PET 子程序库

广西计算中心译

一九八二年十二月

译 者 序

《PET子程序库》给出了一系列适用于CBM微型机的BASIC子程序，特别是绘图程序和排序程序有较强的功能。尽量充分地、灵活地运用这些子程序，可以简化我们的程序设计工作，提高我们的程序设计水平。这本书由尹业民、罗海鹏、陈寿勤、乔中南翻译，符华儿、吴地兴校对。由于译者水平所限，错误之处一定不少，请读者批评指正。

一九八二年十二月

目 录

一、引言

对于一般水平的PET用户，指望能编写一个程序集以便执行，比如说，商业上的应用，那是很难的。许多人甚至试都不试，宁愿雇用别人去做这工作或是买一个现成的标准件。仅仅编写程序倒不难，大概每一个PET用户都为他自己的用途编写过许多简单的小程序。难的是要编写有几千行BASIC代码程序的大程序集。与此关联的是如何及从哪里开始的问题，以及究竟需要程序做些什么。这些问题纯粹是精神上的障碍，因为任何一般智力的人都可以编写并设计这样的程序。首先给出一个逻辑结构，然后在这个基础上按照标准子程序格式插入、添加一些辅助程序语句，编写应用程序的过程就变得容易得多。

1．程序结构的确定

编写和运行程序的过程把人脑对问题或过程的抽象思维定义成机器使用的非常明确的指令序列的过程。最低的然而最精确的级是微处理机使用的二进制代码；这是用于机器编码程序的级。较高一级是Basic的解释程序的指令。后者允许对处理机给出易懂的英语语句指令。完成Basic语句的一条指令可能需要几百个机器代码程序步。紧接着Basic解释程序的上面一级是子程序。这是用Basic编写的完成某一具体功能的代码块。子程序给子程序员的功能比在Basic中可得到的功能更高级。例如，区分一个数组为字母数字型的子程序为程序员提供一个新的命令，概括为：SORT ARRAY X。正如Basic命令相当于几百条机器代码指令一样，子程序可能由几百条Basic命令组成。在子程序上面的一级是由子程序集组成的程序。在程序上面的是最高

等级的程序组，这一级是若干个程序的集合，这些程序由公共数据库来联结并且共同完成一个应用。例如：备料控制。

层次概念是非常有用的，因为它允许将编写程序的过程分割成一系列易于定义的级（即层次）。这首先给予程序员逻辑上的操作顺序，其次是使与最低一级有关的大量工作能自动完成。这是靠使用高级语言，如Basic，来完成的。这就排除了用最低级亦即机器代码级编写程序的必要性。而且，由于使用标准子程序，所需写的Basic代码的数量可减少相当多。程序组中全部程序甚至可以是标准的，因此可用于许多不同程序组。

编写一个大的应用程序的第一阶段是规定程序组确切的功能。在这个阶段，确定程序使用什么硬设备，了解该硬设备的局限性和性能也是重要的。知道正在使用的是什么硬设备是必要的，因为它确定程序的体积、程序所用的数据文件的性质和构造。例如，对32K PET使用3040磁盘和3022打印机编写程序所采取的方法，与在一个带有盒式磁带机的8K PET上写同一用途的程序所采取的方法是完全不同的。由于这种情况，第一阶段是要写一个有关这种应用的简要描述，其中有输入输出和数据文件的定义。

假设一个用32K PET，2040双面软盘和3022打印机的用户想用这个系统实现一个图书馆的参考书目的数据库。系统设计的第一阶段是用户正确地决定程序必须完成的事情。在这种情况下，要求程序通过主题内容或作者，从有500种书名的小图书馆中查找某一本书或若干本书。用户需要打印他所感兴趣的某一主题的书名，而计算机则造一个包括有关该主题信息的书名目录表。同样地，用户也需要造由某一具体作者编写的图书目录表。从存贮在磁盘上的数据库中存取和检索数据是相当简单的事情。可以把程序分成四部分：

1）数据输入：经常记入图书馆所增添的新的书目。

2）数据修改：需要修正数据库的错误或删除已从图书馆中去掉的那些书目。

3）数据存取：这部分程序响应某一具体输入，完成用户从数据库存取数据的要求。

4）数据文件维护：允许用户制做数据文件的可靠的备用付本。它也完成诸如将数据文件按字母顺序分类的功能。

你会注意到这程序的四部分中有三部分与数据文件的维护有关。这是使用磁盘数据库的所有程序的特征。这四个程序部分的每一部分是独立于其它部分的，而且仅通过数据库与其它各部分互相影响。由于各部分是无关的，它可以象独立的程序一样来编写，对于程序员来说，这就简单多了——一次只需写程序的一部分。各部分可以象独立的程序一样存在磁盘上，而当需要时由用户装入执行。象这样的程序的集合构成程序组。

2．文件结构

把所有程序联合成一组的原因是它们都使用同一数据库。在写任何程序前必须确定数据的性质和格式。这大概是程序设计中最关键和最复杂的部分。拙劣的文件结构是造成大量低劣程序的原因。遗憾的是没有简易的经验法则，可以用来选择一种最好的数据文件，以便用于某一程序。值得记住的唯一规则是：如果数据文件很大，所谓很大指的是一个文件包含的数据比机器内存中可存入的数据要多。那么那个文件在多数情况下应该是一个随机存取文件。可被输送到数组并可存贮在内存里的短文件最好作为顺序文件存到磁盘上。用这个规则的理由是它使数据存取时间相当快并且也使文件维护容易得多。

有三种主要方法可以存贮数据：顺序磁盘（或磁带）文件，直接存取磁盘文件，以及作为数据语句存放在程序内的数据。这些方法各自有它的优点和缺点，而所采用方法的选择取决于找出最适合应用的那一种。顺序文件是用起来最简单的，因为操作系统自动组织信息。在顺序文件中，数据项一个接一个地被记录，第一项记录在头上，最后一项记录在链尾，以便按顺序建成一个数据链。顺序文件的长处是数据项可以有任意长度，因而不致浪费磁盘或磁带存贮空间。顺序文件的不利条件是，每次读一个记录必须读该记录前面的整个文件，因为没有办法使程序能确定某一个具体记录的开始位置。同样，在顺序文件的某个地方插入或修改某一项的唯一可行办法是靠写整个文件的修改版本。

直接存取文件和顺序文件差别在于前者能指定磁道、该磁道上的区段以及记录在其上的那个区段上的字符。如果各记录存贮单元的内容保持不变，那么直接存取文件允许人们直接查阅一个记录，而不用读取文件中的所有前面的记录。因而直接存取文件允许不按顺序即随机存取信息。（为此，直接存取文件通称随机存取文件。）修改直接存取文件上的记录不必重写整个文件。然而，随机存取文件上的记录必须具有固定长度，否则记住各个记录的位置的工作变得太复杂。使用固定长度记录意味着存贮同样数据时，直接存取文件比顺序存取文件使用空间要多。随机存取文件的主要优点是存取迅速。PET上直接存取的缺点是需要依赖机器里配置的Basic和磁盘操作系统的版本。为了建立和使用直接存取文件，使用Basic 3.00和DOS（磁盘操作系统）1.00的机器要求相当复杂的程序设计。这已在后来使用Basic 4.00和DOS2.00的机器上得到纠正。这些机器拥有简单的用于相对记录文件的命令。相对记录就是由记录号来存取的数据。例如记录50是

文件上第五十个记录。在大量数据打算由程序贮存和读取的地方，对于所有数据库文件，直接存取是主要的选择。

有时作为数据语句在程序里直接贮存数据是合乎需要的。它具有迅速存取的优点，而在以PET系统为基础的磁带中特别有用。使用自动数据语句生成子程序可以添加数据或修改数据。数据语句可用顺序检索或相对记录方法来存取。作为数据语句在程序里贮存数据的明显缺点是：由于所有数据存放在内存中，因而数据文件的最大体积受到有效内存容量的限制。与存贮在数组里的顺序文件共同具有的另一个缺点是：在新程序或数据保存到磁带或磁盘上之前，一旦机器掉电，那么被输入或修改的数据可能会失去。

仔细考虑上面提出的例子，我们发现该程序需要三个主要数据文件。最大的是包括书目和内容细节的主数据库文件：这个文件被构造成随机存取文件。另外两个文件都是顺序的：一个作者文件，一个主题文件。在数据存取期间，两个较短的文件中有一个被检索。就是说，每个记录包括由一系列指针跟着的作者名，这些指针指向随机存取主数据库文件中的记录。这些指针只不过是记录号：在随机存取文件系统中，记录号是从文件中存取某一具体数据块所需要的全部信息。使用包括索引和指向较大文件的一个或多个指针的短文件，其优点首先是速度快，而且分类短的索引文件也比长的随机存取文件要容易得多。使用短的索引文件的另一个优点是它可以具有多个索引文件，正如这里我们有一个作者文件和一个主题文件一样。

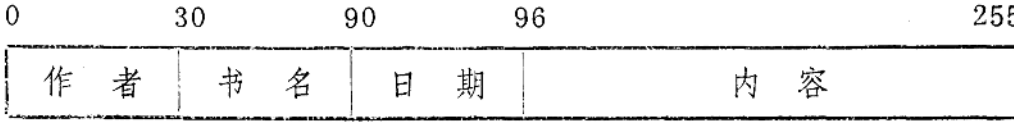
在已经确定了程序组中使用的文件类型之后，我们必须选定用于这些文件的数据格式。如果最佳用法是由机器存贮容量决定了的，那么对待数据格式常常要非常仔细考虑。首先让我们来看随机存取文件。随机存取文件包括一个记录集：可能有上百、上千、甚至上万个

记录，这个数字随应用而定。在此例中书名的最大数量是500，因而我们不要求此随机文件的记录比500更多。每个记录具有相同长度。一般记录长度为128或256字节，或是2的某次幂的值。使用256字节长的记录更可靠，因为这是计算机系统的自然结构。由于PET磁盘是按256字节为一块来划分的，所以每个随机存取记录用一个字块是最容易的，这样一个磁盘可供给的实际最大记录数为640个。第一步草草记下你打算存入每个记录的各种数据的清单，并且对照各项记下存贮它所需字节的最大数，同时记下数据是数字型的还是字母型的。运行时所用字节的总数应该保持不变，因为总数不应该超过我们为记录所设置的最大值（在此为256字节），这点很重要。还要特别注意的是在一个记录中的各数据项即字段应有足够的空间。这空间必须给数据项提供所需的可能有的最大长度，以保证该字段中所有要求的信息的传输。在我们的例子中清单可能象这样：

AUTHOR（作者）	30	Alpha （字母型）
TITLE （书名）	60	Alpha （字母型）
DATE（日期）	6	Numeric（数字型）
CONTENTS 1（内容1）	80	Alpha （字母型）
CONTENTS 2（内容2）	80	Alpha （字母型）

这记录中有四个字母字段和一个数字字段。记录应该这样来存贮，就是使它的所有字段都具有相同的变量类型。在本例中，把日期输入到记录中之前应将其转换成字母串。日期字段占6个字节，而其他四个字母字段具有不同长度；上表中引用的字节数是各项的最大值。比最大值短的各项，在字串末尾用空格填充以便使它达到最大长度。例如Shakespeare，w将做为SHAKESPEARE，w．----------来存放（破折号代表空格）。使用这个方法，我们知道书名字段总是

在记录的第31字节开始而在第90字节结束，这使得用Basic的字串操作命令更容易从某一个记录中检索各数据项。



由于设想我们的实例随机存取文件将有500个记录，每个记录256字节，所以整个数据库将占有128K字节的存贮空间。这恰好能装在一个磁盘上。在设计程序时，总是力图使数据使用一个磁盘驱动器而程序存贮使用另一个。还有，在程序执行过程中换磁盘是不可取的，因为这常常会导致不可靠性并且大大增加了丢失数据的机率。

虽然随机存取记录必须有固定长度，但是顺序文件记录却可有固定长度也可有可变长度。选择哪一种取决于应用情况。在多数情况下，数据总是有相同的长度，这就使得逻辑上选择固定长度和格式上顺序的文件。而正如在我们的例子中的随机存取文件的情况一样，每个顺序文件记录可包括多个字段，因而我们的例子是选择固定格式、固定长度的记录。分析它们比可变长度记录容易得多。但是有些应用（我们的例子也是其中之一），在这些应用中固定长度的顺序记录是不适用的。两个顺序文件的每一个包括可变长度的记录数。每个记录包括一个索引（作者的名字或他的书目）以及一个数字型指针数，这些指针的数量是可多可少的，其作用是指出随机存取文件里的记录。为了在此应用中使用固定长度记录，应该要求在每个记录中配置足够的空间以便提供最大长度的索引字以及最大指针数。由于文件要做为数组存放在内存中，而使用固定长度的记录浪费掉大量空间（也即内存），因而使此完全不能实现。当记录不具有固定长度时字段之间必须使用分界符，普遍最常用的记号是星号。为了分析该记录类型，从

左到右先检索第一个标记字符（即分界符）。消除掉标记右边的部分就得到记录的第一个字段，再找下一个标记就可得第二个段，等等。这个方法不如使用固定长度固定格式的记录便利，但完全有效。在我们的实例中作者文件记录可以存贮如：

SMITH.J＊14＊56＊79＊125＊256＊428＊0

第一个字段是索引——作者的名字——SMITH·J，紧接的以及随后的那些字段是指向随机存取文件的指针，说明这个作者写的书名被存在记录14，56，79，125，256和428中。零用来作为记录结尾的记号。使用这个方法，作者的名字可以如需要的那么长并且可以象要求的那样后面跟着许多参数（记录的最大长度仅由80字符的缓冲器限制）。

无论使用哪种文件类型，如果希望能够把进一步的数据加到文件中去，而不冒把已存的数据擦掉的危险的话，这就是很必要的了。在随机存取文件中，可以指定零号记录用于此目的。程序员必须记住该记录仅用于存放下一个自由记录号。或者这个信息可以同其它对于系统的正确操作是必不可少的信息一起存入某一指定的参数文件。参数文件在任何程序组中都是非常有用的，因为它不仅允许变量在程序之间传送或在运行当中存贮，它们还允许改进某一程序包以便适合某一具体用户的需要。在一个顺序存取文件中，最容易的方法是用一个指定的结束记录当做该文件的最后一个记录。使用此方法，人们可以简单地继续读文件中的记录直到遇到结尾记录为止。常见的结尾记录形式是用Z字串填满该记录。例如：

ZZZZZZZZZZZZZZZZZZZZZZZZZZZ

检查记录的（比如说）前三位是否为“ZZZ”是容易的事情，因为正文中自然出现这种组合的可能性是很小的，所以它是文件结尾的可靠

标识。当读文件时，通过计算记录数，可以得到顺序文件的实际记录号。由于文件存在磁心里，使用分类过程可以很容易加上记录，并将其插入到正确位置。用这个方法建的新文件就重写在老文件上面。如果文件比有效数组范围大，必须把它分成两个子文件，正如我们处置作者书目和文件的情况一样（如果磁心空间允许，可以把它们合并为一个文件）。

3．程序设计说明

在第一阶段中，我们确定了数据如何由程序存贮以及程序需要执行哪些功能。程序设计说明可根据这些决定写出。这是程序的框架，围绕它写出实际的代码程序。每一个细节都要正确，这点很重要。因为可以证明这一阶段的错误在后来改正非常困难。这些说明也形成了可围绕它写成最后的用户文件的基础。说明应包含程序组中各程序操作的完整描述，从而说明操作员需要什么数据、将要输出什么数据以及存取哪个数据文件。这个成文的描述应该带有说明数据流的图表。说明的第二部分应该是一个程序组使用的数据文件结构的完整的表格。第三部分以及最后部分是关于如何组织数据输入和输出（或是在荧光屏或是在打印机上）。这看起来似乎是表面的工作，然而操作员之所以能容易地使用系统，几乎完全取决于说明的这部分的思想。

在程序各个阶段，荧光屏上的输入输出格式和/或打印机输出格式应该拟订在一张方格纸上。应该把整个程序使用的输入的各种标准形式列在一个表里。全部输入应当用同一变量类型和格式构成，因为有些输入用回车号来终止，而另一些没有使用回车来终止，就会干扰操作员。人们可以做一个决定，使所有输入为具有固定长度的字串形式，此决定可以提示并且捡出无用项、非法回车等等。至于象输入核

实的方法以及错误检查的步骤，比如校验字的使用，也应该在这一阶段作出决定。

在计算机程序设计的每一阶段使用程序流程图是个好办法。用直观形式来显示过程，在逻辑流程中发现错误常常容易得多。应该对程序定义的各级层次结构绘制说明其顺序和逻辑关系的流程图。当定义变得更明确时，在通常定义的流程图上的处理步骤可以用更加精密地定义了的该处理步骤的流程图模块来代替。相当于标准子程序的处理步骤当然不必象程序的其他部分那样明确地定义。标准子程序的广泛使用有助于减少最终的流程图的复杂性。在最终的程序流程图上各步将对应一个子程序或相当一个或多个Basic 命令。当编写程序代码时可使用程序流程图的最后方案作为指南。

4．子程序

写了说明之后，程序设计就可以开始了。当使用程序组时最好以数据项开始，因为接着可以用这个程序来建立测试该组中其它程序所需要的数据文件。程序组中的各个程序都可以划分成一组子程序。子程序是为完成某一特定功能而写的程序，在没有主程序支配的情况下，它可以单独运行。使用标准子程序编制程序的极大的优点是，这些子程序不必专门用于某个程序，而可在任何需要它们的程序中重复使用之。这意味着一旦写成一个好的子程序，就可以在其它程序中反复使用，从而大大简化程序编制工作。标准子程序的广泛使用是迅速而容易地编写好的程序的关键。

5．参　数

为了最大限度利用标准子程序，程序员必须采用关于使用变量名

的严格规定。这是必不可少的，因为子程序通过输入和输出变量的方法和程序的主体（也就是把子程序连结在--起的代码）通讯。显而易见，为此目的必须保留一组专用变量。这些变量通称为参数。在一个子程序和主程序中或者在一个子程序和另一个子程序中传送参数的方法有几种。在此书中我们将使用两种传递参数的方法。第一种通称为“用值和结果调用”，而第二个称作“标准调用”。传递参数的有效方法是需要的。因为子程序中使用的变量名不大可能和主程序中使用的那些变量名相同。比如，子程序中使用的变量可能称为PX，而PX的值是从主程序获得的。这个子程序可能在一个程序中被调用几次，但传送到PX（这称为形式参数）的值可能在第一种情况存入变量A，而在第二种情况存入变量Q（A和Q都是实际参数），等等。除非从主程序把值放入形式参数中，使其等于主程序中使用的实际参数，否则转移到子程序的结果将是错误的。

用值和结果调用是在用Basic写成的子程序之间传递参数的最普通的和最简单的方法。这需要两个参数传递操作——一个在转移到子程序之前，而另一个从子程序返回之后。第一个操作中实际参数的值被送到形式参数（这些是子程序使用的参数）。第二个操作中把子程序设置的参数的值从形式参数送到实际参数。例如：我们打算调用子程序对两个变量完成一个算术操作，而产生的结果做为第三个变量。（三个变量的结果）。子程序使用的两个实际参数这时被存入变量A和B，实际结果参数在变量C。子程序使用形式参数是变量PX和PY，子程序应置值的形式参数是变量PZ。

操作1——PX=A:PY=B

转移到子程序。

操作2——C=PZ