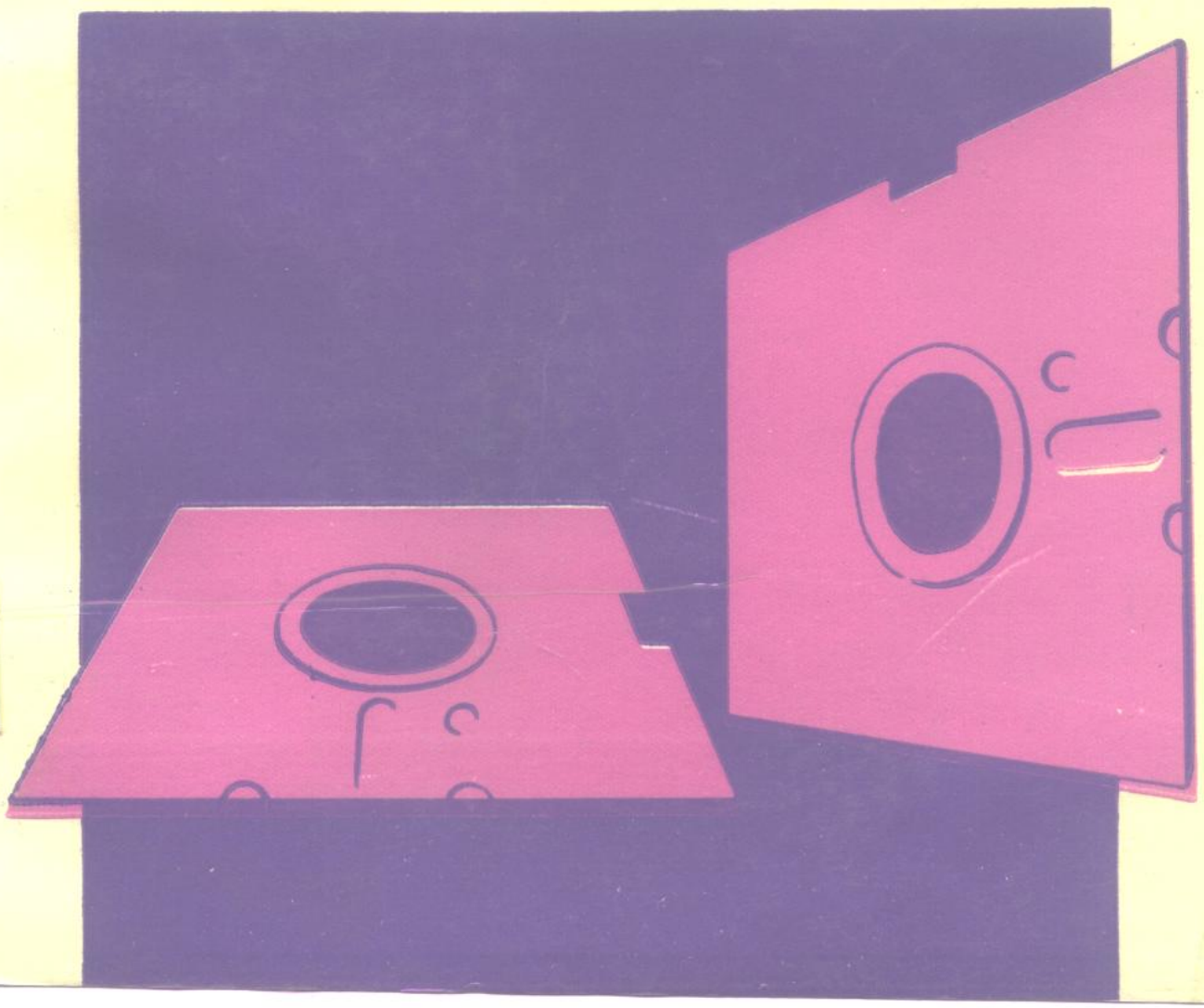
PASCAL Chengxu Sheji Fengge ji Yingyong

PASCAL 程序设计风格及应用

梁光春 罗玉芬 编

重庆大学出版社

PASCAL



PASCAL 程序设计风格及应用

梁光春　　罗玉芬　　编

重 庆 大 学 出 版 社

内 容 提 要

目前已出版的PASCAL程序设计教材都是把重点放在语言的语法规则上，忽视了从程序设计的角度来介绍程序设计技术。本教材以PASCAL为工具，突出程序设计风格，全面介绍程序设计领域内的基本概念和技术。

全书共分九章，第一、二章介绍编程之前的问题定义和算法拟定的准备工作。第三、四、五章介绍PASCAL的基本内容，并以较快的速度介绍这些章的句法说明。第六章讨论调试、测试、文件编制及维护等技术问题。第七、八章讨论PASCAL的数据结构和子程序技术。第九章是本书最重要的一章，讨论和处理大量的实际问题和高质量的编程技术。

为了突出程序设计的风格，把大量的程序设计风格的讨论分散在各章中进行，让学生养成良好的程序设计风格。同时，每章以后附有大量的习题和重点习题参考解答，很适合于计算机专业本科(或专科)的PASCAL程序设计课教材，也可以作为PASCAL的初学者入门的自学参考书。

PASCAL 程序设计风格及应用
梁光春　罗玉芬　编
责任编辑　韩洁
*
重庆大学出版社出版发行
新 华 书 店 经 销
威 远 县 印 刷 厂 印 刷
*
开本：787×1092　1/16 印张·15.5 字数：386千
1994年5月第1版　1994年5月第2次印刷
印数：2001—5000
标准书号：ISBN 7-5624-0675-8/TP·37　定价：9.80元
(川)新登字020号

目　录

第一章　概　述 …… (1)
§1.1　引言 …… (1)
§1.2　计算机程序设计步骤 …… (1)
§1.3　问题的定义阶段 …… (3)
§1.4　例子——查表 …… (4)
练习 …… (6)
第二章　算　法 …… (7)
§2.1　引言 …… (7)
§2.2　拟定算法 …… (9)
§2.3　算法的效率 …… (16)
§2.4　递归算法 …… (21)
§2.5　小结 …… (23)
练习 …… (23)
第三章　PASCAL 的基本概念 …… (27)
§3.1　引言 …… (27)
§3.2　数据类型的概念 …… (27)
§3.3　标准纯量数据类型 …… (28)
§3.3.1　整型 …… (28)
§3.3.2　实型 …… (29)
§3.3.3　字符型 …… (30)
§3.3.4　布尔型 …… (31)
§3.3.5　常量说明 …… (32)
§3.4　PASCAL 中的名字 …… (33)
§3.5　纯变量 …… (34)
§3.6　PASCAL 程序结构 …… (36)
练习 …… (39)
第四章　PASCAL 基本程序设计 …… (42)
§4.1　算术表达式 …… (42)
§4.2　标准函数的使用 …… (44)
§4.3　布尔表达式 …… (45)
§4.4　赋值语句 …… (46)
§4.5　输入和输出 …… (47)
§4.5.1　读与行读语句 …… (47)
§4.5.2　写与行写语句 …… (49)
§4.6　程序实例 …… (52)
§4.7　运行程序 …… (53)
练习 …… (53)
第五章　控制流 …… (56)

§5.1 引言 …… (56)
§5.2 复合语句 …… (56)
§5.3 循环语句 …… (57)
§5.3.1 WHILE 语句 …… (57)
§5.3.2 REPEAT 语句 …… (59)
§5.3.3 FOR 语句 …… (64)
§5.4 条件语句 …… (68)
§5.4.1 IF—THEN 语句 …… (68)
§5.4.2 IF—THEN—ELSE 语句 …… (71)
§5.4.3 CASE 语句 …… (73)
§5.5 无条件转移 …… (76)
§5.5.1 语句标号 …… (76)
§5.5.2 GOTO 语句 …… (77)
§5.6 分析问题 …… (81)
练习 …… (88)
第六章 故障检测和调试 …… (92)
§6.1 引言 …… (92)
§6.2 处理程序 …… (92)
§6.3 故障检测 …… (93)
§6.3.1 语法错误 …… (95)
§6.3.2 运行时的错误 …… (98)
§6.3.3 逻辑错误 …… (100)
§6.4 程序测试 …… (105)
§6.5 文件编制和维护 …… (108)
§6.6 小结 …… (110)
练习 …… (110)
第七章 PASCAL 结构数据类型 …… (115)
§7.1 纯量类型 …… (115)
§7.1.1 用户自定义的纯量数据类型 …… (115)
§7.1.2 子界类型 …… (118)
§7.2 结构数据类型 …… (119)
§7.2.1 数组 …… (120)
§7.2.2 多维数组 …… (125)
§7.2.3 记录 …… (129)
§7.2.4 集合 …… (135)
§7.2.5 文件 …… (138)
§7.3 指针类型 …… (141)
§7.4 问题分析 …… (145)
练习 …… (153)
第八章 函数和过程 …… (158)
§8.1 引言 …… (158)
§8.2 函数 …… (161)

§8.2.1 函数说明 …… (161)
§8.2.2 函数执行 …… (161)
§8.3 过程 …… (164)
§8.4 参数 …… (166)
§8.5 分程序结构 …… (169)
§8.6 函数和过程可作参数 …… (173)
§8.7 递归 …… (175)
§8.8 外部子程序 …… (179)
§8.9 问题分析 …… (180)
§8.10 小结 …… (188)
练习 …… (189)
第九章 构造优质程序 …… (195)
§9.1 引言 …… (195)
§9.2 面向人的程序设计 …… (195)
§9.3 程序的大小 …… (196)
§9.4 可读性程序设计 …… (197)
§9.5 结构程序设计简介 …… (199)
§9.6 计算机程序设计规则 …… (201)
§9.7 结构码分析 …… (203)
§9.8 数据结构的作用 …… (214)
§9.9 优质的最终含义 …… (214)
§9.10 小结 …… (216)
练习 …… (216)
附录 …… (220)
附录A PASCAL 语法 …… (220)
附录B 标准PASCAL 标识符 …… (225)
附录C 程序设计风格讨论 …… (228)
部分练习题参考解答 …… (230)

第一章 概 述

§1.1 引 言

这里使用的计算机"程序设计"指的是"在计算机上解题中所包括完整的一系列步骤",然而程序设计这个词往往与编码等同,它又是用某种存在的计算机语言书写语句过程的同义语。任何可以解答的实际问题编程之前都必须做大量的准备工作,它主要包括:准备定义想做的是什么?并去掉问题语句中的任何二义性和不可靠的成分来确定解法,用一种简便的方法粗略拟定出求解框架等这些步骤;只有这个准备工作完成以后,才能进入到编程阶段,即将问题求解过程转换为适应语法的某种特殊语言的语句。

另外,在编程完成后和编程前一样还必须花费很多时间和精力,设法找出其错误并纠正其错误,编制文档、测试、验证和程序的维护。

我们想说明的是计算机程序设计是一个由许多独立过程组成的一个极其复杂的任务,其中每一个过程都是重要的,对问题的求解都有贡献。不要把一些简单的过程与程序设计的概念混淆起来,例如编程不能算是程序设计。我们希望能够解释,为什么第一个完整的程序不出现在第一章而是在第四章?在构成自己的程序时,在PASCAL 编程之前一定要做大量的工作,要阐明、组织、构造和反复表达你的解答过程,不理解这个原理,那将是学习计算机程序设计的第一个最大错误。

§1.2 计算机程序设计步骤

在引言中我们反复强调程序设计包括许多步骤,现在对这些步骤做详细而明确的描述。

1. 问题的定义

开始做事之前必须准确地知道要想做些什么? 但是这个阶段往往被程序设计人员忽略或视而不见。如果带着充满了二义性和不可靠的问题开始工作是危险的,准确地了解什么是创造出可靠性解答的必要条件,定义问题的任务还在其余节中进行讨论。

2. 拟定出求解的框架

除非常简单的程序外,一般程序都不是由单一任务组成,而是由许多相互联系的任务所组成。例如一个计算机化的工资管理系统,决不能看成是一个单一的程序,相反,它包括多个程序,能够输入数据、管理文件、计算并打印出工资单,工资分析报告,错误记录以及保存信息的时间等。尤其是在大型程序中,除了给出每一个任务的职能外,还要给出它们的联系和相互作用,这一点显得特别重要,这样,就可以把它们组成一个有机的整体。

当然,本书中前面部分的程序都是很简单的并且由单一任务组成,这些程序由于不太复杂,画框图是可省的。因此可以被看成是为了教学的需要。本书将以大量篇幅对程序结构、大型管理、实际程序设计的技巧等进行讨论。

3. 算法的表达和选择

为了求解问题,规定了各种任务和子任务。对每一个任务都知道将选择什么信息,以及希望得到什么结果。算法是用于解题的特殊方法,它可以是已经编制好的或在文献中发表过的;

或用于交流的；或是我们自己设计的。一开始将算法的非规则性定义直接编码成某种语言的程序是一个很不好的习惯，其原因将在以后讨论。相反，同一种不依赖于任何计算机的算法表达方法描述解答的细节才是极大的优越性。在下一章将讨论算法及其编制，另外，将详细描述表达算法的一种特殊方法。

4. 编程

只有当问题的定义无二义性，组织好解答，一步一步地拟定出算法说明以后，才开始考虑编码的问题。选择计算机语言大多数是由以下 3 种特殊原因来决定：a. 问题的性质；b. 适应于计算机的语言；c. 计算机设备的特殊要求和限制。

有些程序设计语言应用范围很广，而另一些仅适应某些特殊问题，有些语言具有广泛的适应性，有些则仅仅在少数计算机上运行。下面列出了几种常用的程序设计语言。

语言	出现日期	应用范围
FORTRAN	1957	面向数字语言，最适合于科学、数学和统计等问题。使用范围广且有广泛的适应性。
ALGOL	1960	面向数字的语言，但具有新的特性，欧洲广泛使用。
COBOL	1960	使用最广泛的面向商业的计算机语言。
LISP	1960	特殊目的语言，基本上是为表和符号处理设计的广泛用于智能模拟领域。
SNOBOL	1962	特殊目的语言，主要用于字符串处理，包括课文编辑、语言处理和书目工作。
BASIC	1965	一种简单的交互式程序设计语言，在高等院校以及学院中普遍用于程序设计教学。
PL/1	1965	一种极其复杂而且普通目的语言，将 FORTRAN 的数字能力，COBOL 的商业能力以及其它特性的综合为一体的语言。
APL	1967	一种交互式语言，在广泛范围内具有新的运算和语言特性。
PASCAL	1971	一种普通目的语言，最适合于计算机程序设计概念的教学和对大规模程序能有效地执行而设计的一种语言。

5. 调试

初学的程序员很快地可以体会到程序的编制和运行很难使问题得到解答，因为程序有不可避免的错误，这是一个费时而令人烦恼的事情，为了便于调试，在 § 6.3 节中讲述。

6. 测试与修正

程序不一定都能得到正确的结果，为了保证得到正确的结果，必须确信自己的程序在所有的情况下都能产生正确的结果，即使是没有经过仔细测试的也应如此，在 § 6.4 将致力于计算机程序测试与修正的问题。

7. 说明

程序的文件编制是一个连续不断的过程，程序的说明从第 1 步到第 3 步算法的表达以及第 4 步本身都可以成为程序文档编制的一部分，我们必须保证文档的编制是全面的，完整的和

可以用的形式，它包括了程序员级技术文档和用户级使用文档。§6.5包括了这两级文档编制的准则和标准。

8. 程序的维护

这本书涉及的是人与人之间的有效通讯，而不是人与计算机间的通讯，显然，从程序的明了性、可读性和文档的编制都可以看出这点，当新的问题需要求解或增加新的设备时，就会变成过时的东西。那怕是几周，几月甚至是几年前写的程序，也几乎总是需要反复考虑，然后进行修改。即使在写程序时非常小心，并以一种清楚的、系统的、易读的形式写程序，要修改就可能失败甚至不可能，因此，我们也和其他人一样，需要一个写得简单明了，组织完善而结构优美的程序。

从这种意义上讲，我们可以说，本书的全部内容是致力于简化程序维护的目的。

最后，要强调以上程序设计过程的8个步骤大多数是相互交错的，例如文档就是在整个程序设计期间写成的，步骤中的大多数需要反复进行，又如当在调整中暴露出的错误时，就要回过头来对程序重新编写或重新考虑这个解答。这里讨论的要点是程序设计确实是一项重复的工作，对学完PASCAL规则的人很容易成为一个很好的编程员。

§1.3 问题的定义阶段

问题定义阶段包括了拟定和弄清问题的准确的含义，简单地说必须准确给出我们必须做些什么？这是一个非常明显的步骤，所以许多程序员完全跳过它并忽视了对他们求解的问题进行认真思考和了解，其结果是他们常常是用 定义有毛病和设想有错的情况下开始了程序设计。这就导致了混乱，不可靠以及各种恶劣的成分，得到的是错误的解答。

程序员对问题的定义常常处理得很糟糕的另一原因，是对这个问题在程序设计书中或在程序设计的课堂教学中都有被轻视的倾向。问题定义阶段是由学生或教师或其他人分别进行的。一个定义好的问题应清楚地印在一张纸上或一个副本上发给学生，并且要求叙述十分明确，比如输入A，B，C的值，做I，J，R运算，得到X，Y，Z值。对这样的叙述很不明确，学生很容易挑剔出不足之处，比如"从哪里得到A，B，C的值？""为什么要进行I，J，R运算而不选择L，M，N运算？"，"如果不可能，怎样才能成功地运算"，"你想过也应为用户提供W的值吗？"，如果这些问题是重要的，应该得到明确的回答。

这些相互作用的最终结果是写出一组问题说明，说明中应用清楚的无二义性的语言详细地说明了我们要求解的准确问题。这些问题说明包括了三类重要信息，虽然这三类信息不需要逐条进行讨论，但必须包括下列内容：

1. 输入说明，这节是描述程序的输入，应包括回答下列的问题：

a. 给程序提供哪些特殊值？

b. 这些值是什么格式(顺序，间隔，精度)？

c. 对每个插入可以设定值的真正范围是多少？

d. 使用这些值有什么限制？我们可以修改输入吗？当做完后可以去掉输入吗？

e. 在什么时候完成了输入？输入结束符是用特殊符号呢还是由我们自己来规定？

2. 输出说明：正象我们需要给出程序提供的规定的输入一样，必须详细地描述产生的输出，输出说明必须回答下列类型的问题：

a. 产生什么样的值?

b. 值的格式是什么(有效位数,十进制精度,在每页的位置)?

c. 在完成的报告中需要什么特殊的注释,表头或标题?

d. 还有在什么说明要作为输出量的吗?

特别地指出,表达输出说明的最有效的方法之一,是给出一个程序将产生的输出报告的详细例子。

3. 特殊处理:如果每件事都很顺利地进行,我们已经具备了所有的信息,就可以开展下一阶段的程序设计工作。出现的特殊情况或需要单独处理的非常情况,或者考虑改正和再生的错误,都应包括在问题的说明之中。

输入输出和特殊处理的说明表达了在问题定义阶段所收集的最重要的信息,却忽略了任何问题求解的最基本部分——求解方法。虽然规定了从什么地方开始并希望以什么结束,但一点也没给出如何完成这个任务的说明。

这个忽略是有意识的,用于求解问题的算法说明的一步一步地描述通常不属于写说明的部分,如果过早地确定如何写算法的低级说明,这样把程序员限制在特殊的应用中,那就缺少求解这个特殊问题的最好技术的灵活性。

问题定义与算法间的明显区别在课堂内不是永远存在的,在问题的定义中包含着问题的说明和求解该问题所需要的算法,通常这种选择已作为一种特殊程序设计概念讲授给学生。

在实际生活中常见的问题是问题的组织者仅仅与得到的结果有关,而与如何取得其结果无关。只要完成的程序大体上不是效率很低,一但拟出一个相应满意的问题说明,就能较容易选择解决所述问题的最好算法。选择和表达计算机算法的方法和效率比较规则地构成了问题求解过程的第二阶段,这将在下一章讨论。

为了使你取得一些程序设计的经验,书中后面有些练习有意留下一些不完善的地方。并用跟在问题语句后的圆括号内的注释来识别。这些注释在有意忽略了的重要说明处提示你一下。对这忽视不存在单一的正确答案或纠正的方法,必须列出并判断可能的所有抉择,然后选出你觉得最合理的解答。

程序风格 1:思考在前,编程在后。

Henry Ledgard 在他的程序设计原理书中(Hayden Book Co. Inc. 1975)Murphy 程序设计定律叙述了一个简单而很有趣的推论:"你的程序编写工作开始得越早,花的时间就越长。"虽然这个格言的真实性没有作出形式上的证明,但人们的经验早已指出了它的正确性。要想对任何问题都写出一个简单的程序而又没有组织好解答框架,这就好象用锤子、钉子和木材修房子而又没有图纸很快就会混乱不堪一样。

§1.4 例　子——查表

让我们用一个非常简单,且又是计算机程序设计中非常重要的问题——查表,作为第一个例子,这个特殊应用不是以一个完整的问题出现的。但是,它可以作为一个大型程序设计项目以一个单一的任务出现。例如,工资单程序中,我们需要在一张工资比率表中查找某一个职员的工资比率;在语言的翻译程序中,我们需要在词典中查出某一词对应于另一语言中等价的

词;在典型的罪犯审查中,我们需要在一个被盗汽车表中查出某一执照的牌号。

这类问题常常表示成以下形式短语:给出一个表的值,从表中查出所希望的值。并印出特殊值在表中或不在表中,若在表中,位于表中的什么位置。

这样简单的语句看起来似乎十分清晰,但是,如果在处理过程中进行正规的讨论,我们会发现实际极其粗劣和根本不可能接收的一个查表问题的描述。对下面的问题都未得到回答:

1. 表中的值会有哪些特殊类型?
2. 如何知道表中有多少值?
3. 这个特殊值是什么类型?
4. 表中可能的值的范围是多少?
5. 表中是否有重复的值?
6. 如果这个值找到了,希望产生什么样的准确信息?
7. 如果这个值没有找到又希望产生什么样的准确信息?
8. 对于上述两种信息之一还希望印出其它的附加信息吗?

所有这些问题的答案都应该提供。如果以一个汽车被盗问题作为一个特殊例子,就可以发现与专家们讨论会把我们引向下面类型的问题语句:给你一张下面形式的 6 个字符的执照牌的标识符表。

第一、二字符:取 A～Z 中的字符。第三至六字符:取 0～9 中的数字。表中最后一个执照牌是特殊值"AA0000"它唯一地用来表示表的结束。你得到的一个关键值与表中的其它值具有同样的形式。

设计一个程序查找这个关键值是否在表中某个地方,如果在表中印出下列信息:

PLATE NO."ccnnnn"REPORTED STOLEN

这个问题语句比前一个更为具体了,但这个问题语句也是很不完全的。再者,前面定义的语句,它忽略了下面情况时,我们应该做些什么。

1. 我们要查找的特殊值的错误形式,例如 ABC123。
2. 关键值在表中不只出现一次,查到的是第一次出现的还是全部?
3. 处理完全单一的关键值后是停止呢? 还是有一组不同的关键值还要处理?
4. 这个表是直接查找还是要从某一特殊输入设备将其读入?

对上述问题的正确答案应由正确抉择的讨论来决定,例如处理无效数据问题可以简单地加上下述语句得到解决。

如果关键值不符合执照牌表所规定的形式时,印出下面的信息。

cccccc

INVALID FORMAT——PLEASE CHECK AND RE-ENTER

跳过该数据处理其余部分。

第二个问题是重值问题,处理的选择性很大,一个正确的而又简单的是给出一个他们需要查找的信息的修正值。一个可能的办法是修改以前的定义,叙述如下:写一个程序如果关键值表中至少出现了一次时则印出下面的信息:

PLATE NO."ccnnnn"REPORTED" nn"TIMES

这里的 nn 是指执照牌在表中出现的总"nn"次数。这个问题的完全定义如下:

给你一个具有两个不同部分的数据集合,第一部分是被盗汽车牌号的主表,每一条数据都

包含下面形式的 6 个字符的执照牌标识符，第一、二列：A～Z；第三至六列：0～9。

表中最后的执照牌号是特殊值“AA0000”，它唯一地标志表的末尾，整个表当前存储在一个标有 PLATES 的磁盘的文件上。数据的第二部分在键盘上，包含了上述的同样形式的执照牌号，如果这个数据集合包含有一个错误形式的值印出下面的信息：

cccccc

IMPROPER FORMAT—PLEASE CHECK AND RE—ENTER 并忽略这条数据的处理。

设计一个程序能从键盘上输入一个执照牌号，并查出该号码在主表中所有出现的次数。如果该执照牌号在主表中未出现那就印出下面的信息：

PLATE NO."ccnnnn" NOT REPORTED STOLEN.

如果该号码在主表中至少出现了一次则印出下面信息：

PLATE NO."ccnnnn" REPORTED "nn" TIMES

这里"nn"是执照牌号出现在主表的总次数。对所提供的每一个数据重复这个过程。

最后一个问题语句与前面给出的问题语句比较，得出的结果是定义了一个要求解的问题是清楚的、无二义性的语句，需要的输入和将产生的输出得到了清楚的描述。如果这个语句是可以接收的，则它将是下一步程序设计的基础，不论是哪种情况下有这样一个完整的定义都是极其有用的。

对于实际形式，这里描述的查表操作的计算机算法的设计将在下章进行讨论。

练　习

❶

1. 对下列每一个一般性问题，为了使问题语句无二义性，试叙述问题还需要回答些什么。然后，任给一个必要的假设，拟出一个可接受的问题定义格式。

*a. 找出前 N 个素数。

b. 生成一张某英语课文中所有字母的频率表。

c. 计算期末测验的平均分数。

d. 将一张表的数，分类成数字序列。

e. 根据每天的时间，计算一周的实得工资。

2. 一个货物仓库，希望它的存货清单控制系统计算机化，仓库老板希望有一个程序，能始终监视现有的每 3000 件批发销售的价格和每一件的另售价格以及有关的信息。分析这种情况并表达出这种问题可能的输入输出定义，并考虑以下的一些情况：

a. 货物老板需要的信息。

b. 承订商需要的信息。

c. 股东或政府所需要的逐年逐月增加的信息。

d. 所需要的信息怎样收集和准备。

---

❶ 带有星号的数或字母表示该题的解答在书末的解答中已给出。

第二章 算 法

§2.1 引 言

算法这个概念是计算机科学中的一个基本概念，通常可以把算法看成是解题的一组方法或者是一组说明，或者是一个怎样得到所希望的结果的秘诀。我们的工作时时刻刻都在和秘诀打交道，有时，我们常常不把它们正规地称为算法。如下面的洗发剂标签上的说明也可以称为一个“洗发算法”。

1. 浸湿头发；
2. 涂肥皂沫；
3. 漂洗；
4. 重复进行。

由于它在某些方面与算法的定义不完全相符，所以实际上是不正确的。

更一般地，把算法定义为“一个运算序列”，当该序列执行时，总要产生一个结果，并在一定时间内结束。

一个运算序列指的是，对于算法中的每一个步骤，其后一步都是明确定义的，即是说，完成了一步后应该必须知道到那里去找下一步。例如排列顺序可以通过整数，表明用步骤来规定并遵守下面的规则：

除了告诉了做其它步或已做到最后一步外，在完成了第(i)步后就继续做第(i+1)步，i=1,2,…，换句话说，执行顺序可以用位置隐含地规定，完成了第i行的运算后就执行下一行(i+1)行的运算，有时不希望按照这个规定的顺序执行。事实上在算法中某些步不能继续到(i+1)步，而应接着去作第(K)步，不管怎么表示这个说明，下一步的标志应该是极其明确的。前面讲过的“洗发算法”的一个问题的第四步是模糊的，因为没有明确表明哪些处理步骤是可重复的。

运算序列概念是从头到尾的固有想法，一个算法必须总是一个明显的起始点和一个更明确的终点，起始点可以暗示，通常假设从第一步开始，如果多于一个起点时，从那里开始运行将产生二义性。这就打乱了顺序，然而，在算法中有一步或多步作为停止的标志是可以接受的，甚至是希望的，因为经常是，根据所处理的特殊值，把问题可以分成两个或多个不相交的段，但是不管执行哪一段，都希望执行完这段后就停止。

1. START

2 决定把问题分为可能的三种情况

CASE1	CASE2	CASE3
3	6	9
4	7	10
5STOP	8STOP	11STOP

这个框架仅仅有一个确定起始点和三个确定的终点，这组执行的步骤是(1,2,3,4,5),(1,2,6,7,8),(1,2,9,10,11)。然而一个或多个停止标识符的存在不一定保证真正能停止。例如有组

无意义的(从程序设计的观点来看是可怕的)运算,是很明显的。

```
      ⋮
5  GOTO  STEP7
6  STOP
7  GOTO  STEP5
      ⋮
```

要想有一个正确的算法,我们必须保证执行该算法时能最终在一个结束标识符上停止,在有限步之后停止。理论上其条数是1条或者是10的1000次方条都无关紧要,当然,实际上可以拒绝一个花时间太多的正确算法,因为它是不实际的或者是无效的。例如在国际象棋上,从一个给定的位置把所有可能对策的详细探索的算法能正确地规定出来,大概要花费大约10的40次方年的时间,才能研制出这种对策的计算机。另外,对任何随机的数据集,不管是怎样的病态集,算法必须在有限的时间内结束。

有关算法的基本问题,关系到在算法的每一步中所能写出的运算类型。即是说,应该构成什么样的模块我们才能组成一个算法?这些构成的模块通常称为原始部分,必须对人或执行该算法的机器是易理解的和完全正确的。例如下面是苹果的烹调法。

1. 做一个硬盒;
2. 制作苹果的填料;
3. 将填料装入硬盒中;
4. 用37.5度的温度烘烤30分钟。

对于一个十分熟悉制作的人也许是可接受的,然而,第一步和第二步对大多数没有烘烤经验的人来说是不明确的。对于这些人来说那两步是不可接受的,因为烘烤原物应该象这样操作:

把X和Y相混;
测量X的配料份数;
搅拌X;
以Y度温度燃X,Z分钟/小时;
烧X直到规定的条件为止。

其意义更为明确一点,对大多数人来说,它们是烧原物的实际例子。然而,甚至这些简单语句,如果存在二义性和不清楚的地方时,也可能是不可接受的。

现在从更为正规的方法定义原始操作,执行算法的人或机器可以直接了解并完成,不必再分解成一些基本的步骤,它可以组成最完善和复杂的操作。下一节的主要目的之一是设计一组可以接受和满足要求的原始操作,将在计算机算法设计中使用他们。

总之,一个算法,就是完成特殊任务的过程,该过程必须有下列性质:

1. 每执行完一步后,总是应知道下一步执行的标志;
2. 有一个明确规定的起始点和一个更为明确的终点;
3. 在所有的情况下,算法都在有限条步骤的执行后终止;
4. 算法由一些意义清楚,并对他执行的人或机器都是明了的基本部分所组成的。

再看洗发算法,可能发现它是不正确的,其原因有两个:其一,第四步中重复的步骤是模糊的,返回到第一步、第二步还是第三步呢?其二,没有规定算法的停止。它将无休止地重复,或

者最后直到我们用尽洗发剂、热水、或全部精力！下一节将提供必要的工具，使得写出的算法适合上述需要。

§2.2 拟 定 算 法

拟定一个求解特殊问题的算法的过程，是一个需要大量尝试的试凑过程，在解答和为测试其正确性的检查时，程序员要进行一些初步的尝试。他们发现的错误常常导致现有的算法进行插入、删除或修改，并重新开始。这种精练过程要继续到提问人满意为止。这时的算法才基本上是正确的，才能准备执行。在拟定算法中获得经验越多就越接近于正确的解答，而很少需要修改。然而，无论如何，程序员一开始就应把拟定算法作为一个单一的步骤来考虑，这一点需要再次强调，因为书中的算法都能简单地表示成正确的解答。然而，对于提问者或其它人在适合于算法的设计中他们表达的是最后一步。这个设计是通过由不可靠开始和不能再改动为止完成的，其间经过了无数次错误的教训，这些错误对每一个人来说都是明显的，事实上，算法的最后方案本身仍常常含有错误，这些错误要直到作为计算机实际程序运行时才能发现。

例：查表

介绍算法过程的最好方法，是通过例子来进行。在第一章，对一个非常普通的查表的程序设计问题拟出了定义。为了正确地求解此问题，对此拟出一个算法。在本例中我们使用的是最简单和最明显的顺序查找技术。首先，查看第一个元素，接着第二个元素、第三个元素，找到我们需要的那一个元素或到表末尾为止。如果表没有特殊的顺序（比如按数字排列或按字母排列）则顺序查找可能是最好的技术，然而，如果这个表已被组织成某种格式，则这种没有发挥其格式的优点的顺序查找法常常是效率最低的技术。这一点很重要，将在后面详细地讨论它。

对于这个例子，假定包括了有 N 个成分，分别为 a1，a2，…，an，并且要查找的特殊值命名为 KEY（关键值）

第一次尝试：

a1 等于关键值吗？

a2 等于关键值吗？

⋮

an 等于关键值吗？

看起来，似乎是最简单易行而又符合逻辑的解答，但当表变得非常大时，必要的语句的数目变得使人难以容忍。这忽略了程序设计最基本的重复技术，应写出仅有一条而能连续执行多次的操作来代替写 n 条不同的操作，这种算法表示了相对地，不依赖于正在使用的特殊数据集合。例如在例子中，可以用一个变量 i 作为一个指示器或者一个索引，用以指出表中常数的位置。当改变 i 的值时，表达式 ai 就依次指向 a1，a2，…，an。

假定存在一个算法的基本部分称为 START，它必须作为算法的第一行出现一次，而第一个可执行的单元部分则紧跟在 START 之后，由于算法也必须最终停止，类似地采用一种约定，即存在一个被称为 STOP 的基本部分，它可以在算法中出现一次或多次，当执行到 STOP 时，算法的执行就停止。另外，我们还应设置一个称为 END OF THE ALGORITHM 的基本部分，它必须放在最后的一行，START 和 END 部分总是要大写，醒目地表示算法的开始和末尾。

```
START
    语句 1
    语句2
      ⋮
    语句 N
END  OF  THE  ALGORITHM
```

下面仔细观察初始尝试中的第一行，和它们的每一行都表示为表的特殊值等于关键值吗？如果相等我们怎么办？不等怎么办？这些都没有说清楚，我们需要的是它对下一步有什么影响的回答。实现这点最普通的方法是用 IF－THEN－ELSE 基本部分。

```
IF    condition    THEN    operations
                  ELSE    operations
```

在 IF 部分规定的条件，表达了能执行算法的人(或机器)了解和求值，并产生真值和假值的任何条件。如果指明条件为真，则执行包含 THEN 子句的全部运算，而跳过 ELSE 子句中的全部运算，如果条件是假则跳过 THEN 子句的运算，而执行 ELSE 子句中的运算。如果两个子句中任何一个或两个的运算组很长时，则为了避免混淆，我们可以使用一个括号“[”将所有属于同一子句的部分逻辑地组成一组。

```
IF  condition  THEN  [ s1
                     ⋮
                     sn

ELSE                 [ s1
                     ⋮
                     sm
```

下面是使用这个基本部分的例子：

```
IF  x<0    THEN  计算√(-x)
           ELSE  计算√x
```

如果条件 X＜0 为真，就执行 then 子句，求－x 的平方根(负数的负数为正)，如果条件 x＜0 为假，就执行 else 子句，即计算 x 的平方根。

另外打印出算法所产生的结果的基本部分是很需要的，输出部分是很简单。

```
write  values
```

并且约定在引号“ ”内的值作为直接打印的信息，不在引号内的字母和数字表示当前印出的值。如果 r 是想印出的计算半径，则输出语句为：

```
write“the  radius  is”,  r
```

最后假定计算机具备了所有算法逐算的功能，于是下面的命令：

```
Add        a    to      b          a 加 b
Subtract   a    from    b          a 减 b
Multiply   a    times   b          a 乘 b
Divide     a    into    b          a 除以 b
Decrement  a                       a 减 1
Increment  a                       a 增 1
```

或任何与此类似的东西都被认为是算法的基本部分。

现在用已讨论过的内容再次写出其算法：

第二次修正，

```
START
If  ai  equals  the  key
Then [write"found  at  location",i
      stop
Else [increment  i
      repeat
END  OF  THE  ALGORITHM
```

这里采用了以前我们感兴趣的计算机算法形式，重复地执行一组语句。但该算法还不是很正确的，因为在算法的第二行涉及到 a 的第 i 个元素。然而，无论是索引值还是表的值都未作明确的规定，未初始化的错误(即在未给一个变量或一个记录的值之前就企图使用它)。任何一个算法都应在使用一个记录之前，明确规定怎样将一个值赋给它，两种普通的赋值方法是通过以使用内部和外部的数据，内部数据是由算法的本身所产生，外部数据是指数的特殊值，它独立存在于执行算法的人或者机器外部输入设备中。这些值可以由人或机器通过从这个装置读入数据来获得的，我们假定这两种方法为运算的基本部分。

内部数据：将变量置一个值；

例如　将 x 置 1；

将索引值置成 a 与 b 之和(当然，a、b 的值必须预先规定好)。

外部数据：读入值

例如　读入 x，y 和 z 的值；

读入元素 a1，a2，…，an。

再看算法的最后一行，发现重复语句是不能接受的，正如与"洗发算法"一样，这既没有指明重复什么，也没有指明重复多少次。

算法的特征是频繁地使用重复，实际上，写出的每个算法都是包含有一组重复若干次的语句，在拟定算法时，特别要考虑循环的基本部分，规定一组执行语句和执行多少次的标准结构是令人感兴趣的，使用重复的大多数方法循环次数都是固定的，或循环到某一个值为止。对这两种情况提供基本部分。

a)

```
While  condition  do
        s1
        s 2
        ⋮
        sn
End  of  the  loop
```

在执行时，首先检查条件，如果条件为真则执行所有的语句 s1，…，sn，一直执行到条件值为假时为止。如果一开始条件值就为假则一组语句一次都不执行。

b)Repeat　the　following　count　t　ime