标准 FORTRAN 77 程序设计

高文培
马速成 郭玉田

长春地质学院

标准FORTRAN77

程　序　设　计

高文培
马速成　郭玉田

长春地质学院

前　言

今天计算机科学已深入到人类社会生活的各个领域，因此，在我国高等学校深入普及计算机科学的教育尤为迫切。遵照计算机科学的基础教学应以数据结构、算法和程序的设计为中心的指导思想，本书加强了数据结构和算法设计的内容。过去一般把程序设计和用某种语言编程序等同起来，这是不对的。程序设计应从问题分析开始；确定解决问题的方法；划分模块；确定每个模块的算法；选择一种语言来编程序；程序的调试；程序正确性的测试、验证；最后是文件编制，形成清晰的程序文件，以便提供使用和维护。程序设计在很大程序上与选择的语言无关，比如，问题分成多少模块，确定引入哪些个过程等等，所以讲授程序设计无论采用哪一种语言都没有关系。因为所有的程序设计语言都差不多，它只是用来描述算法的工具，故本书强调了程序设计的方法和算法。当然选择通用的，功能强的高级语言和能够提供程序设计实践的方便环境更好。总之，任何人只要精通一门算法语言之后，当需要使用其他语言编程序时，通过自学是可以做到的。

本书的突出特点是在各章中均强调了程序应具有简单性、易读性、可移植性和可维护性。为了达到上述要求，书中重点描述了处理、判定、循环的三种算法结构。对所有语句做了评述，对哪些认为不宜使用的语句集中列在书中最后一章介绍，这样就突出了主要语句的功能，使程序结构简明清晰。书中内容比较新颖，列举了大量实例，尤其是数据处理方面的例题较多，这对读者是有益的。因为今天计算机的应用范围最广的已不是科学计算，而是数据处理，所以书中也强调了文件系统的使用和字符处理的功能等。每个例题都先给出算法的描述，然后给出程序，以突出算法在程序设计中的作用。每章后面附了大量练习题，书后附录7给出了详细答案。

为了方便读者学习FORTRANⅣ，书中凡是FORTRAN77专有的语句和语法都做了说明，并在书后附录8中给出了FORTRANⅣ的语句表，供读者参考。

本书可作为高等院校学生、研究生、教师的教材和教学参考书，及函授大学、业余大学、广播电视大学的教学用书，亦可供程序设计人员、科研人员、工程技术人员、管理人员以及通用的FORTRAN用户学习使用。

参加本书编写工作的还有邹春贵、范兴业、傅春久、李德昌同志。

由于编者的水平所限，加之时间仓促，书中的缺点和错误在所难免，诚恳地希望广大读者批评指导，以便今后进一步修改。

长春地质学院
软件室
地质部长春计算站

1982.8.1

目　　录

第一章 FORTRAN语言发展概述

FORTRAN语言最初是为解决数值计算问题而发展起来的程序设计语言。今天，它已成为科学技术领域里使用最广泛的程序设计语言，在科学计算的程序设计世界中它一直占据着统治地位，并且科学家和工程师的确把它看作是必不可少的有力工具。与此同时，程序设计语言专家们也指出了它防碍好的程序设计实践的缺陷，并且痛惜由于广泛地使用了FORTRAN而影响现代更先进的程序设计语言的发展。为了弄清为什么FORTRAN语言流行的如此广泛以及为什么它又受到批评，了解一下FORTRAN的历史是必要的。

FORTRAN的研制工作可以追溯到1950年，例如Sammet于1969年引证IBM公司数学公式翻译系统的初级报告资料，论述了FORTRAN于1954年基本形成，1957年在IBM 704机上实现了FORTRAN编译系统。1958年6月IBM704机的FORTRAN Ⅱ被正式发行，这个新文本比原来的文本做了许多重要的改进，如子程序的概念。1958年到1960年期间，在IBM709、650，1620和7070系统上发行了与此类似的FORTRAN文本。

从1960年初开始，在各种计算机上实现FORTRAN编译系统迅速发展，各个计算机工厂都纷纷要求为他们自己的计算机提供FORTRAN编译系统，于是在科学程序设计世界中，FORTRAN语言迅速地发展起来。在发展过程中，对原文本不断地修改和扩充。IBM公司本身也在不断地发展FORTRAN，他们首先在IBM7090机上研制了FORTRAN Ⅳ编译。FORTRAN Ⅳ比FORTRAN Ⅱ增加了许多新的内容（如复型和双精度型运算），并且也去掉了那些依赖于机器的语句，这些改进是非常受欢迎的。

1966年标准

由于FORTRAN在最初的几年发展的很快，因此，早在1960年以前就新提出对FORTRAN系统标准化的问题。1960年美国国家标准化协会（ASA）成立了计算机和信息加工委员会，它隶属于商业装备制造商协会，从事一般程序设计语言标准化方面的工作。1962年5月该委员会成立一个工作组，以开展美国标准FORTRAN语言的规划工作。于1966年发表了两个标准文本，即美国标准FORTRAN和美国标准基本FORTRAN。从历史上看，标准FORTRAN相当于FORTRAN Ⅳ，标准基本FORTRAN相当于FORTRAN Ⅱ。然而大多数计算机工厂和用户对FORTRAN Ⅱ是不感兴趣的。

回顾1966年标准文本令人失望的地方有两点：第一，在文本中隐藏着少量的二义性和不合理的限制。第二，1966年标准没有使用公式化的方法来描述FORTRAN语言。没有参考在几年前就用公式化的方法来定义ALGOL60语言的成功经验。而采用英语散文来定义，容易造成二义性和误解。1966年标准没有给出清晰的语法定义，使许多编译者遇

到一些困难。关于对FORTRAN标准的说明，请读者参考美国国家计算中心1970年出版的标准FORTRAN程序设计手册，它全面地阐述了FORTRAN。

1977年标准

在1970年，美国标准化协会的X3J 3 技术委员会着手对新的 FORTRAN标准公式的研究工作，使用了下列五项原则做为研制准则：

（a）程序的可移植性，即FORTRAN程序在不同的处理系统之间可以互换；

（b）现存的标准和实际的具有一致性。即与FORTRAN66相容；

（c）增加符合需要的新特性，新功能；

（d）具有高效运行的能力；

（e）允许扩充，即允许此语言将来的发展。

（f）用户满意，能为绝大多数用户所接受。

在这个委员会的努力下，于1977年对于FORTRAN77 的新标准取得了一致意见。1978年得到美国国家标化协会批准，并再一次描述了美国国家标准FORTRAN77全集和子集。

FORTRAN77全集语言包括了一些1966年标准没有的特点，这些特点又很有意义，同时又从1966年标准的不合理的限制中解放出来，使程序设计者感到方便。

结　论

FORTRAN作为科学程序设计语言来说是使用最广泛的语言。由于下述原因可以预见在未来也必将如此。

（a）FORTRAN编译系统是最广泛的，甚至没有FORTRAN 编译的微型计算机都是罕见的。

（b）在其它各学科中已经使用 FORTRAN 的人不愿意花费时间和精力去学习其它语言，那怕是现代先进的程序设计语言。何况就现代技术发展的水平来说程序设计语言还达不到科学家和工程师所需要的那样方便。如复型和双精度型的运算。

（c）在过去的20年期间，使用FORTRAN的科学和技术项目已经付出了大量的投资。许多程序仍然在运行，许多程序进行了修改并从一个机器移植到另一个机器上去花费了很大的投资。因此，重写如此大规模的程序是一件可怕的事情，并需付出高昂的代价。

（d）数值计算和统计学软件库已经成型，它们是科学计算界的必不可少的组成部分，它为用户获得高质量的程序提供了有利的条件。这些程序已花费了许多人年写成了FORTRAN程序库的形式，所以既使存在一些缺点，人们也不愿意花费很大代价去改变它。另外FORTRAN程序比用其它语言写的程序更容易实现从一种计算机移植到另一种计算机上去。

然而，不能由FORTRAN统治科学计算世界而推出FORTRAN是理想的程序设计语言。相反，按现代标准检验一个好的程序设计语言时，FORTRAN存在严重的不足之

处，即：

（a）FORTRAN语言还不够简明，不是逻辑结构。

（b）对结构程序设计来说FORTRAN还不是一个理想的工具。

（c）FORTRAN 含有很多的特殊情况，太多使人烦脑的限制以及做同一种事情提供了太多的方法。当研制FORTRAN程序时，这些大大助长错误的发生。总之，尽管FORTRAN存在很多缺陷，但它像一些强劲的野草一样长期地幸存下来。

虽然FORTRAN77保存了早期文本的一些缺陷，但是，新文本在许多有意义的方面做了显著地改进。例如增强了控制结构，使77文本比66文本更有利于结构程序设计，并引进了变量表的直接输入／输出，这两点是很受欢迎的。

FORTRAN77采用了一些现代程序设计语言的一些新特点。因此，重要的是那些用FORTRAN编程序的人应尽快地使用新标准来编程序。

第二章 程 序 设 计

程序设计并不等于用某种语言编程序，程序设计应包括从原始问题的说明到满意地解决这个问题为止的全过程。用某种语言编程序仅是程序设计中的一步，即把算法翻译成程序语言。

用FORTRAN语言编的程序叫源程序。源程序输入到计算机中，经过一个叫做编译程序的软件将源程序翻译成机器语言程序（即目的程序），最后机器执行目的程序为用户处理问题。因此，我们说FORTRAN不是面向机器的而是面向问题的一种高级算法语言。

在讨论用FORTRAN77语言进行程序设计之前，我们先看一下用计算机处理一个实际问题的过程。虽然计算机本身并不能改变处理问题的精度，然而要求设计一个令人满意的程序是十分重要的，可是我们经常忽视这一点，以为能算出结果就是一个好程序。所以本章先讨论一下包括从原始问题说明开始设计一个程序的步骤。

设计一个程序可分五个阶段：

（a）确定问题；

（b）分析问题；

（c）设计解决问题的算法；

（d）把算法译成一种合适的程序设计语言；

（e）调试和维护。

实际上，以上这些阶段都是独立的，但是在达到满意的结果之前，为了进一步研究和修改，程序员经常要回到前一阶段甚至原始的规格说明阶段。

对这些阶段的粗心和认识不足将导致大量意想不到的错误和额外的工作。尤其在着手编程序之前，设计一种恰当的算法是非常重要的，否则最终这个程序得不到满意的结果，这是很简单的道理。c和d两个阶段要求不同的思想方法，一个是解决问题的设计方法，另一个是提供给计算机的算法，它们之间又是互相影响的。

确定问题

确定问题阶段主要是把要处理的问题说明清楚，如果问题说明不清，很明显其结果程序是不会令人满意的。因为提供问题说明的人可能没有计算机专业知识。问题不清经常来源于人的说明，因此，在设计好的程序之前，必须保证向程序员提供关于问题的足够的信息。在第一、二阶段中可能要求和用户讨论几次，最终结果必须使用户满意。下面我们举一个简单的由于人为造成问题说明不清而导致错误的例子：

“打印1——100数的平方表”

这可以理解为整数1，2，3，……，99，100的平方表，也可以理解为1，1.5，

2，2.5，…或1，3，5，…以至1，2，…，9，10，15，20，…，45，50，60，70，…，90，100的平方表。用户站在自己的地位上认为它是“自然数”的平方表，但实际上没有说清楚。因此，程序员在编程序之前必须把问题弄清楚。

分析问题

分析问题是算法设计的第一阶段。一般来说，它包括解法的确定及如何把问题分解成若干个较小的问题。如果现存的程序中有合适的程序，则可直接利用现成的程序。

算法设计

这是最困难的阶段，为了设计一个合理的算法，应按灵活方便的方法提供数据集合，使用户容易做到并感到满意。注意，满意不仅包含着用适当的方法提供正确的结果，也包含运行程序的花费。因此，一个好的算法是在使用计算机资源时有一个最佳值。

由算法到编程序要考虑加工的数据。加工处理数据的能力直接影响程序的功能。因此，设计阶段要考虑给出多少加工数据是最合适的和最好的结构，这一点是很重要的。以及选用什么样的描述语言描述算法也是重要的。我们通常采用流程图的描述语言，但是描述过程必须使误解减到最小。使之有利于编程序。

编 程 序

这一阶段比较容易，但要特别细心。在上述阶段已指明了解决问题的方法，程序员应在所提供的算法基础上最好地利用计算机资源，集中力量编制好的程序。对于一个程序员来说把算法描述成所使用的程序设计语言是十分明确的任务，这一点并不难。不同的程序设计语言有不同的特色，如果选择的语言是可用的，最重要的是把语言的特点应用的最好。

调试检查

如果上述各阶段都做得很认真很细致，那么调试检查阶段暴露的错误和缺陷不会很多。但应该记住程序必须接受所有提供给它的数据的检查，保证正确处理各种情况下设计的检查数据，并得到满意的结果。

找出这一阶段浪费机器时间的部分，修改程序中的错误。因此，再一次强调，如果弄懂了问题并精心地设计出好的算法，多数错误发生在编程序阶段，然而找到这种错误是不困难的，因为编译程序会向你提供产生错误的信息。

维 护

一个已投产的生产性程序，在反复使用中可能出现三个问题要解决：

（a）发现了新的错误；

（b）用户想做一些修改，扩大一些功能；

（c）用户想提高程序的质量。

因此，程序必须维护，最好是原程序设计者维护，转交给别人维护时，重要的是要有上述各阶段的足够的资料，即有清晰的文件资料，用最少的力量去解决上面的问题。然而意外的是程序的创建人往往忽略了他所承担的程序细节，造成难以维护。维护阶段是程序设计中最容易被人们忽视的阶段，可能因为它不是创造性的劳动，不引起人们的注意。不过，维护是使一个程序有无生命力的重要组成部分，一个程序维护的好坏命力就强，维护的不好就没有生命力。进一步说，简单的维护是如何发展良好的程序设计风格，使所编的程序具有易读性和可维护性。

结　论

程序设计不单是用一种喜爱的计算机语言书写程序，设计过程是非常复杂也是非常困难的阶段，如果不这样地看待它，其结果程序就不可能达到要求，有效地运行和易于维护。在程序设计阶段更是如此。

设计阶段

有许多设计算法的方法，但是，一个非常成功的方法是众所周知的自顶向下的设计方法。

自顶向下设计方法

这个方法要求程序员站在最高点向下看，按最高水平考虑整个问题（如描述），然后把问题分解成若干个逻辑部分，每一部分是独立的并可能进一步把它再分解成若干个下一级的逻辑部分，这一分解的过程，是自顶向下的过程。一直分解到编程序这一级为止。注意在每一级上都有一个解决问题的算法。该模核对每一级上解决问题的算法，使程序员确信对于最终这一级的各个逻辑部分他都能编出一个有效的程序。在编程序过程中可能产生小的错误，如笔误，但应有把握确信整个逻辑结构是正确的。注意，使逻辑错误局部化，这样容易发现错误。

模块化

自顶向下设计的方法产生出若干个“逻辑部分”或者叫模块。每个模块是独立的，在不影响其它模块的情况下，要求模块去处理已知的输入数据并产生已知的输出结果，也就是每个独立模块分别调试，使它们都能产生正确的结果。

程序模块化便于阅读和维护。模块化的方法很多，通常用算法语言的一个过程构造一个模块（第十二章）。虽然一些模块可能用于程序的若干部分，但它在程序中只出现一次。

风　格

由于在设计中没有引起足够的重视，一个程序员可能把程序编的非常复杂，这种程序更应加强维护。把程序编的简单是不容易的，而使它复杂化却很容易。因此，养成良好的程序设计风格是十分重要的。初学者应阅读一些典型的高质量的程序。因为良好的程序设计风格来自于经验和简单的模块设计。程序设计者应尽可能地多掌握一些典型程

序的算法。

算法描述语言

对一个算法有许多不同的方法，但在描述方法中有两点是必须做到的。

（a）说明必须明确；

（b）符合自顶向下的模块设计法。

此外，普遍认为算法描述语言需要三个基本的积木：

（1）处理语句；

（2）判定结构；

（3）循环结构。

这三个积木用于设计算法过程的所有阶段。下面我们介绍这些积木的功能，它们贯穿在本书的各章中。

处理语句

处理语句表明一些处理单位的功能的语句。它依赖于上面分解成逻辑部分的级别，它可能描述一个复杂的操作（如，设计一座桥梁）或一个简单的操作(如，求平方数)。一个处理语句可以写成一个简单的句子或由类似程序设计语言的语句写成。在进一步分解加细时，一个处理语句可能包括一系列的处理语句、判定和循环结构。

判定结构

判定是在计算中取得给定数据的当前值（如，当前值是否是0），判定结构有两种形式：

（a）如果——则型语句（if——then），这个语句表示条件成立时则执行某种操作否则不执行。

（b）如果——则——否则型语句（if——then——else），这个语句是在两种操作中选择一个。

判定结构是由一个比较（一个逻辑表达式——见第九章和第十六章）和一或二个处理语句组成。即：

（a）if 逻辑表达式 then 处理语句

（b）if 逻辑表达式 then 处理语句A

else 处理语句B

显然判定结构必须利用足够的信息先计算逻辑表达式的值然后决定下一步做什么。随着对判定结构中某些部分的分解加细，判定结构的某些部分又可以包含处理语句、判定和循环结构。

循环结构

某些问题要求重复执行若干次一系列操作，例如，要查看一个名字表，要求依次检查表中的每一项登记。一系列操作的重复执行叫循环。循环结构有两种：

（a）loop while语句，表明一系列操作重复到条件不成立时为止。

（b）loop for语句，表明一系列操作重复确定的若干次。

（a）一个loop while语句由一个逻辑表达式和处理语句组成，即

注 意

(a) loop while 逻辑表达式 do 处理语句

先计算逻辑表达式的值，然后决定是否循环。

（b）一个loop for语句由一个循环控制变量和它所能得到的一组值以及处理语句组成。例如：

loop for i = 1，2，3，……，n do 处理语句

循环控制变量先获得确定的值，然后去执行do后面的处理语句，直到完成给定的循环次数为止。

循环结构的一些部分也可以包括处理语句、判定和循环结构。

总之，以上三种积木均具有嵌套的结构。

练 习 *2*

1、已知用日，月，年的形式给出20世纪中某一天的有效日期，设计一个算法打印出已知日期的下一天的日，月，年。例如

15，3，42（即1942年3月15日）下一天是16，3，42；

28，2，76（即1976年2月28日）下一天是29，2，76；

31，12，29（即1929年12月31日）下一天是1，1，30；

2、设计一个算法判定输入的日期数据是否是有效的日期，假定输入的日期数据应该包含在1950年1月1日至2050年12月31日期间。已知输入数据的形式是：日，月，年。

3、A是m个数的向量，B是n个数的向量，在输入以前两组向量的元素都是按递增的顺序排列，并假定每个向量中都没有重复值，设计一个算法把向量A和B的元素合并到新的向量C中，要求向量C的元素也按递增的顺序排列，并不包含重复数值。例如：

如果A={1，2，4，5，10}，B={2，6，10，12}则向量C的结果是

C={1，2，3，4，5，6，10，12}

第三章 FORTRAN 的基本组成

FORTRAN字符集

FORTRAN字符集由26个字母、10个数字和13个专用字符组成。

字母是下列26个字符之一：

ABCDEFGHIJKLMNOPQRSTUVWXYZ

数字是下列10个字符之一：

0 1 2 3 4 5 6 7 8 9

专用字符是下列13个字符之一：

字　符	字　符　名
空　白	
=	等　号
+	加　号
−	减　号
*	星　号
/	斜　线
(	左括弧
)	右括弧
,	逗　号
.	小数点
$	美元符号
'	撇　号
:	冒　号

FORTRAN程序纸和卡片

FORTRAN源程序必须按规定的格式书写。为了避免写错，采用印好固定格式的程序纸来写源程序。程序纸每行固定80列，其中前72列中的字符是有效字符，输入到计算机中。73列到80列仅供程序设计者写说明和标识用，一般用来编卡片的序号，这8列的字符不输入到机器中。源程序中的每一行对应一张穿孔卡片，卡片上的格式和程序纸上的格式一致，将源程序用卡片穿孔机穿出一组源程序卡片，即将纸上的信息转换成计算机所能识别的信息，然后将源程序卡片放在卡片输入机上，计算机通过卡片输入机把源程序读入计算机中。有的机器没有卡片输入机，只有穿孔纸带输入机，源程序也必须按

规定的格式穿在纸带上，然后通过纸带输入机将源程序输入到计算机中。在一些微型机和计算机终端上，即没有卡片输入机也没有纸带输入机，仅有一台控制台终端打字机，源程序通过打字机送入到计算机中，这时也必须按上述规定的格式输入。

FORTRAN源程序是块状结构的，它是由一个主程序块或者加上若干个辅程序块组成的。辅程序包括函数辅程序和子例程辅程序，统称为过程辅程序，还有一个辅程序叫数据块辅程序。

程序块又称为程序单位，程序单位由语句序列和任选的注解行的序列组成。一个程序单位或者是一个主程序或者是一个辅程序。每个程序单位中由若干行组成，每行由若干列组成。程序单位中的行是一个不超过72个字符的序列。在一行中的字符位置称为列，并且按从左到右的顺序排列，因此，程序单位是由整个有序字符集组成。

程序单位中的行分为：

注解行

注解行是在第1列有字母C或星号*或在第1列至第72列仅有空白符的任何行。第1列有C或* 的注解行，在第2列至第72列中可以包含计算机处理系统能够表示的任意字符。在注解行中一般允许使用小写字母和小写字。注解行可以出现在程序单位中的任何地方，它不参与程序的执行，仅为程序提供资料说明，便于阅读程序。它可以被输入输出。

始　行

始行为不是注解行的任何行，而且第6列必须是空白或数字0。第1列至第5列可以含有语句标号或空白符。

继续行

继续行为第6列含有除空白符或数字0以外的字符集中任意字符的行，而且第1列至第5列必须是空白符。继续行是上一行的继续，一个语句最多可有19个继续行。

结束行

每个程序单位必须以END结束，END语句仅写在7到72列上。

总之每个FORTRAN语句必须写在第7列至72列上，且一个语句不能多于1320个字符。任何语句的前面可以带有标号，用来标识这个语句的名字和位置，语句标号的形式是1到5个数字的序列，其中一个必须是非0。语句标号可以放在语句始行的第1列到第5列的任何地方，标号的前0和空白无意义。在一个程序单位中，同一语句标号不能给与一个以上的语句。

FORTRAN77程序是最简单的“功能块结构”，由各种功能块组合成更复杂的程序实体，直到产生完整的FORTRAR77程序为止。对于读者来说直到现在还不知道标准的FORTRAN77程序是什么样，为了介绍FORTRAN77的基本概念使读者了解它的结构，我们介绍一个完整的FORTRAN77程序，使读者先有一个感性认识。

读入三角形三个边长，计算它的面积和三个角的大小。为了说明方便，在程序的每行左边印上了行号，这与实际程序无关。

下面给出了这个问题的完整的FORTRAN程序，称它为可执行程序。这个程序由两个程序单位组成，一个程序单位是从第1行至第17行称为主程序，它以 PROGRAM 开始，程序名字是TRIANG。另一个程序单位是从第18行至第22行称为函数子程序，它以