当代计算机应用丛书

MC

微计算机量子力学

〔英〕J.P.基林贝克著

上海翻译出版公司

《当代计算机应用丛书》

微型计算机量子力学

〔英〕 J.P.基林贝克 著
陆继宗 冯承天 译

上海翻译出版公司

内 容 简 介

本书在介绍微型计算机及BASIC语言的基础上，着重介绍如何将一些基本的计算方法应用于处理量子力学问题，是一本很好的入门书。作者特别强调要巧妙地使用计算机，并介绍了他自己在这方面的许多心得体会，对于希望将微机应用于量子力学的大学物理、化学、数学等专业的教师、学生、研究生和科技工作者来说，颇有启发和参考价值。

J.P. Killingbeck, DSc

Microcomputer Quantum Mechanics

Published by Adam Hilger Ltd (1983)

微型计算机量子力学

〔英〕J.P.基林贝克 著

陆 继 宗
冯 承 天 译

上海翻译出版公司

（上海武定西路1251弄20号）

新华书店上海发行所发行 南汇县南华印刷厂印刷

开本 787×1092 1/16 印张 8 字数 195,000

1987年3月第1版 1987年3月第1次印刷

印数 1—3500

统一书号：13311·31 定价：2.60元

译者前言

目前，介绍微型计算机的书籍甚多，但是详细介绍微机在某一专门学科中的具体应用方面的书尚不多见．在教学和科研的实践中，我们甚感需要此类书籍．英国 Hull 大学物理系 J.P. Killingbeck 博士所著 *Microcomputer Quantum Mechanics* 一书，正好就是一本介绍微机在物理学中特别是量子力学中的具体应用的书．我们阅后感到该书对于如何将微机与传统的课程结合以更新教材内容很有启发和帮助，因此不顾自己学识浅薄，贸然将它译出，供读者参考．

本书的主要内容大致有三：一是介绍几种微型计算机（器）的性能和具体使用方法；二是介绍一些基本的计算方法，如迭代法、有限差分法、数值积分法、*Padé* 逼近法和矩阵计算等，用的是 *BASIC* 语言；三是介绍如何将这些方法用于处理量子力学问题．本书可作为大学物理专业计算物理课程的参考书．所用到的量子力学知识是极为基本的，实际上仅限于 Schrödinger 方程而已．在数学上讲，这只是一类偏微分方程，所以本书对数学、化学以及工科的某些专业来说，也可作为计算方法课程的参考书．

由于作者既熟悉量子力学（他曾写过 *Techniques of Applied Quantum Mechanics* 一书），近年来又活跃于数值计算研究领域的前沿（见有关参考文献），所以本书对于从事科研的读者也有参考价值．作者特别强调要巧妙地使用计算机，不要作冗长的"笨算"，书中阐述了作者这方面的不少心得体会，例如介绍了如何把本来要在大型机上用矩阵语言计算的问题，简化得只要在微机上甚至在可编程序计算器上就能计算．这就使学生可以能动地学习计算机的使用，提高其实际的计算能力．而且这对于希望在计算物理方面作进一步研究的科研工作者和研究生也是甚有帮助的．作者在结束语中给出了六个尚未解决的问题，这六个问题本身就是很好的研究课题．如经解决，那便是很有价值的研究成就．

由于本书涉及面甚广，加上我们学识有限，译文中谬误在所难免，敬请读者不吝指正．书中原有错误，经发现者均已改正，凡涉及内容者，均在文中注明．对于一些明显的印刷错误，改正后就不再一一指出了．如有误改之处概由译者负责，在此一并说明．

译　者

一九八五年十月

于上海师范大学

序　言

本书的大部分内容是关于如何在科学工作中巧妙地使用微型计算机，因此，这对于广大的学生和研究工作者来说，应该是有所裨益的．开头的几章论述了应用和调试微型计算机的一般方法，这对于在教学中开始使用微机的教师来说，可能是颇有价值的．不过，为了有所侧重，我选取了微型计算机在普通力学中的简单应用，特别是在量子力学中的应用作为后几章的内容．因此，这些章节对于有点量子力学知识的读者来说，或许是特别适合的．譬如讲，这些内容或许可构成一门与量子力学基本原理课程同时开设的有用的"数值应用"课程．我深信一本关于计算技术的书，只有当它真正地表明如何把计算方法应用于一些真实问题时，才是有力量的．现在不少学生学了有关"计算"方面的一些必修课程，往往是语言倒学会了一种，却无用武之地（这犹如"盛装打扮，却无处可去"）．我试图在本书中阐明一种将计算技术和理论分析有机地结合起来的方法，并把它作为统一处理某一科学问题的一个组成部分．我是通过给出许多实例研究来做到这一点的，在这些具体讨论中，我一步一步地陈述了能够处理种种问题的方法．

我力图使得需用到的数学尽可能地简单，即使得读者并不需要大量的数学知识就能阅读本书．我所强调的是简单的原理在各种不同情况中的应用．例如，在本书的许多不同场合，我重复地应用逆的迭代计算的简单公式和一阶能量修正的公式；还反复地讨论递推关系以及Richardson外插法的应用．采用的数学符号都是标准的，只是在个别地方为了保持方程的简洁性而有所变动（例如，使用符号 y' 或 Dy）．

虽然我是根据我个人的经验和兴趣来选择论题的，但可以看出我所选择的材料在某些方面与J.C.Nash的著作 *Compact Numerical Methods for Computers*（Bristol: Adam Hilger,1979）中的取材正好相互补充．例如，Nash比较详细地讨论了矩阵本征值的计算；而我对数值积分问题，以及微分方程的本征值计算等问题的讨论则更为详细．这两本书合在一起，囊括了内容广泛的材料，对于小计算机的使用者来说，这些材料是很有价值的．

在整本书中，我用到了自己编制的许多程序；有时也注明一些近代的参考文献，读者可从中找到经过检验的BASIC程序．虽然我用几种型号的计算机来表明各种方法是如何使用的，但是必须强调指出，重要的是这些程序具有简单的结构，以致它们几乎能适用于所有的计算机．（那些我在CBM Pet机上试用过的、用到很大数组的程序，只要加以修改，就能在配备任一廉价的16 K的随机存取存贮器的Sinclair ZX-81机上运用．）

在本书中，我列举了许多例子，并给出了许多有解答的习题，以帮助读者提高解析的和数值计算的技巧．我也给出了一些参考书和文章，读者可在这些文献里找到有关专题的更为详尽的论述．我并不假定读者能熟练地编制BASIC程序，但假定他们已仔细读过所用微型计算机的使用手册．尽管，几乎每种计算机至少都与我的四种样机中的一种相类似，但只要读者能熟悉所使用的计算机，就能判断出如何在计算机上最完美地执行我所描述的任意一般过程．

因为我集中于论述如何使用简单的计算机方法去计算各种量，所以在某些地方，就不得不假定读者已具备初等量子力学的一些知识．那些希望能进一步深入到量子力学中的变分法的基础理论、微扰论和群论中去的读者，可在我以前出版的 *Techniques of Applied Quantum Mechanics*（Butterworths,1975）一书中找到一个包罗万象的概述．（下略）

目　录

第一章　微型计算机和 BASIC

§1.1　什么是微型计算机

计算机硬件工业的发展，现已使小计算机具有愈来愈大的功能. 通常在计算机前面加以定语"小型"和"微型"，究竟是按其体积大小，抑或是按其计算容量大小(以 RAM 的千字节 K 计)区分并不总是十分清楚. 实际上，"小型"和"微型"能否交换使用也是不清楚的. 就本书的目的而言，我随意地把 RAM 容量等于或小于 8 K 的计算机规定为微型计算机. 这并不是关于微型计算机的一个武断的定义，而只是对于某一类能用来处理本书所描述的一些简单方法的机器所作的一个说明. 许多计算实际上能够在存贮容量为 1 K 的机器上进行，为了使人们相信这确是如此，就设计了这样的算法，并且已用一些典型的机器作了检验. 这些机器是：得克萨斯仪器公司的 TI-58 可编程序计算器，Sharp PC-1211 袖珍计算机，基本 1 K 型的 Sinclair ZX-81 计算机以及 8 K 型的 CBM Pet 计算机. 后三种机器能用 BASIC 语言编制程序. BASIC 是广泛用于小计算机的一种算法语言. 科学工作者常常在大型计算机上使用 ALGOL 语言和 FORTRAN 语言，也有些人认为诸如 PASCAL 和 COMAL80 在某些方面要比 BASIC 更适用于小计算机. 虽然某些计算机(如 Apple 和 Pet)还有适用其他语言的机型，但是大多数想买"现成的"家用计算机的人仍会发现这些机器是按使用 BASIC 语言而装置的. 因此，在本书的大部分章节中我使用 BASIC 语言. 当然，一当理解了这些算法的基本结构后，要把它们改编成其他语言也并不难. 实际上正如 Alcock 在他写得极好的那本入门书[1]中指出的那样，即使是 BASIC 语言也会随着机器的不同而有一些微小的变化.

那些具有条件转移和子程序装置的可编程序计算器(诸如TI-58)也被我列入计算机一类[2]. 我把机器所存贮的程序能够包含借助于条件转移、循环等来控制计算的流程的指令作为由计算器跃为计算机的重要界限. 当然，大部分的计算器只采用编号式的存贮，而不用字母式的存贮. 例如在使用 BASIC 的计算机中，我们可以用变量 X，Y 和 Z 来计算，而能在程序里列出这样的语句

$$\text{LET Z=X+Y,}$$

则计算机将取出 X 和 Y 并贮存其和 Z，而我们并不知道这些量存放在什么地方，机器将为我们做簿记操作. 但是在 TI-58 的某一程序中作此计算时，就必须确定把 X 放在(譬如说)存贮器 1 中，把 Y 放在存贮器 2 中以及把 Z 放在存贮器 3 中，并且必须认真地记住这些情况. 为了计算 $Z=X+Y$，我们将使用下列指令

$$\text{RCL 1+RCL 2=STO 3}$$

这里的簿记操作是由程序编制人员明晰地作出的. 尽管有这些额外的问题，而且与大部分计算机相比较 TI-58 速度也较慢，但对本书中的数种计算来说，它仍不失为一种有使用价值的仪器，因为它有 13 位数字的精确度. 此精确度要比大部分计算机在使用 BASIC 语言时所能达到的精确度高得多. 对于某些类型的迭代计算法(在其中每多一次计算循环，就更的数值一步)来说，联合使用计算器和计算机(例如用 TI-58 和 Pet)有时是很有

Pet 机快速地做多次循环运算以得到一个较好的答案，接着仅用较慢的计算器来做最后的循环运算，就能得到非常精确的结果.

本书篇幅较短，不可能做到包罗万象. 我毫不怀疑一位机器语言专家能够把 8 位二进制字节纳入 ZX-81 或 Pet 机中以得到多倍精确度运算并增加运算速度. 在编写本书时，我已考虑到这种可能性. 然而，我决定不扯到这一领域中去，而只是集中于讨论基本理论的分析能有助于提高运算的速度和精确度的那种方法. 在我所叙述的大多数计算步骤中，有着许多乘、除运算，把它们写成机器代码是非常冗长的，因为大部分的芯片只把加、减运算作为其基本指令. 因此，我并不去走那对我来说是漫长的弯路；不过，我深信一本有价值的书(在某种意义上与本书互为补充的书)正有待于一位既是机器代码专家又是物理学家的作者去写. 可编程序计算器不能做而 BASIC 语言计算机能做的一件事是处理字符串(譬如说，按字母次序来整理词). 此差别对本书的材料并不太重要，因为我将要论述的主要是数值算法而不是有关信息和数据处理等内容.

§1.2 人机配合和迭代

一些使用计算机的科学家有这样的观点：一个"好"的程序是完全自动的. 操作员必须提供一些必要的数据，但此后的所有计算和判定均受程序控制. 操作员可以去打一场高尔夫球，然后回来取出结果. 现在有许多程序库的程序就以这种方式在大型计算机上(以及在微型计算机上)自动运行. 不过我相信在微型计算机上仍然有人机配合计算的很大余地. 就其本质而言，程序库中的程序通常并不是由设计它们的人员所使用，所以(程序的和人的)宽容限度是极端重要的. 例如，在我参加的一次会议上[3]，几位软件设计师特别提到科学家正在某些场合错误地使用程序库中供解微分方程的程序，在这些场合，作为程序的基本理论表明其精确度是有问题的. (这使我想起，通常在别的场合下使用的众所周知的一种说法：天才的设计，愚蠢的操作.)在我研究的量子力学领域里，也常常看到有人在大型计算机上用矩阵方法去求解简单的问题. 很明显，最好的解释是因为程序库里"有"处理大型矩阵的程序(这真是自找麻烦!). 稍加考虑就可明白，在电子计算器上使用简单的非矩阵方法有时候可能会得到更好的结果. 但是，正如一位计算机工业界人士向我指出的那样，一个人的重要性 I 被假定是与他所使用的计算机的时间 T 成正比的. 对我自己的作用来说，我认为把这一公式改成 $I=AT+BT^{-1}$ 则更为恰当，式中 T^{-1} 项是"天才项".

于是在本书中，我所想处理的计算是那些能在微型计算机上通过人机配合进行的计算. 操作员将不时地停止和重新开始计算，以及可能需要监控一些输出数据. 如果计算看来是"错"了，他能在程序的任一步上随意插入一个输出指令以监控中间结果，并当纠错完成后能去除这一指令. 如果能以这种方法跟踪并控制某一计算，则许多科学家确实会对此计算感到更有把握. 在物理学中，有许多真正需要大型计算机的问题(如大原子的 Hartree-Fock 计算，晶体 X 射线数据分析)，我们可以放心地把这些留给大型计算机去完成. 但是也有许多虽然小一些然而很重要的问题，这些问题能以一种更为密切的人机配合的方式在微型计算机(诸如§1.1 中列出的那四种类型)上处理. 人机配合计算的目的是把操作员的经验和判断(这些在自动程序中是难于获得的)与计算机的巨大计算速度(除 Zerah Colburn 和 Carl Friedrich Gauss 外. 这是人力所不及的)结合起来. 于是，在此意义上，操作员加上微型计算机形成了"完整的计算机". 我以前也曾提到过这种观点[2]，这乃是几年前读了一篇科学幻想小说之后产生的. 书中

的主人公在试图设计一台用来处理通常语言含义不清问题的计算机而遭到挫败后，就把自己造到机器里去了[5].

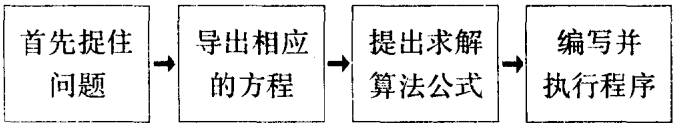
我在本书中所论述的问题选自经典物理和量子力学. 物理学的这些分支为举出有关微型计算机程序编制和数值分析的令人感兴趣的例子提供了足够的问题，以致我所讨论的课题在科学和数学的许多领域都有着它的类似情况. 我要强调的是：只要行得通就用简单的数学；或者，象我有时候说的那样，认真地用简单的数学. 现在的问题是，在当前的科学文献中有种倾向，认为只有那些看起来是很深奥的数学才是"重要的". 在众多的例子中只举一个来看看：最近我看到一篇十页左右的论文，其中充满了非常巧妙的回路积分，这要使读者花上几个星期才能解决. 最终结果是一个与量子力学有关的数，其精确度为百分之五左右. 而应用一个简单的计算器上的窍门，可以只花两页纸证明，并且不需要任何回路积分，就能以百万分之一的精确度得到该数. 如果科学论文的作者相信(如我那样)，写文章的目的是为了交流，特别是为了增加广大读者的知识和科学研究的能力，那末很明显，他的最好方式是使用简单的数学和简短的论证. 数学的部署可能是有独创性的，但是其宫殿应是由普通的砖块建造的. (大部分物理学家都记得 Weyl 曾指责 Dirac 在一次应该是初等的讲演里暗中使用了群论，对此 Dirac 则回答说，他的方法不需要有关群论的任何预备知识[6].)

如果一数学问题是被转变成一种与 1 K 左右的微型计算机存贮量相适合的算法，那末它在任何场合都不会过于复杂和冗长，虽然在如下的意义上来说它可能是十分"巧妙"：为了看出可以如此简单地得到所求的结果，就需要进行一些仔细的分析(§ 6.4 和§ 7.4 将给出这方面的例子). 事半功倍的通常做法是将某种计算转换为使用迭代算法或递归算法即能完成的形式. 只需要对一个迭代循环编制程序，给出一个简短的程序，而这种计算就只是简单地多次重复这一循环. 如果这种计算是以人机配合的方式进行的话，那末操作员就能亲眼看到迭代何时收敛. 在这里，某些纯粹主义者再一次会说停止迭代过程应是自动执行的：当相继的两个迭代与某一特定的精确度相符时，计算机会停止. 但如果我们有一个方程，譬如说，其根是 0.0011 和 8315.2，我们指定"当两个估计值的差小于 10^{-4} 时停止"呢，还是"当其结果之差小于 10^4 中的一个单位时停止"？在某些情况下，会得出过早的"伪极限"，而实际要求的极限则要经过很长的运算才能达到. 于是就清楚地知道：如何编写一组自动的程序指令去处理即使象 END 这种简单的命令也不总是明显的. 当然一个足够长的程序能顾及大多数不测情况，但这将花很长的时间去编写，以及它还将包含好几个逻辑判断步骤，这就将减慢程序的运算速度. 通过简单地进行相互配合的计算以及在显示屏上监控这一计算来消除我们对结果真实性的怀疑，常常是一种较好的做法. 诸如 ZX-81 和 Pet 等机器具有能包含在程序内的打印位置控制指令，因此所要求的量(例如 X)的输出值，在每一迭代循环后能被打印在显示屏的一固定位置上. 当值 X_1, X_2 等收敛到极限值时，被显示数的头几位数字将凝固不动，而后几位数将迅速跳动并一位接一位地凝固到它们的收敛值. 因为多个迭代循环的整个计算中只用了显示屏上的一行，所以几个相继计算的结果就能被同时显示在屏上. 对于 TI-58 和 Sharp PC-1211，在任何情况下，每次都只能见到一行，虽然 PC-1211 在这一行上可以显示不止一个数字，例如，它能肩并肩地显示 X_n 和 X_{n+1}. 这两种机器都有一 PAUSE(暂停)指令，该指令将在中止计算的同时，显示所要求的数约一秒钟. 在 TI-58 机上可以连续用几次暂停以按所需时间的长短来显示其数字(例如，长得足以在记录簿中记下该数).

§1.3 理论的作用

在量子力学的研究中使用微型计算机对教学和科研两者都是有价值的.在教学上,这使学生们能看到当采用数值计算时,诸如变分理论和微扰理论等方法是如何给出结果的;还使人们弄清楚:当真正要把数字代入方程时,正式教科书上的哪些数学内容是易于应用于实践的.通过引导学生认识到表示一个方程或其解的"最佳"形式在解析和在数值上可能是不同的,从而促使人们在使用数学方程本身时采用一种更为灵活的方式.这样一来,对一个积分值就不是想方设法去找一个复杂的明显表达式,而更有价值的,也许只是计算它的数值解.当然,问题也还有另一面:有时候如果你确实知道了一个问题的精确的(以及可精确计算的)解析解,那你可以把它作为一个校验问题来检验计算机上所采用的一个数值计算过程是否足够精确和稳定.可以把这种方法作为有价值的双向反馈过程来使用.例如,由使用中点积分法则来求各种简单函数的积分,我的一些大学一年级的学生就发现对于小的积分步长 h 来说,使用中点法所得的值与精确的解析值相差一个与 h^2 成正比的数.从一系列的检验问题中发现了这一规律,于是他们就能用它来求出那些不能解析积出的积分的精确值.这种逐步地从已知过渡到未知的能力是学习的一个重要部分,而微型计算机能用来帮助建立这个过程.在某种意义上说,它允许理论和实验相混合,尽管这只是在整个物理学所包含的大天地中的一个很小的圈子里才能这样做.

在后面的章节中,我试图解释如何使用在微型计算机上进行的经验性的"试验"来提出理论研究感兴趣的课题.我也将反复指出,对于某一问题的背景数学理解得愈透彻,常常会使得这个问题的数值计算程序愈完美.在许多情况下,物理学家遇到的数值问题是他正在处理的某个总问题的一部分,所以只要有可能,我就通过对专题的研究来论述.这些专题研究着重表明了在实际计算中代数技巧和数值技巧通常是紧密结合的那种方法.Fox 和 Mayers[7]在他们那本深受称赞的书中强调了考察问题的这种综合方法.我完全赞同他们的如下观点:在编写计算机程序之前,首先把分析搞正确是整个过程的一个重要部分.在下列步骤中,每一步都是重要的.首先必须"捉住"问题,即得到问题的某种有明确定义的形式.不过,所研究的问题往往在开始时总有定义得很差的麻烦,只是在经过多次直观的试验和误差分析后,才能结晶出一个明白无误的数学问题.其次,必须在清楚地知道什么是已知的,什么是待求的情况下,给出有关的代数,虽然在最初阶段也许还不能在数值上给出这些输入和输出的要求.(例如,我们可能会说"这一量不能从此推导出来";其后,这句话就变成边界条件 $\psi=0$.)第三和第四阶段是得出求解的算法和程序.要把这两阶段截然分开几乎是不可能的,因为所用的算法可能必须要去与用来作数值计算的计算机的能力相适应.特别是,如果我们要想使程序简短,就或许希望建立一个使用迭代或递归过程的算法,即便此时存在理论上完全能给出解的其他类型的算法.甚至在可行的迭代算法中,当我们考虑到有效位数和计算机速度时,某一些算法可能比另一些算法更有效些.当然在此情况下,可以求解一些检验性问题以获得一些用以判别所提出的两种算法的优缺点的感性知识.



大多数物理学家都会把上面所概述的半经验方法看成是"显然要做的事",我同意这种看

法，虽然我也看到了严格的数值分析家观点的正确性，他们认为对于此类半经验的方法，除非有严谨的数学支持它，否则是不值得重视的．他们苛刻的批评之所以是中肯的，原因之一是任何一个尚未卷入到整个问题中去的人或许不能充分地看出某些情况改变的含意．前面我提到过的错误地使用程序库的程序就是这方面的一个例子．另一个原因是与归纳逻辑的古老问题有关的：只是从一个方法在不多的几个试验中有效这一点，还不能得出我们可以信赖它的结论．我猜想大多数物理学家都倾向于不太尊重这类纯粹主义的告诫，所以我想从亲身经历中举出一些例子来吹毛求疵一番．使用可编程序计算器的学生一开始往往不愿意自己编写实验程序来进行数值积分，因为正如他们所说的："我的计算器内已经有了计算积分的程序片了．"进一步研究揭示出不论在计算器的说明书中还是在学生们的头脑中都没有任何关于步长 h 的变化对结果的精确度影响的信息．（当然，我尽力促使学生自己去发现的正是这种信息．）假如学生使用他的程序片，通常是 Simpson 法则程序，去积 x，x^2 和 x^3，他将得到三个精确的值，除了或许会有的一个小的舍入误差以外．我知道学生（实际上，还有一个大学讲师）由此就得出了一个无根据的结论：该程序片给出了精确的积分值．实际上，对于小的 h 值，理论证明误差随 βh^4 变化，不过当被积函数形如 $x^n(n=0,1,2,3)$ 时，β 碰巧是零．

在上面概述的这类研究中，可清楚看出至少需要一点理论分析来帮助我们监控经验积分．从所有这些情况引出的唯一教益是带有普遍意义的：总要设计尽可能多的交叉检验以避免错误，总要能以足够灵活的方式去创新，即使正统的教条是禁止这样做的．微型计算机的一个有用的特性是，它能让研究工作者作出一个快速试验，即使是对那些数年前由于希望不大而放弃了的"愚蠢的"概念．本书中的一些非常简单的方法都是以这样的方式产生的．作为一个需要某些答案的物理学家，我只是着手设计并检验一些解决我的问题的方法，这些方法在我看来是最简单和最直接的．有些方法是如此之简单，以致于那些作为科学杂志审稿人的"考究形式的"数值分析学家要使我相信这些方法显然是不起作用的．当然现在已积累了足够的证据来为我的异端邪说辩解．（说是只可能有七颗行星，但实际上却有九颗．事实胜于雄辩．）我的一些早期程序是为存贮量极小的可编程序计算器编制的，这就迫使我尽量使计算紧凑，而大型计算机就允许使用者在这些问题上偷些懒．这种限制对独创性所产生的激励，在人文科学中也有相似的情况．Richard Wilbur[8]说过，按规定的格律吟诗会激发诗人的创造力．

§1.4 种种 BASIC

仅仅使用 BASIC 语言中的不多几个基本部分也十分容易进行复杂的运算，这正如仅选用 FORTRAN 中的有限词汇也能凑合过去[9]．或者，事实上象 C.K. Ogden 或 C.Duff 的工作所表明的那样：在人类的数种语言中，人们只要使用极有限的词汇（一千字左右）便能应付[10]．我在本书中所关注的是要使计算的构造尽可能地简单，所以我将凑合着使用那种"随着我们的讲述便能学会的"BASIC．不过，许多人是在使用某种特殊的微型计算机时才首次碰到 BASIC，并且每一家制造商总倾向于有他们自己的语言变种，这就使得在一台机器上能用的语言可能在另一台上就不能用．在这些情况下，我们就应该查阅供某一种特殊机器用的使用说明手册，不过诸如 Alcock[1]写的那本书会对于将遇到的大多数 BASIC 语言的可能变种给出一个全面的评述．不同的机器之间最明显的差别是有些机器需要语句定义符 LET 而另一些则不需要．重复一下我在§1.1 中给出的例子，对于语句

LET Z=X+Y

在 ZX-81 微型计算机中需要 LET，而在 PC-1211 或 Pet 机中则不需要(然而，对于后一类机器，写入 LET 也能运行，而对于前一类机器如果略去 LET，则将出现语法错误符号.)

几乎现在市场上供应的所有微型计算机都有一套错误信息符号. 当机器不能接受操作员的程序时，机器将发出这些符号，通常还指出错误出现在程序的哪一行中. 对于学习使用一种新机器的人来说，这是很有用的，虽然有时可能会引起微机和操作员之间的一场猜谜游戏. 当某一冒犯机器的语句看来充满语法错误(有的机器还有一些在使用手册中未列出的精妙的错误信息符号)时赶紧放弃(以免多受损失)，并重新开始、重新排列运算，这往往是合算的. 就象在写作时，把那些不易修改的长句子重新改写一下可能会更好些. 通常避免错误或确定错误位置的简单方法，是使用我所称为的海明威风格的方法[11]，即不去写包含许多嵌套括号和算术运算的冗长的表达式，而是将计算分解成一些较短的语句. 如果一句冗长的语句包含某一难以捉摸的错误，那末把它拆开，并给每一部分一个语句标号，将有助于精确地找出错误. 当确定了错误的位置并经改正之后，如果需要的话，就可以把原来较长的语句重新构造出来. 这样做的好处是：譬如说，在图像显示时，每一行都写得相当满，而使得整个过程可以立即在屏幕上看出.

我再强调一下：在后面的章节中，我所主要论述的是数值计算，所以没有花很多篇幅去讨论各种微型计算机卓越的数据处理能力. (这可能就是我在§1.1 中提及的有待于人们去写的关于理想机器码的书的内容中的一部分.)根据我选择的讨论范围，我列出了几种典型的微型计算机所具有的一些有用的特性. 为了表明各种可利用的功能，我还汇编了一张表，从中可以看出我们特定的四种机器具有哪些特性，我同时又简述了给每种机器以"个性"的那些性质.

特性一览表

1. 能贮存程序(在磁带上或在固态永久性存贮器中).
2. 能接受带 LET 和不带 LET 的两种程序语句.
3. 一行上可以接受几个由冒号(：)分开的语句.
4. 对所有的指令(RUN, PRINT, LIST 等)都可实现单键操作.
5. 能接受普通科学函数(如 EXP, COS 等).
6. 能接受自定义函数.
7. 有 BODMAS 算法. 例如 3+4 * 2　按 3+(4 * 2)计算.
8. 执行溢出停止程序并给出错误符号.
9. 运算到限定的数位.
10. 具有 GOTO，条件转移以及子程序等功能.
11. 在任何一行都可由 RUN 启动程序.
12. 变量在指令 STOP 和 CONT 之间能用手动改变.
13. 有固定位置打印，可以给出迭代过程的静止显示.
14. 能用指定形式如 M(I,J)那样的矩阵阵列进行运算.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
TI-58C	√	–	–	–	√	–	√	IE100	13	√	√	√	–	×
PC-1211	√	√	√	√	√	√	√	IE100	10	√	√	√	–	×
ZX-81	√	×	×	√	√	×	√	IE38	9	√	√	√	√	√
Pet	√	√	√	×	√	√	√	IE38	10	√	√	√	√	√

Texas Instruments TI-58C

TI-58C 与 TI-58 的差别仅在于 58C 具有能保存程序的连续存贮. 这种计算器有一个可以随机存取的存贮器,它最多可有 480 程序步或 60 个数据寄存器,这两者能以 8∶1 的比率相互转化. 该计算器有一程序库组件,此组件包含为多种运算预先编制好的程序,这些程序由一个相应于各个程序的编码来调用. 当数字贮存在数据寄存器中时,也能对它们起作用,例如 15 SUM 1 SUM 5,则把 15 加到寄存器 1 和 5 中的数字上去. 在作插入或修改时,一次可以列出程序中的一步. 条件转移是由比较显示出的数字与特定的 t 寄存器中的数字来作出的,即利用 $x \geqslant t$ 来决定是否作这种转移. 可能的转向地址可由绝对步数或由字母 A, B, C 等来指定. 诸如计算 $\sqrt{-5}$ 这样的错误,将由现出闪烁的 $\sqrt{5}$ 而不是现出错误符号来表示. (在大多数微机上要想求出 SQR(—5)时,机器将停止运行并现出错误符号.)虽然显示出来的只是十位数字,但如果需要时,供内部计算使用的整个十三位数字都可提取出来.

Sharp PC-1211

这种袖珍计算机有一可随机存取的存贮器,它具有 1424 步或 178 个存贮器,能以 8∶1 的比率相互转换. 第 1 个至第 26 个寄存器由字母 A 至 Z 来标记,而这些字母必须用来标记变量(例如使用 AA 将给出语法错误符号). 整个语句可以指派到一个键上,例如 Z 能够用来表示 PRINT,或者,譬如说 X * X+EXP(X)−Y * Y,这样编写程序就更快了. 该机能够处理七个字符长的字符串. 当用 BASIC 来写诸如 $x+2y$ 那样的代数表示式时,一个常犯的错误是把应写成 X+2 * Y 的式子写成 X+2Y,这将导致语法错误和程序停止. 值得注意的是 PC-1211 能把表达式 2Y 正确地解释成 2 * Y. 此外,它能接受表达式作为输入,例如 $\sqrt{2}+3$ * B 是一个可接受的输入,而这在我所知道的其他任何计算机上将会给出错误符号. 带有运算的 GOTO,例如 GOTO 10 * N,也是可以的. PC-1211 机上 1424 步所代表的容量比 1.5K 略大,后者是微机所通常采用的术语.

Sinclair ZX-81

这种小型机与带有连续调谐控制的便携式电视机相连时,操作起来很理想(我将我的计算机与一架 Bush Ranger 3 型电视机连用,发现在超高频 36 频道中能很方便地调到该计算机的频带). ZX-81 有一个 QWERTY 打字键盘,不过每一个键都有多种用途(例如 LET, INPUT, RUN, GOSUB, COS 被指派在特定的键上),这就使得不需要把控制词一个一个字母地打印出来. 数组的下标,例如象在 M(I, J)中的,可为 1, 2, 3 等. 但不能为零. 这种计算机可以两种速度运行,由指令 FAST 和 SLOW 所控制. 虽然一行只允许有一语句,但这一语句中可以包含一个很长的可溢出到屏上的下一行去的数学表达式. (在大多数微型计算机上,名不副实的"行"会在显示屏上占去数行*.)ZX-81 能产生随机数,这在使用概率模型的计算中是有用的. 它也能接受简单的表达式,例如 5+(2 * 3),作为变量的输入值.

CMB Pet

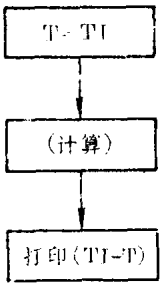
在 Pet 键盘上,常用的指令 PRINT 是由键 ? 给出的. 数组可以是多重的,例如立方数组 M(I,J,K)能被存贮和使用. 数组的下标可以是 0, 1, 2 等. 在 RUN 的指令下变量就预置于数值零,并不需要在一程序开始时加以说明. 例如语句(不需要 LET)

---

* 为此,习惯上把行译为语句,例如,第 50 行相应地译为语句 50——译注.

$$M=N+23$$

会被执行，即使 M, N 是在此程序中首次出现. 存贮器会分配给 M, N, N 的初值为零，M 终止时的值为 23. Pet 机能产生 0 和 1 之间的随机数. 该机内部装有一时钟，这对于不同的算法进行速度检验是有用的(见§ 2. 3). 计算的计时可如下进行：



变量 T 被预置于初始时间；计算进行了，然后将结束的时间减去初始时间就得耗用的时间(以 1/60 秒为单位).

Harrison[12]在其新近出版的一本书中指出，如果应用电视机的帧计数器，并用那些将数字放入特定位置的指令 POKE 或从其中复制它们的指令 PEEK(许多微机都有)的话，则 ZX-81 也能用作计时器. 这样一来，譬如说，在作指令 POKE 16436,200 后再作指令 POKE16437,100，则通过低字节和高字节的配置，会使计时器预置于 200+256×100=25800. 每过 1/50 秒此数字减少 1，所以在后一时刻，量 PEEK16436+256 * PEEK 16437 就给出当时的计数. 从 25800 中减去此数即给出以 1/50 秒为单位的耗用时间. 利用这种方法，我发现下列形式的延迟循环(参见第二章解答 2)

```
10    FOR N=1 TO Q
15    NEXT N
```

在 2X-81 上用 SLOW 模式运行时，花了 Q/50 秒. 在 FAST 模式中，显示屏被关闭了，所以帧计数器的读数固定不动，因而不能记时. (或许仍然有更巧妙的方法得到计算机的内部时钟，不过现在我还不知道.)FAST 和 SLOW 模式之间的速度比约为 4∶1，所以我能由使用 SLOW 模式来估计两种程序的相对运行时间.

当一种计算准备进行多次时，知道变量的值在指令 RUN 启动一次新的运行时将发生什么变化，这一点很重要. Pet 机置所有的变量为零，而且对新出现的变量也按上述那样的方式处理. 如果我们在一台 ZX-81 机上试用语句

```
LET M=N+23
```

(这里 M, N 是第一次出现)，则它会停机，因为它不能找到 N 的值. 但是如果在前面已确定了 N 的值(例如 LET N=0)，则它将为 M 置设一个存贮器，并将 M=23 贮存其内. 这样一来，在=的左边能有新名称，而在右边却没有，故我们必须用某种语句明显地给予每一变量以初值. 在 PC-1211 机上，M 和 N 的作用颇象 STO 10 和 RCL 14 在 TI-58 机上的作用：(A 至 Z)≡(1 至 26). 不过指令 RUN 并不搞乱变量的值，这些将是它们上次运算的结果，或许是上星期运算的结果，因为 PC-1211 具有永久存贮功能.

习题

1. 计算器 TI-58 在计算实数的平方根时，若 $x>0$，则给出 $\sqrt{x}$；若 $x<0$，则显示出闪烁的 $\sqrt{-x}$. 这在求实系数二次方程的根时有什么用处?

2. 在 PC-1211 机上计算 $\sqrt{2}$，其结果是 1.414213562. 假定 $\sqrt{2}$ 能写成 1.414+H，试导出 H 满足的二次方程. 用标准公式解此方程又将涉及到求 $\sqrt{2}$. 另一种方法是给出 H 这样的方程，在此方程中 H 在左、右两边都出现，因此开始在右边输入 H=0，我们用迭代法就能求得正确的 H 值. 证明这样得到的 $\sqrt{2}$ 的值能比上面所给的值多三位数字. 你能否看出如何将此过程转变为一种一般的算法，这种算法将给出从 1 到 10 的，有十个数位的高度精确的平方根?

3. 如果在 BASIC 程序中有这样一个语句

$$\text{LET X}=\text{X}/5+3,$$

它将使微机从 X 的存贮单元中取出 X，进行右边的运算，然后将结果作为新的 X 存贮到原 X 的存贮单元中去. 如果同样的语句（当然没有 LET）出现在代数中，我们就当它是一个关于 X 的方程，其解是 X=3.75. 作为一个有趣的练习，看你能否想出一种简单的方法使计算机将此语句当作一个方程来处理并给出 X 的值 3.75?

4. 包含 IF 的逻辑检验（例如，IF A>4 THEN GOTO 100）对于控制计算的流程是有用的. 在 Pet 机上，如果愿意的话，词 GOTO 可以省略. 大多数微型计算机接受多重条件转向，如

$$\text{IF R}>=0\text{ AND R}<=1\text{ THEN GOTO 100.}$$

很明显，为了达到一个程序具有能从一种 BASIC 机器转换到另一种机器的合理转用能力，那末，LET，THEN GOTO 等应完整地写入该程序. 即使这将使该程序包含一些对某些机器不需要的词，而这些机器仍然接受这样一个带有多余词的程序. 为了安全起见，我们可用"最小公分母"型的那种处理方法. 许多能在一台 BASIC 计算机上做的事，在 TI-58 计算器上也都能做，不过为了执行条件转移，我们要用 t 寄存器. 计算某一数 R，而后它能给人以深刻印象地被显示出来（鉴于考虑该计算就象对它进行笔算一样）. 如果 t 寄存器包含 0，则程序步

$$\text{2nd } x \geqslant t \text{ A}$$

将使此程序转移到步 A；只要 R（即显示的数）大于或等于 t 数（在此情况下为零），否则计算就简单地按部就班进行. 你能否编写上面那种 BASIC 双重条件语句，使得它能在 TI-58 机上被译为一个单一的 t 寄存器检验?

5. 如已指出的那样，PC-1211 和 ZX-81 机接受表达式作为变量的输入值. 在 PC-1211 机上，该表达式中能包含函数以及其他变量的值，我们曾举过 $\sqrt{2}+3$ * B 的例子. 在 TI-58 上按键 R/S 将暂停程序等待输入. 这使机器回到手动状态，以致我们能在按下键 R/S 继续进行计算之前，算出输入表达式. 例如，倘若知道 B 值在存贮器 2 中，便可键入

$$3\times\text{RCL 2}+\sqrt{2}=\text{R/S,}$$

在 Pet 机上，语句

$$\text{INPUT "A" ; A}$$

使得屏上显示出 A? 这需要一个数值输入. 如果输入一个表达式，将出现 REDO FROM START（从头开始）字样的回答. 试对 TI-58 的那个例子，请编制一种方法，使得表达式成为能接受的输入.

解答

1. 当该计算器计算 $\sqrt{\text{b}^2-4\text{ac}}$ 时，它或许给出一个稳定的显示，或许给出一个闪烁的显示. 这就显而易见地告诉我们根是实的还是虚的. 下面的程序将做此项运算，所以选用它是为了表明 TI-58 的一些特性. 输入数据，a 在存贮器 1 中，b 在 2 中，c 在 3 中.

程序

$$\text{RCL 2}\div 2\div\text{RCL 1}=\text{STO 4} \quad (1)$$

$$\text{RCL 2}x^2-4\times\text{RCL 1}\times\text{RCL 3}= \quad (2)$$

$$\sqrt{\ }\div 2\div \mathrm{RCL}\ 1=\mathrm{STO}\ 5\ \mathrm{R/S} \tag{3}$$

2nd LblA

$$\mathrm{RCXL}\ 5-\mathrm{RCL}\ 4=\mathrm{R/S} \tag{4}$$

$$-2\times\mathrm{RCL}\ 5=\mathrm{R/S} \tag{5}$$

2nd LblB

$$\mathrm{RCL}\ 4\pm\mathrm{R/S} \tag{6}$$

(1) 将 b/2a 存入存贮器 4，(2)和(3)算出$\sqrt{b^2-4ac}$并将它除以 2a. 按下 R/S(运行—停止)，结果便显示出来. 如果显示的数字是闪烁的，则它是这两个根的虚部. 按下键B显示出其实部. 如果(3)中的显示是稳定的，按下A给出第一个实根，再按 R/S 给出第二个根.

2. 令$\sqrt{2}=1.414+H$，并对两边平方得方程

$$H=0.00604/(2.828+H),$$

右边从$H=0$开始，经过三个循环，会得出一个稳定的H值. 这给我们

$$\sqrt{2}=1.414+H=1.4142135623731$$

为了得到计算其他数目的同样过程，我们可在 PC-1211 机上使用如下 BASIC 程序

```
10  INPUT X
20  A=INT(1000 *√X)
30  B=A/1000:R=X-B * B:H=0
35  FORN=1 TO 3
40  H=R/(2 * B+H)
45  NEXT N
50  PRINT B:PRINT H
60  GOTO 10
```

语句 20 是取$\sqrt{X}$，将它乘以 1000，然后取其整数值，就得出"截断"值；这类似本例中的 1.414. 语句 30 算出了类似 0.000604 的值. 语句 40 至 50 给出了一个循环，该循环执行三次以求出H值. 语句 50 显示了该高精确度平方根值的两个部分. 语句 60 使我们能计算下一个 X. (如我将在第二章中指出的那样，此程序在 Pet 和 ZX-81 机上不能很好地执行，所以必须对它稍加修改.)

3. 如果我们使计算机只是重复这一语句，则在X位置上的数字将增大，最后达到 3.75. 这是多项式方程的收敛迭代解法的一个例子(见§ 3.2). 如果 X 的起始值为 0，在 PC-1211 机上对 X 要进行 17 次循环运算，才能得到稳定值 3.75. 虽然若右边的函数更为复杂，此方法有时也能用，但是它并不总是有效，例如当右边为 2 * X+3 时就不行. 请试一下看看! 为了使这一语句重复多次，一种方法是将它放入一个类似于习题 2 中所用的那种循环中去.

4. 仅当 R>0 以及 R<1 时，函数 R(R−1)为负. 用下列语句即已足够

```
LET RR=R *(R-1)
IF RR<0 THEN GOTO 100
```

(名称不必一定是 RR，但必须在此程序中没有用到过的).

在 TI-58 机上进行下列各步乃是一个好办法: