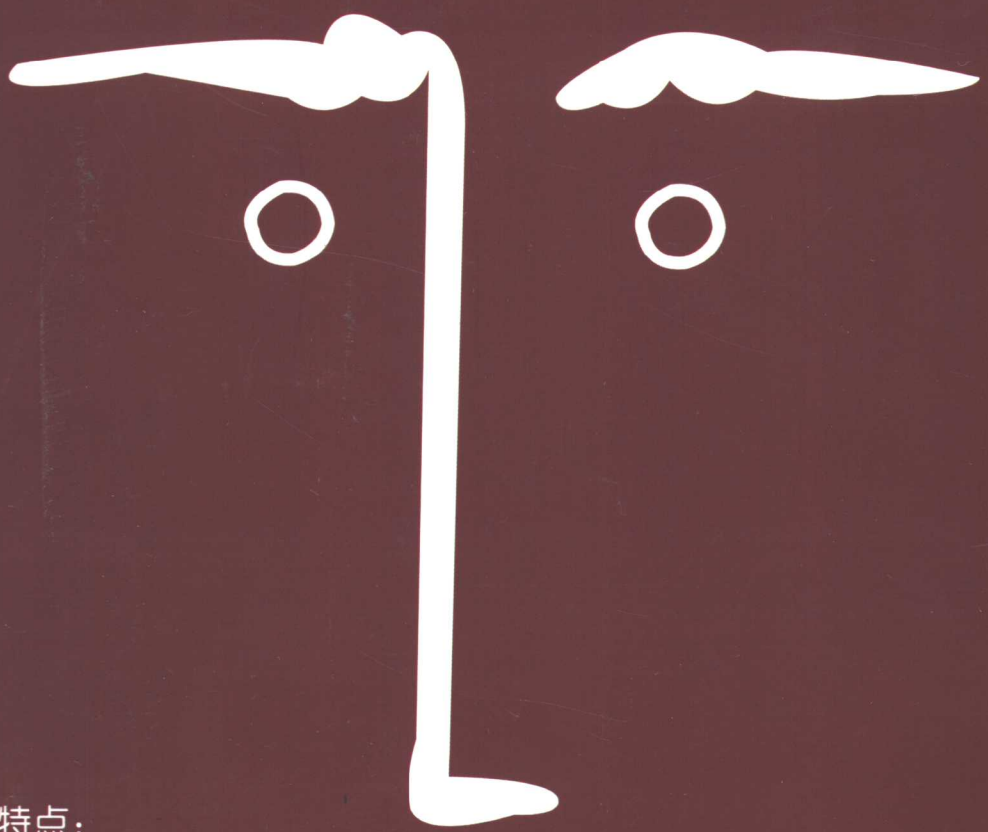
C语言名题精选百则

技巧篇

冼镜光　编著



本书特点：

◎ 本书体例别具一格

◎ 囊括100则经典C语言程序设计题

◎ 精辟文字全面解析C语言程序设计技巧

◎ 免费提供完整的源程序下载

机械工业出版社
China Machine Press

C语言名题精选百则

技巧篇

冼镜光　编著

机械工业出版社

本书收集了100则C语言程序设计题，共分9类。第一类比较简单，主要希望读者了解到本书的题目、解法与其他书籍之间的差异；第二至六类分别是关于数字、组合数学或离散数学、查找、排序、字符串等方面的题目；第七类列出了一些不太容易归类的题目，如Buffon丢针问题、Dijkstra的三色旗问题等；第八类则收录了一些有趣的、娱乐性的题目，如魔方阵等；第九类题目相对较难，且多数是程序设计的名题。

本书在组织方式方面别具一格，问题与解答分开介绍。前半部分，把所有问题按归类的方式收纳在一起，说明问题的内容和分析；后半部分是问题的解答。另外，每个题目还包含习题、程序和参考文献，习题部分要求读者延续解答中未完成的工作，而程序为读者提供了一个很好的参考。详细完整的源程序放于网站中方便读者下载学习使用，而参考文献是读者进行进一步学习论证的资料来源。

本书内容新颖，实用性强，可作为高等院校师生学习C语言的参考书，也是C语言爱好者的自学读物。

本书版权登记号：01-2003-7483　图字：01-2003-7483

图书在版编目（CIP）数据

C语言名题精选百则技巧篇 / 冼镜光编著. -北京：机械工业出版社，2005.6
ISBN 7-111-16376-1
Ⅰ.C…　Ⅱ.冼…　Ⅲ.C语言-程序设计-习题　Ⅳ.TP312-44
中国版本图书馆CIP数据核字（2005）第024545号

机械工业出版社（北京市百万庄大街22号. 邮政编码 100037）
责任编辑：王　平　　版式设计：李永梅
三河市宏达印刷有限公司印刷 • 新华书店北京发行所发行
2005年7月第1版第1次印刷
787mm×1092mm 1/16 • 32印张 • 786千字
0001-5000册
定价：44.00元

凡购本图书，如有缺页、倒页、脱页，由本社发行部调换
本社购书热线电话：（010）68326294
封面无防伪标均为盗版

序

一年多前，作者就有了编写一本C语言问题集的念头，其中几度动笔，但终究还是把手稿悉数毁掉，始终不能确定为什么一直不能满意自己的成果，直到某天突然发现问题的症结就在于“读者要一本什么样的问题集”之后才恍然大悟，正是因为想在书中放入太多不兼容的东西，才造成数次易稿的问题。

读者究竟要一本什么样的问题集？作者思索再三，也读过一些相关的著作，发现读者不能只有一本问题集。首先，目前市面上的问题集大致可以分成两种（不以使用的语言来分）：第一种比较简单，目的是用一些程序来说明某种语言的用法，因此定位于入门学习程序语言的读者；第二种的内容比较丰富，从语言的用法一直到最后一部分的数值分析，显然定位于想深入学习的读者。但作者认为，一本这样的书对于想深入学习的读者而言，可能有一半（讲解语言的部分）没有用，而对刚入门的初学者来说，同样也有一半（数值分析的部分）没有用，即对两者都是浪费。如果书中还有基于某一科目知识的题目，那么读者所面临的就不仅仅是开发程序的问题，还有了解这门知识时的困难。

如果一本问题集会出现上述的问题，那为何不把它分开来做呢？但分开之后又要从哪一部分开始呢？环顾目前已经出版的类似书籍，其最终的目的都不过是教您如何编程而已。当然，正确性是程序的第一要点，但并不是惟一的；一旦入门且能够编写正确的程序之后，效率（Efficiency）就是另一个必须追求的目标，但这一点却鲜有程序设计的书籍探讨。这里所讲的“效率”，并不是指如何安排程序中的语句来加快速度；这样做固然可能会让程序的运行速度加快，但却不是一个治本的方法。就好像在骑自行车时，把身体练得强壮些可以骑得更快，但就长远来看，恐怕就不如一辆小小的摩托车了；换句话说，固然把程序语言学得很精通，可以迅速编写一个程序，但是如果采用了一种很慢的方法，则不如一个采用很快的方法，但没有用极佳的语言功能来编写的程序。本书列举的这种例子很多，读下去自然就会看到。所以，本书就从这一步出发，通过各个题目的讲解，教读者如何设计一个有效率的程序。书中有很多题目是读者耳熟能详的，那么是否想到过更好的做法呢？

收集在本书中的题目，有许多是知名的问题，它们散见于各类书本、教科书与期刊、杂志，有一些是学校的考题，另一些源于个人的研究，还有一些则出自名家的专栏，当然更有一些是名著的片段。在取材时，把它限定于中等程度以上，而且不包含任何数据结构的论题，因此最深的数据结构就是堆栈（Stack）、队列（Queue）以及排序中用到的堆积（Heap），对于与树状结构（Tree）、图（Graph）等有关的题目，一概不用，而留到后续的书中处理。至于那些简单的题目，一般教科书与参考书已经有很多，所以也不采录，但若采用，那么在解法上必定有独特、值得讨论的地方，或者有一种更有效率的做法。

本书绝大多数的题目都取自其他书、期刊论文，因此凡是能查出题目的来源和作者的，都在该题的参考文献中说明，以表示对原作者的敬意。如果您觉得某些题目很好，希望在自己的著作中引用，也请您能注明出处。

虽然本书是一本问题集，但并不希望读者看了题目就马上去读解答，显然没有经过深思就直接看解答不会有很大的作用，所以，本书的做法是一分为二，前半部分是问题与问题的说明，后半部分才是问题的解答与程序。另外，书中所有程序均放于网站中以方便读者下载学习使用。正因为如此，书中许多程序的主程序都没有列出来，在编程时已经考虑到这一点，这些程序的主程序只是提供一些测试资料以及显示结果而已，所有重点都放到各个函数中。本书所提供的程序免费提供读者参考、抄录，可以随意修改来配合学习，更可以提供给他人使用，但是读者不应该以它牟利，这正是公共程序（Public Domain Software）的意义；换句话说，未经本书作者允许，不能采用本书中任何程序或片段从事营利的行为。

本书是作者的一个新尝试，希望可以为学习编程的人员提供更高一层的信息与技巧，也希望您能在阅读、使用本书之余提供更多的改进意见。如果大家觉得这样的题材有趣而且实用，那么浩翰的书海中还有更多素材可以帮助我们再接再厉出版其他论题的专集。

尽管本书最后的程序设计工作只用了三个多月时间，但是过去一年多在图书馆中寻幽访胜的工作使我受益匪浅，最后要感谢妻子在整理这些资料时所付出的精力，与伴我昼夜不分甚至颠倒日夜工作的心意。

沈镜光

前　言

从某个角度来看，计算机科学是为编程服务的（吸收并分析来自各行各业、科目和范畴的问题，并且可能的话，找出一个高效率的算法来解决问题），而有了算法之后，编程就不再是一件难事了。所以解决问题的算法才是编程的关键所在，一个程序只不过是在实现它所使用的算法而已。如果程序有了问题，也就是有了 Bug，可能是对所使用的语言的语法、语意不甚了解而出了差错，或者是程序逻辑有了漏洞，其后者就是算法可能有问题；当然还有第三种可能，就是从算法到实现它的程序之间出了问题，原因是对所使用的算法了解不够而产生了误解，也可能是对语言了解不够导致使用错误而无法完全实现算法的功能。总之，要编写一个程序，除了要彻底了解所使用语言的特性之外，还需想出一个解决问题的算法。

当然首先不要被“算法”这个名词吓住了，它也并不是高不可及。事实上，有不少科学家正在孜孜不倦地研究、设计各式各样的算法。作为一个程序员，即便是一个刚学习程序设计的人，在把程序语句写下来之前，必定会先构思一番。例如，解 $ax^2+bx+c=0$ 这道一元二次方程式的题目，他可能会想到先算 b^2-4ac 是否不小于 0，如果是，再用课本中的公式来解，这就是算法。

本书并不像其他“问题集”、“问题演练”的书一样，把针对程序语言的问题与设计算法的问题，甚至于数值分析的问题都集中在一起，而使读者觉得语言问题不值一提（如果是一个编程高手），或者觉得解方程式等问题高不可攀（如果根本不懂数值分析）。本书的做法是，针对语言的问题，读者可以去读程序语言的教材、参考书，在那些书中自然会有详细的说明；对于一些有特殊背景的问题，如数值分析等也不作介绍，于是书中的问题都可以说是独立的，所无法独立的就是对问题的解法。因此，对于这一类型的题目，将在书中各处多安插几个，让读者见到它们的机会大一些。总之，本书所着重讲述的不是程序语言的问题，而在于如何设计一个解决问题的算法。

但是，算法也有好坏之分，如果在一本入门书中作者不慎选用了一个差劲的算法，而使读者对它有了第一印象，以后再想扭转就很难了。有一个很好的例子，求组合系数 C(n,r)，它的定义是：

$$C(n,r)=\frac{n!}{r!(n-r)}$$

很多作者的做法是先算出 n!、r!与 (n−r)!，再把 C(n,r) 求出来，这是个很自然的解法，但是作者却觉得这个方法不够美妙，而用：

$$C(n,r)=C(n-1,r)+C(n-1,r-1)$$

因此，程序就可以写成递归的形式，先以递归的方法算出 C(n−1,r)，再用递归的方法去算 C(n−1,r−1)，最后把结果相加。程序比较短小，看起来也很漂亮（寥寥数行）。但又有新的问题出现，在执行程序时付出的代价可就太大了。当 n 较大，而 r 接近于 n/2 时，用

这种方法会比用第一种方法慢很多。不仅如此，许多人认为第二种方法只用加法，而第一种方法还要用乘法，因此第二种方法不但“漂亮”而且“快速”，但事实上这是个“主观”而不正确的结论。因此，就回到“有效率”的算法的课题上。

什么样的算法才算是有效率的呢？简单地说，它要快，而且使用空间要小。这里的“快”不是指 CPU 时间。因为，CPU 时间是相对的，在一部 Cray 上，上面的第二种方法可能要比在 PC 上用第一种方法快许多倍，所以不能用 CPU 时间来度量。一般而言，科学家会根据算法的程序一共进行了多少个运算（+、–、*、/、比较等），或处理了多少个数据作为评估的标准。本书不深入探讨这个问题，但在需要的地方也会稍微进行介绍。本书的程序都会选用一个较快并且一般读者都能接受的方法，有时还会介绍一个比较慢的方法与快的方法之间的差异。一般而言，我们着重于用较快也就是效率高的算法来解决问题。

总之，本书放弃了那些专门的解说程序语言用法的简单题目，而倾向于程序设计的内涵，也就是如何设计一个算法。对于有多种方法可以使用的题目，采用效率高的那一种。正因为如此，本书将不仅仅是一本入门书，还需要读者对程序语言、程序设计具有一定的基本知识。本书结合 C 语言来展开介绍，但 C 语言并不是一个很适合用来做这些实例的语言，其独特且有些怪异但却很方便的语法，往往会使一个读程序的初学者无所适从，因此在适当的地方，都作了提示。

本书的安排方式

本书共收集了 100 个中等难度以上的程序设计题目，可以粗略地分成 9 类（也许分类方式并不很适当，但这是作者觉得比较合理的分法），每一类中的题目在观念、解法上存在一些类似的地方。

第一类主要希望读者能够了解到本书的题目、解法与其他“问题集”书籍之间的差异。这几个题比较容易，但是有些程序员就是不会活用题目所给的条件，经常设计出一些“正确”但“效率”差的程序或算法。这几个题具有很好的代表性，希望能够做到本书所要求的境界，而不致于写出没有效率的程序。

第二类则进入正题，这十几个问题的对象是数字。例如，如何填数值的字谜？如何求出 2～N 的质数？如何求 x 的 n（正整数）次方？如何计算 Fibonacci 数，$f_n = f_{n-1} + f_{n-2}$？如何求组合系数 C(n,r)？如何求 n!等。乍看之下，似乎都会了，确实每一个题目都是熟悉的，因为都是程序设计课中的论题，但是在重做这些题目之前，首先看看是否能达到本书的要求。

第三类则比较少见，属于组合数学（Combinatoric Mathematics）或离散数学（Discrete Mathematics）的题目，包含了产生一个集合的所有子集、Gray 码、产生 1～n 的所有排列、产生 n 个元素中任取 r 个元素的所有取法、产生 n 个元素的所有 k 个元素的子集、n 个元素集合的所有分割（Partition）方式，以及如何把一个正整数写成若干个正整数的和等。其中有些题目，如排列方式、组合方式已经很普遍了，但有一些则不是，相信这部分对学习组合（或离散）数学的朋友会有所帮助。

第四类的主题是查找（Search），重点是在一群数中找出满足条件的元素。这一部分基本上是二分查找法（Binary Search）的各种变形，有些题目则是推广，而且难度也有一些

提高，但解法十分完美。

第五类的主题是排序（Sorting）。为方便起见，把几个有名的方法都写成了程序供读者参考。详细介绍：如何加快插入式排序（Insertion Sort）；如何在快速法中处理值相等的元素，使得排序完成之后值相等的元素仍然维持原来的顺序；如何让快速排列法在排正整数时不必递归，也不用堆栈；如何找出中位数；堆积法（Heap Sort）与它的变形；合并排序法（Merge Sort）；桶子法（Bucket Sort）等基本技巧。此外，还有不少应用到排序的程序也收集在此。例如，若数组中元素大多数相同时如何处理等。排序在计算机科学中是一个相当重要的技巧，但一般书都只介绍如何排序，本书则希望通过这些应用实例，使读者知道排序的应用不止是排列成绩、订单这些简单数据而已。

第六类是字符串（String）方面的课题。对很多人来说，这可能是最困难的一部分，因为大多数的书在这一类型的问题上都介绍不多。在该部分，通过问题与解答，介绍了Knuth–Morris–Pratt 方法与化简的 Boyer–Morre 方法，这两种方法在寻找字符串 s 中是否有一个特定的字符串时有相当杰出的表现。此外，还有几个有关部分序列（Subsequence）的问题，例如找出两个字符串的最长共同序列、把一个字符串编排成另一个字符串等。

第七类是一些不太容易归类的题目，如 Buffon 丢针问题，Dijkstra 的三色旗问题、找零钱问题、背包问题、无限位数的整数运算、最短路径问题等。这些可以算是杂项，相当富有变化，既有很单纯的（比如文字到整数的转换），又有含相当技巧性的（比如向量的分类）等。

第八类收录了一些有趣的、娱乐性的题目，如魔方阵（Magic Square）、N 后问题、武士巡逻、环游世界、一笔画、河内塔、生命游戏等。这些题目并不是广义的、两人对局的游戏，因此说成娱乐性的题目还恰当些，但也不排除以后会添加对局的游戏题目的可能性。

最后一类对读者来说是一个挑战。与第一类正好相反，这些题目都比较难，大多数是程序设计的名题。例如，要求找出圆上的一组点有没有 4 个可以组成长方形、机器人走完一段固定路程之后回到原点一共做了多少次 360° 的旋转、一组数中连续若干个的和与积的最大值、如何找出名人、如何找出投标结果中得票过半数的人、如何找出一对一函数，以及找出最长的递增部分序列等。

除了第一类的题目建议每道都做一次之外，其他的题目没有必要一题接一题地依次演练，读者可从中选择想做的题目来磨练一番。另外也需注意，分类的目的是方便把题目收集在一起，并不是指哪一类题目特别难、哪一类题目比较简单，而且同一类题目的难易程度也并不是愈在后面就愈难。作者认为，难易之分并没有绝对的标准，往往因人而异，也因人的学习背景而异。因此，如果您一下就把题目破解了，做得至少与本书的解答相当，那就恭喜了；但如果一时做不出结果，也不要灰心，多思考、多转几个弯、多变换几个招式，或许就会得到好的结果。这些思路历程就构成您以后解题、就业、工作的经验，切记千万不要看了题目就去看解答。

题目的组织方式

本书虽然也是一个问题集，但在组织方式方面却与其他同类书籍不太相同。虽然书中也是一个题目接一个题目讲解，但是问题与解答是分开的。在前半部分，把所有问题分类

收纳在一起，而问题的解答、说明与程序则是在下半部分。这种做法，就是希望能先消化了题目，经过相当程度的思考之后，觉得实在有必要或者自己也做出了结果，希望与本书的做法作个比较之后，才去翻阅解答的部分。

每一个题目都从问题的部分开始，在此说明该问题是什么，要做些什么事。接着就是问题的说明，因为做题目的人背景不同，也许问题部分过于简略而要作进一步的解说，这就是“说明”部分的工作。通常，在此会对问题作进一步的解释、分析，可能的话，还会指出适当的提示，好让读者有一个下手的地方。有些题目中，也会介绍一个大多数人都会想到的方法，分析此做法效率不高，而要求寻找更好的方法；换句话说，程序的正确性固然重要，但到了某个阶段之后，效率就是一个重要的课题，而要求的重点就是效率。

后半部分就是各个问题的解答，在解答中会把问题部分重复一次，但不会再讲说明的部分，而是立刻进入解答问题。绝大多数的解答都由解答、习题、参考文献、程序 4 部分构成，但是也有一些题目会视具体情况而有所增减。解答部分的目的，是告诉读者编写这个程序背后的想法、技巧、数据结构等要件，通常会很仔细地讲解要如何考虑这个问题，用什么数据结构会简化程序设计的工作。重要的是，绝大多数的题目到最后都会就所用的方法作一个效率上的分析，这些分析都是非正式的，在此也避免使用令人困扰的符号。另外，还可以帮助读者了解如何分析一个程序的效率进而改善自己的程序。

习题部分通常是要求延续解答中未完成的工作，例如证明一些在解答中不会做到但却有用的简单结果，就解答的效率作进一步分析或修改程序使其可以做更多的工作、多输出些有用的结果等，通常只要模仿或修改本书提供的程序就可以完成，但还是值得花点工夫去完成；另外还有极少数的题目，本身就是一个独立的具有挑战性的问题。一般而言，较难的习题书中都加上了提示，但是不用提示也应该可以解答，因为解法通常不止一个，所以提示部分仅供参考。

参考文献部分是本书的特点之一，在此可以知道这个题目是谁想出来的，解法是谁做的；如果这种方法并不是最好的，可以在什么地方找到更好的解法；也或者，有人在什么时候有过类似或其他的解法。因此通过这些参考文献，以此为起点，就可以发现更多的资料与另一个天地。

程序部分就是该题目的实际程序，它完全反映了解答部分的讨论。但需要注意，绝大多数的程序都已经写成函数的形式而由主程序调用，主程序方面只不过提供一些执行时的测试值而已，所以在这种情况下，为了节省篇幅，主程序并没有附在书上。除非可以完全不必改变而使用本书中的其他函数，所有的函数都会列出来供读者参阅。如果想知道主程序的细节，可以查看光盘中的内容。本书中的所有程序也都分类存储在光盘中，请看下面的说明。

本书中没有的内容

第一，强调本书并不讨论如何运用某一种程序语言，所以不可能在本书找到。第二，本书既然指明了“技巧篇”，所以问题就以发挥技巧为主，因此那些一眼就可以看出结果、比较没有变化的题目就排除在外了。第三，也是最重要的一点，本书的题目限定在数据结构以下，也就是说解这些题目可以不要数据结构的技巧，因而有比较广泛的读者群。事实

上，本书中所用到的技巧中最难的就是堆栈（Stack）、队列（Queue）而已，而且维数很低；倒是在第五类排序中的内容比较多，但对大多数人而言，这些都是耳熟能详的内容。至于在数据结构以上以及一些更为基本的题目，考虑在其他书中介绍。

本书下载文件

http://www.cmpbook.com/download/16376.rar

或 http://www.cmpbook.com/jk_xz.asp

目　录

第一部分　问　　题

第1章　序　　曲

问题 1.1　最长平台（PLATEAU.C）

已知一个已经从小到大排序的数组，这个数组中的一个平台（Plateau）就是连续的一串值相同的元素，并且这一串元素不能再延伸。例如，在 1,2,2,3,3,3,4,5,5,6 中 1,2.2,3.3.3,4,5.5,6 都是平台。试编写一个程序，接收一个数组，把这个数组中最长的平台找出来。在上面的例子中 3.3.3 就是该数组中最长的平台。

【说明】

这个程序十分简单，但是要编写好却不容易，因此在编写程序时应该考虑下面几点：

（1）使用的变量越少越好。

（2）能否只把数组的元素每一个都只查一次就得到结果？

（3）程序语句也要越少越好。

这个问题曾经困扰过 David Gries 这位知名的计算机科学家。本题与解答取自 David Gries 编写的有关程序设计的专著。

问题 1.2　支配值数目（GT_COUNT.C）

已知 f[ ]与 g[ ]两个整数数组，元素都已经从小到大排列，试编写程序算出 f[ ]中每一个元素比 g[ ]中元素大的个数的总数。换句话说，f[0]比 g[ ]中多少个元素大、f[1]比 g[ ]中多少个元素大等，这些值的总和就是所要求的答案。

例如，如果 f[ ]中有 1,3,5,7,9，而 g[ ]中有 2,3,4,7,8，比 g[0]大的有 f[1]～f[4]，比 g[1]大的有 f[2]～f[4]，比 g[2]大的有 f[2]～f[4]，比 g[3]大的是 f[4]，比 g[4]大的是 f[4]，因此答案是 4+3+3+1+1=12。

【说明】

与问题 1.1 一样，需要特别注意数组 f[ ]与 g[ ]已经排序。如果问题 1.1 能够做出完美的解答，那么本题也不难，相似的方法就可以得到高效率的程序。

问题 1.3 等值数目（EQ_COUNT.C）

已知两个整数数组 f[ ]与 g[ ]，它们的元素都已经从小到大排列好，而且两个数组中的元素都各不相同。例如，f[ ]中有 1,3,4,7,9，而 g[ ]中有 3,5,7,8,10。试编写程序算出这两个数组之间有多少组相同的元素。

就上例而言，f[1]与 g[0]为 3 是第一组；f[3]与 g[2]为 7 是第二组。

【说明】

建议不要使用下面的方法来编程：

（1）固定 f[i]。

（2）对于 f[i]而言，检查 g[ ]中是否有与 f[i]相同的元素。

（3）处理下一个 f[i]，即 f[i+1]。

因为 f[ ]与 g[ ]都已经从小到大排列好，所以应该活用这一个很强的特性。一个好的程序员绝对不应用上面的笨拙方法来编写程序，这样会做太多无意义的事。

为什么呢？因为 g[ ]的元素都相异，对于 f[i]而言，最多只会找出一个与它相同的元素，最坏的情况是把 g[ ]全部查完才找出相同元素（如果采用上面的方法），如果 g[ ]中有 n 个元素，需要查 n 次；但是若 f[ ]中也有 n 个元素，那么需要把 g[ ]查 n 遍，一共作 n^2 次比较才能找出结果。

试着找出一种方法，把 f[ ]与 g[ ]各查一次就可以得到答案（记住，活用 f[ ]与 g[ ]已经从小到大排列的特性）。

做完这一题后，建议继续做下一题。

问题 1.4 两数组最短距离（MINDIST.C）

已知两个元素从小到大排列的数组 x[ ]与 y[ ]，请编写一个程序算出两个数组元素彼此之间差的绝对值中最小的一个数，此值称做数组的距离。

【说明】

如果 x[i]与 y[j]是两个元素，那么|x[i]-y[j]|就是这两个元素之间的距离，所有这些距离的极小值，称为数组的距离。比如说 x[ ]有 1,3,5,7,9，y[ ]有 2,6,8，那么最短距离就是 1，因为 x[0]与 y[0]、x[1]与 y[0]、x[2]与 y[1]、x[3]与 y[1]，还有 x[4]与 y[2]的距离都是 1。