录变量。使用 LSET 把一个记录变量分配给一个不同的用户定义数据类型记录变量。

3．缓冲区数据写入磁盘语句 PUT *

格式：

PUT［*］〈文件号〉［，［〈记录号〉］［，〈变量〉］］

功能：把一个变量或随机缓冲区中的一个记录写到指定的文件。

说明：〈变量〉是含有记录的内容，并向文件输出的变量。PUT 语句把该变量中的内容写入文件。

[例 12-25] 下面程序演示使用 LSET 和 RSET 语句的情况。

```
OPEN"FILEDAT.DAT"FOR RANDOM AS #1 LEN=10
FIELD #1,5AS Ls1$,5 AS Rs1$
LSET Ls1$="LSET"
REST Rs1$="RSET"
PUT #1,1
CLOSE #1
OPEN"FILEDA.DAT"FOR RANDOM AS #1 LEN=10
FIELD #1,5 AS Ls2$,5 AS Rs2$
GET #1,1
PRINT"*"+Ls2$+"*"+Rs2$+"*"
CLOSE #1
```

[例 12-26] 建立一个学生名单随机文件，每个记录由姓名（6 字节）、性别（2 字节）、年龄（2 字节）、入学成绩（4 字节）组成。

```
OPEN"student.dat"FOR RANDOM AS #1 LEN=14
FIELD #1 6 AS Fxm$,2 AS Fxb$,2 AS Fnl$,4 AS Frxcj$
DO
READ xm$,xb$,nl%,rxcj
IF xm$="END"THEN EXIT DO
LSET Fxm$=xm$
LSET Fnl$=MKI$（nl%）
LSET Frxcj$=MKS$（rxcj）
PUT #1
LOOP
DATA 吕德华，男，25，180，周曼玉，女，22，160.5
DATA 高学友，男，21，161，END…
```

12.4.5 读取随机文件中的数据

从随机文件中读取数据应按以下步骤：

（1）用 OPEN 打开随机文件。

（2）使用用户定义数据类型或 FIELD 语句定义各个字段。

（3）用 GET #语句从指定的文件中读取一个记录。

1. 读取随机文件数据语句 GET

格式：

GET［#］〈文件号〉［，［〈记录号〉］［，〈变量〉］］

功能：从指定的文件读取一个记录到随机缓冲区或变量中。

说明：〈记录号〉是随机访问文件读入记录的编号。〈变量〉是从文件中接收输入的变量，这是量通常是一个用户定义类型的变量。

2. 把字符型还原为数值型数据函数

格式：

CVI（〈2 字节数字字符串〉）

CVL（〈4 字节数字字符串〉）

CVS（〈4 字节数字字符串〉）

CVD（〈8 字节数字字符串〉）

功能：把相应的数字字符串还原为数值型数据。

说明：CVI 函数返回一个整型数值；CVL 函数返回一个长整型数值；CVS 函数返回一个单精度数值；CVD 函数返回一个双精度数值。

[例 12-27] 读取例 12-26 建立的随机文件 student.dat 并在屏幕上显示出来。

```
OPEN "student.dat"FOR RANDOM AS #1 LEN=14
FIELD #16 AS Fxm$,2 AS Fxb$,2 AS Fnl$,4 AS Frxcj$
PRINT "姓名"，"性别"，"年龄"，"入学成绩"
FOR i%=1 TO LOF（1）/14
GET #1,i%
xm$=Fxm$
xb$=Fxb$
nl%=CVI（Fnl$）
rxcj=CVS（rxcj$）
PRINT xm$,xbE,nl%,rxcj
NEXT i%
CLOSE #1
END
```

12.4.6 用记录类型处理随机文件

用前面介绍的方法处理随机文件虽然能够解决问题，但要用 FIELD 语句定义记录的各个字段，用 MKI$（）、CVI（）等函数转换类型，用 LSET 或 RSET 语句向缓冲区传送数据，使用起来很不方便。

使用 Quick BASIC 提供的记录类型（即用户定义数据类型），则可很方便地处理随机文件。建议使用记录类型处理随机文件。

[例 12-28] 使用用户定义类型建立一个随机文件。记录由 Student（学生名）和 Score（得分）两个字段组成。

```
TYPE TestRecord
Student AS STRING *20
Score AS SINGLE
END TYPE
```

```
DIM MyClass AS TestRecord
OPEN"FINAL.DAT"FOR RANDOM AS #1 LEN=LEN（MyClass）
MyClass.Student="MarySa"
MyClass.Score=99
PUT #1,1,MyClass
GET #1,1,MyClass
PRINT"STUDENT:",MyClass.Student
PRINT"SCORE:",MyClass.Score
CLOSE #1
KILL"FINAL.DAT"
```

[例 12-29] 随机文件的主要优点之一就是可以通过记录号直接访问文件中的任一个记录，从而大大提高存取速度。在用 PUT 语句向文件写记录时，就把记录号赋给了该记录。在读取文件时，通过把记录号放在 GET 语句中可以人随机文件中取回一个记录。下面是通过记录号读取随机文件 stock.dat 中任一条记录的程序。

```
CLS
DEFINT A－Z
TYPE stockitem
partnumber AS STRING*6
description AS STRING*20
unitprice AS SINGLE
quantity AS INTEGER
END TYPE
DIM stockrecord AS stockitem
OPEN "stock.dat"FOR RANDOM AS #1 LEN=LEN（stockrecord）
numberofrecords=LOF（1）/LEN（stockrecord）
getmorerecords=-1
DO
PRINT"Enter record number for part you want to see（0 to end）:";
INPUT"",recordnumber
IF recordnumber>0 AND recordnumber<=numberofrecords THEN
GET #1,recordnumber,stockrecord
PRINT"Part number.",stockrecord.partnumber
PRINT"Unit price:",stockrecord.description
PRINT"Quantity:";",stockrecord.quantity
PRINT
ELSEIF recordnumber=0 THEN
getmorerecords=0
ELSE
PRINT"Input value out of range."
END IF
LOOP WHILE getmorerecords
CLOSE #1
END
```

上面程序在打开文件后，先计算原来文件中最后一个记录的编号，然后执行读操作。文件的最后一个记录号由下面语句计算得到：

```
numberofrecords=LOF（1）/LEN（stockrecord）
```

LOF（1）函数计算出 stock.dat 文件的总长度，即全部字节数。如果 stock.dat 是一个新文件或者文件没有记

录，则 LOF（1）返回 0。LEN（stockrecord）函数计算出一个记录的字节数。

最后记录号的计算公式：

最后记录号＝文件中全部字节数/一个记录中的字节数

在随机文件的存取中，最后记录号有着重要的作用，因为它是文件中记录的个数。

随机文件建立后，可以从文件中读取数据。从随机文件中读取数据有两种方法，一种是顺序读取，一种是通过记录号读取。由于顺序读取不能直接访问任意指定的记录，因而速度较慢。

12.5 文件与目录维护语句

Quick BASIC 中提供了几个用来维护磁盘文件及目录的语句。

1．MKDIR、CHDIR、RMDIR 语句

格式：

MKDIR〈路径名〉

CHDIR〈路径名〉

RMDIR〈路径名〉

FILES［〈文件说明〉］

功能：MKDIR 建立一个子目录，CHDIR 改变当前目录，RMDIR 删除一个子目录。FILES 显示当前工指定的目录。

说明：如果省略〈路径名〉，则会产生错误信息。〈路径名〉最多可以包含 127 个字符。如果省略〈文件说明〉，则显示当前目录下的所有文件。

示例：

```
MKDIR"C:\TEMP\TEST"
CHDIR"C:\TEMP"
FILES
RMDIR"TEST"
FILES"C:\DOS\*.COM"
FILES"D:\"
FILES"C:\QB\*.*"
```

2．KILL 语句

格式：

KILL〈文件说明〉

功能：删除磁盘上的文件。

说明：如果文件已经打开则不能删除，必须先把它关闭。

示例：

```
KILL"C:\QB\STUN3.TXT"
```

在程序中输入要删除的文件：

```
INPUT "File to delete:";f$
KILL f$
```

3．NAME 语句

格式：

NAME〈旧文件名〉AS〈新文件名〉

功能：改变磁盘上的文件名。

说明：〈旧文件名〉和〈新文件名〉必须包含相同的路径名。

示例：

```
NAME"C:\QB\EG12-25.BAS"AS"C:\QB\EG12-32.BAS"
```

在程序中输入要改换前后的文件名：

```
INPUT"Old Name:";OldFN$
INPUT"New Name:";NewFN$
NAME OldFN$ AS NewFN$
```

4. SHELL 语句

格式：

```
SHELL［〈命令〉］
```

功能：暂停执行一个 Quick BASIC 程序，运行一个 DOS 命令或批处理文件。

说明：〈命令〉是 DOS 命令名和批处理文件名。当 DOS 命令执行完成后，将返回 Quick BASIC 环境。如果省略〈命令〉，SHELL 将转至 DOS 状态下并且显示 DOS 提示符待命，用 EXIT 命令从 DOS 状态返回 Quick BASIC 环境。

示例：

```
SHELL "DIR"
```

5. SYSTEM 语句

格式：

```
SYSTEM
```

功能：关闭所有打开的文件，返回 DOS 操作系统。

12.6 本章小结

通过本章学习，主要应该掌握数据文件的基本概念及其建立、存取和修改的各种方法。本章所使用的语句和函数如表 12-4 所示。

表 12-4 数据文件 I/O 语句及函数

语句或函数	功 能
OPEN 语句	打开指定的磁盘文件，为其分配内存缓冲区，指定存取方式和记录长度。
CLOSE 语句	关闭指定的磁盘文件，释放所分配的缓冲区。
INPUT#语句	读取磁盘文件记录，并将各域数据赋给各内存变量*
WRITE#语句	对内存中已被赋值的各域，以记录为单位，写入磁盘文件*
FIELD#语句	分配一个记录中各域的大小。* *
GET#语句	从磁盘文件读取一个记录的数据。* *
PUT#语句	将一个记录的数据存磁盘文件。* *
EOF 函数	检测是否已读完文件中的全部数据。* *
LOF 函数	给出被打开文件的总字节数。* *

注：*用于顺序文件。* *用于随机文件。

习 题

1．下面程序读取 5 个学生的数据，分别为学号、姓名、语言、英语、数学，并计算每个学生的平均成绩。

2．将一个包含 n 位学生数据的记录数组，按平均分数从大到小排序，然后显示排序结果。

3．假设某单位有 10 名职工，试编制程序，将他们的工作编号、姓名、性别、工资输入到一个文件名为“ zgqk.dat ”的文件。

4．求学生记录最高分，并输出与最高分相差在 10 分以内的学生记录。

5．已知磁盘上存放着某单位全年每次报销的经费（假定为整数），试编写一个程序，从磁盘上读入每次报销的经费，计算其总和，并将结果存入另一新建文件中。

6．假定磁盘上有一个学生成绩文件，存放着 100 个学生的情况，包括学号、姓名、性别和 5 门课程的成绩，试编写一个程序，建立以下 2 个文件：

（1）女生情况的文件；

（2）按 5 门课程平均成绩的高低排列的学生情况的文件（须增加平均成绩一栏）。

7．用随机文件建立一个人事档案管理文件。文件包括编号、姓名、性别、年龄、工作单位、职称等字段。

输入若干男女职工档案，并使其完成下述功能：

（1）计算男女职工各多少人。

（2）列出全部师档案。

（3）根据编号查找某人档案。

第十三章　综合程序设计概要

到目前为止，已经学习了用 Quick BASIC 语言进行程序设计的基本方法和一些最常用的基本算法。本章首先总结了结构化程序设计的方法，然后提供了一些典型实例，目的在于通过实践，提高读者解决实际问题的能力。

本章的主要内容有：

- 结构化程序设计方法总结
- 程序设计综合举例

13.1　结构化程序设计方法总结

早在第二章就已介绍了结构化程序设计方法的概念，它主要包含：自顶向下、逐步求精、模块化、结构化编码。并且，以后各章通过例题的示范和习题的练习，对此已有一定的了解，这里将对结构化程序设计的方法加以总结。

13.1.1　自顶向下设计

自顶向下设计（top-down design）是一种从总体出发，逐层分解和逐步细化，直至使整个系统设计达到足够简单、明确、清楚和详细为止。这种方法如同撰写文章，先确定题目写什么，再拟订出大纲以及每段的大意，最后再逐字逐句书写。

现在来看一个例子。本例将要较甲乙两班学生的四级英语考试成绩，首先对这个问题进行分析，然后总结出算法。

1. 问题分析

首先，用 V1，V2 分别代表甲乙两班的平均分数值，那么，比较结果不外乎以下 3 种情况：

V1＞V2，则甲班的平均分数比乙班高；

V1＜V2，则甲班的平均分数比乙班低；

V1＝V2，则甲乙两班的平均分数相等。

其次，进一步需要了解如何得到 V1、V2。

我们知道：

V1＝甲班的总分数/甲班总人数

V2＝乙班的总分数/乙班总人数

可以归纳出一个公式，即计算平均分数的公式：

V（平均分数）＝参加考试人的总分数/参加考试的人数

再次，考虑到一般情况下，甲乙两班参加考试的人数不会相等，为了使计算 V 的过程通用化，输入数据时，选择－1 作为终止标志。

最后，输出结果：若 V1＞V2，就打印“甲＞乙”；若 V1＜V2，就打印“甲＜乙”；若 V1＝V2，就打印“甲＝乙”。

2. 算法设计

用自顶向下、逐步求精的设计方法来描述这一算法，表示如下：

顶　层：

（1）计算出 V1

（2）计算出 V2

（3）比较 V1、V2

第 2 层：

（1）设计一个主程序 main

（2）设计一个子程序 Average 计算 V

（3）设计一个比较 V1、V2 的子程序 Compare

第 3 层：（1）设计主程序

① 输入数据，用 READ 语句读数，DATA 语句放置以下两组数据：

70,80,60,43,…,-1

60,85,80,32,…,-1

② 调用子程序 Average 求 V1；

③ 调用子程序 Average 求 V2；

④ 调用子程序 Compare 进行比较，并输出结果

（2）设计一个子程序 Average 计算 V

① S＝0，总分数置 0

② N＝0，总人数置 0

③ 读一个分数 X

④ 若 X＝-1，就转⑧

⑤ S＝S+X，总分数加 X

⑥ N＝N+1，总人数加 1

⑦ 转③

⑧ V＝S/N

⑨ 返回

（3）设计比较 V1、V2 的子程序 Compare

① 若 V1＞V2，则打印出“甲＞乙”，转④

② 若 V1＜V2，则打印出“甲＜乙”，转④

③ 若 V1＝V2，则打印出“甲＝乙”

④ 返回

通过这个例子可以看到，它是从最高一层开始一层一层地展开，一步一步“精细化”，直到精细到根据经能直接写出 BASIC 语句为止。所以这种方法称作“自顶向下，逐步求精”的设计方法。逐步求精是结构化程序设计方法的核心。

这种程序设计方法，能够使人们思路清晰，有条不紊地进行工作，用较短的时间设计出正确的程序，并容易验证程序的正确性，便于维护。但是，采用这种设计方法，要求在每一步要检查其算法的正确性，即每次在向下一层精细化之前应确保上一层算法是正确的，否则精细化后的结果将会是错误的，这是需要特别注意的。

13.1.2 模块化

在程序设计中，将一个复杂的算法（或程序）分解成若干个相对独立、功能单一的模块，并利用这些模块组合成所需要的全局算法（或程序），这种设计程序的方法称为模块化程序设计（modular programming）。

在模块化结构中，整个系统尤如搭积木一样，是由各模块适当组合而成的，如图 13-1 所示。矩形块表示各模块，矩形框内写有按功能定义的模块名，各矩形块之间的流向线表示调用关系。整个程序按调用关系分成若干层次，每一层次由若干个模块组成。

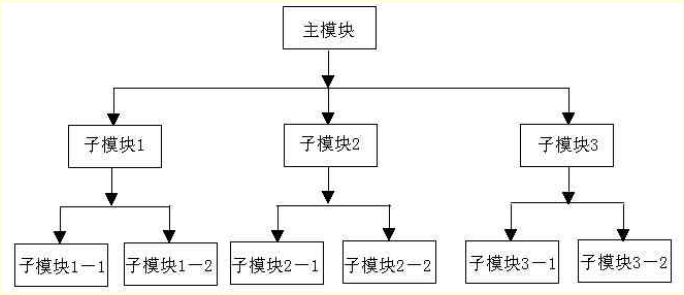


图 13-1　模块化结构

这种方法设计出的程序具有以下优点：

（1）使复杂的软件件研制工作可以化整为零，分而治之，便于多人分工编写，从而缩短开发周期，节省开发费用，提高软件质量。

（2）可以设计一块，调试一块，设计完成，调试也随之完成，十分方便，有效地防止模块间错误的扩散，增加整个系统的稳定性与可靠性。

（3）结构灵活，便于组装；层次分明，便于维护；条理清晰，容易理解。其优越性随着程序规模的增大而愈加明显。

13.1.3　结构化编码

一个算法，不仅可用文字描述，还可用流程图表示；可以用 BASIC 语言编写程序，也可以用 C、FORTRAN、PASCAL 等语言编写程序。本书介绍了 Quick BASIC 的各种功能（语句和函数），用它就能方便地编写结构化的程序。

编写程序时，以下的问题值得注意：

（1）划分程序模块时，最好一个模块实现一个功能。模块由子程序或函数来实现，如用 Quick BASIC 语言编写程序，可用 SUB 子程序和 FUNCTION 子程序实现。

（2）使用见名知义的变量名。例如：用 COUNT 代表计数，用 TIME 表示时间，用 WAGE 代表工资等。即用英文单词或相应汉语拼音作为变量名。

（3）要用注释语句简明地说明主程序、子程序、程序段或者个别语句的功能。

（4）采取缩进格式对程序进行编排。在循环（或块 IF）中将循环体（或块 IF）内的语句向右边缩进若干格。当有循环嵌套时，同一层循环体（或块 IF）语句按列对齐，这样层次分明。

（5）循环控制变量应尽量用整型变量。

（6）输出计算结果时，应加以文字说明。

（7）对复杂的多功能应用程序，最好采用“菜单技术”，以方便使用。

（8）用 FOR－NEXT 语句完成已知循环次数的循环，用 WHILE－WEND 或 DO－LOOP 语句完成未知循环次数的循环。尽量用 SELECT CASE 语句来完成多路选择结构的程序设计。

（9）对多次重复出现的表达式，可以先求出其值放在一个变量中以便多次使用，减少重复计算。

综上所述，在具体编码中既要使程序的执行效率高（速度快），也也使程序简单朴实，便于阅读和使用。

13.1.4　软件工程的概念

在本书的学习中，大家对从问题分析、建立模型、确定算法和数据结构、画流程图、编码、调试、编写程序说明书这一解题全过程已经比较熟悉，这对处理较简单的问题是适合的。但是，计算机的迅猛发展对软件的需求越来越大，用个体手工方式编写程序的办法编写出的软件质量差，成本高，研制周期长，可靠性难以保证，管理与维护困难。因此，人们就运用工程学的基本原理和方法来组织和管理高质量的软件产品的研制，像工厂生产产品一样，软件的生产也分几个阶段，每段都有严格的质量管理，这就形成了软件工程。

一个软件开始提出任务直到研制完成、交付使用、最后该软件被淘汰这一全过程称为软件的生命周期，它由如下 3 个阶段组成：

（1）软件定义阶段；

（2）软件开发阶段；

（3）软件维护阶段。

软件定义阶段的主要任务是：软件项目规划和需求分析。

软件项目规划的任务是确定总的目标和功能，进行技术、经济、社会的可行性研究，并对可利用的资源、开发成本、开发进度进行估计，制定实施计划，最后进行评审。

需求分析着重解决该软件应该什么的问题，这就要对软件产品的用户需求进行详细精确的分析，写出功能说明书。它是下个阶段进行软件设计的基础，也是软件产品验收的依据。

软件开发阶段，这个阶段要进行设计、编码和高度，设计包括划分模块、定义接口、确定数据结构、规定标记等，接着对每个模块进行过程设计、编码和单元调试，最后进行组装调试，对每一调试及结果都要评审。

软件维护阶段，在程序交付使用之后，在实践中必然会出现一些问题。或根据用户需要，增加一些新的功能；或由于环境的变化（如：计算机系统的更换），需要对软件作某些修改；或由于软件调试中未能发现软件中的潜伏的错误而需要改正等。这就是软件维护时期的任务。这个时期是很长的，直到该软件被淘汰为止。

应当指出，上述各阶段的划分并不是绝对的，一成不变的。一般意义上的程序设计指的是：软件开时期中的设计和编码，包括概要设计、详细设计和编码 3 个阶段。

软件工程中各个阶段分别由不同的人去完成。一般地说：问题定义与需求分析是由系统分析员完成的，概要设计高级程序员完成的，详细设计和编写程序是由程序员完成的。测试工作是由有经验的软件人员完成的。作为程序设计者应当发解在整个软件开发中自己工作的位置与要求。

13.2 程序设计综合举例

程序设计要综合用到算法、数据结构、结构化程序设计方法以及计算机语言工具这 4 个方面的知识。本书已介绍了多种算法，如穷举法、递推法、迭代法、递归法、对分法等，介绍了数组和自定义类型的数据结构，在此基础上，就可以采用结构化程序设计的方法，并用 Quick BASIC 语言来进行综合的程序设计。因此，本节将介绍一些常用的曲型实例。

13.2.1 用高斯消元求逆矩阵

矩阵运算是数值运算中的一个重要部分，如求句矩阵的和、差、积，求矩阵的逆、矩阵置零、矩阵转置等。在有的 BASIC 版本中有专门的矩阵语句，实现矩阵运算十分方便。如没有矩阵语句，可用循环和二维数组编写实现矩阵运算的程序。但矩阵逆运算比较复杂，也比较有用，因此本例介绍一种求逆方法。

[例 13-1] 用高斯消元法求矩阵 A 的逆矩阵 X。

设：$A=\begin{bmatrix}3 & 2 & 1\\ 1 & 1 & 1\\ 2 & 3 & -1\end{bmatrix}$

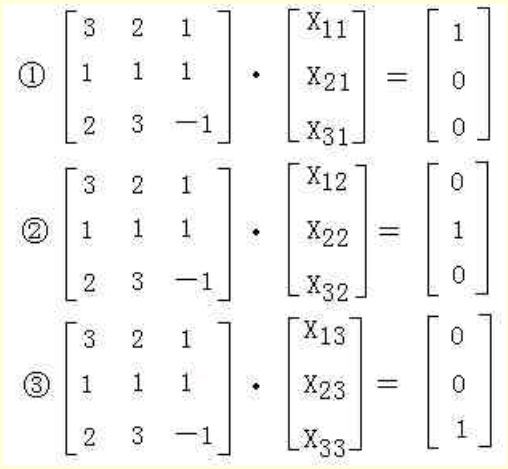
1. 问题分析

根据矩阵运算规则有 $A * A^{-1}=A * X=C$

这里 X 是 A 的逆矩阵，C 是单位矩阵。即有：

$$\begin{bmatrix}3 & 2 & 1\\ 1 & 1 & 1\\ 2 & 3 & -1\end{bmatrix}\cdot\begin{bmatrix}X_{11} & X_{12} & X_{13}\\ X_{21} & X_{22} & X_{23}\\ X_{31} & X_{32} & X_{33}\end{bmatrix}=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

根据矩阵的乘法规则可有：



由①得一个三元一次方程组如下：

$3X_{11}+2X_{21}+X_{31}=1$ ……（1）

$X_{11}+X_{21}+X_{31}=0$ ……（2）

$2X_{11}+3X_{21}-X_{31}=0$ ……（3）

同样，由②、③又可得两组三元一次方程组。因而，矩阵求逆的问题，就归结为解多元一次方程组的问题。

2．算法

顶层：

（1）输入矩阵 A，生成单位矩阵 C。

（2）根据矩阵的维数 N，组成循环，求解 N 遍多元一次方程组。

（3）输出 A 的逆矩阵 X。

第 2 层：

（1）输入矩阵 A，C

① 定义数组 A（N，N+1），X（N，N），C（N，N）

② 生成单位矩阵 C

③ 把 A 的元素，按行放在 DATA 中，用 READ 语句读入。

（2）用高斯消元法求解方程组

① 判断方程中每一列元素是否全为 0。

A：如全为 0，则方程组无解，停止执行。

B：如果不全为 0，再判断主元素（主对角线上的元素）是否为 0。

如果为 0，则将同一列的其他不为 0 的那一行，与主元素所在行交换；否则不交换。

② 逐个消去各元素：

A：把主元素置成 1。

B：使主元素所在列的其他元素置成 0。

③ 输出结果。

第 3 层：

（1）判断多元一次方程组是否有解

A：设一计数器 Q，判断某列元素为 0 时，计数器计数，当计数器 Q 与判断元素的个数相等时，表明该列元素全为 0，打印“无解或不定解”。

B：判断主元素 A（J，J）=0？如等于 0，与同列中不为 0 的元素所在行交换。

（2）消元

A：主元素置 1，该行乘以 1/A（J，J）。

例如本例的（1）式，变为

$$X_{11}+\frac{2}{3}X_{21}+\frac{1}{3}X_{31}=\frac{1}{3} \quad (1a)$$

B：使闰元素所在列的其他元素置成 0，例如消去（1）式中的 X_{11}，则用 X_{11} 的系数的负值（即－1）去乘式（la），和（2）式相加，即：

$$-X_{11}-\frac{2}{3}X_{21}-\frac{1}{3}X_{31}=-\frac{1}{3} \qquad (la')$$

$$+\quad X_{11}+X_{21}+X_{31}=0 \qquad (2)$$

$$\frac{1}{3}X_{21}+\frac{2}{3}X_{31}=-\frac{1}{3} \qquad (2a)$$

要消去（3）式中的 X_{11}，用 X_{11} 的系数的负值（即－2）去乘式（la），和（3）式相加，即

$$-2X_{11}-\frac{4}{3}X_{21}-\frac{2}{3}X_{31}=-\frac{2}{3} \qquad (la'')$$

$$+\quad 2X_{11}+3X_{21}-X_{31}=0 \qquad (3)$$

$$\frac{5}{3}X_{21}-\frac{5}{3}X_{31}=-\frac{2}{3} \qquad (3a)$$

于是第一次消元的结果得到了：

$$X_{11}+\frac{2}{3}X_{21}+\frac{1}{3}X_{31}=\frac{1}{3} \qquad (la)$$

$$\frac{1}{3}X_{21}+\frac{2}{3}X_{31}=-\frac{1}{3} \qquad (2a)$$

$$\frac{5}{3}X_{21}-\frac{5}{3}X_{31}=-\frac{2}{3} \qquad (3a)$$

重复上述消元步骤，得到第 2 次消元的结果：

$$X_{11}-X_{31}=1 \qquad (1b)$$

$$X_{21}+2X_{31}=-1 \qquad (2b)$$

$$-5X_{31}=1 \qquad (3b)$$

再进行第 3 次消元，最后得：

$$X_{11}=0.8$$

$$X_{21}=-0.6$$

$$X_{31}=-0.2$$

源程序：

```
'EXAMPLE 1
INPUT "N=";N                          '输入矩阵 A 的维数
DIM SHARED C(N,N),A(N,N+1),X(N,N)
PRINT   "Mat C"                       '生成和打印矩阵 C
FOR I=1 TO N
FOR J=1 TO N
C（I，J）=0
IF I=J THEN
C（I，J）=1
END IF
PRINT C(I，J);" ";
NEXT J
PRINT
NEXT I
PRINT"The Table of Coefficients"
```

```
FOR Z=1 TO N                    '读入和打印矩阵 A
FOR I＝1 TO N
FOR J＝1 TO N
READ A（I，J）
PRINT A（I，J）；" ";
NEXT J
A(I,N+1)=C(Z,I)
PRINT A(I,N+1)
NEXT I
RESTORE
FOR J=1 TO N
IF A(J，J)=0 THEN               '主元素是否为 0
Q＝1                            '记录为 0 的元素个数
I＝J
DO WHILE I＜＝N
IF A（I，J）＝0 THEN
Q＝Q+1
END IF
I＝I+1
LOOP
IF Q＝N－J－1 THEN
PRINT "No Unique Solution";"Or No Solution"
STOP                            '该列元素全为 0，停止运行
END IF                          '当主元素为 0 时
END IF
I＝J－1
DO
I＝I+1                          '找一个同列中不为 0 的元素
LOOP UNTIL I＜＝N AND A（I，J）＜＞0
FOR K＝J TO N+1
SWAP A（J，K），A（I，K）       '与主元素行交换
NEXT K
C=1/A(J,J)                      '主元素置 1
FOR I＝J TO N+1
A（J，I）＝A（J，I）* C
NEXT I
FOR I＝1 TO N
IF I＜＞J THEN
C＝－A（I，J）                  '消去同列其他元素
FOR S＝J TO N+1
A（I，S）＝A（I，S）+C * A（J，S）
NEXT S
END IF
NEXT I
NEXT J
FOR I＝1 TO N
```

```
X（I，Z）＝A（1，N+1）
NEXT I：PRINT：PRINT
NEXT Z
PRINT "Mat X"                    '输出逆矩阵 X
FOR I＝1 TO N
FOR J＝1 TO N
PRINT X（I，J）："   ";
MEXT J
PRINT
NEXT I
DATA 3,2,1,1,1,1
DATA2,3,－1
END
```

运行结果：

```
N＝3
Mat C
1    0    0
0    1    0
0    0    1
Then Table of Coefficients
3    2    1    1
1    1    1    0
2    2    －1   0

3    2    1    0
1    1    1    1
2    3    －1   0

3    2    1    0
1    1    1    0
2    3    －1   1
Mat    x
.8    －1    －.2
－.6    1    .4
－.2    1    －.2
```

13.2.2 打印万年历

每 400 年中只有 97 个闰年，凡不能被 4 整除的年份是平年；凡不能被 100 整除，但能被 4 整除的年份为闰年，如 1980 年，1996 年；能被 100 和 4 整除，但不能被 400 整除的年份为平年，如 1800 年，1900 年；闰年的 2 月份为 29 天，平年的 2 月份为 28 天；大月为 31 天，小月为 30 天。

根据年份按下式求 E1，E2（单，双月的 1 号是星期几）：

```
Z＝Y+（Y－1）\4－（Y－1）\100+（Y－1）\400+1
E1＝Z－（Z\7）  * 7
E2=T1+E1－（T1+E1）\7 * 7          （其中，T1 为单月天数）
```

[例 13-2]　**按图 13-2 的格式输出一张万年历表，要求只给出年号就打印出结果。**

1．算法设计

顶层：

（1）确定年历打印格式；

（2）计算元月 1 日是星期几；

（3）判断是否是闰年以确定 2 月份是 28 天还是 29 天。

第 2 层：

（1）打印格式画分为 4 个模块

① 打印年份的牌头；

② 打印月份的牌头；

③ 打印每月的历的每一行；

④ 打印表格的分隔线。

（2）计算元月 1 日是星期几，用以下公式：

Z＝Y－1+INT（（Y－1）/4）－INT（（Y－1）/100）+INT（（Y－1）/400）+1

（3）判断是否是闰年

① 凡能被 4 整除，但又不能被 100 整除的，是闰年；

② 凡能被 4 整除，又能被 400 整除的，是闰年。

第 3 层：各模块级设计

（1）主模块

① 定义各子模块；

② 键盘输入年份 Y；

③ 调子程序 NIAN，打印年份牌头；

④ 计算元月 1 日是星期几和单月 1 号是星期几；

⑤ 计算双月 1 号是星期几；

⑥ 打印月份牌头，调用子程序 YUET；

⑦ 分别调用子程序 YUEL 打印单月和双月的月历。

（2）SUB 子程序 NIAN。用 TAB 函数定位打印如图 13-2 所示的年份牌头。

（3）Function 子程序 PRT$，打印一条分隔线。

（4）SUB 子程序 YUEL，用以打印每月的月历，按它的格式为 7 列（0,1,2,3,4,5,6）代表星期几，每月的行数是不等的，少则 4 行，多则 6 行。因此，设一变量 K，把当月星期几的值作为初值赋给它，开始打印当月第 1 周的星期几，当 K＞6 时，让它又重新置 0。

（5）SUB 子程序 YUET

① 打印如图 13-2 中每日的牌头。包括两条分隔线和表示星期几的代号。

② 判断是否闰年，以确定 2 月份的天数。

2．源程序

```
'EXAMPLE 2
COMMON SHARED E1,E2,Y                    '定义 E1，E2，Y 为全局变量
DECLARE SUB YUET（YUE，T2）
DECLARE SUB NIAN（）
DECLARE SUB YUEL（K,R,R2,X1,X2,I）
DECLARE FUNCTION PRT$（）
CLS：INPUT"INPUT YEAR:";Y               '输入年份
CALL NIAN                                '打印年份牌头
Z＝Y+（Y－1）\4－（Y－1）\100+（Y－1）\400+1
E1＝Z－（Z\7） * 7                       '计算元月一日是星期几
FOR YUE＝1 TO 6
READ T1，T2
```

```
CALL YUET（YUE，T2）                    '计算双月一号是星期几
D=1:S=1
DO WHILE T1＞0 AND T2＞0
CALL YUEL（E1,D,S,T1,T2.1）             '打印单月日历
IF T1＝0 THEN EXIT DO
PRINT TAB（33）；"$";
CALL YUEL（E2,S,D,T2,T1,38）            '打印双月日历
LOOP
NEXT YUE
DATA 31,28,31,30,31,30,31,31,30,31,30,31
END
SUB NIAN                                '打印年份牌头子程序
PRINT TAB(25)：STRING $(16,"$")
PRINT TAB(25);"$";TAB(30);Y;TAB(40);"$"
PRINT TAB(25);STRING $(16,"$")
PRINT
END SUB
FUNCTION PRT$
PRT $＝"------$------"
END FUNCTION
SUB YUEL(K,R,R2,X1,X2,1)                '打月历子程序
DO
IF K＜＝6 THEN
IF R＞X1 THEN EXIT DO
PRINT TAB(1+4 * K);R;
R=R+1:K=K+1                             'R 为打当月的累计天数
IF R＞6 THEN K＝0                       'R2 为同排另一月的天数
IF R＞X1 AND R2＞X2 THEN X1＝0；E1＝E2
END IF
LOOP UNTIL K＝0
IF X1＝0 AND I＝1 THEN PRINT TAB（33）；"$"
IF X1=0 THEN PRINT:PRINT PRT$
END SUB
SUB YUET(YUE,T2)                        '打月份牌头子程序
PRINT TAB(10);"--(";YUE;")--";TAB(33);"$";
PRINT TAB(47);"  --(";YUE+1;")--"
PRINT PRT $
F=1                                     '打星期标志
DO WHILE F＜39
FOR I＝0 TO 6：PRINT TAB（F+4 * I）；I；：NEXT1
F＝F+37：IF F＝38 THEN PRINT TAB（33）；“＄”；
LOOP
PRINT PRT $
IF YUE=1 THEN                           '判断是否闰年
IF Y/4＝Y\4 THEN
IF Y/400＜＞Y\100 THEN F＝0
```

```
    IF Y/400＝Y\400 THEN F＝0
    END IF
    IF F＝0 THEN T2＝29
    END IF
    END SUB
```

3. 运行结果

```
INPUT YEAR：？2001
```

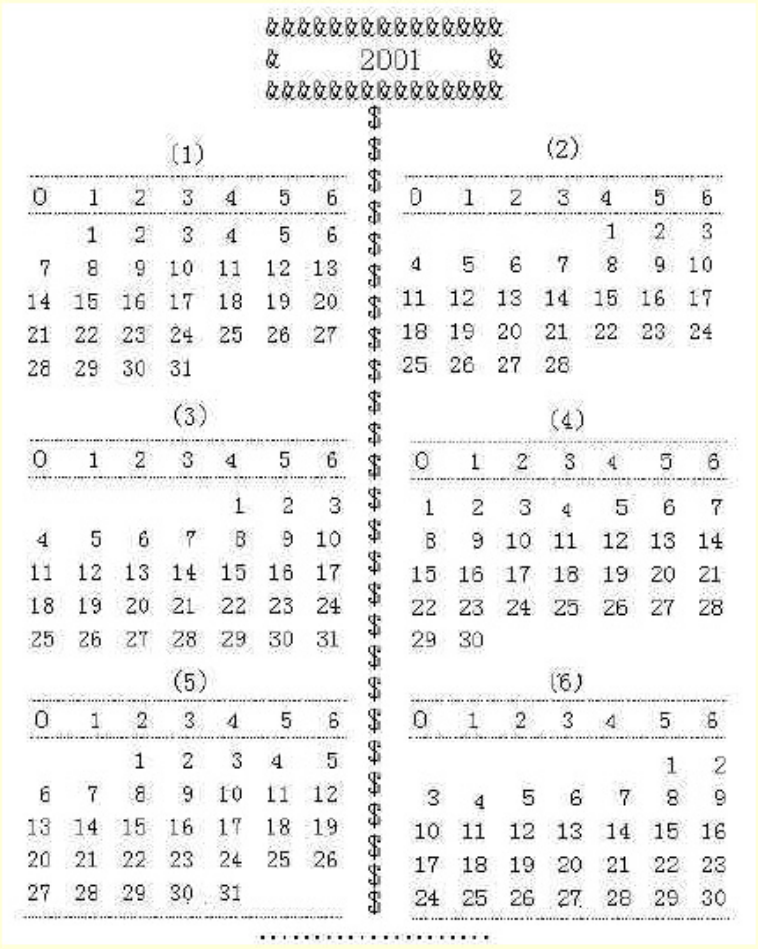


图 13-2　日历

4. 程序说明

本程序共有主模块，打年份牌头块，打月牌头模块及打月历等 4 个模块，各主要语句作用在程序的注释中已作了简要的说明，这里再作如下补充。

打单月或双月的日历时，都要调用 SUB YUEL 子程序，其输出顺序是星期天、星期一、……、星期六，变量 K 是控制此顺序的，因此 K 不能大于 6，否则将 K 重新置为 0。R 就设置 T1＝0 作为标志符，这时将 E2＝＞E1，以确定下一排单月的一号打在什么位置。

13.2.3　“菜单”技术

用户使用计算机时总希望操作方便，菜单工作方式就是一种深受用户欢迎的计算机输入输出的接口。这种方式把一个计算机软件（或程序）的各种功能按不同层次列出各种选择项，使用时，屏幕上就显示出这些选择项，并约定一个代号（如一位数字或一个英文字符），用户输入一个代号，就进入执行该选择项，并约定一个代号（如一位数字或一个贡文字符），用户输入一个代号，就进入执行该选择项的功能，有如饭店提供一个菜谱，让顾客需要什么菜就点什么菜，所以，计算机的这种工作方式，称为菜单工作方式。

下面我们来看一个菜单的使用方法和如何编制菜单程序。

图 13-4，就是在屏幕上显示出的一个菜单，共有 5 个选择项，每一项的第一个字母作为代号，为引人注目，这个字母大写并加一右括号，与其余字符分开，选择项的下面有一行提示字符：

G A D P Q→

光标停留在该提示符的箭头号右边，这表示等待用户的输入，而且告诉你，在这 G、A、D、P、Q 五个字母中，你决定选取其中一个字母输入，就是你所选中的选择项。

例如输入一个 G，屏幕下方接着就又一行信息：

<Enter>to confilm;<ESC>to Redo

它的意思是进一步确认，你是不是真的要进入 G 选择项，如果是确定要选 G，就按 Enter 键；如果是由于疏忽按错键，或者临时改变主意，又不选 G 项了，那么，就按<ESC>键，退回到主菜单，重新再选。

如果按了 Enter 键，则屏幕上又显示：

The Get data option

表示就进入所选的选择项了。至于这个选择项的具体功能程序，本例中没有写，所以显示一个提示信息‘ Press Spacebar to continue ”，只要按一次格键，又返回初始选择画面。

下面将介绍菜单程序的设计方法。

[例 13-3] 设计一个菜单测试程序 Menu Test，菜单的显示屏幕如图 13-3 所示。

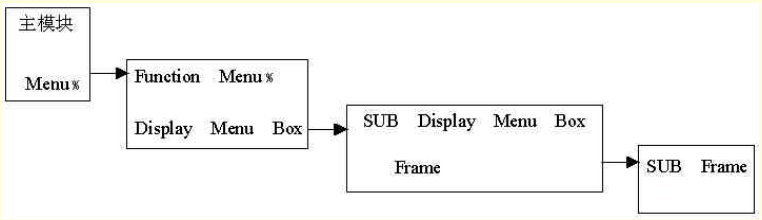


图 13-3 模块的相互关系

算法设计

顶层：

（1）设计显示屏幕，确定图框的形状、位置；安排选择项的位置，代号如何向用户作出提示。

（2）设计对用户选择的处理，用户正确选择了一个选择项后，计算机该转到什么功能块，如果选错了，又怎样处理等等。

第 2 层：模块划分

（1）设计一个主模块 Menu Test。用多路选择的结构，分别进入 5 个选择项；5 个选择项的确定由一个函数子程序 Menu 实现。

（2）设计一个函数子程序 Menu，它的功能是：

① 在屏幕上显示菜单，该功能再调用一个子程序 DisplayMenuBox 来实现。

② 接受用户的选择，等待用户输入；用户输入后，进行识别，分别不同情况向用户提示。

（3）设计一个屏幕显示菜单的子程序 DisplayMenuBox，它的功能有二：

① 画一个方框，调用画方框的子程序 Frame；

② 显示 5 个选择项和提示字符串。

（4）设计一个画方框的子程序 Frame，按给定的尺寸画一个矩形的双线框。各模块之间的相互关系如图 13-4 所示。

第 3 层：各模块的详细设计：

（1）主模块

① 定义一个字符型数组 MenuOptions$，存放 5 个选择项的题目。（如“ The Get data option ”）

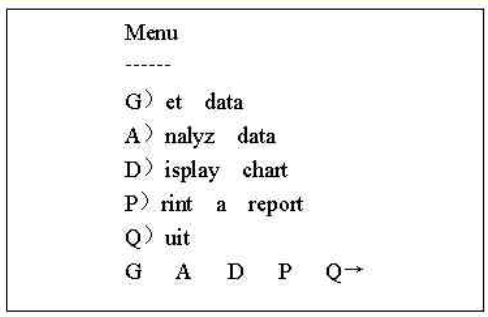


图 13-4 菜单提示

② 用 SELECT CASE 语句，按 5 个分支进入 5 个子功能模块，该语句调用 Function 子程序 Menu%，分别进入 SELECT 语句的 5 个入口。Menu%子程序返回的是整数，调用时，把数组 MenuOptions$作为实参传给 Menu%子程序的形参数组 Choices$()。

（2）Menu%函数子程序

① 调用 DisplayMenuBox 子程序，输出显示屏幕；

② 等待用户键盘输入，用函数 INKEY$扫描键盘的输入，用户可以随便输入大写或小写字母，程序都把它转换成大写字母，一旦有字符输入，计算机就判断它是否是合法的选择项目，方法是用字串查找函数 INSTR，在存有“ GADPQ ”的字符串 OKSTR$中查找；

③ 如果没选对，则发出一声蜂鸣，以提醒用户，并继续等待；

④ 如果选对了，就进一步确认，如果是确定要进入的话，按 Enter（ASCII 码 13）键，如果要退出，按 ESC 键（ASCII 码 27）。同样用一字符串 controlkeys$，存放这两个字符，当用户输入后，程序判断，是否是选对了，如没选对，发出一声蜂鸣声，以提示用户，并等待输入。

如果按了 Enter 键，则将存放选择项的 inchoice%的值返回主程序，执行 SELECT 语句，转入相应入口，显示选择项题目，并进一步显示；

“ Press the spacebar to continue ”。

如果按了 ESC 键，则将光标移在提示字符串的后面，等待重新选择。

（3）SUB 子程序 DisplayMenuBox

① 把每一个选择项的第 1 个字母转换成大写字母，并加一括号。

② 把 5 个选择项的第 1 个字母，构成一个提示字符串 prompt$（“ GADPQ→ ”）和一个字符串 OKSTR$（“ GADPQ ”）供 Menu%程序查找选择字母时用。

③ 根据选择项的长度 Longchoice%决定画面矩形的尺寸：Leftcoord%，rightcoord%，topcoord%，Bottom%的值。

④ 将上述 4 个值传给子程序 Frame，调用它画出矩形框。

⑤ 显示框内标题“ Menu ”及各选择项和提示字符串。

（5）SUB 子程序 Frame

① 先画矩形的 4 个角（用 ASCII 码 201、187、200、188）

② 画矩形的两条垂直线

③ 画矩形的两条水平线

源程序：

```
'EAMPLE 3
'Menu Demonstrates the Menu% function          '主程序
CONST false%=0,true%=NOT false%
DECLARE FUNCTION Menu%(choices$())
DECLARE SUB DisplayMenubBox(choices$(),leftCoord%,prompt$,ok$)
DECLARE SUB Frame(left%,right%,top%,bottom%)
DIM MenuOptins$(5)
DATA Get data
```

```
DATA Analyze data
DATA Display a chart
DATA Print a report
DATA Quit
FOR i%=1 to 5
READ menuOptions$(i%)
NEXT i%
CLS
DO
SELECT CASE Menu%(menuOptions$())      '调用函数子程序
CASE 1                                 '返回值（1,2,…,5）作多路选择
CLS：PRINT "The Get data option."
CASE 2
CLS:PRINT "The Analyze data option."
CASE 3
CLS:PRINT "The Display a chart option."
CASE 4
CSL:PRINT "The Print a chart option."
CASE ELSE
done%=true%
END SELECT
IF NOT done% THEN
PRINT
PRINT "Press the spacebar to continue."
DO
ch$=INDEY$
LOOP UNTIL ch$＝" "
END IF
LOOP UNTIL done%
END
FUNCTION Menu%(choices$()) STATIC     '函数子程序
listLength%＝UBOUND(choices$)
DisplayMenuBoxchoices$(),leftMargin%,promptStr$,okStr$
controlKeys$＝CHR$(13)+CHR$(27)    '调用显示子程序
DO
LOCATE,,1
charPos%＝0
DO
answer$＝UCASE￥（INKEY$）          '等待并扫描键盘输入
IF answer$<>" " THEN
charPos%＝INSTR（okStr$,answer$）
IF charPos%＝0 THEN BEEP
END IF
LOOP UNTIL charPos%>0
PRINT answer$
LOCATE 11+listLength%,23,0
```

```
PRINT "<Enter>to confirm;<Esc>to redo."
inChoice%＝charPos%
charPos%＝0
DO
answer$=INKEY$
IF answer$<>" " THEN
charPos%0＝INSTR（controlKeys$,answer$）    '确认选择
IF charPos%＝1 THEN BEEP
END IF
LOOP UNTIL charPos%>0
IF charPos%=1 THEN
done%＝true%
CLS
ELSE                                        '按 ESC 键，光标回到提示行
done%＝false%
LOCATE 11+listLength%,23:PRINT SPACE$(35)
LOCATE9+listLength%,lefMargin%+3+LEN(promtStr$):PRINT" ";
LOCATE,POS(0)-1
END IF
LOOP UNTIL done%
Menu%＝inChoice%
END FUNCTION
SUB DisplayMenuBox(choiceList$(),leftCoord%,prompt$,ok$)    '显示子程序
numChoices%＝UBOUND(choiceList$)
prompt$＝" "
ok $＝" "
longChoice%＝0
FOR i%＝1 TO numChoices%
first$=UCASE$(LEFT$(choiceList$(i%),1)
ok $＝ok$+first$
prompt$=prompt$+first$+" "          '生成提示字符串
longTemp%＝LEN(choiceList(i%))
IF longTemp%>longChoice%THEN longChoice%＝longTemp%
NEXT i%
longChoice%＝longChoice%+1          '确定选择项最大字符数
prompt$＝prompt$+"——＞"
IF LEN(prompt$)＞＝longChoice%THEN longChoice%=LEN(prompt$)+1
leftCoord%=37-longChoice%\2         '确定矩形的 4 个角的坐标
rightCoord%=80-leftCoord%
topCoord%=3
bottomCoord%=10+numChoices%
Frame leftCoord%,rightCoord%,topCoord%,bottomCoord%     '调用画框子程序
FOR %＝1TO numChoices%
LOCATE 6+i%,leftCoord%+3
PRINT UCASE$.(LEFT$(choiceList$(i%),1)+")"+MID$(choiceList$(i%),2))
NEXT i%
```

```
LOCATE 4,38:PRINT "Menu"          '显示框内各项内容
line$＝STRING$(longChoice%,196)
LOCATE 5,leftCoord%+3:PRINT line$
LOCATE 7+numChoices%,leftCoord%+3:PRINT line$
LOCATE 9+numChoices%,leftCoord%+3:PRINT prompt$;
END SUB
SUB Frame(left%,right%,top%,bottom%)STATIC     '画框子程序
LOCATE TOP%，left%:PRINT CHR$(201)              '画矩形的四个角
LOCATE top%,right%:PRINT CHR$(187)
LOCATE bottom%,left%:PRINT CHR$(200);
LOCATE bottom%,right%:PRINT CHR$(188);
FOR vert%=top%+1 TO bottom%-1                   '画垂线
LOCATE vert%,left%:PRINT CHR$(186);
LOCATE vert%,right:PRINT CHR$(186);
NEXT vert%
horiz%=right%-left%-1                           '画水平线
hline$=STRING$(HORIZ%,205)
LOCATE top%,left%+1:PRINT hilne$
LOCATE bottom%,left%+1:PRINT hline$
END SUB
```

13.2.4 陷阱技术

计算机在程序执行过程中，总会完成某一些功能的操作，比如计时、按键、演奏音乐、操作游戏棒等等，我们统称之为事件（event）。计算机可以自动地检测这些事件是否发生。如果没有发生，就正常地执行程序；如果有这类事件发生，就能自动中断正常程序的执行，转到一个处理事件的子程序去处理，处理完后，又自动回到中断点继续程序的正常执行。这种功能就是陷阱技术。

与此类似，程序执行过程中，有时会出现错误，计算机也能自动检测错误是否发生。如果没有发生错误，程序正常进行，如果发生了错误，就中断程序执行，转到错误处理程序，经过处理后，再回去执行正常程序。

对于错误的陷阱，又称错误捕捉（Error Trapping）,对于事件的陷阱，又称事件捕捉（Even Trapping）程序中的陷阱功能，要通过程序设计来实现。下面就这两种捕捉的编程进行介绍。

1．时间捕捉（事件捕捉之一）

它要使用以下两个语句

（1）事件设置语句

语句格式：

ON TIMER（n） GOSUB <标号>

其中 TIMER 是一个时间函数，它可以给出从午夜以来所经过的秒数。

n 为用户指定的时间长短，用秒作单位。

<标号>是事件处理子程序的入口标号。

功能：设置时间陷阱

（2）时间启停控制语句

语句格式：

TIMER{ON | OFF | STOP}

其中参数 ON、OFF、STOP，三者任取其一

功能：ON：表示允许时间捕捉。

OFF：表示禁止时间捕捉。

STOP：表示暂停时间捕捉。

举例：

[例 13-4] **在屏幕右上角显示当天的日期和时间，每一分钟修改一次显示。**

```
'EXAMPLE 4
TIMER ON                              '允许时间捕捉
ON TIMER（60）GOSUB Display           '设置一分钟的时间陷阱
DO WHILE INDEY $＝" ":LOOP            '敲任意一键，停止程序执行
END
Display:Oldrow＝CSRLIN                '存贮光标当前行
OLDCOL＝POS（0）                      '存贮光标当前列
LOCATE 1，60                          '指定显示位置在右上角
FOR I＝1 TO 300
PRINT DATE $：TIME$                   '显示日期和时间
NEXT 1
LOCATE OLDROW，OLDCOL                 '光标返回原来位置
RETURN
```

如果在程序执行过程中，不想显示时间，就可以用 TIMER STOP **使其暂停。以后还可以再次用** TIMER ON **打开。**

2．**功能键捕捉（事件捕捉之一）**

它要使用以下两个语句

（1）事件设置语句

语句格式：

ON KEY（n）GSUB <**标号**>

其中：n **的值和键盘上功能键的对应关系如表** 13-1 **所示：**

表 13-1 **对应表**

N	1~10	11	12	13	14	15~20
对应功能键	F1~F10	↑	←	→	↓	**自定义功能键**

功能：

设置一个功能键陷阱，即在程序执行过程中，只要按一次该指定的键，就调用一次＜括号＞指定的处理子程序

（2）功能键控制语句

KEY {ON ｜ OFF ｜ STOP}

ON：**允许功能键捕捉**

OFF：**禁止功能键捕捉**

STOP：**暂停功能键捕捉**

[例 13-5] **设有一程序，运行时在屏幕中显示时间（**TIME$**）的变化，但要求按** F5 **键时在第** 15 **行上显示“** TRAPED **”。程序如下：**

```
'EXAMPLE 5
CLS
ON KEY(5) GOSUB PRT
K:KEY(5) ON
KEY(5) STOP
LOCATE 12,36
PRINT TIME$
KEY(5) ON
GOTO K
PRT：LOCATE 15，30
```

```
PRINT "TRAPED"
RETURN
```

从上述两个事件捕捉的例子，可以看到，对事件捕捉有 3 个必要的步骤：

①要编写一个事件处理子程序；子程序的第一语句应有一个＜标号＞，以便调用。

②用一条事件设置语句

```
ON ＜事件＞ GOSUB ＜标号＞
```

当＜事件＞发生时，把控制转移到事件处理子程序去。

其中＜事件＞可以是 TIMER、KEY，还可以是其他事件如 PEN、PLAY、STRIG 等。

③要有一条控制事件捕捉的语句

```
＜事件＞ON 允许事件捕捉
＜事件＞OFF 禁止事件捕捉
＜事件＞STOP 暂停事件捕捉
```

3. 错误捕捉

在程序执行过程中，总是免不了要出错，当发生错误时，有时会显示出错信息，并停止执行；有时会进入死循环，不显示任何信息，甚至造成“死机”。如果采用出错陷阱技术，可以捕捉到预想到的错误，这时程序就能执行错误处理程序，处理完之后，程序又恢复正常运行。

执行错误捕捉有关的语句有两个：

（1）出错陷阱设置语句

```
ON  ERROR  GOTO  ＜行号或标号＞
```

此语句一般放在程序的前面。在程序执行过程中，出现错误时，程序就转到指定的行号或标号的语句去执行。

如果行号是 0，则并不表示要转到行号 0 去，而是关闭所有的错误捕捉。

例如：

```
ON  ERROR GOTO  1000
```

如出现错误，程序就转到行号 1000，它是错误处理程序的第 1 个语句。错误处理程序的任务是分析错误、处理错误并通知用户，继续执行程序等。在分析错误时，要利用 ERR，ERL 两个函数；

```
ERR 函数给出出错的错误代码；
ERL 函数给出发现错误行的行号。如果程序中无行号，则 ERL 回送 0
```

（2）继续程序执行语句

```
RESUME＜参数＞
```

功能：执行了错误处理程序后，继续程序的正常执行。当语句后省略参数时，则在产生错误的地方恢复程序执行；当参数为 NEXT，则从出错行的下一语句起开始继续执行；参数可以是行号或标号，但应避免使用。

[例 13-6]　一个执行程序错误捕捉演示程序。

```
'EXAMPLE 6
ON ERROR GOTO handler
DEF FNTEST(A)=1-SQR(A)
FOR I=2 TO -2 STEP -1
PRINT I,FNTEST(1)
NEXT 1
END
handler:PRINT   "No Negative arguments"
PRINT   "ERR=";ERR,"ERL=";ERL
RESUME NEXT
RUN
z       -0.4142136
l     0
```

```
0    1
－1   No Negative arguments
ERR=5   ERL=0
－2   N 为 Negative arguments
ERR=5    ERL=0
OK
```

当 I＝－1 和 I＝－2 会打印出“ No negative arguments ”，最后结束。

13.2.5 快速排序

第八章中，已经学过用比较法和选择法对数组元素进行排序。数据个数不多的时候，这些方法还是可以用的，但是在数据量很大的情况下，排序的效率低，花的时间就相当的长，有时是不能满足实际的需要的。所以人们又研制了效率更高、速度更快的一些排序方法，这里再介绍两种：Shell 排序的 Quick 排序。

[例 13-7] Shell 排序

算法设计：

设一数组 A，有 N 个元素，比较大小时，不是用相邻的两个元素比较，而是用相隔一定距离的两个元素比较，如取 N 的一半为相隔的距离；比完一次以后，再把距离缩短一半，一直到距离为 1。用图 13-5 来说明排序过程。

（1）首先取间隔 H1＝4，每两项比较，如 2 与 7 比较，不对调；6 与 4 比较，对调；1 与 5 比，不对调。

（2）只要发生了对调，就再按 H1 的距离检查一遍，然后进行下一步。

（3）缩短间隔，取 H2＝H1/2＝2，进行如上的比较。

（4）如有对调，用 H2 重复一遍。

（5）再把距离缩短一半，H3＝H2/2＝1 进行上述比较。

（6）如有对调，用 H3 重复一遍，一直到没有对调，排序完成。

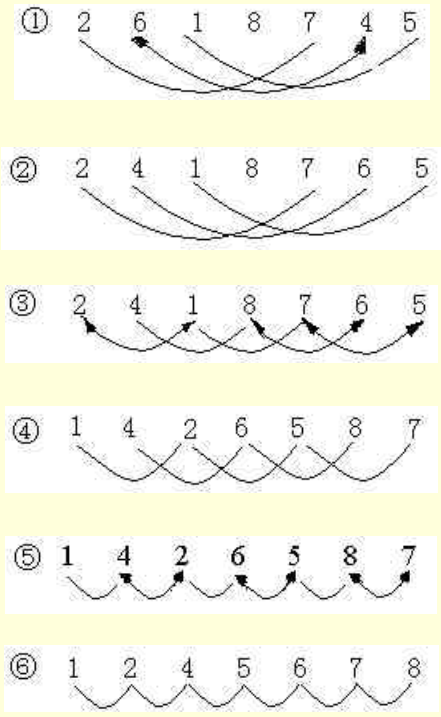


图 13-5 Shell 排序

源程序：

```
'EXAMPLE 7
Declare SUB ShellSort(x(),m)
CONST false=0,true=NOT false
```

```
INPUT "The dimension of array is:"n
DIM SHARED a(n)
FOR i=1 TO n
INPUT a(i)
NEXT i
ShellSort a(),n
FOR i=1 TO n
PRINT a(i),
NEXT i
END
SUB ShellSort(x(),m)
h=(m+1)\2
DO WHILE h>0
k=m-h
DO
switch=false
FOR i=1 TO k
IF x(i)>x(i+h)THEN
SWAP x(i),x(i+h)
switch=i
END IF
NEXT i
LOOP WHILE switch
h=h\2
LOOP
END SUB
```

[例 13-8] Quick 排序

算法设计：

算法的基本思想是利用分组概念把一个数组分而治之。因此，如何分组就成为这个算法中的一个重点。假设选其中一个元素作为分割点，把这个元素称为“比较子”（Comparand），然后把所有大于或等于比较子的元素放在一部分，小于比较子的元素放在另一部分，这样，数组就一分为二了。以后对每部分再分为两组，新分成的组又再一分为二，这样依此类推，一直到把整个数组排序完成为止。

如何选择比较子的问题，方法比较多，如取数组中间的元素作比较子；随机选取一个元素作比较子，取某一小部分元素的平均值作比较；甚至选一端的元素作比较子，这要看数组的具体情况而定。

这种快速排序的方法是一个递归进行的过程，所以完全可以用 Quick BASIC 的递归功能，这是很有用处的。

源程序：

```
'EXAMPLE 8
DECLARE SUB quicksort(sort$(),left%,right%)
DIM SHARED sort(1 to 50)
DEFINT AZ:CLS
DEF FNrndchr$(strng$)=MID(strng$,INT(strng$)*RND(1))+1,1)
letter$="abcdefghijklmnopqrstuvwxyz"
CLS
RANDOMIZE TIMER
INPUT "number of elements";count
FOR num=1 TO count
```

```
word$=“ ”
FOR addletter=1 TO 5
word$=word$+FNrndchr$(letter$)
NEXT addletter
sort$(num)=word$
PRINT word$
NEXT num
PRINT
LOCATE 24,2；PRINT“press a key to sort the words”
ky$=INPUT $(1)
LOCATE 24,2:PRINT SPACE$(30)：PRINT
CALL quicksort(sort$(),1,count)
PRINT
FOR num=1 TO count
PRINT sort$(num),
NEXT num
END
DEFINT AZ
SUB quicksort(sort$(),left%,right%)
IF right>left THEN
count1=left-1:count2=right
DO
DO
count1=count1+1
LOOP WHILE(sort $(count1)<sort$(right))
DO
countz=count2-1
LOOP WHILE sort$(count2)>sort$(right) AND count2>0
temp$=sort$(count1)
sort$(count1)=sort$(count2)
sort$(count2)=temp$
LOOP WHILE count2>count1
sort$(count2)=sort$(count1)
sort$(count1)=sort$(right)
quicksort sort$(),left,count1-1
quicksort sort$(),ount1+1,right
END IF
END SUB
```

13.2.6 模拟技术

计算机模拟就是利用计算机对一个现实系统的状态、功能，用一个简化的理想的模型去代替。人们用构造的模型来作试验，这是一种有用的研究方法，比如对卫生运行轨道的研究，河流泥沙淤积的研究、生态系统的研究、人口增长规律的研究等等。计算机模拟可以节约研究经费，缩短研究周期，不受空间限制，所得结果，在一定精度范围内能满足实际的需要。因此，获得了广泛的应用。

计算机模拟的关键是要对现实系统进行抽象，从而构造出相应的数学模型，有了数学模型，就可以用计算机来实现模拟。一般描述的数学模型可以分为连续系统模型和离散系统的模型。这里举两个例子：一个是确定

性模型，一个是概率型模型。

1．确定性模型

确定性模型的特点是模型中的各参数均取确定值，函数与自变量之间有一一对应的关系。

这里介绍一个简单的生态系统的模拟。

[例 13-9] 狐狸和野兔构成一个生态系统的模拟，狐狸捕食野兔，野兔吃草，假定地面的草是无限的。根据生存竞争的相互关系，如果狐狸太多，野兔被大量吃掉，以致野兔不能满足狐狸的需要，将会导致狐狸也要减少。相反，野兔如果很多，有利于狐狸的生长，因此狐狸数量将随之而增加。这种相互关系可以用一个数学模型来描述：

$$\frac{dr}{dt} = A * r - B * r * f$$

$$\frac{df}{dt} = C * r * f - D * f$$

式中，r 代表野兔的数量，f 代表狐狸的数量，A、B、C、D 是常数，其值为：

A＝0.5，B＝0.01，C＝0.001，D＝0.4

源程序：

```
'EXAMPLE 9
SCREEN 2,0:CLS
DEF FNdr(r,f)=a*f-b*r*f
DEF FNdf(r,f)=c*r*f-d*f
a=0.5
b=0.01
c=0.001
d=0.4
tmax=20
dt=0.1
FOR n=1 TO 4
INPUT “r,f=”;r,f
FOR t=1 TO tmax STEP dt
PSET(r,f)
r=r+FNdr(r,f)*dt
f=f+FNdf(r,f)*dt
NEXT t
NEXT n
END
```

2．概率性模拟

概率性模拟用于概率模型描述的系统。这种系统是一种随机系统。其特征是系统所处的状态是离散的而不是连续的。例如，车站的候车人数，电话总机处的电话呼叫次数，抛 1000 次硬币正反两面出现的次数等。要模拟这些事件的随机性，要用计算机中的 RANDOMIZE 函数来产生随机数。因此，概率模拟的实质就是用随机函数来模拟系统的随机性。

[例 13-10] 用蒙特卡洛法（Monte-Carlo Method）求π值。

蒙特卡洛方法是用计算机产生随机数进行数值估计的方法。

用此方法求π值的具体方法如下：产生若干组随机数 X 和 Y，它们的值均在（0，1）之间，且为正值，每一对 X，Y 对应一个点 T（X，Y），T 与原点的距离是 SQR（X*X+Y*Y）。若 T＜1，表示该点在半径 R＝1 的圆内。统计 N 组 X，Y 中有几组落在圆内。如有 M 组落在圆内，则第 1 象限中的 1/4 圆面积与小正方形面积之比为 M/N。小正方形面积为 1*1，整个大正方形面积为 4，因此圆面积应为 S＝4*M/N，见图 13-6。

源程序：

```
'EXAMPLE 10
'compute the PI with the Monte－carlo method
M=0
INPUT "How many spots";N
FOR I=1 TO N
X=RND；Y＝RND
IF X*X+Y*Y＜＝1 THEN M＝M+1
NEXT I
PRINT "PI=";M/N*4
END
RUN
N=?100
PI=3.36
N=?1000
PI=3.172
N=?2000
PI=3.17
N=?5000
PI=3.1488
N=10000
PI=3.1468
N=?100000
PI=3.1424
```

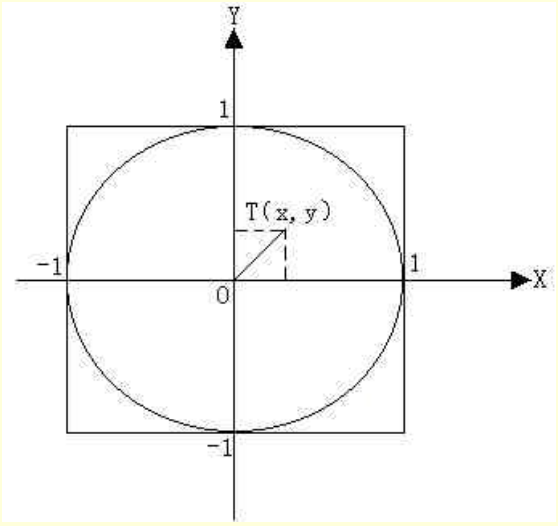


图 13-6　蒙特卡洛法求π值

可以看出：概率模拟得到的 PI 值只是一个估计值，而且每一次也不是确定的。

如果模拟的次数 N 愈大，则所得到的 PI 值也越接近其准确值。当然并不要用它来求π的精确值，而这种方法具有普遍意义，对不能求得精确解的随机问题十分有用。

习　题

1．采用结构化程序设计，应经历哪几个步骤？

2．用 Shell 排序法对 40 个学生的计算机应用基础课程成绩进行排序，要求按分数由高到低打印出各学生的学号和成绩。

3．模拟玩骰子的游戏，骰子是个正 6 面体，每面分别刻有 1、2、3、4、5、6。现有两个骰子，问出现两个“ 6 点 ”的机会是多少？

4．设计一个时钟计时显示程序，每秒显示一次时间，每隔 1 分钟发现一次响声（用 BEEP 语句或 PRINT CHR￥（7）发出音响）。

5．设计一个有陷阱的程序，用来处理“ 未打开打印机而用了 LPRINT 语句 ”的情况。未打开打印机的错误代码为 27。当发生此错误后，错误处理程序能给出有关信息，而后等待操作人员打开打印机并按任一键后恢复执行。

6．设计一个学生成绩管理系统的随机文件，用菜单方式给出功能提示。设有成绩输入、成绩打印、成绩统计、成绩更正 4 大功能。每个记录中包括学号、高等数学、英语、计算机应用基础 3 门课程的成绩。

附　　录

附录一　习题答案

第一章　练习题

1．

（157）$_{10}$=（10011101）$_{2}$　　（21.125）$_{10}$=（10101.001）$_{2}$

（1106）$_{10}$=（10001010010）$_{2}$　　（10.3）$_{10}$=（1010.0100011）$_{2}$

（216.005）$_{10}$=（11011000.0000000101）$_{2}$

2．

（01000）$_{2}$=（8）$_{10}$　　（1101）$_{2}$=（13）$_{10}$

（1101001）$_{2}$=（105）$_{10}$　　（100111101.1111）$_{2}$=（317.9375）$_{10}$

（110100111.001）$_{2}$=（423.125）$_{10}$

3．

（E）$_{16}$=（1110）$_{2}$　　（2C）$_{16}$=（1011000）$_{2}$

（FFFF）$_{16}$=（1111111111111111111）$_{2}$　　（A21）$_{16}$=（101000100001）$_{2}$

（BC.2A）$_{16}$=（101111100.0011）$_{2}$

4．

（1011）$_{2}$=（B）$_{16}$　　（11000101010）$_{2}$=（62A）$_{16}$

（100110001）2=（131）$_{16}$　　（11001000.001）2=（C8.2）$_{16}$

（1011110001.1）2=（2F1.8）$_{16}$

5．略

6．略

第二章　练习题

略

第三章　练习题

略

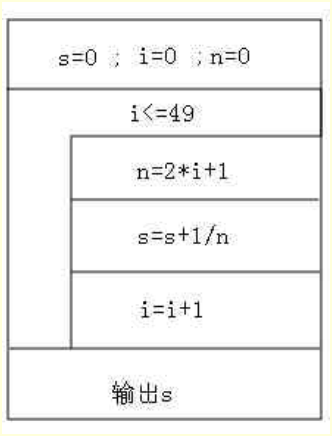
第四章　练习题

1．略

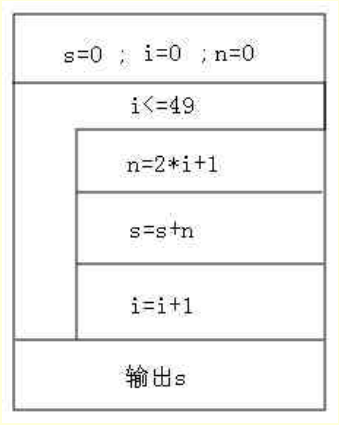
2．略

3．N-S 图为：

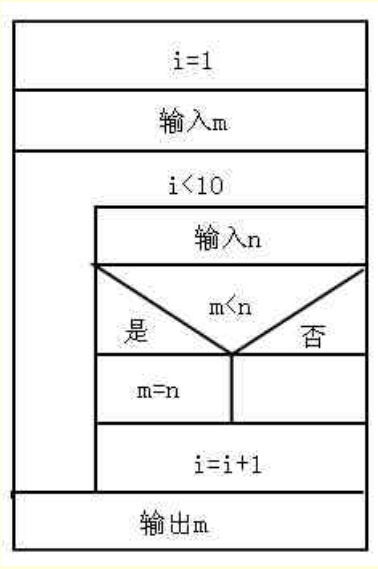
(1)



(2)



(3)



(4)

第五章　练习题

1．**略**

2．**略**

3．**程序为：**

```
REM 输出图案
CLS
xing$="*"
```

```
PRLNT TAB(20);xing$
PRLNT TAB(19);xing$;TAB(21);xing$
PRLNT TAB(18);xing;TAB(22);xing$
PRLNT TAB(17);xing;TAB(23);xing$
PRLNT TAB(16);"＊＊＊＊＊＊＊＊＊"
PRLNT TAB(17);"＊＊＊＊＊＊＊"
PRLNT TAB(18);"＊＊＊＊＊"
PRLNT TAB(19);"＊＊＊"
PRLNT TAB(20);"＊"
END
```

4．程序为：

```
PEM 计算购物总金额
CLS
INPUT"请输入第一种商品的单价、购买数量:",a1,a2%
INPUT"请输入第二种商品的单价、购买数量:",b1,b2%
INPUT"请输入第三种商品的单价、购买数量:",c1,c2%
x=al*a2%+b1*b2%+c1*c2%
PRINT"总金额为: "x;"元"
END
```

5．程序为：

```
CLS
INPUT"请输入单价为 2.50 元的读物册数: ";a
INPUT"请输入单价为 3.20 元的读物册数: ";b
INPUT"请输入单价为 3.98 元的读物册数: ";c
x=2.50*a+3.20*b+3.98*c
PRINT"购买这些读物，共需";x;"元"
END
```

6．程序为：

```
REM 摄氏温度向华氏温度转换
CLS
INPUT"请输入一个摄氏温度: ",ss
hs=ss*5/9+32
PRINT"华氏温度为: "hs
END
```

7．程序为：

```
REM 计算鸡兔只数
CLS
h=71
f=158
x=(4*h-f)/2
y=(f-2*h)/2
PRINT"共有鸡: "x;"只","共有兔: ";y;"只'
```

```
END
```

8．程序为：

```
CLS
PRINT TAB(10);"姓名",TAB(20);"性别",TAB;"年龄",TAB(40);"工作单位"
PEAD xm1,xb1,n11,gzdw1
PRIN TAB(10); xm1,TAB(20);xb1,TAB(30);n11,TAB(40);gzdwl
READ xm2,xb2,n12,gzdw2
PRINT TAB(10);xm2,TAB(20);xb2,TAB(30);n12,TAB(40);gzdw2
PEAD xm3,xb3,n13,gzdw3
PRINT TAB(10);xm3,TAB(20);xb3,TAB(30);n13,TAB(40);gzdw3
READ xm4,xb4,n14,gzdw4
PRINT TAB(10)l;xm4,TAB(20);xb3,TAB(30);n14,TAB(40);gzdw4
DATA  张红，男，27，图片工作室
DATA  李骊，女，34，北京大学
DATA  刘英，女，28，地质学院
DATA  张晴，男，30，国家物勘局
END
```

9．程序为：

```
REM  求最佳付款方案
DEFLNG A-Z
CLS
INPUT"应提取存款数额（单位为分）x=",x      'x 为钱款数量，单位：分，如 158816
yuan100=x\10000                          '求百元票张数,100 元=10000 分
x=x-10000*yuan100                        '求剩余款额
yuan50=x\5000                            '求伍拾元票张数,50 元=5000 分
x=x-5000*yuan50                          '求剩余款额
yuan10=x\1000                            '求拾元票张数,5 元=500 分
x=x-1000*yuan10                          '求剩余款额
yuan5=x\500                               '求伍元票张数,5 元=500 分
x=x-500*yuan5                             '求剩余款额
yuan2=x\200                               '求贰元票张数，2 元=200 分
x=x-200*yuan2                             '求剩余款额
yuan1=x\100                               '求壹元票张数，1 元＝100 分
x=x-100*yuan1                             '求剩余款额，单位：分
jiao5=x\50                                '求伍角票张数，5 角＝10 分
x=x-50*jiao5                              '求剩余款额
jiao2=x\20                                '求贰角票张数，2 角＝20 分
x=x-20*jiao2                              '求剩余款额
jiao1=x\10                                '求壹角票张数，1 角＝10 分
x=x-10*jiao1                              '求剩余款额
fen=x\5                                    '求伍分票张数
x=x-5*fen5                                 '求剩余款额
fen2=x\2                                   '求贰分票张数
x=x-2*fen2                                 '求剩余款额
```

```
fen1=x                                          '求壹分票张数
PRINT"应付 100 元票: ";yuan100;"张"
PRINT"应付 50: ";yuan50;"张"
PRINT"应付 10 元票: ";yuan10;"张"
PRINT"应付 5 元票: ";yuan5;"张"
PRINT"应付 2 元票: ";yuan2;"张"
PRINT"应付 1 元票: ";yuan1;"张"
PRINT
PRINT"应付 5 角票: ";jiao;"张"
PRINT"应付 2 角票: ";jiao;"张"
PRINT"应付 1 角票: ";jiao;"张"
PRINT
PRINT"应付 5 分票: ";fen5;"张"
PRINT"应付 2 分票: ";fen2;"张"
PRINT"应付 1 分票: ";fen1;"张"
END
```

第六章　练习题

1．略

2．略

3．程序为：

```
INPUT"请输入 x 的值:",x
IFx>=0 THWN
y=1+x
ELSE
y=1-2*x
ENDIF
PRINT"y=";y
END
```

4．程序为：

```
INPUT"请输入 3 个数 a,b,c=";a,b,c
IF ABS(a)>ABS(b) ABD ABS(a)>ABS(c)THENT"绝对值最大者为:";a
IF ABS(b)>ABS(a) AND ABS(b)>ABS(c)THINT"绝对值最大者为:";b
IF ABS(c)>ABS(a) AND ABS(c)>ABS(b)THINT"绝对值最大者为:";c
```

5．程序为：

```
REM  计算 y 值
INPUT "x=";x
IF x<0 THEN
y=LOG(-x)
ELSE
IF x<5 THEN
y=x+LOG(5+x^3)
ELSE
```

```
y=EXP(x)
END IF
END IF
PRINT "y=";y
END
```

6．程序为：

```
REM 计算邮费
INPUT"请输入邮包重量：",w
INPUT"请输入邮包个数：",gs%
PRINT"需作快件投递吗？";
PRINT"(输入非零数表示需快递，输入 0 表示不快递)"
INPUT kd%
IF w<=20 THEN
x=w*0.85
ELSE
x=20*0.85+(w-20)*(0.85+0.15)
END IF
x=x+gs%*0.5
IF kd% THEN
x=x*(1+0.2)
END IF
PRINT"应付邮费:";x;"元"
END
```

7．程序为：

```
REM 三个数排序
INPUT"a=";a
INPUT"b=";b
INPUT"c=";c
IF a>b AND a>c THEN
IF c>b THEN SWAP b,c
END IF
IF b>a AND b>c THEN
IF a>c THEN
SWAP a,b
ELSE
SWAP a,b
SWAP b,c
END IF
END IF
IF c>a AND c>b THEN
IF b>a THEN
SWAP a,c
ELSE
SWAP a,b
```

```
SWAP a,c
END IF
END IF
PRINT a,b,c
END
```

8．程序为：

```
INPUT"请输入两个数 a,b(b 不能为 0):"a,b
INPUT"请输入一个运算符(+.-.*./):",ysf$
SELECT CASE ysf$
CASE"+"
y=a+b
PRINT"和为";y
CASE"_"
y=a-b
PRINT"差为";y
CASE"*"
y=a*b
PRINT"积为";y
CASE "/"
y=a/b
PRINT"商为";y
END SELECT
END
```

9．程序为：

```
REM 判断能否获得一等奖学金
INPUT"请输入 6 门课程的成绩:",cj1,cj2,cj3,cj4,cj5,cj6
cj=cj1+cj2+cj3+cj4+cj5+cj6
tj1=cj1>85 AND cj2>85 AND cj3>85 AND cj4>85 AND cj5>85 AND cj6>85
IF tj1 AND cj>520 THEN
PRINT"能获得一等奖学金！"
ELSE
PRINT"不能获得一等奖学金！"
END IF
END
```

第七章　练习题

1．略

2．程序为：

```
CLS
s%=0
FOR x%=3 TO 100 STEP 2
PRINT x%,
s%=s%+x%
NEXT x%
```

```
PRINT"3~100 之间的奇数和为:";s%
END
```

3．程序为：

```
i=1                          '计数器赋初值
INPUT"输入非零数个数 N＝，n
DO WHILE i<=n
INPUT"请输入非零数:",fls
SELECT CASE fls
CASE IS>0
zs=zs+1                      '正数个数累加
zsh=zsh+fls                  '正数和累加器累加
CASE 0
EXIT DO
CASE IS＜0
fs=fs+1                      '负数个数累加
fsh=fsh+fls                  '负数和累加器累加
END SELECT
i=i+1                        '计数器累加 1
LOOP
PRINT"正数个数:";zs,"正数和:",zsh
PRINT"负数个数:";fs,"负数和:";fsh
END
```

4．程序为：

```
CLS
FOR i=1 TO 5
PRINT TAB(6-i);
FOR j=1 TO 2*i-1
PRINT "*";
NEXT j
PRINT
NEXT i
END
```

5．程序为：

```
CLS
FOR i=1 TO 6
FOR j=6 TO i STEP-1
PRINT"*";
NEXT j
PRINT
NEXT i
END
```

6．**程序为：**

```
CLS
FOR i=-3 TO 3
PRINT TAB(ABS(i)+2);
FOR j=1 TO 7-2*ABS(i)
PRINT "*";
NEXT j
PRINT
NEXT i
END
```

7．**程序为：**

```
x=1
FOR n%=1 TO 2 STEP-1
x=(x+1)*2
NEXT n%
PRINT"第一天共摘";x;"只桃子"
END
```

8．**程序为：**

```
PRINT"公鸡","母鸡","小鸡"
FOR x%=1 TO 19
FOR y%=1 TO 33
z%=100-x%-y%
IF 15*x%+9*y%+z%=300 THEN PRINT x%,y%,z%
NEXT y%
NEXT x%
END
```

9．**程序为：**

```
FOR x%=1 TO 100
j%=10
IF x%>=10 THEN j%=100
IF x%>=100 THEN j%=1000
y%=x%^2
IF y% MOD j%=x% THEN PRINT x%,y%
NEXT x%
END
```

10．**程序为：**

```
f1=1
f2=1
PRINT f1,f2,
FOR i%=3 TO 20
f3=f1+f2
PRINT f3,
```

```
f1=f2
f2=f3
NEXT i%
END
```

第八章　练习题

1．程序为：

```
CLS
DIM a(100,100)
INPUT"输入矩阵行数：",m
INPUT"输入矩阵列数：",n
FOR i=1 TO m                          '行数控制
FOR j=1 TO n                          '列数控制
PRINT"输入第";i;"行、第";j;"列的元素：";
INPUT a(i,j)
NEXT j
NEXT I
max=a(1,1)                            '设 a(1,1)为初始最大数
p1=1:p2=1
FOR i=1 TO m                          '行数控制
FOR j＝1 TO n                         '列数控制
IF max<a(i,j)THEN
max=a(i,j)                            '求得当前最大数
p1=i:p2=j                             '标记当前最大数的位置
END IF
NEXT j
NEXT i
PRINTF"最大元素为：";max"在第";p1;"行、第"p2;"列"
END
```

2．程序为：

```
REM  求获奖学生
DIM name$(10),number$(10),points(10)      '定义姓名、学号、成绩数组
FOR i=1 TO 10                             '输入各学生数据
PRINT"输入第";i;"个学生的姓名：",
INPUT name$(i)
PRINT "输入第";i;"个学生的学号：",
INPUT number$(i)
PRINT"第";i;"个学生的成绩
",
INPUT points(i)
s=s+points(i)                             '求总成绩
NEXT i
s=s/10                                    '求全班平均成绩
first=s * 1.1                             '求一等奖标准
second=s*1.05                             '求二等奖标准
```

```
FOR i=1 TO 10
IF points(i)>first THEN
PRINT name$(i);number$(i);points(i);"获一等奖"
ELSEIF points(i)>second THEN
PRINT name$(i);number$(i);points(i);"获二等奖"
END IF
NEXT i
END
```

3．程序为：

```
REM 求转置矩阵
CLS
INPUT"请输入行数 N＝";n
INPUT"请输入列数 M=";m
DIM a(1 TO n,1 TO m),b(1 TO m,1 TO n)
FOR i=1 TO n
FOR j=1 TO m
PRINT"输入第";i;"行，第";j;"列的数值：";
INPUT a(i,j)
NEXT j
NEXT i
PRINT"矩阵 A 为："
FOR i=1 TO n
FOR j=1 TO m
PRINT TAB(4*j);a(i,j);
b(j,i)=a(i,j)
NEXT j
NEXT i
PRINT"A 的转置矩阵 A 为："
FOR i=1 TO n
FOR j=1 TO m
PRINT TAB(4*j);b(i,j);
NEXT j
PRINT
NEXT i
END
```

4．程序为：

```
DIM  tv(5,3),s(3),t(5)
REM   输入电视机销售量，电视机价格
PRINT"请按输入 5 个百货公司的 3 个牌号电视机的销售量"
FOR i=1 TO 5
FOR j=1 TO 3
INPUT tv(i,j)
NEXT j
NEXT i
```

```
PRINT"请输入 3 个牌号电视机的价格"
INPUT"兰花牌电视机的价格：",s(1)
INPUT"梅花牌电视机的价格：",s(2)
INPUT"菊花牌电视机的价格：",s(3)
REM  统计营业额
FOR i=1 TO 5
t(i)=tv(i,j)*s(j)+t(i)
NEXT j
NEXT i
REM  输出输入的数据和统计的结果
PRINT
PRINT"                百货公司电视机销售量及营业额"
PRINT"百货公司","兰花牌","菊花牌","销售额（元）"
FOR i=1 TO 5
PRINT"第";i;"百货公司",tv(i,1),tv(i,2),tv(i,3),t(i)
NEXT i
PRINT
PRINT"兰花牌","梅花牌","菊花牌"
PRINT s(1),s(2),s(3)
END
```

5．程序为：

```
REM  统计人口数
CLS
DIM a(150)                          '定义数组
z=0
FOR x=1 TO 150                      '各年龄人口数初值为 0
a(x)=0
NEXT x
INPUT"请输入一个年龄：",x
DO WHILE x<>z                       '当该地区人口尚未统计完毕时
a(x)=a(x)+1
INPUT"请输入一个年龄：",x
LOOP
FOR x=1 TO 69                       '各年龄组人数输出处理
PRINT x;"岁年龄组人数：";a(x)       '输出 x 岁年龄组的人数
NEXT x
SLEEP                               '暂停，分页输出
FOR x=70 TO 139
PRINT x;"岁年龄组人数：";a(x),
NEXT x
END
```

6．程序为：

```
REM   求众数
CLS
```

```
DIM a(2000),b%(2000)                    '定义实数存放数组和次数统计数组
INPUT"请输入实数个数（1~2000）：",n
FOR i=1 TO n
INPUT"请输入实数：",a(i)
b%(i)=1                                 '次数统计初值
NEXT i
FOR i=1 TO n-1                          '各数出现次数统计
FOR j=i+1 TO n                          'a(i)之后各实数位置控制
IF a(i)=a(j)THEN b%(i)=b%(i)+1
NEXT j
NEXT i
max=b%(1)
place=1
FOR i=2 TO n                            '选取出现次数最大者
IF b%(i)>max THEN
max=b%(i)
place=i
END IF
NEXT i
pinlv=b%(place)/n
PRINT"众数为：";a(place);"众数次数：";b%(place);
PRINT"众数频率：";pinlv;"首次出现位置：";place
END
```

7．程序为：

```
REM  平均成绩名次表
CLS
DIM xh$(100),cj(100),xm$(100)           '定义学号、平均成绩、姓名数组
INPUT"请输入学生人数（1~100）：";m
INPUT"请输入课程门数（1~20）：";n
FOR i=1 TO m                            '学生人数控制
PRINT"输入第";i;"个学生姓名：";
INPUT xm$(i)                            '输入第 i 个学生姓名
PRINT"输入第";i;"个学生学号：";
INPUT xh$(i)                            '输入第 i 个学生学号
zf=0                                    '每个学生总分赋初值
FOR j=1 TO n                            '课程门数控制
PRINT "输入第";j;"个成绩：";
INPUT x                                 '输入第 i 个学生第 j 门课程分数
zf=zf+x                                 '求第 i 个学生总分
NEXT j
cj(i)=zf/n                              '求第 i 个学生平均成绩
NEXT i
FOR i=1 TO m                            '对平均成绩排序
p=i                                     '择换法排序（从大到小）
FOR j=i+1 TO m
```

```
IF cj(p)<cj(j) THEN p=j
NEXT j
IF p<>i THEN
SWAP xm$(p),xm$(i)
SWAP xh$(p),xh$(i)
SWAP cj(p),cj(i)
END IF
NEXT i
j=1:i=1                                    '标记第 1 名
PRINT"名次","姓名","学号","平均成绩"
PRINT j,xm$(i),xh$(i),cj(i)                '第 1 名信息输出
FOR i=2 TO m                               '按名次排列输出处理
IF cj(i)<>cj(i-1) THEN j=j+1
PRINT j,xm$(i),xh$(i),cj(i)                '第 j 名信息输出
NEXT i
END
```

8．程序为：

```
REM 统计旅客人数，查询某一房间住宿数
CLS
DIM a(3,2,5)
'三维数组的输入处理
FOR i=1 TO 3                               '楼层控制
FOR j=1 TO 2                               '排数控制
FOR k=1 TO 5                               '房间控制
PRINT"输入第";i;"层、第";j;"排、第";k;"号房间的旅客人数：";
INPUT a(i,j,k)
NEXT k
NEXT j
NEXT i
'住宿旅客人数统计处理
zrs=0
FOR i=1 TO 3
FOR j=1 TO 2
FOR k=1 TO 5
zrs=zrs+a(i,j,k)                           '逐层逐排逐号房间住宿人数统计
NEXT k
NEXT j
NEXT i
PRINT"总人数为：";zrs
'查询某房间旅客人数处理
DO
PRINT"你要查询某一房间的旅客人数吗？（输入 0 或负数结束查询）"
PRINT"输入需要查询的楼号："
INPUT i
PRINT"输入需要查询的排号："
```

```
INPUT j
PRINT"输入需要查询的房号：";
INPUT k
PRINT"现在该房间旅客人数为：";a(i,j,k)
LOOP UNTIL i<=0 OR j<=0 OR k<=0
END
```

第九章　练习题

1．程序为：

```
DECLARE FUNCTION age(n%)
PRINT age(5)
END
FUNCTION age(n%)
IF n%=1 THEN
age=10
ELSE
age=age(n%-1)+2
END IF
END FUNCTION
```

2．主程序如下：

```
DECLARE SUB zdgys(a,b,c)
INPUT "m,n=";m,n
CALL zdgys(m,n,c)
PRINT"其最大公约数为：",c
END
```

子程序 zdgys 为：

```
SUB zdgys(a,b,c)
IF b<>0 THEN
CALL zdgys(b,a MOD b,c)
ELSE
c=a
END IF
END SUB
```

3．主程序为：

```
DECLARE SUB sort(x!(),n%)
CLS
DIM x1(1 TO 10)
RANDOMIZE TIMER
FOR i=1 TO 10
x1(i)=RND
NEXT i CALL sort(x1(),10)
FOR i=1 TO 10
PRINT x1(i)
NEXT i
```

```
END
```

子程序为：

```
SUB sort(x(),n%) STATIC
FOR i=1 TO n%-1
FOR j=i+1 TO n%
IF x(i)<x(j) THEN SWAP x(i),x(j)
NEXT j
NEXT i
END SUB
```

4．程序如下：

```
DECLARE FUNCTION gys%(m%,n%)
INPUT"输入三个整数 a,b,c=";a%,b%,c%
h%=gys(a%,b%)
h%=gys(h%,c%)
PRINT"最大公约数为:";h%
END
```

自定义函数 gys 为:

```
FUNCTION gys%(m%,n%)
IF m%<n% THEN SWAP m%,n%
r%=m% MOD n%
DO WHILE r%<>0
m%=r%
r%=m% MOD n%
LOOP
gys%=n%
END FUNCTION
```

5．程序为：

```
DECLARE FUNCTION gys(m,n)
DIM SHARED a,b,gbs
INPUT"请输入 a,b=";a,b
m=a:n=b
PRINT"最大公约数为:";gys(m,n)
PRINT"最小公倍数为:";gbs
END
```

自定义函数 gys 为:

```
FUNCTION gys(m,n)
IF m<n THEN SWAP m,n
r=m MOD n
DO WHILE r<>0
m=n
n=r
r=m MOD n
LOOP
gys=n
```

```
gbs=a*b/n
END FUNCTION
```

6．**函数** ymj **为**:

```
FUNCTION ymj(r) STATIC
a=3.14159*r^2
n=n+1
PRINT"调用次数 n=";n,"面积为:";a
ymj=a
END FUNCTION
```

主程序调用函数 8 **次，主程序为：**

```
DECLARE FUNCTION ymj!(r!)
CLS
sum=0
FOR i=1 TO 8
s=ymj(i)
sum=sum+s
NEXT i
PRINT"圆面积和为:";sum
END
```

7．**自定义函数为**:

```
FUNCTION day$(n%)
SELECT CASE n%
CASE 1
day$="星期一:Monday"
CASE 2
day$"星期二:Tuesday"
CASE 3
day$"星期三:Wednesday"
CASE 4
day$"星期四:Thursday"
CASE 5
day$"星期五:Friday"
CASE 6
day$"星期六:Saturday"
CASE 0
day$"星期日:Sunday"
CASE ELSE
day$"输入错误!"
END SELECT
END FUNCTION
```

8．**主程序为**:

```
DECLARE FUNCTION lettercnt%(st$,letter$)
CLS
```

```
str 1$="SELECT CASE-CASE 1-CASE 2-CASE 3-END SELECT"
str 2$="E"
PRINT lettercnt%(str 1$,str 2$)
END
```

自定义函数为:

```
FUNCTION lettercnt%(st$,letter$)
cnt%=0
IF LEN(letter$)>1 THEN letter$=LEFT$(letter$,1)
FOR c%=1 TO LEN(st$)
IF MID$(st$,c%,1)=letter$ THEN cnt%=cnt%+1
NEXT c%
lettercnt%=cnt%
END FUNCTION
```

第十章　练习题

1．程序为：

```
PRINT "请按门牌号、路名、城市名输入地址:";
LINE INPUT x$
y=INT(VAL(x$)/2)*2
IF y=val(x$) THEN
PRINT"在路南"
ELSE
PRINT"在路北"
END IF
END
```

2．主程序为：

```
DECLARE SUB dap 1(n)
DECLARE SUB dap 2(n)
INPUT "请输入一个字符串:"x$
m=LEN(x$)
FOR i=1 TO m
n=ASC(MID$(x$,i,1))
IF i MOD 2<>0 THEN
CALL dap 1(n)
ELSE
CALL dap2(n)
END IF
MID$(x$,i,l)=CHR$(n)
NEXT i
PRINT x$
END
```

子程序 dap 1 **为：**

```
SUB dap 1(n)
IF(N>=97 AND n<=122)OR(n>=65 AND n<=90) THEN
n=n+2
```

```
IF(n>90 AND n<=92) OR n>122 THEN n=n-26
END IF
END SUB
```

子程序 dap 2 **为：**

```
SUB dap 2(n)
IF(n>=97 AND n<=92)OR n>122 THEN n=n-26
END IF
END SUB
```

子程序 dap 3 **为：**

```
SUB dap 2(n)
IF(n>=97 AND n<=122) OR (n>=65 AND ＜＝90) THEN
n=n-2
IF n<65 OR (n<97 AND n>=95) THEN n=n+26
END IF
END SUB
```

3．**程序为：**

```
x$="You are a student"
PRINT"未改动前的字符串:";x$
x1$=LEFT$(x$,8)+MID$(x$,10)
PRINT"删除第 9 个字符 a 后的字符串:";x1$
x2$=LEFT$(x$,8)+"two"+MID$(x$,10)
PRINT"在 a 原来位置加入 two 后的字符串:";x2$
x3$=x2$+"s."
PRINT"在尾部加入 s.后的字符串:";x3$
END
```

4．**程序为：**

```
x$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
y$="FLQCOSNZEYVRXHTBWAJDMIGUPK"
INPUT"请输入任一密码串：",z$
FOR i=1 TO LEN(z$)                    '将该密码从左到右逐个处理
a$=MID$(z$,i,1)                       '取出第 i 个字符放入 a$中
j=INSTR(x$,a$)                        '寻找密码字符在 x$中的位置
IF j=0 THEN
PRINT a$;                             '若没找到，显示原字符
ELSE
PRINT MID$(y$,j,1);                   '若找到，转换成明码输出
END IF
NEXT i
END
```

5．**程序为：**

```
CLS
DO UNTIL r>0 AND r<24                 '输入的行数应为 1 到 23
INPUT"请输入行数（1-23）:",r
```

```
LOOP                                    '否则会重复执行
WHILE n=0
INPUT"请输入任一字符串：",x$
n=LEN(x$)
WEND
CLS
c=INT((80-n)/2)                         '计算屏幕上显示字符串起始域
LOCATE r,c                              '将光标移到第 r 行第 c 列
PRINT x$
END
```

第十一章　练习题

1．程序为：

```
REM 定义记录
TYPE studentrec
id AS LONG
names AS STRING*6
chi AS INTEGER
eng AS INTEGER
mat AS INTEGER
avg AS SINGLE
END TYPE
REM 定义记录变量 stu
DIM stu AS studentrec
CLS
PRINT STRING$(30,"*")
PRINT"学号      姓名      语言      英语      数学      平均分数"
f$="######\        \###     ###     ###     ###.#"
FOR i=1 TO 5
READ stu.id,stu.names,stu.chi,chi.stu.eng,stu.mat
stu.avg=(stu.chi+stu.eng+stu.mat)/3
PRINT USING f$;stu.names;stu.chi;stu.eng;stu.mat;stu.avg
NEXT i
PRINT STRING$(30,"*")
DATA 2000101,"刘小雨",92,89,86
DATA 2000102,"丁蓉蓉",100,90,91
DATA 2000103,"张芳艳",76,72,90
DATA 2000104,"杜无庸",68,78,69
DATA 2000105,"孙　强",85,79,65
END
```

2．程序为：

```
REM 定义记录
TYPE studentrec
stunum AS STRING *6
names AS STRING*8
```

```
avg AS SINGLE
END TYPE
REM 假设有 5 个学生
n=5
REM 定义记录数组 stu
DIM stu(1 TO n) AS studentrec
REM 把数据读入记录数组中
FOR i=1 TO n
READ stu(i).stunum
READ stu(i).names
READ stu(i).avg
NEXT i
REM 显示排序前记录数组中的内容
CLS
PRINT"排序前记录数组中的内容"
PRINT STRING$(22,"8")
FOR i=1 TO n
PRINT stu(i).stunum,stu(i).names,stu(i).avg
NEXT i
REM 冒泡排序法
FOR i=1 TO n-1
FOR j=1 TO n-i
IF stu(j).avg<stu(j+1).avg THEN
SWAP stu(j),stu(j+1)
END IF
NEXT j
NEXT i
PRINT
REM 显示排序后记录数组的内容
PRINT"排序后记录数组的内容"
PRINT STRING$(22,"*")
FOR i=1 TO n
PRINT stu(i).stunum,stu(i).names,stu(i).avg
NEXT i
DATA 20001,"刘小雨",92
DATA 20002,"丁蓉蓉",100
DATA 20003,"张艳芳",76
DATA 20004,"吴小丽",68
DATA 20005,"许大强",85
END
```

3．程序为；

```
REM 将基本情况输入 zgqk.dat 顺序文件
OPEN"zgqk.dat"FOR OUTPUT AS #1
FOR i=1 TO 10
READ num,name$,sex$,mon
```

```
WRITE # 1,num,name$,sex$,mon
NEXT i
CLOSE #
DATA 98001,Wang,f,543,98002,Li,t,654
DATA 98011,Huang,t,443,98012,Liu,f,554
DATA 98043,Zhang,f,678,98054,Chen,t,456
DATA 98064,Chang,t,576,98078,Gou,f,556
DATA 98084,Ding,f,789,98097,Sun,f,456
REM 从 zgqk.dat 中读入全部职工
DIM num(10),name$(10),sex$(10),mon(10)
OPEN"zgqk.dat" FOR INPUT AS #1
FOR i=1 TO 10
INPUT #1,num(i),name$(i),sex$(i),mon(i)
NEXT i
REM 将女职工存入 zgqknv.dat 中
OPEN "zgqknv.dat" FOR OUTPUT AS #2
FOR i=1 TO 10
IF sex$(i)="f" THEN WRITE #2,num(i),name$(i),sex$(i),mon(i)
NEXT i
CLOSE #1,#2
END
```

4．程序为：

```
TYPE goods
xm AS STRING *6
xb AS STRING *2
nl AS INTEGER
fs AS SINGLE
END TYPE
DIM da(5) AS goods
OPEN"xsda.dat" FOR RANDOM AS #1 LEN=LEN(da(1))
n=LOF(1)/LEN(da(1))
FOR i=1 TO n
GET #1 i,da(i)
NEXT i
FOR i=1 TO n-1
k=i
FOR j=i+1 TO n
IF da(k).fs<da(j).fs THEN k=j
NEXT j
IF k<>j THEN SWAP da(i),da(k)
NEXT i
FOR i=2 TO n
IF da(1).fs-da(i).xm,da(i).xb,da(i).nl,da(i).fs
END IF
NEXT i
```

```
CLOSE #1
END
```

5．程序为：

```
DEFLNG A-Z
OPEN "bxjf.dat"FOR INPUT AS #1
DO WHILE NOT EOF(1)
count=count+1
INPUT #1,xm$,je
total=total+je
LOOP
PRINT"共有报销单据";count;"张","总金额为";total;"分"
END
```

6．程序为：

```
CLS
OPEN "xscj.dat" FOR INPUT AS #1          '学生成绩文件
OPEN "xs-girl.dat" FOR OUTPUT AS #2      '要建立的女生文件
'  建立女生文件
PRINT"显示女生记录"
DO WHILE NOT EOF(1)
INPUT #1,xh$,xm$,xb$,yw,wl,zz,yy
IF xb$="女" THEN
WRITE #2,xh$,xm$,xb$,yw,sx,wl,zz yy
PRINT xh$;TAB(8);xm$;TAB(16);xb$;
'  显示女生记录
PRINT TAB(20);yw;TAB(30);sx;TAB(40);wl;TAB(50);zz;TAB(60);yy
END IF
LOOP
CLOSE #2
PRINT"显示排序前的平均成绩"
'  计算平均成绩、排序并建立文件
DIM xh$(1 TO 100),xm$(1 TO 100),xb$(1 TO 100),yw(1 TO 100)
DIM sx(1 TO 100),wl(1 TO 100),zz(1 TO 100),yy(1 TO 100),yy(1 TO 100),pj(1 TO 100)
SEEK #1,1                                '记录指针指向文件
i=0                                      '从磁盘读入数组以便排序
DO WHILE NOT EOF（1）
i=i+1
INPUT #1,xh$(i),xm$(i),xb$(i),yw(i),sx(i),wl(i),zz(i),yy(i)
pj(i)=(yw(i)+sx(i)+wl(i)+zz(i)+yy(i))/5
PRINT xh$(i);TAB(8);xm$(i);TAB(16);xb$(i);
'  显示排序前的记录
PRINT TAB(20);yw(i);TAB(30);sx(i);TAB(40);wl(i);
PRINT TAB(50);zz(i);TAB(60);yy(i);TAB(70);pj(i)
LOOP
PRINT"显示排序后的记录"
```

```
OPEN "xs-sort.dat"FOR OUTPUT AS #3          '排序后建立的文件
n=i                                         '按平均成绩从高到低排序
FOR i=1 TO n-1
FOR j=i+1 TO n
IF pj(j)>pj(i) THEN
SWAP xh$(j),xh$(i):SWAP xm$(j),xm$(i):SWAP xb$(j),xb$(i)
SWAP yw(j),yw(i):SWAP sx(j),sx(i):SWAP wl(j),wl(i)
SWAP zz(j),zz(i):SWAP yy(j),yy(i):SWAP pj(j),pj(i)
END IF
NEXT j
'    写入一条记录
WRITE #3，xh$(i),xb$(i),yw(i),wl(i),zz(i),yy(i),pj(i)
PRINT xh$(i);TAB(8);xm$(i);TAB(70);pj(i)
NEXT i
'  写入最后一条记录
WRITE #3,xh$(i),xm$(i),xb$(i),yw(i),sx(i),wl(i),zz(i),yy(i),pj(i)
PRINT xh$(i);TAB(8);xm$(i);TAB(16);xb$(i);
PRINT TAB(20);yw(i);TAB(30);sx(i);TAB(40);wl(i);
PRINT TAB(50);zz(i);TAB(60);yy(i);TAB(70);pj(i)
CLOSE #3
END
```

第十二章　练习题

2．程序如下：

```
DO
SCREEN 13
CLS
FOR i%＝1 TO 100
x%=RND*350
y%=RND*200
PSET(x%,y%)                 '画星星
CIRCLE(200,50),15,14,4.2,2  '画月亮
CIRCLE(195,50),14           '给月亮填色
NEXT i%
FOR i%=1 TO 100             '用空循环实现延时
NEXT i%
LOOP UNTIL INKEY$<>""       '按任一键结束
END
```

3．程序为:

```
SCREEN 7
CLS
FOR i=1 TO 5                '画 5 个同心圆
CERCLE(150,100),10*i,3
NEXT i
FOR i=1 TO 4                '涂上不同的颜色
```

```
PAINT(165+10*(i-1),100)i,3
NEXT i
END
```

4．主程序为：

```
DECLARE SUB hua(R!,x0!,y0!,c!)
DECLARE SUB mo(R!,x0!,y0!,c!)
SCREEN 12
CLS
DO
r=20
c=1
x0=320
y0=200
FOR i=1 TO 15
CALL hua(r,x0,y0,c)                    '调用画圆子程序
CALL mo(r,x0,y0,c)                     '调用抹圆子程序
r=r+i
c=c+1
NEXT i%
LOOP UNTIL INKEY$<>""                  '按任一键结束
END
```

抹圆子程序为:

```
SUB mo(R,x0,y0,c)
FOR b=0 TO 2*3.14 STEP 0.01
x=COS(b)*r+x0
y=SIN(b)*r+y0
FOR m=0 TO 10
preset (x+m,y+m)
NEXT m
NEXT b
END SUB
```

画圆子程序为:

```
SUB hua(R,x0,y0,c)
FOR a=0 TO 2*3.14 STET 0.01
x=COS(a)*r+x0
y=SIN(a)*r+y0
FOR j=0 TO 10
PSET (x+j,y+j),c
NEXT j
NEXT a
END SUB
```

5．程序为：

```
SCREEN 2                               '设置屏幕方式
WINDOW(-30,-30)-(300,120)
```

```
LINE(0,-10)-(0,100)              '画 Y 轴
LINE(-10,0)-(280,0)              '画 X 轴
'    标 Y 轴的刻度
LOCATE 5,20
PRINT"成绩分布比例情况"
LOCATE 4,3:PRINT"100%_"
LOCATE 8,3:PRINT"80 _"
LOCATE 11,3:PRINT"60 _"
LOCATE 15,3:PRINT"40 _"
LOCATE 18,3:PRINT"20 _"
FOR x=30 TO 260 STEP 60       '画条形
READ D$,M
LINE(x-22,0)-(x-2,M),,BF
LOCATE 23,(x \ 4+5):PRINT D$     '标 X 轴的刻度
NEXT x
DATA<60,10
DATA 60-70,20
DATA 70-80,50
DATA 80-100,20
END
```

用饼形图开画出各范围内的分布情况，程序如下

```
SCREEN 2
WINDOW(0,0)-(300,200)
PRINT"成绩分布比例情况"
REVF＝2*3。14159
CIRCLE(75,100)-(125,100)
angle=0
FOR C=1 TO 4
READ d$,M
angle=angle+(M/100)*REV
X1=50*COS(angle)
Y1=50*SIN(angle)
LINE(75,100)-(75+X1*23/25,100+Y1*23/25)
NEXT C
LOCATE 13,13:PRINT"80-100"
LOCATE 12,28:PRINT"<60"
LOCATE 16,25:PRINT"60-70"
LOCATE 10,23:PRINT"70-80"
DATA<60,10
DATA 60-70,20
DATA 70-80,50
DATA 80-100,20
END
```

6．先画出一条帆船，程序为:

```
REM  画一条帆船
```

```
CLS
SCREEN 1,0
DRAW"L60 E60 D80 L60 F20 R40 E20 L20"
END
```

如果要逆时针旋转 45 **°，只需要设置角度，可用** TAn **来实现。**

修改后的程序为:

```
REM  逆时针旋转 45 °后的帆船
CLS
SCREEN 1,0
DRAW"TA45 L60 E60 D80 L60 F20 R40 E20 L20"
END
```

如果要画出 3 **条大小不同的帆船，可以利用比例因子** Sn **来实现。**

修改后的程序为:

```
REM  画 3 条大小不同的帆船
SCREEN  1,0
CLS
COLOR 1,1
PSET(260,120)
DRAW"S1 L60 E60 D80 L60 F20 R40 E20 L20"
PSET(160,100)
DRAW"S2 L60 E60 D80 L60 F20 R40 E20 L20"
PSET(70,85)
DARW"S2 L60 E60 D80 L60 F20 R40 E20 L20"
END
```

7．略

附录二　ASCII字符编码表

ASCII值	控制字符	ASCII值	字　符	ASCII值	字　符	ASCII值	字　符
0	NUL	32	(space)	64	@	96	、
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VF	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	S0	46	.	78	N	110	n
15	S1	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v

23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	：	90	Z	122	z
27	ESC	59	；	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	―	127	DEL

附录三　PRINT USING 语句的格式码

字符	功能说明	举　例
！	只输出字符串的第一个字符	PRINT USING“ !”;“ Main ”“ Storage ” RUN MS
\ \	两斜杠之间有 n 个空格，则显示字串的 2+n 个字符；如 n＝0，无空格时，只显示 2 个字符	PRINT USING“\　\”,“ BASIC ”, PRINT USING“\　\”;“ STATMENT ” RUN BASIC ST
&	字串按原样输出	PRINT USING“ & ”;“ CHINA ” RUN CHINA
#	指定输出数字的长度	X＝2357.1 PRINT USING“# # # # ”; X RUN 2357
	指定小数点的位置	X＝3.1415926 PRINT USING“# #　　# # ”; X RUN 3.14
＋	指定显示数值的符号（加号或减号）	A＝56.3：B＝－7.28 PRINT USING“ + # #　 # ”; A，B RUN +56.3　－7.3 56.3+ 7.3
－	放在格式字串尾部，当显示负数时，负号在数字尾部	X＝5.01：Y＝-0.13 PRINT USING“ * * # # ”; X，Y 5.01　0.13－
* *	让数字前的空格填以“ * ”，“ * * ”位置上也可用数字代替	X＝5.01：Y＝－0.13 PRINT USING“ * * #　　# # ”; X，Y RUN * * 5.01　* －0.13
$－$	让数字前面有一个“ $ ”号，“ $－$ ”位置上也可以用数字代替	X＝5.01：Y＝－0.13 PRINT USING“ $－$ #　　# # ”;X,Y RUN $ 5.01　－$ 0.13
* * $	同时显示“ * ”和“ $ ”符号	X＝56.7　Y＝－1.25 PRINT USING“ * * $ # #　　# ”;X,Y RUN * $ 56.7　*－$1.3
,	逗号在小数的左边把数字按三位数分节；逗号在字串末尾，原样输出逗号	A＝123.76：B＝21724.12 PRINT USING“ * * $ # # #　 # #, ”;A,B PRINT USING“ * * $ # # #，　# # ”;A,B ” RUN * $ 123.76, $21742.12, * $ 123.76 $21,724.12
^ ^ ^ ^	指定指数格式	A＝175.314 PRINT USING“# #　 # # # ^ ^ ^ ^ ”; A RUN 1.753E+02

_	表示下一个字符原样输出	A＝21.5 PRINT USING“A＿＝##　　#”;A PRINT USING“＿##　　#”;A RUN A＝21.5 ＿21.5
%	显示的数超过了 表示给定的长度	X＝112.3 PRINT USING“#　　##”;X RUN %112.3

附录四　Quick BASIC 语句和函数一览表

一、语句

1. 说明语句

关 键 字	主 要 功 能
CONST	定义符号常数
DEFINT	定义整型变量类型
DEFLOG	定义长整型变量类型
DEFSNG	定义单精度变量 类型
DEFDBL	定义双精度变量类型
DEFSTR	定义字符串变量类型
TYPE…END　TYPE	定义用户自定义数据类型
DIM…AS…	定义变量类型
DIM	定义数组的维数和大小
REDIM	重新定维
OPTION BASE	选择数组的下界
DEF FN	定义用户自定义函数
DECLARE	说明模块中使用的过程
DIM SHARED	定义各过程共享的数组或简单变量
COMMON	定义各模块共享的全局变量
STATIC	说明静止变量
SHARED	在过程中引用模块级的参数（该参数未在过程中说明）

2. 输入输出语句

关 键 字	主 要 功 能
LET	给变量赋值
INPUT	程序执行时，从键盘上给变量赋值
READ/DATA	用 DATA 置数在内存，用 READ 读给变量
RESTORE	恢复读数指针
LINE INPUT	从键盘输入一行的数据给字符串变量
SWAP	交换两个变量的值
PRINT	输出到显示器
PRINT USING	按指空格输出到显示器
LPRINT	输出到打印机
LPRINT USING	指定格式输出到打印机
WRITE	输出到显示器

3. 流程控制语句

关 键 字	主 要 功 能
IF＿THEN＝ELSE	两路分支选择的行结构
IF…END IF	两路分支选择的块结构
SELECT CASE	多路分支选择
WHILE…WEND	WHILE 当型循环

FOR/NEXT	计数循环
DO…LOOP	DO 循环
EXIT DO（FOR）	中途退出循环
EXIT FUNCTION （SUB、DEF）	中途退出过程
GOTO	无条件转向
GOSUB/RETURN	转子和返回语句
ON GOTO；ON GOSUB	多路分支转移或转子
END	程序结束
STOP	程序暂停

4. 作图语句

关键字	主要功能
WIDTH	定义屏幕宽度
LOCATE	指定光标的位置
CLS	清除屏幕
SCREEN	定义屏幕模式
COLOR	选择屏幕上显示颜色
PSET/PRESET	在屏幕上画点
LINE	在屏幕上画线
DRAW	在屏幕上连续画线
CIRCULE	在屏幕上画园
PRINT	绘图形着色
WINDOW	定义窗口的大小
VIEW PRINT	设置屏幕文件显示器的上下线
PCOPY	将屏幕的一页拷贝到另一页
PALETTE	改变调色板中的一种颜色
PALETTE USING	改变调色板中的多种颜色
GET（图形）	把屏幕图形存入指定数组
PUT（图形）	把由 GET 存放的图形输出到屏幕上

5. 声音语句

关键字	主要功能
BEEP	使扬声器发声
SOUND	发出一定频率和长度的声音
PLAY	按字符串指定来演奏音乐

6. 文件管理语句

关键字	主要功能
OPEN	打开文件
CLOSE	关闭文件
FIELD	指定一个记录中的域
INPUT#	从顺序文件中读出数据
LINE INPUT#	从顺序文件中读出一行送给字串变量
PRINT#	把数据写入到顺序文件中去
GET（文件 l/O）	从随机文件读数据到变量
PUT（文件 l/O）	将变量写入随机文件
RSET	将数据从内存移到缓冲区，字串右对齐
LSET	将数据从内存移到缓冲区，字串左对齐
WRITE#	把数据写入顺序文件
RESET	关闭所有磁盘文件
CLEAR	初始化变量，关闭文件
SEEK	为下一个读或写设置文件的位置
LOCTL	向一个设备驱动程序发送控制字串
LOCK/UNLOCK	用于网络环境对文件的全部或部分锁存
FILES	显示文件目录
NAME	更改文件名
KILL	删除文件
MKDIR	建立一个新的文件目录
CHDIR	改变当前缺省目录
RMDIR	取消一个现存的目录

CHAIN	连接文件，把控制权传到另一程序
OPEN COM	打开并初始化一个 l/O 通讯通道

7. 过程语句

关键字	主要功能
FUCTION…END FUCTION	定义函数子程序
SUB…END SUB	定义 SUB 子程序
CALL	调用 SUB 子程序

8. 出错处理语句

关键字	主要功能
ERROR	模拟错误，定义出错代码
ON ERROR	允许产生错误自陷，指定错误处理子程序的第一行
ON EVENT	允许事件陷阱子程序第一行
RESUME	指定事件陷阱子程序第一行
PEN ON/OFF/STOP	允许、禁止或暂停光笔事件 自陷
PLAY ON/OFF/STOP	允许、禁止或暂停演奏事件自陷
STRIG ON/OFF/STOP	允许、禁止捕捉游戏棒
TIMER ON/OFF/STOP	允许、禁止计时器事件捕捉
COM ON/OFF/STOP	打开、关闭或暂停指定口的通讯事件自陷
KEY ON/OFF/STOP	开始或停止指定键的自陷

9. 其他语句

关键字	主要功能
BSAVE	将内存某一区域的内容送到输出设备
BLOAD	把 BSAVE 建立的文件送到内存中
OUT	送一个字节到 l/O 端口
POKE	把一字节写入存贮器地址中去
WAIT	在监视一个机器输入端口状态期间，暂停运行
KEY	为功能键指定软键串值
ENVIRON	修改 DOS 环境中表中的参数
SHELL	退出 BASIC，执行 DOS 命令等，再返回下一语句
DATE$	设置当前日期
MID$	用一个字串取代另一字串
SYSTEM	关闭所有打开的文件，返回到操作系统
RUN	重新开始当前内存中的程序或运行一个指定的程序
ERASE	删除数组
RANDOMIZE	打开随机数发生器
REM	注释语句

二、函数

1. 数值函数

函数名	功 能
ABS	求自变量的绝对值
ATN	求自变量的反正切
COS	求自变量的余弦值
EXP	求自变量的以 e 为底的指数
LOG	求自变量的自然对数
RND	取 0~1 之间的单精度随机数
SGN	求自变量的符号
SIN	求自变量的正弦值
SQR	求自变量的平方根
TAN	求自变量的正切值
CDBL	将自变量转换成双精度
CINT	将自变量四舍五入转为整型
CLNG	将自变量四舍五入转为长整型
CSNG	将自变量转换成单精度型
CVSMBF、CVDMBF	把含有 Microsoft 二进制格式的串转换成 IEEE 格式的数字
FIX	截去小数取整

HEX$	十进制数转换为十六进制数的串
INT	取不大于自变量的一个整数
MKSMBF$、MKDMBF	把 IEEE 格式数转换为二进制串
OCT$	十进制数转换为八进制数的串

2. 字符串函数

函数名	功　能
ASC	求字串表达式第一个字符的 ASCII 代码
CHR$	把自变量的值转换成 ASCII 代码
INKEY$	从键盘读一个字符
INSTR	在一个串查找子串第一个字符的位置
LCASE$	转换为全部用小写字母的位置
LEFT$	从左边取一个子字符串
LEN	测试字串的长度
LTRIM$	除去字符串的前导空格
MID$	从字串任意位置取一个子串
RIGHT$	从字串右边取子串
RTRIM$	除去字符串的尾端空格
SPACE$	取 n 个空格的字串
STR$	把数值转换成字串
STRING$	生成一个字串
VAL	把字串转换成数值
UCASE$	转换成全部用大写字母的字串

3. 有关输出（包括图形、声音、时间、光笔等）的函数

函数名	功　能
DATE$	返回系统当前日期
LBOUND	返回数组指定维的下界值
UBOUND	返回数组指定维的上界值
LPOS	返回行打机打印头的当前位置
PEN	读光笔坐标的值
PLAY	读取乐曲中当前的音符值
POINT	读取图素的颜色号或图素的坐标
SPC	在 PRINT 语句中跳过几个空格
STICK	取两个游戏棒的 X、Y 坐标
TAB	在 PRINT 语句中跳过几个空格
TIME$	返回系统当前时间
TIMER	取午夜以来经过的秒数
VARPTR$	取在 DRAW 和 PLAY 中使用的变量的地址，用字串表示
CSRLIN	返回光标当前的行位置
POS	返回光标当前的列位置
SCREEN	读屏幕上字符的 ASCII 码或颜色

4. 有关文件的函数

函数名	功　能
CVI、CVS、CVL、CVD	把字符串中的数字字符转换成数值
EOF	检测文件结束标志
FREEFILE	返回下一个自由 BASIC 文件号
FILEATTR	返回一个打开文件的信息
INP	从 I/O 读入一个字节
INPUT$	从指定文件中读入一个字串
IOCTL$	从设备驱动程序中接收一个控制字串
LOC	返回文件中的当前位置
LOF	返回命令文件长度的字节数
MKD$、MKI$、MKL$、MKS$	把数字值转换成字符串
SEEK	读取当前文件位置

5. 其他

函数名	功　能
COMMAND$	用于引用程序的命令行
ENVIRON$	从 DOS 环境串表中返回一个环境串

FRE	返回有效内存的大小
PEEK	从内存地址中取一个字节
PMAP	逻辑坐标表达式对应物理位置，或物理坐标表达式对应到逻辑坐标位置
SADD	取指定字串表达式的地址
SETMEM	改变堆栈使用的内存容量
VARPTR,VARSEG	取变量的地址
ERDEV,ERDEV$	返回出错后的设备状态
ERR/ERL	返回错误状态