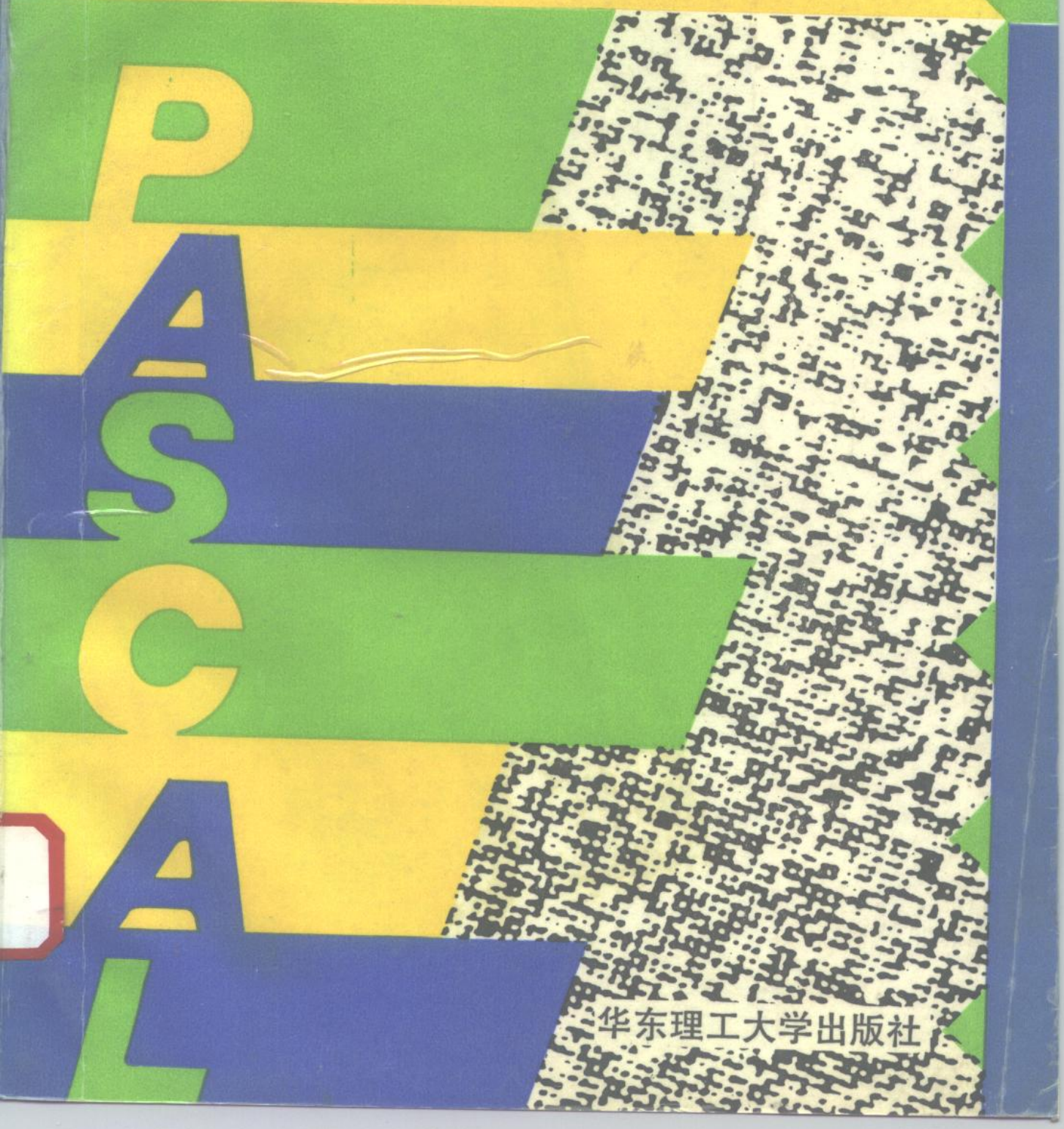
PASCAL 程序设计

黄润发　袁荣喜　主编

计算机应用基础教学系列丛书

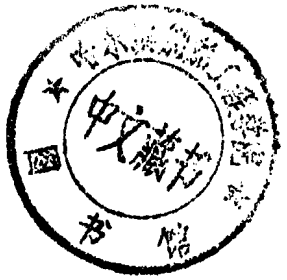


华东理工大学出版社

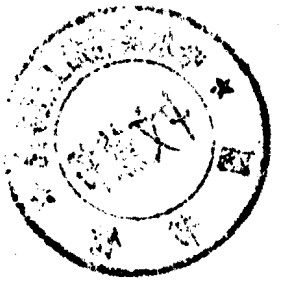
计算机应用教学丛书

PASCAL 程序设计

黄润发　袁荣喜　主编



华东理工大学出版社



(沪)新登字208号

计算机应用教学丛书
PASCAL 程序设计
PASCAL Cheng Xu She Ji
黄润龙、[illegible]主编
华东理工[illegible]出版发行
[illegible]
新华书店上海[illegible]发行经销
常熟文化印刷厂排版
上海群众印刷厂嘉定分厂印刷
开本 787×1092 1/16 印张 19 字数 453 千字
1994 年 7 月第 1 版 1994 年 7 月第 1 次印刷
印数 1—14000 册

ISBN 7-5628-0473-7/TP·63 定价 15.00 元

上海市高校计算机应用教学丛书编委会

序

随着计算机硬件和软件的迅速发展，计算机在各行业中已得到普遍的使用，应用水平也不断地提高，计算机的应用能力已成为衡量科技人员和企事业管理人员素质的重要标志之一。

1990年上海市高等教育局决定建立上海普通高校非计算机专业学生计算机应用知识和应用能力等级考试制度，并于1992年3月组织了首次考试，以后每年都要进行这样的考试。

为了进一步提高高校非计算机专业的计算机教学质量，上海市高等学校计算机教学协作组(非计算机专业)组织编写了这套"计算机应用教学"丛书。

本丛书的作者均是各高校长期从事计算机教学第一线的骨干教师，教学经验丰富，实践能力强。

本丛书的主要对象是高等学校非计算机专业的学生，也可作为对科技人员、管理人员进行计算机应用知识和应用能力培训的教材和自学参考书。

热忱欢迎广大读者对本系列丛书提出宝贵意见。

上海市高校计算机教学协作组

1994.1

前　言

PASCAL 语言是由瑞士 N.Wirth 教授于 70 年代初提出的，它具有定义严格、结构严谨、数据类型丰富、程序可读性好、易于体现结构化程序设计思想等一系列特点，它不仅是开发系统软件和应用软件的理想工具，也是被国内外广泛采用的程序设计教学语言。

本书针对非计算机专业学生学习计算机基础知识的特点，自始至终围绕“怎样设计 PASCAL 程序”这个中心，通过大量的例题，形象地、深入浅出地、详细介绍了PASCAL语言、程序设计的基本概念和方法，以及数据结构的初步知识。内容充实、文字流畅，便于读者自学。

全书共分十一章和两个附录，各章末均有小结，并配有适量的习题，最后两章除了给出一个结构化程序设计实例外，还列举了若干常用的算法与程序，并遵照循序渐进的原则，系统地设计了与各章内容紧密配合的实习内容，以便教师根据不同的对象组织教学内容。

本书由黄润发、袁荣喜主编。第一、九章由袁荣喜编写，第二、三章由强莎莎编写，第四、五、六章由周余洪编写，第七章由黄润发编写，第八章由乔沛荣、黄润发编写，第十章由黄润发、袁荣喜编写，第十一章实验部分由忻秀珍编写，全书由黄润发统审、修改定稿。

李玉茜教授仔细地审阅了书稿，提出了许多宝贵意见，在此表示衷心地感谢。在编写过程中还得到钱迺裕高级工程师以及黄桃云、时玉秋等同志的帮助，在此一并致谢。

限于编者的水平，书中不当之处在所难免，竭诚欢迎读者批评指正。

编　者

1994年2月

目　　录

1 计算机程序设计的基本知识

电子计算机的诞生是当今世界上最辉煌的科学技术成就之一。事实证明，将近半个世纪以来所取得的许多先进科学技术成果，如果不借助于电子计算机技术是难于想象的，特别是体积小、价格低、可靠性高、功能强、使用方便的微型计算机（俗称微电脑）的出现，引起了人们的极大兴趣，“计算机热”迅速波及世界，计算机的应用早已远远超出最初纯数值计算的范畴，扩大到金融管理、家用电器、仪器仪表、交通、能源、广播通讯、信息处理、工业控制、人工智能、办公自动化、教育以及文化生活等各个领域，几乎无所不包，这就给人一种印象：似乎电脑是无所不能的。其实，从本质上说，计算机只是人类在进行各种创造性劳动中的一种高效的通用工具，计算机本身不会（至少目前不能）自觉地去做什么，它的操作是由人指挥的，一个人要想使用计算机来帮助自己解决各种问题，他必须至少具备两方面的能力：一个是确定解决问题的方法，并将它们分解成一连串对应的操作步骤；另一个是掌握某一种能与计算机沟通的语言，以便将解决问题的操作步骤用这种语言加以描述，使计算机能够接受，并按要求完成指定的任务。前者属于设计“算法”问题，只有通过各种有关课程的学习，不断研究、总结提高、积累经验，才能拟定出有效的解题方法和步骤；后者指的是人与机器交互的工具，只有掌握了语言的规则和使用方法，才能正确地运用它去描述算法，编写出正确的程序，从而指挥计算机完成人们所要求的各种任务。

本书将通过对 PASCAL 语言各种成分的介绍，使读者掌握用 PASCAL 语言进行程序设计的方法。

1.1 算法的基本概念和特性

1.1.1 算法的概念

在科学研究、生产实际和各种管理中出现的问题往往是多种多样的，解决这些问题的方法也各不相同，甚至对同一个问题也会有多种解决方法。例如，要解决工矿企业中的“三角债”问题，某厂主可能这样做：对欠款者一律停止供应产品，并上告法院，要求将对方现有的流动资金全部用来偿还自己欠别人的债，用这种方法很可能导致工厂倒闭；另一种方法是，对欠款者按时间的先后、金额的大小列出重点去催讨，同时继续适当供应产品，并有计划地抽出部分流动资金逐步还清自己欠别人的债，这样做，可以维持工厂正常生产，债也可逐步理清。可见解决问题的方法通常不是唯一的，有一个正确选择的过程。

又如，要求前一百个自然数之和，即求 $\sum_{i=1}^{100} i$，也有几种方法，一种是：1+2，再+3，再+4，…，直至加到100，即进行99次加法；另一种是：(1+99)，(2+98)，…(49+51)，再+50，+100。前49次加法的结果都一样，均为100，共进行51次加法和1次乘法；第三种

方法是利用等差级数求和公式：

$$和=(首项+末项)\times 项数/2$$

即$(1+100)\times 100/2=5050$，这里只需要进行一次加法、一次乘法和一次除法。当然，我们总希望采用方法简单、运算步骤少的方法，这就要进行适当的选择，选择的首要标准是算法正确而且易于实现，其次是算法的效率。例如上述求和问题，所举的三种算法都是正确的，但若我们使用计算机只具有加法和减法功能，而做乘除运算必须用程序来模拟实现时，那么第一种算法可能是选择的对象，虽然它步骤较多，但用程序来模拟乘除法可能效率更低、更麻烦。

1.1.2 算法的描述及举例

仍以上述求和问题为例，将三种算法分别描述如下：

方法一：

步骤1：[初始化]将 s 置0，i 置1；

步骤2：[累　加]$s+i$ 送 s，$i+1$ 送 i；

步骤3：[判　别]若 $i\leqslant 100$，转步骤2；

步骤4：[结　束]输出累加和 s，算法结束。

必须注意，在一般情况下，算法步骤是顺序执行的，除非遇到了明确指出要转向哪一个步骤的指令。例如，在步骤3中，当 $i\leqslant 100$ 时，要求转到步骤2去执行，而当 $i>100$ 时，则顺序执行步骤4。在整个过程中，步骤1和4各执行1次，而步骤2和3则都执行了100次。

为了使描述更为简单明了，可用一些符号来代替某些文字说明，例如：$S_{1\sim 4}$(Step)表示步骤；A(Algorithm)表示算法；⇒表示送。于是上面的描述可改写为：

A1：

S1：[初始化]$0\Rightarrow s$，$1\Rightarrow i$；

S2：[累　加]$s+i\Rightarrow s$，$i+1\Rightarrow i$；

S3：[判　别]若 $i\leqslant 100$ 转 S2；

S4：[结　束]输出 s，算法结束。

中括号内是对该步骤所起的作用加以说明，对于各个步骤内的具体操作内容，如有必要可作进一步注释以便阅读和查错，这种注释可以写在步骤后的大括号内。

方法二(A2)：

S1：[初始化]$0\Rightarrow s$，$0\Rightarrow i$，$100\Rightarrow j$；

S2：[累　加]$s+i+j\Rightarrow s$，$i+1\Rightarrow i$，$j-1\Rightarrow j$；

S3：[判　别]若 $i<50$ 转 S2；

S4：[加末项]$s+50\Rightarrow s$；

S5：[结　束]输出 s，算法结束。

方法三(A3)：

S1：[初始化]$1\Rightarrow F$，$100\Rightarrow E$，$100\Rightarrow n$；{F 表示首项，E 表示末项，n 表示项数}

S2：[计　算]$(F+E)\times n/2\Rightarrow s$；{按公式计算}

S3：[结　束]输出 s，算法结束。

下面再举两例：

[例 1.1] 求两个正整数 M, N 的最大公约数。

解决这个问题可用著名的辗转相除法，算法描述如下：

S1: [初始化]输入 M, N, 若 $M<N$, 则 $M\Rightarrow T$, $N\Rightarrow M$, $T\Rightarrow N$; {保证将两者中较大的一个数放在M中}

S2: [求　余]$M\div N$ 的余数$\Rightarrow R$; {即 $M \bmod N\Rightarrow R$}

S3: [判　别]若 $R\neq 0$, 则 $N\Rightarrow M$, $R\Rightarrow N$, 转 S2;

S4: [结　束]N 即为所求的公约数，算法结束。

[例 1.2] 判别一个正整数N是否为素数。

素数是指除了1和其本身外不能被任何其他数整除的数，因此，可将N作为被除数，将2～$\sqrt{N}$之间的每一个数作为除数，进行除法运算，若均除不尽，则N就是素数。

S1: [初始化]输入N的值, $2\Rightarrow i$, $\sqrt{N}\Rightarrow M$;

S2: [相　除]$N\div i$, 余数$\Rightarrow R$;

S3: [判　别]若 $R=0$, 则输出"不是素数"，算法结束;

S4: [修　改]$i+1\Rightarrow i$;

S5: [判结束]若 $i\leqslant M$, 转 S2, 否则输出"N 是素数"；算法结束。

算法的例子举不胜举，我们将在以后各章中对若干典型问题的算法逐个加以介绍，只是描述的形式有所不同。

1.1.3 算法的特性

从上述的示例中我们可以得出算法的一些特点。

1.1.3.1 有穷性和有效性

任何一个算法只能包含有限个操作步骤，不能是无限的。例如在例1.1中，S2, S3虽然被重复若干次，但总是有穷的，即使M, N本身互质，辗转相除所得的余数也将会不断减小(R必定小于N)，总有一个时刻使得$R=1$，然后再相除一次必然出现$R=0$，从而不再重复S2，只转向S4。但是"有穷"应在一个合理的范围内，如果要运算若干年以后才能获得结果，则这个算法就很难说是有效的了。当然，如果出现诸如除数为0的除法也是无效的。

1.1.3.2 确定性

算法的每一步骤都必须有明确的含义，绝对不能出现"二义性"，否则计算机将无所适从，很难获得正确的结果。

1.1.3.3 有若干个(可以是零个)输入

这是明显的，例如求最大公约数的M和N，判素数的N，人们必须给出这些原始数据，否则算法无法实现。当然，对于那些本身已给出明确数据的问题，如求1～100的整数和，就不再需要输入数据了。

1.1.3.4 有若干个输出

这同样是明显的，人们必须以各种方式了解所求的结果，如果在算法中没有作出诸如显示打印等输出的安排，计算机是不会自动给出结果的。

1.2 程序的概念

人与计算机之间交换信息总是要采用某种方式，这种方式就是使用人和机器都能理解的语言，这种语言包括一定数量的符号、语句、语法和规则，人们只要按照这种语言的规定，根据解决问题方法的要求编写程序，然后输入计算机，计算机就能理解并执行该程序，完成指定的任务。计算机理解这种语言是靠存放在计算机内的语言处理程序来实现的，它把人用该种语言编写的程序翻译成一条条计算机能直接理解执行的指令。因此，当我们说在某计算机可以使用某种语言时，首先是指在该计算机内可配置某种语言的处理程序，然后计算机才有可能执行使用该语言编写的程序。

1.2.1 计算机语言

计算机使用的语言有多种；现简要介绍如下：

1.2.1.1 机器语言

机器语言实际上就是计算机的指令系统；每台计算机都有自己的一套指令系统，它包括几十条乃至几百条指令，每一条指令都对应着一个或几个基本操作，而所有的基本操作几乎都是由逻辑电路完成的。因此，机器语言也有对应的处理程序，不过这个程序并非由某种其他语言编写的，而是由计算机内的运算器和控制器电路组成的，就是说，如果用机器语言（指令）编程序，计算机可以直接理解和执行，这是一种编程效率最高的语言。但是，由于机器指令是由一系列0和1组成的二进制代码，而代码长度一般都在8位以上，有的达32位，甚至64位，这就给编程、识别、记忆它们带有很大困难，因而早期的计算机只有计算机专业人员才会使用。另外，由于机器语言直接和计算机的硬件（设备）相对应，用某台计算机的机器语言编写的程序往往在另一台不同类型计算机上不能用，使得推广应用非常困难，从这一点上说，机器语言并不是人与计算机交换信息的理想工具。然而，机器语言必竟是计算机唯一能直接理解的语言，不管其他语言多么高明，最后总要被翻译成机器语言才能执行。因此，机器语言是任何一台计算机必备、也是最终实际指挥计算机工作的语言（又称目标语言）。

1.2.1.2 汇编语言

由于机器语言烦琐、难记，人们想到了用一些符号来代替它，于是就出现了汇编语言。在这种语言中，每条指令的操作代码都由一串确定的字母符号来代替。例如：ADD表示加法，SUB表示减法，LDA表示取，STA表示存，等等。利用汇编语言编写的程序易读、易查、易改，而且执行的速度与机器语言程序相近。所以，在实时控制、生产检测，特别是由单板计算机、单片机组成的这些系统中，汇编语言应用很广。但是，计算机并不能直接理解汇编语言中的符号，必须要有一个汇编语言处理程序——即“汇编程序”或“汇编器”将汇编语言符号翻译成对应的机器语言，指令才有可能在机器中执行。当然，“汇编程序”本身必须是由机器语言组成的可执行程序，否则就不能完成翻译任务。

虽然汇编语言已比机器语言进了一步，但各类计算机的汇编语言也存在很大的差别，而且用汇编语言编写程序时要求人们对计算机的内部结构有较多的了解，因此，汇编语言的推广普及仍然是困难的。

1.2.1.3 高级语言

由于机器语言和汇编语言都与计算机的指令系统密切相关，对机器的型号依赖性极大，因此，通用性很差，而且编程时要求技巧性较高，对于编写比较复杂的计算程序难度很大。为了从根本上解决这些问题，人们迫切要求设计一种与具体的计算机指令系统无关、而表达方式则接近于人类的自然语言和数学语言、而且又容易为人们所掌握和书写的语言。例如，用 $y=x*x+3/x$ 表示 $y=x^2+\frac{x}{3}$，用 PRINT x, y 表示打印输出 x 和 y 的值等，这样人们在编写程序时就不必考虑计算机内部构造和在哪一类计算机上运行，这就是高级语言。（又称算法语言）。经过人们的努力，这个愿望终于在50年代中期实现了，到目前为止，世界上已研究出多种类型和功能不同的高级语言，其中最常用的有：

BASIC——这是一种容易学习和使用，而且有一定实用价值的语言，可用于数值运算，也可用于中小型的事务处理和数据处理；

FORTRAN——它是世界上最早出现，而且使用最久的一种高级语言，适用于大型科学计算和工程计算

COBOL——这是一种广泛用于商业、金融和交通等行业的高级语言；

PASCAL——这是最早出现的结构化语言，逻辑性强、结构严密、思路清晰，广泛用于教学、科学计算以及计算机系统软件的设计；

ADA——这是一种功能强，主要用于国防科研领域的工程化语言；

PL/1——是一种大型的、兼有数值计算和数据处理能力的高级语言；

C——是一种高效、灵活的系统软件开发工具，它既具有高级语言的功能，又具有机器语言的一些特征。

不论哪一种高级语言，都必须要有一种与它相对应的语言处理程序，通过处理程序将用高级语言编写的程序（一般称源程序）翻译成计算机能直接理解执行的机器语言程序（称目标程序）。正是有了这种处理程序，使得高级语言具有很好的通用性，因为不论何种类型的计算机，只要遵守某高级语言的规则和要求，用各自的机器语言写成对应的高级语言处理程序，那么，用这种高级语言编写的源程序在任何计算机上都能运行。

1.2.1.4 非过程化语言

上面介绍的各种语言都属于过程化语言，为了求解某一问题，人们使用它们时必须一步一步地将解题的全过程描述出来，即不仅要告诉计算机“做什么”，而且还要告诉计算机“怎么做”。使用非过程化语言编程序时，人们就不必描述为解决某问题的全过程，只需提出“做什么”，至于如何做的细节则由语言系统本身去处理并给出所要求的结果。如目前广泛使用的各种数据库语言：DBASE、FoxBASE、SQL、DL/1、QEB 等均属非过程化语言，它们为计算机用户开发应用系统提供了很大的方便。

目前使用的计算机语言多数是过程化语言，熟悉这类语言是进一步学习和掌握计算机应用开发技术的重要基础，有了这一个基础，要掌握非过程化语言是不困难的。

1.2.2 计算机程序

前面我们把计算机语言分为四大类，每种语言都有自己一套完整的符号以及严格的语法规则和明确的语义。所谓程序，是指为了解决某一特定问题（或实现某一算法）而从一种语言中选取必需的语言成分（又称指令）组成的有序的指令序列。因此，每个程序都是完成