

高等院校计算机教材丛书

FORTRAN

语言程序设计

王肇荣 姚全珠 编

(第二版)

COMPUTER



西安电子科技大学出版社

71312
✓V480
(2)

438299

高校计算机教材丛书

FORTRAN 语言程序设计

王肇荣 姚全珠 编

(第二版)



西安电子科技大学出版社

1997

(陕)新登字 010 号

JS/38/06

内 容 简 介

本书以国家标准 GB 3057—82“程序设计语言 FORTRAN”为依据,详细地介绍了 FORTRAN 77 的基础知识和程序设计方法,并对 FORTRAN 高级编程给予简述。全书有大量的例题和应用举例,其中既有数值计算的,也有非数值计算的。每章都安排了习题,并在书后给出了参考答案。本书所有例题与习题都在 Microsoft FORTRAN 5.10 编译系统上进行了验算与通过。

本书可作为普通高校有关专业的教材或供计算机应用人员参考使用。也可作为成人教育和职业培训的教学参考用书。本书既便于教学也适合自学。



高校计算机教材丛书
FORTRAN 语言程序设计
(第二版)

王肇荣 姚全珠 编

责任编辑 李纪澄

西安电子科技大学出版社出版

空军电讯工程学院印刷厂印刷

陕西省新华书店发行 各地新华书店经售

开本 787×1092 1/16 印张 16 8/16 字数 384 千字

1984 年 11 月第 1 版 1997 年 10 月第 2 版 1997 年 10 月第 3 次印刷 印数 11 001—15 000

ISBN 7 - 5606 - 0483 - 8/TP · 0222(课)

定价: 16.50 元

序 言

人类即将进入 21 世纪,那是一个信息化社会。计算机技术是信息化社会的核心技术之一,因此加强学生的计算机基础知识与应用能力的培养是社会发展的必然、学科发展的需要;也是学生提高个人素质的自觉要求。国家教委把加强计算机基础教育作为近年来重点抓的一项工作,各院校也都加大了投入,学生更是倾注了空前的热情。

为了把我省普通高校的计算机基础教育提高到一个更高水平,发挥陕西作为一个教育、科技大省的整体优势,省教委曾多次组织省内从事该项工作的有丰富教学经验的专家、教授共商提高我省计算机基础教育水平的大计。在目前,教材建设仍是一项当务之急的基础性工作。1994 年以来,我们曾组织出版过一套“陕西省普通高校非计算机专业计算机教材丛书”,发行数十万,对于推动我省计算机基础教育起到了良好作用,受到各院校师生欢迎。

计算机技术日新月异,教学内容必须更新,因此原有的那些教材已远远不能适应教学要求,广大教师迫切需要一套符合技术发展的新教材。去年以来,省教委通集了省内 10 来所院校的专家、教授开始酝酿修订和编写新一轮教材的大纲、物色作者,经过一年多的努力,这套教材已基本完成。

本丛书修订和编写遵循以下原则:

- 保证高质量。专家组聘请作者,教材大纲和书稿均由专家组审定。
- 取材要有先进性。成熟的、已进入使用的新技术要反映到教材中去,以引导教学不断更新内容,提高实验水平。
- 要适当考虑省情,增加可操作性和可使用性。
- 尽量出齐,成熟一本出版一本。

这套丛书包括《计算机应用基础》、《计算机软件基础》、《C 语言程序设计》、《PASCAL 语言程序设计》、《FORTRAN 语言程序设计》、《BASIC 语言程序设计》、《微型计算机原理及应用》、《单片微型计算机原理及应用》、《FoxPro 原理及应用》等 9 种。大部分教材可望在年内出版。我们希望这套教材在提高我省的计算机基础教育水平方面发挥作用,同时也希望广大师生在使用过程中提出宝贵意见,以便再版时改进,使这套教材成为计算机基础教材的精品。

本编写组是在陕西省教委组织和领导下工作的，省教委高教处对计算机等级考试大纲和教材编写、修订的原则作了明确指示，并在遴选编、审者，确定书目和内容，以及协调出版等方面，为提高这套丛书的编写、出版质量，尽快与读者见面作了大量卓有成效的工作。

陕西普通高校非计算机专业

计算机教材丛书编写组

1997年7月

第一版前言

本书以我国国家标准 GB 3057—82“程序设计语言 FORTRAN”(1982—05—12 发布, 1983—05—01 实施)为依据, 按照陕西省高校非计算机专业计算机应用知识和应用能力等级考试大纲的要求, 介绍了 FORTRAN 77 语言及其程序设计方法。在学习本书之前要求先掌握《计算机应用基础》(理工科类)一书的内容。

本书编写力求做到概念准确、由浅入深、循序渐进、前后呼应, 把 FORTRAN 77 的基本内容介绍给读者。全书有大量的例题, 其中既有数值计算的也有非数值计算的。每章都安排了习题, 并在书后给出了参考答案。本书所有的例题与习题都在 Microsoft FORTRAN 5.00 编译系统上进行了验算与通过。

全书共分十章, 第 1、2 章由王肇荣编写, 第 3、4、5 章由董宏编写, 其余五章由姚全珠编写。全书由王肇荣统编和定稿。

由于我们水平有限, 书中一定有缺点和不足之处, 甚至可能存在错误, 敬请读者批评指正。

本书由陕西师范大学曹豫荻教授审稿。我们认真研究了曹教授在审稿中提出的宝贵意见, 并对原稿做了多处修改, 在此向曹豫荻教授表示深切的谢意。

编 者

1994 年 3 月

于西安

第二版前言

本书以我国国家标准 GB 3057—82“程序设计语言 FORTRAN”(1982—05—12 发布, 1983—05—01 实施)为依据, 按照陕西省高校非计算机专业计算机应用知识和应用能力等级考试大纲的要求, 详细地介绍了 FORTRAN 77 语言的基础知识及其程序设计方法, 并专用一章对 FORTRAN 高级编程给予简述。在学习本书之前要求先掌握有关计算机应用基础方面的知识。

本书编写力求做到概念准确、由浅入深、循序渐进、前后呼应, 把 FORTRAN 77 的基本内容介绍给读者。第 10 章 FORTRAN 高级编程指南列为选学内容, 可供学时数较多且学生基础较好的院校选用。全书有大量的例题和应用举例, 其中既有数值计算的, 也有非数值计算的。每章都安排了习题, 并在书后给出了参考答案。本书所有的例题与习题都在 Microsoft FORTRAN 5.10 编译系统上进行了验算与通过。

本书的第一版为 1994 年由西安电子科技大学出版社出版的《FORTRAN 语言程序设计》, 由王肇荣、姚全珠、董宏编写。第二版由王肇荣(第 1~5 章)、姚全珠(第 6~11 章、习题答案和附录)负责修订。主要修订的内容有: 第 3 章对各种编辑符的功能和使用方法做了详尽的介绍, 增加了较多的例题; 第 4 章和第 5 章都增加了一节“应用举例”, 以帮助学生牢固地掌握并灵活地应用学过的知识; 第 7 章增加了“记录类型数据”一节; 为了提高学生的 FORTRAN 语言水平, 新增加了第 10 章 FORTRAN 高级编程指南。

这次虽然对第一版中存在的缺点和不足之处, 做了认真的修订, 但限于水平, 肯定还会有疏漏, 敬请读者批评指正。

本书由陕西师范大学曹豫菽教授审稿。我们认真研究了曹教授在审稿中提出的宝贵意见, 并对原稿做了多处修改, 在此向曹教授表示深切的谢意。

编 者

1997 年 5 月

于西安

目 录

序言

第二版前言

第一版前言

第 1 章 FORTRAN 语言概述	1
1.1 发展概况	1
1.2 FORTRAN 源程序基本结构	2
1.3 程序流程图	4
1.4 结构化程序设计	7
习题一	11
第 2 章 FORTRAN 数据类型与赋值	12
2.1 常数	12
2.2 变量及其类型说明	15
2.3 赋值语句	18
2.4 内部函数	27
2.5 DATA 语句和 PARAMETER 语句	29
2.6 STOP 语句和 PAUSE 语句	32
习题二	33
第 3 章 输入与输出初步	38
3.1 输入、输出的概念	38
3.2 基本的输入/输出语句	39
习题三	55
第 4 章 分支结构	58
4.1 GO TO 语句	58
4.2 算术 IF 语句与逻辑 IF 语句	61
4.3 块 IF 结构	65
4.4 多路判定与块 IF 的嵌套	69
4.5 应用举例	73
习题四	81
第 5 章 循环结构与数组	84
5.1 循环结构概念	84
5.2 数组的定义及引用	90
5.3 DO 循环	94

5.4 多重循环	99
5.5 隐含 DO 循环	101
5.6 应用举例	104
习题五	113
第 6 章 过程	117
6.1 过程的概念及分类	117
6.2 语句函数	118
6.3 外部函数	121
6.4 子程序	126
6.5 虚元和实元的结合	134
6.6 利用过程实现结构化程序设计	144
习题六	151
第 7 章 FORTRAN 数据结构	156
7.1 概述	156
7.2 数值型数据之间的类型转换与运算	157
7.3 字符型数据	161
7.4 记录类型数据	170
习题七	174
第 8 章 数据联系	177
8.1 EQUIVALENCE 语句(等价语句)	177
8.2 COMMON 语句(公用语句)	181
8.3 数据块辅程序	186
习题八	187
第 9 章 文件	189
9.1 FORTRAN 文件分类	189
9.2 辅助文件操作语句	191
9.3 文件输入/输出	196
9.4 文件的使用	199
习题九	202
第 10 章 FORTRAN 高级编程指南	204
10.1 FORTRAN 与汇编语言混合编程	204
10.2 FORTRAN 与 C 语言混合编程	206
10.3 图形的功能	208
10.4 FORTRAN 中的汉字处理技术	218
第 11 章 上机操作指南	228
11.1 系统安装	228
11.2 输入与修改源程序	230
11.3 编译与连换	232
11.4 执行程序	233

习题答案.....	234
附录 A	242
附录 B	246
附录 C	250
参考文献.....	252

1 FORTRAN 语言概述

1.1 发展概况

FORTRAN 是“FORmula TRANslation”(公式翻译)的字首组合词。它是 1954 年被提出来的,1956 年美国首先在 IBM 704 型计算机上实现了 FORTRAN I 语言的编译程序。

FORTRAN 是目前国际上广泛流行的一种高级程序语言,最初是为科学计算而设计的,至今仍然主要用于求解数学、工程与科学方面的问题。但是,该语言实际上并无什么地方要强施这一专门性,目前也用于非数值计算,例如用于管理上和商业上。随着 FORTRAN 语言的发展,它的用途也更为广阔。当前,FORTRAN 广泛地作为向学生讲授计算机应用和程序设计的一种工具,因为它不用太高深的数学基础且很容易掌握。

FORTRAN 语言问世 40 年来发展很快,先后推出了不同的版本。1966 年美国国家标准协会(American National Standards Institute,简称 ANSI)公布了两个 FORTRAN 标准文本:

美国国家标准 FORTRAN X 3.9 - 1966(习惯上称为 FORTRAN IV)。

美国国家标准基本 FORTRAN X 3.10 - 1966(习惯上称为 FORTRAN I)。

1972 年国际标准化组织(International Standardization Organization,简称 ISO)在美国 FORTRAN 标准文本的基础上,稍加修改后公布了 ISO FORTRAN 标准,即《程序设计语言 FORTRAN ISO 1539 - 1972》,它分为以下三级:

基本级 FORTRAN(相当于 FORTRAN I)。

中间级 FORTRAN(介于 FORTRAN I 和 FORTRAN IV 之间)。

完全级 FORTRAN(相当于 FORTRAN IV)。

1976 年 ANSI 对 FORTRAN X 3.9 - 1966 进行了修订,在功能上做了许多改善和扩展,公布了一个 FORTRAN 新的标准草案。1978 年 4 月由 ANSI 正式公布作为新的美国国家标准,称为美国国家标准 FORTRAN X 3.9 - 1978(习惯上称为 FORTRAN 77)。1980 年, FORTRAN 77 被接受为国际标准,即《程序设计语言 FORTRAN ISO 1539 - 1980》。

1982 年 5 月 12 日,我国公布了中华人民共和国国家标准 GB 3057—82“程序设计语言 FORTRAN”,并于 1983 年 5 月 1 日开始实施。

为了适应社会发展的需要, FORTRAN 语言还在不断发展中。1985 年 8 月推出的是 Microsoft FORTRAN 77 V3.31, 1989 年推出了 Microsoft FORTRAN 5.00, 继之又推出 Microsoft FORTRAN 5.10。随着版本的不断更新, FORTRAN 语言的功能越来越强。例如, Microsoft FORTRAN 5.10 版本提供了丰富的图形库, 支持 OS/2、Windows 系统; 提供了丰富的接口, 可以方便地进行混合语言编程; 还提供了一些新的数据结构和若干新的控制结构。这些使得 FORTRAN 语言日臻完美。1991 年 5 月国际标准化组织(ISO)推出国际标准——Fortran 90。Fortran 90 对 FORTRAN 77 做了较大的扩充和完善, 现在已经开始应用。

本书将以国家标准 GB 3057—82“程序设计语言 FORTRAN”为依据来介绍 FORTRAN 77 语言及其程序设计方法。全书的例题与习题都在 Microsoft FORTRAN 5.10 编译系统上进行了验算与通过。Microsoft FORTRAN 符合 ANSI X 3.9 - 1978 标准的程序设计语言 FORTRAN 77。

一般来说, 目前计算机上的 FORTRAN 文本是不违背标准文本的。但是, 各种机型的硬件(特别是输入和输出)有所不同, 以及相应的编译系统实现的方法也不同。因此, 在不同计算机上实现的 FORTRAN 语言可能会有一些具体规定或少量的修改。所以, 读者在计算机上使用 FORTRAN 77 之前, 最好先阅读该机上的 FORTRAN 文本, 尤其要注意那些与标准文本不一致的地方。

1.2 FORTRAN 源程序基本结构

一、FORTRAN 字符集

FORTRAN 字符集由 26 个字母, 10 个数字和 13 个特殊字符组成。

1. 字母

字母是下列 26 个字符之一:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

2. 数字

数字是下列 10 个字符之一:

0 1 2 3 4 5 6 7 8 9

对数字串要解释为数值时, 应解释为十进制数。

3. 特殊字符

特殊字符是下列 13 个字符之一:

空格(本书中用—表示))	右括号
= 等号	,	逗号
+ 加号(正号)	.	小数点
- 减号(负号)	'	撇号
* 星号(乘号)	:	冒号
/ 斜线(除号)	\$	币号

(左括号

在 FORTRAN 77 源程序中,除一些特殊规定的情况外,一般只使用这 49 个字符,而且字母的大小写不予区分,空格符也无意义。但对特殊规定的一些情况,空格符和字母大小写均有意义,这在后面遇到具体情况时将予介绍,而且在这些情况中使用的字符可以超出上述字符集的范围,当然以处理系统能接受为限。

二、源程序的书写格式

用 FORTRAN 77 语言编写的程序称为 FORTRAN 77 程序,又称为源程序。FORTRAN 源程序必须严格地按照一定的格式书写。FORTRAN 源程序一行最多可以有 80 个字符。行中的字符位置称为列,一行中各列从左至右依次连续编号为 1 到 80。这 80 列分为四个区,分别书写不同的内容。第 1 至第 5 列为标号区,用来给语句标上标号;第 6 列为续行标志区;第 7 至 72 列为语句区,第 73 列至 80 列为注释区。

在一行的语句区中最多只能写 1 个 FORTRAN 语句。当一个语句在一行中写不下时,可以在下一行的语句区中继续写。称一个语句的第 1 行为始行,它的其它行称为续行。FORTRAN 规定一个语句不能多于 19 个续行,也就是说一个语句不能包含多于 1320 个字符。始行的第 1 列至第 5 列上可含有语句标号或全是空格符,而第 6 列必须是空格符或数字 0;续行的第 1 列至第 5 列必须全是空格符,而第 6 列上必须是除空格符或数字 0 以外的 FORTRAN 字符集的任意字符。始行与它的续行必须书写在连续的行上,中间不允许插入除注解行之外的任何其它语句行。

若一行上的第 1 列是字母 C 或星号*,则表示该行为注解行。注解行的内容写在第 2 至第 72 列上。注解行的内容可以由处理系统能接受的任何字符组成。注解行完全不影响可执行程序,注解行可用于对源程序作一些说明,以增强源程序的可读性。注解行可以出现在源程序的任何地方,可置于一个语句的始行之前,也可置于始行和第一个续行之间,或两个续行之间。

语句标号提供引用语句的标志,或作为控制语句转向的目标。因此,FORMAT 语句(见 3.2 节)和作为控制语句转移到达的语句必须有标号,其它语句的标号可有可无,并不影响程序的执行。语句标号书写在始行的标号区中,可以由 1 到 5 位数字组成,可以是 1 到 99999 的无符号整数。在区分语句标号时,空格或前导零没有意义。例如,下列标号

150 150 0150 150 00150

编译系统认为是相同的标号。在程序中,语句标号出现的次序与它的数值大小无关。在一个程序单位中,同一个语句标号不得标识一个以上的语句。除程序执行控制转向外,程序是按可执行语句在程序单位中出现的次序执行的,与标号值的大小或出现与否均无关。

三、源程序的基本结构

1. 程序单位

一个可执行的 FORTRAN 程序由一个或多个程序单位组成。程序单位由语句和任选注解行的序列组成。一个程序单位或者是一个主程序,或者是一个辅程序。一个程序单位的语句标号仅在该程序单位内有效。程序单位必须以 END 语句结束。

2. END 语句

END 语句的形式是

END

END 语句是可执行语句，它一定是程序单位的最后一个语句，它表明一个程序单位的语句和注解行序列的结束。它必须写在始行的第 7 列到第 72 列上，END 语句不允许有续行。

如果主程序的 END 语句被执行，则整个程序执行就结束；如果函数辅程序或子程序辅程序的 END 语句被执行，则控制返回引用程序。在一个程序单位中不允许有 2 个或 2 个以上的 END 语句。

3. 主程序

主程序一般是以 PROGRAM 语句开头，以 END 语句结束的一个程序单位。PROGRAM 语句的形式是

PROGRAM name

其中，name 是出现 PROGRAM 语句时主程序的符号名，它不应与其它被使用的名字相同。

主程序命名并不是必要的，所以主程序中 PROGRAM 语句可有可无。但是，如果要写 PROGRAM 语句的话，就必须是主程序的第一个语句，而且要起一个名字。

4. 辅程序

辅程序包括下列 3 种：

函数辅程序，它以 FUNCTION 语句开头，END 语句结尾。

子程序辅程序，它以 SUBROUTINE 语句开头，END 语句结尾。

数据块辅程序，它以 BLOCK DATA 语句开头，END 语句结尾。

一个可执行的 FORTRAN 程序，有且仅有 1 个主程序，但可以有零个到多个辅程序。程序的执行是从主程序的第一个可执行语句开始的。

1.3 程序流程图

程序流程图又称为框图，它是用一些图框来表示各种类型的操作，以便形象地描述解决问题的步骤。

美国国家标准协会 ANSI 规定了一些常用的流程图符号，见图 1.1。

例 1.1 某房地产公司有两块长方形的地皮。第一块宽度为 95.8 m，长度为 223.5 m。第二块宽度为 198.6 m，长度为 219.6 m。试画出计算每块地皮面积和两块地皮总面积的流程图。

图 1.2 就是满足上述要求的流程图。

例 1.2 画出求解一元二次方程 $Ax^2 + Bx + C = 0$ 的流程图。见图 1.3。

有些算法画出的流程图会较复杂，为了避免流程线的交叉或过长，常常使用连接点，可以使流程图清晰易读。图 1.4 实现的是一个完整的算法，我们分三块画出。




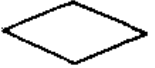


图形符号	名称	意义
	起止框	表示流程的开始、终止或暂停
	输入/输出框	表示输入或输出数据
	处理框	表示一般的处理操作
	判断框	表示要做出判断，在几条可供选择的 路径中，根据条件选取其中一 条路径
	流程线	表示流程的去向
	连接点	表示流程图中流程线的连接点

图 1.1

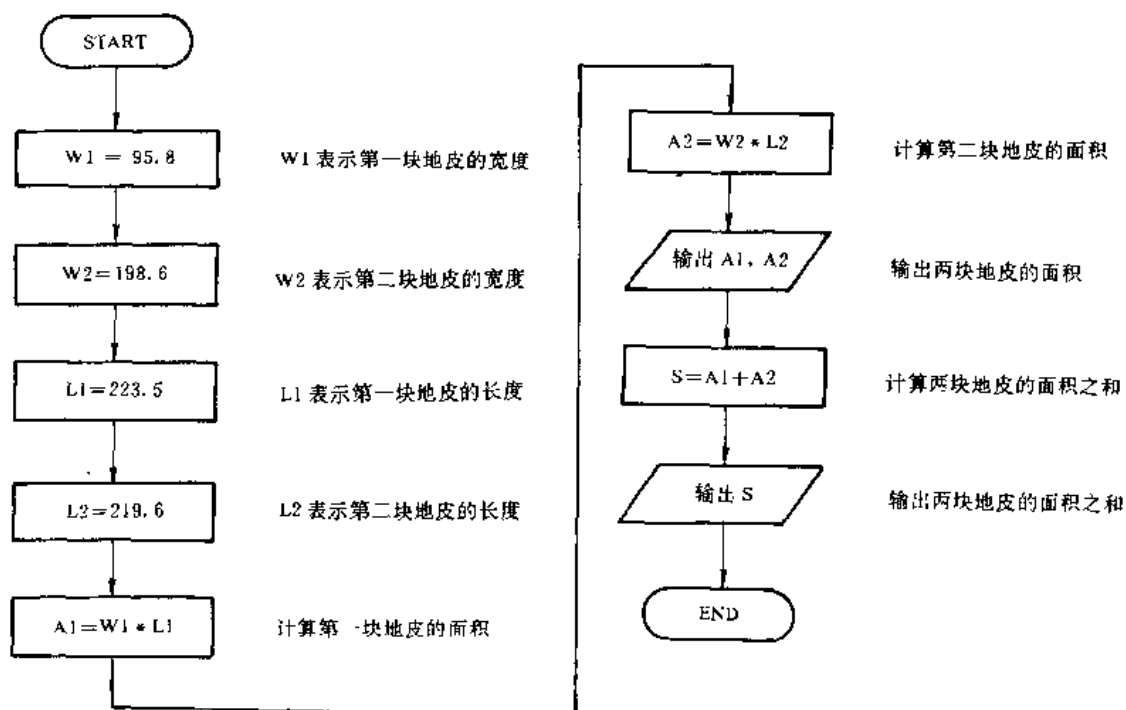


图 1.2

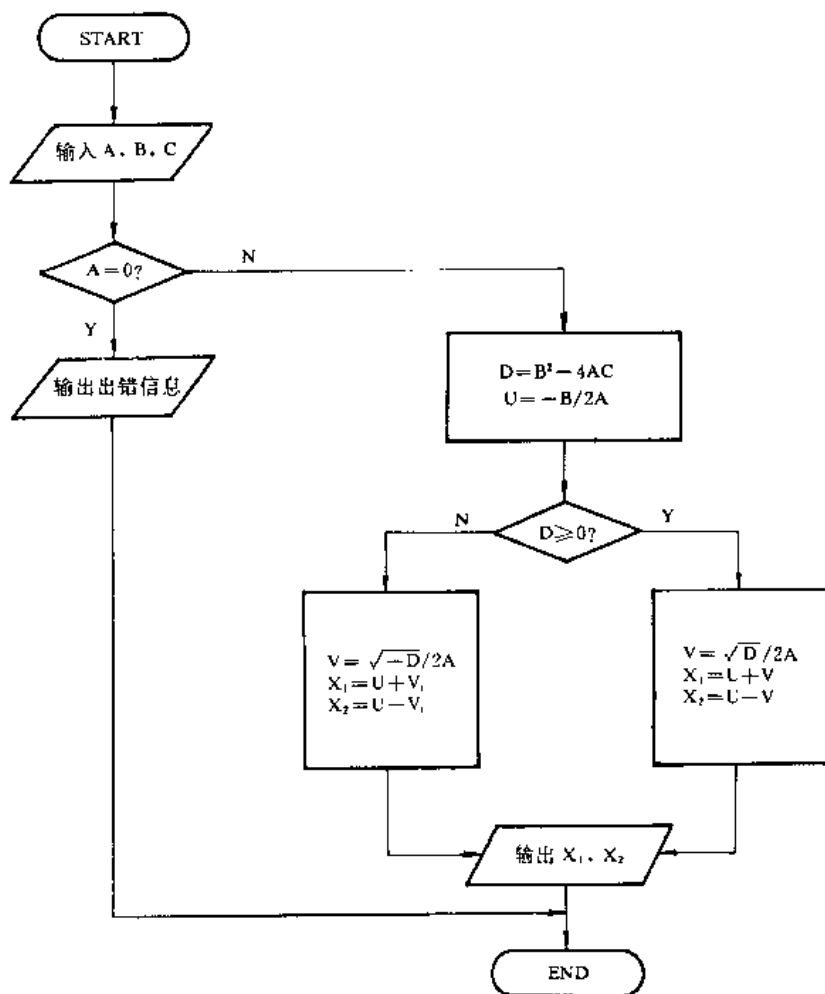


图 1.3

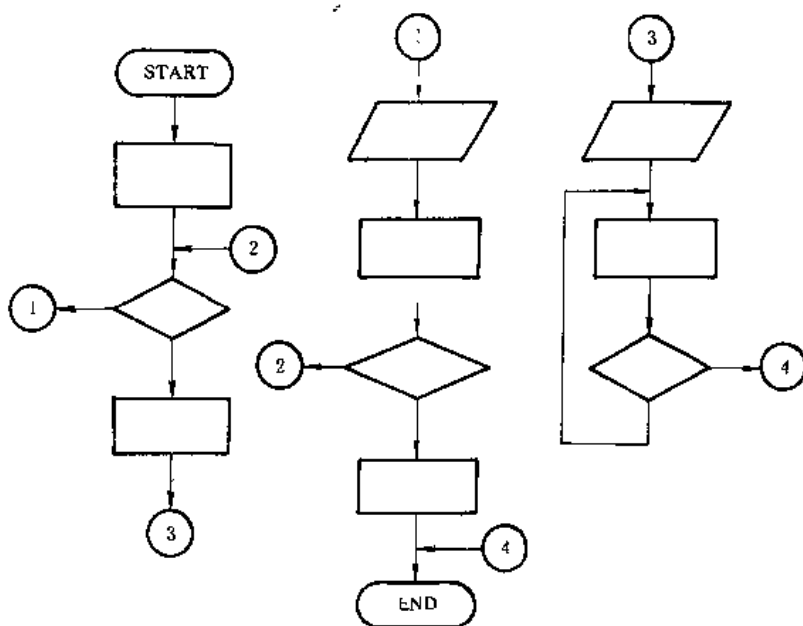


图 1.4

1.4 结构化程序设计

一、结构化程序设计思想的产生

20 世纪 50 年代到 60 年代,一个程序员在掌握了程序设计语言和程序设计的一些基本技巧之后,只要会使用一些算法并经过一定的实践,就可以编出满足当时要求的程序来。当时编出的程序与程序员本人的业务水平关系极大,程序中使用的技巧越多越巧妙,运行速度就越快,所占用的存贮单元就越省。因此,人们常称程序设计既是一门科学也是一门艺术。从 60 年代末期,越来越多的大型科研课题使用计算机,并陆续出现了大型软件系统,如操作系统、数据库等,这些都给程序设计带来了新问题。一个大的程序要花费众多的人力,有时需要几千人年的工作量,因而不能由一个人或少数几个人来编制。但是,当许多人分别编制的风格各异的程序连结在一起的时候,常常是可靠性差、错误多,维护和修改都很困难。隐藏的错误也不易被发现和纠正。为此,就促使人们开始研究程序设计的一些最基本的问题:什么是程序的基本组成部分?应该用什么样的方法来设计程序?程序设计的主要方法和技术应如何规范化和工程化,等等。

1969 年,荷兰数学家 Dijkstra 首先提出了结构化程序设计的概念。他强调了从程序结构和风格来研究程序设计。一个程序结构好,是指程序结构清晰、便于编写、便于阅读、便于验证、便于修改和维护。结构好的程序从效率上看,并不一定是最好的程序。这里所指的效率是从时间和空间两个角度来衡量,即希望编出来的程序越短越好,运行速度越快越好。但是,结构好的程序能减少程序出错的机会、提高程序的可靠性和保证程序的质量。随着硬件技术的发展,当前计算机运行速度大大提高,存贮容量也很大了,因而程序的可靠性和可维护性已成为第一要求。除了系统的核心程序及其它一些有特殊目的(如军事目的)程序外,在通常情况下,宁可降低效率,也要保证程序有好的结构。从此,人们开始研究、总结出一套程序设计的基本原理和方法,使程序设计尽可能减少对程序员个人的风格、技巧的依赖,而逐渐上升为一门科学性的学科。当然程序设计仍然是由程序员来完成的,还是保持着一定程序艺术的特点。目前,普遍使用的和比较成熟的程序设计方法,是结构化程序设计方法。

二、结构化程序设计方法

结构化程序设计方法主要包括以下两个方面的内容。

1. 模块化程序设计

程序中完成一种特定功能的指令集合称为模块。模块化程序设计的基本思想是在进行系统设计时,对总体方案采取自上而下、逐步细分的方法,把一个大而复杂的问题分解为若干个相对独立、功能单一的模块。继之,是对这些单一功能的模块的研制,因而使复杂的研制工作简化了。由于模块的相对独立性也能有效地防止错误在模块之间扩散蔓延,当许多模块连结在一起时,减少了相互之间的干扰和影响,从而提高了系统的可靠性。

考虑了总体方案的程序设计的结构之后,就对每个模块编制程序。那么,整个程序就由一些相对小的程序组成,而这些程序都具有一定的相对独立性,可以分别调试和修改。

这样，一方面可以使许多人分头工作，缩短软件研制的周期；另一方面也使整个系统思路清楚，便于发现问题并及时纠正。一般地，整个系统的模块可用树形图表示。“树根”是主控模块，下连第一层子模块，各个子模块下还可以有更下一层子模块，……直到最底层模块。最底层模块称为“树叶”。图 1.5 表示一个三层的模块结构。

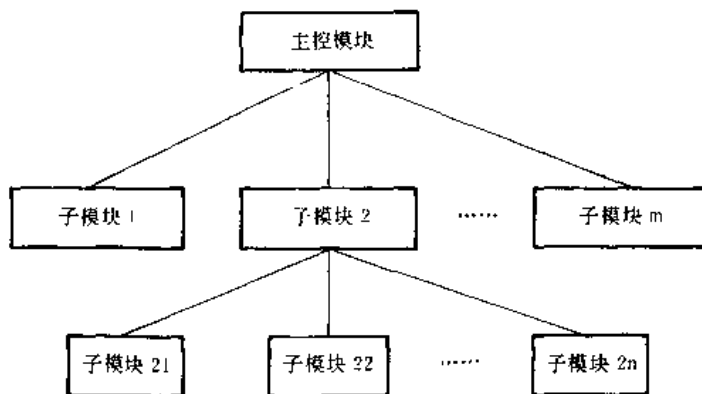


图 1.5

采用模块化程序设计的优点是：

① 由于模块相对独立，可以对每一个模块独立编制、调试和修改程序。这种修改只会对本模块发生影响，而不会产生涉及其它模块的连锁修改。

② 由于模块本身只有一个入口和一个出口，于是模块内部的数据结构只有模块内部提供的操作才能给以改变，因而保证了模块内部数据结构的安全性和完整性。

③ 由于各模块只在上下级之间有接口，同层次之间无接口，因而程序结构清晰，模块的正确性就容易保证。

2. 三种基本结构

这里所说的结构化程序设计是指采用三种基本结构，就能使程序具有好的结构。

1966 年，Böhm 和 Jacopini 证明了程序设计语言中只要有三种基本结构，就足以表示出各种各样的其它形式的结构。这三种基本结构是顺序结构、选择结构和循环结构。

(1) 顺序结构

如图 1.6 所示，虚线框内是一个顺序结构。顺序地执行语句序列 A 和 B，只有当 A 执行完成之后才执行 B。顺序结构是最简单的一种基本结构。

(2) 选择结构

如图 1.7 所示，虚线框内是一个选择结构。此结构必包含一个判断框，当条件成立 (YES, 用 Y 表示) 或不成立 (NO, 用 N 表示) 时分别执行语句序列 A 或 B，二者择其一。不管执行哪一个语句序列，执行结束后都转移到同一出口的地方。A 或 B 两个语句序列中可以有一个是空的，即不执行任何操作，如图 1.8。

(3) 循环结构

循环结构又称为重复结构，即反复执行某个语句序列。这里介绍两类循环结构：

1) 当型 (loop - while 型) 循环结构

见图 1.9，虚线框内是一个当型循环结构。它的功能是：当给定的条件成立时，执行语句序列 A (在循环结构中语句序列 A 称为循环体)，执行完 A 后，再判断条件是否成立，如果仍然成立，再执行循环体，如此反复执行循环体，直到某一次条件不成立时而离开此循

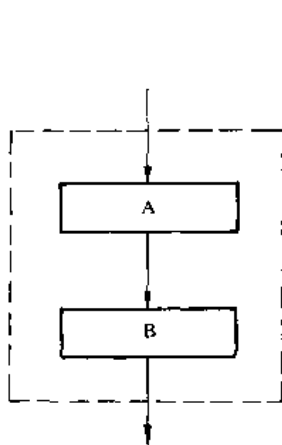


图 1.6

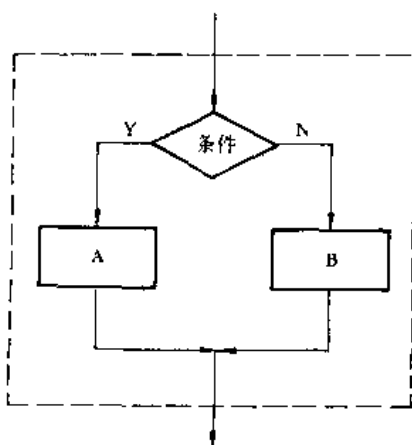


图 1.7

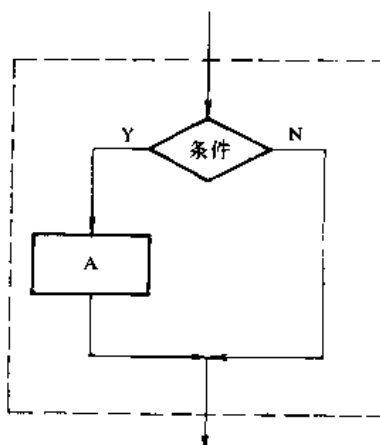


图 1.8

环结构。

2) 直到型(loop - until 型)循环结构

见图 1.10, 虚线框内是一个直到型循环结构。它的功能是: 先执行语句序列 A, 即执行循环体, 然后判断给定的条件是否成立, 如果条件不成立, 则再执行 A, 然后再对条件进行判断, 如果条件仍然不成立, 又执行 A, ……如此反复执行 A, 直到给定的条件成立时而离开此循环结构。

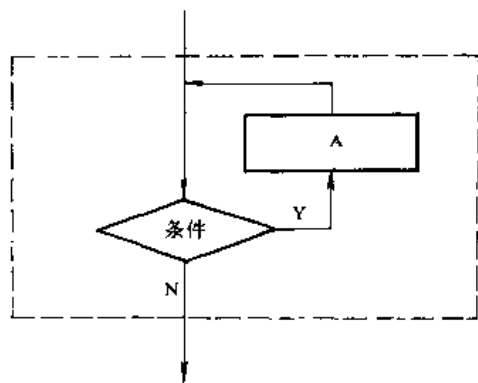


图 1.9

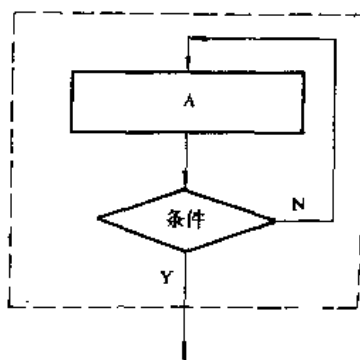


图 1.10

图 1.11 和图 1.12 分别是当型循环和直到型循环的应用例子。

图 1.11 和图 1.12 的作用都是打印 10 个数 1, 2, 3, …, 10。可以看出对同一个问题既可以用当型循环来处理, 也可以用直到型循环来处理。

现在我们来分析一下这两种循环结构的异同。

- ① 两种循环结构都能处理需要重复执行的操作。
- ② 当型循环是一种先判断后执行的循环结构, 当条件不成立时停止循环, 循环体可能一次也不被执行; 直到型循环是一种先执行后判断的循环结构, 循环体至少被执行一次, 当条件成立时停止循环。

- ③ 对同一个问题, 如果分别用当型循环结构和直到型循环结构来处理, 则两者结构中

的判断条件恰为互逆条件。例如，图 1.11 中的条件为“ $m < 10$ ”，而图 1.12 中的条件为“ $m \geq 10$ ”。二者恰好互逆，这是一条很重要的规律。

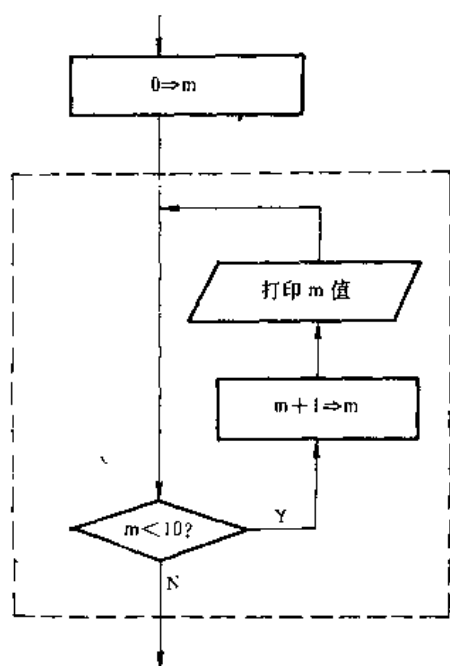


图 1.11

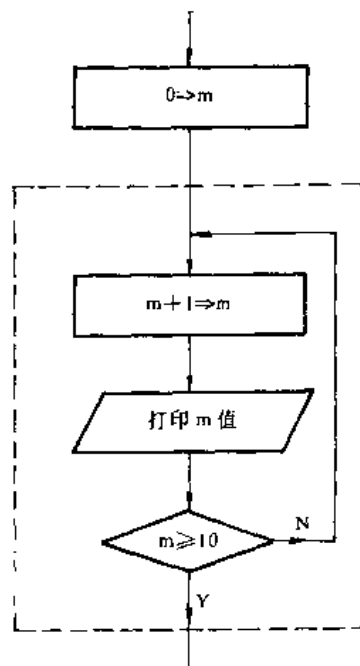


图 1.12

对于一个具体的问题，究竟采用当型循环还是用直到型循环可由编程者选定。由于直到型循环至少执行一次循环体，而当型循环之循环体可能一次也不被执行。所以，当事先不能确定是否至少执行一次循环体时，一般采用当型循环。

从图 1.6 至图 1.9 可以看出，这三种基本结构有一个共同的特点，就是每种结构严格地只有一个入口和一个出口。当然，语句序列 A 或 B 的内部又可以包含这三种结构。如果组成程序的各个分结构都只存在如此简单的接口关系，那么就可以相对独立地设计各个分结构，静态地分析控制关系，并容易验证它们的正确性。同样，如果某一分结构需要修改，只要接口不变，就不会影响到其它分结构乃至整个程序。因此，这种程序具有好的结构。应当注意，一个判断框有两个出口，而一个选择结构只有一个出口，不要将判断框的出口和选择结构的出口混淆。显然，这种结构也不会出现“死循环”（无终止的循环）。图 1.13 就是死循环。

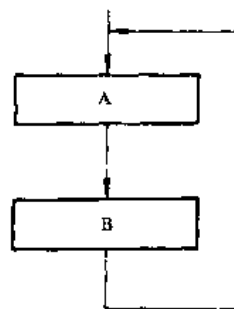


图 1.13

尽管已经从数学上证明程序设计语言只要具备上述三种基本结构就可以进行结构化程序设计，但为了用户方便，各种程序设计语言还常常引进其它各种各样的控制结构。例如 FORTRAN 77 允许使用 GO TO 语句来实现转移。由于 GO TO 语句破坏了语句顺序执行，不符合结构化原则，因此一般不提倡使用 GO TO 语句，通常是把 GO TO 语句限制在一个基本结构内部使用。

习 题 一

1. 什么叫源程序？FORTRAN 源程序一行有多少字符？标号区、续行区、语句区和注释区是如何划分的？
2. 在一个语句区中能写几个 FORTRAN 语句？如何区别始行和续行？在始行与续行之间是否能插入注解行？注解行的标志是什么？注解行在源程序中的位置有何限制？
3. 语句标号的作用是什么？语句标号的取值范围是什么？语句标号必须按数字的大小顺序排列吗？
4. 什么叫程序单位？什么是主程序？什么是辅程序？
5. END 语句的作用是什么？在一个程序单位中是否允许有两个 END 语句？
6. 程序流程图在程序设计中有什么作用？
7. 什么是结构化程序设计？采用哪些基本结构就足以表示各种复杂的结构？这些基本结构的共同特点是什么？
8. 试用流程图描述 $1^2 + 2^2 + \cdots + 100^2$ 的计算过程。

2 FORTRAN 数据类型与赋值

数据类型是程序设计语言所允许的变量种类，也就是说数据类型是变量可能取的值和可能进行的运算的总称。各种程序设计语言所能提供变量类型的多少决定了该语言功能的强弱。FORTRAN 77 提供了 6 种数据类型：

- a. 整型(INTEGER)；
- b. 实型(REAL)；
- c. 双精度型(DOUBLE PRECISION)；
- d. 复型(COMPLEX)；
- e. 逻辑型(LOGICAL)；
- f. 字符型(CHARACTER)。

不同类型的数据有各自的内部表示形式、书写格式和取值范围。前面 4 种类型又称为算术型数据。类型对于数据所参与的运算可能有影响。

2.1 常数

FORTRAN 77 中常数指算术常数、逻辑常数和字符常数。常数的值是不变的。算术常数包括整常数、实常数、双精度常数和复常数。表示常数的字符串形式既指明了它的值也指明了它的数据类型。除字符常数外，出现在常数中的空格字符对该常数的值没有影响。

无符号常数是指没有前导符号的常数。带符号常数是指带有前导加号或减号的常数。任选带符号常数是指或是带符号常数，或是无符号常数。除另有说明外，整常数、实常数和双精度常数可以是任选带符号常数。

一、整常数

整型数据是整数值的精确表示。它可以有正值、负值或零值。整常数是任选的符号后跟以非空数字串。该数字串要理解为十进制数。上述任选符号即指可选负号“—”(表示负

值), 或选正号“+”, 或不选, 后两种情况均表示正值。

下面是一些合法的整常数的例子:

```
237
-82
+1
000
0028
--0
+0
28
```

其中, 000, -0, +0 的值是相同的, 都表示零值, 0028 与 28 是等值的。

下面这些则不是整常数:

```
23.0      (不允许有小数点)
-3,500    (不允许有逗号)
103      (出现了指数)
$203      ($ 作为前导)
3E4       (包含了非数字)
```

FORTRAN 77 规定, 1 个整型数据占有 1 个数值存贮单元, 1 个数值存贮单元通常指连续 4 个字节。在有些 FORTRAN 77 的编译程序中, 1 个整型数据只占 2 个字节。

在计算机中, 每个二进制位称为一个比特(bit), 8 个二进制位称为 1 个字节(byte)。我们以 2 个字节整型数据的存贮结构来看, 最高位为符号位, 0 表示正数, 1 表示负数, 其它各位存放“0”或“1”, 于是 2 字节整型数据所能表达的数值范围为 $-32\,768 \sim +32\,767$ ($-2^{15} \sim 2^{15}-1$)。类似地, 4 字节整型数据所允许的数值范围为 $-2\,147\,483\,648 \sim +2\,147\,483\,647$ ($-2^{31} \sim 2^{31}-1$)。

二、实常数

实常数有三种不同的形式。

1. 基本实常数

基本实常数的形式是依次为 1 个任选的符号、1 个整数部分、1 个小数点和 1 个小数部分。整数部分和小数部分两者都是数字串。这两部分中的任一部分可省去, 但不能两部分都省去。基本实常数要理解为十进制数。

下面是一些合法的基本实常数的例子:

```
2.163
+0.
-.59
+28.
--101.35
```

下面这些则不是基本实常数:

```
2,163.2 (不允许逗号)
```

59 (没有小数点)

2. 基本实常数后跟一个实指数

实指数的形式为字母 E 后跟任选带符号整常数。实指数代表 10 的方幂。所以基本实常数后跟一个实指数这种实常数的值为基本实常数乘以十进制指数的值。例如：

+0.00028E+5 (等于 0.00028×10^5)

-10135E-2 (等于 -10135×10^{-2})

.002163E3 (等于 $.002163 \times 10^3$)

都是正确的实常数。而下面则都不是实常数：

E10 (不允许单个实指数，应写成 1E10)

.59E2.5 (E 后面不是整常数)

-4.9E4+2 (E 后面不是整常数)

3. 整常数后跟一个实指数

整常数后跟一个实指数的值是整常数乘以十进指数的值。例如：

83900E-2 (等于 83900×10^{-2})

79E5 (等于 79×10^5)

1E-8 (等于 10^{-8})

都是正确的实常数，而 E-5, 9E3.4 等都是不合法的实常数。

三、双精度常数

双精度常数的形式有 2 种：

① 基本实常数后跟一个双精度指数；

② 整常数后跟一个双精度指数。

其中，双精度指数的形式是字母 D 后跟任选带符号整常数，它表示 10 的方幂。

双精度常数的值是 D 前的常数乘以 D 后整数表示的 10 的方幂。注意，除去字母 D 代替了字母 E 外，双精度指数的形式和解释完全和实指数的相同。例如：

48.56D2

72D-7

双精度型数据是处理系统对实数值的近似表示，虽然没有指明精度，但其精度必须大于实型的精度。在计算机中一般用两倍于实数的存储单元存放双精度数。双精度型数据可以有正值、负值或零值。

四、复常数

复常数的形式是左括号后跟一对由逗号隔开的有序实常数或整常数，后跟右括号。该对常数的第 1 个常数是该复常数的实部，第 2 个是虚部。那么，常数(7.8, 3.4)表示复数 $7.8 + 3.4i$ 。复数在计算机中是以两个实数(实部，虚部)的形式存放。

五、逻辑常数

逻辑常数的形式和值见表 2.1。

表 2.1

形 式	值
. TRUE .	真
. FALSE .	假

应当注意, 作为分界符的两个圆点切不可省略, 以便有别于其它的标识符。

六、字符常数

字符常数的形式是一撇号后跟非空字符串, 再后跟一个撇号。该字符串可由处理系统所能表示的任何字符组成, 而且字母的大小写与空格都是有意义的。注意, 起分界作用的撇号不算作该常数所表示的数据的一部分。如果字符串中需要含有撇号, 则用两个相邻的其间不插入空格字符的撇号表示。字符常数的长度是分界撇号之间的字符个数, 其中两个相邻的撇号作为一个字符计算, 每个空格符也算一个字符, 表 2.2 给出一些例子。

表 2.2

字 符 常 数	字符常数的值	长度
'3.14159'	3.141 59	7
'FORTRAN77'	FORTRAN77	9
'FORTRAN┐77'	FORTRAN 77	10
'He┐said, "good┐morning!"'	He said, 'good morning!'	23
'Didn't┐you┐see┐my┐book?'	Didn't you see my book?	23

注意, 最后一个字符常数中的问号(?)不是标准 FORTRAN 77 字符集中的字符, 这是允许的。FORTRAN 77 允许字符串由计算机能够接受的任何字符组成。所以应用时, 要了解所使用的机器能接受哪些字符。但为了程序可移植, 通常字符串只由标准字符集中的字符组成。

2.2 变量及其类型说明

变量是在程序运行过程中其值可以改变的量。FORTRAN 77 中的变量与前面讲述的数据类型相对应也有 6 种类型: 整型、实型、双精度型、复型、逻辑型和字符型。

一、符号名

变量名用符号名来标识。符号名采取 1 到 6 个字母或数字的序列的形式, 其中第 1 个必须是字母。

例如 B24AC, D, ALPHA, XEQ16, FNX1, FNX2, M4, K6X29Z 都是正确的符号名。而下面几个则不是符号名:

2X139 (首字符是数字)
 PRODUCT (多于 6 个字符)
 MK3/1 (出现了不是字母或数字的其它字符)

顺便指出,在有些处理系统中符号名可以超过 6 个字符。建议读者还是遵守不超过 6 个字符的限定,以免出错。

为了增强程序的可读性,建议读者为变量命名时,尽量与变量的含义或习惯用法一致。例如用 WATTS、VOLTS 和 AMPS 分别表示瓦特、伏特和安培,用 AREA 表示面积,用 SUM 表示累加和,等等。

FORTRAN 77 规定,一个变量名只在定义它的程序单位中有效。因此,在不同的程序单位中的变量可以同名,但它们的含义可以是不同的。

根据变量的类型在内存中开辟与之相应的存贮空间。因此,一个变量只能存放一个值。这里必须强调,读变量的内容时,变量的值保持不变;如果写入新内容,则新值代替旧值。可记忆为:“取之不尽,一挤就走”。

二、变量的类型说明

FORTRAN 77 中变量的类型说明有如下三种方式:

1. 显式说明

用类型语句(type statement)作显式说明,其形式为

typ v[,v]...

在这种格式中,[]表示其中的内容为可选项,⋯表示还可选用若干个同类型的可选项。本书中一些语句我们都用这种格式描述。

类型语句形式中 typ 称为类型标识符,它是下列单词之一:

INTEGER (整型);
REAL (实型);
DOUBLE PRECISION (双精度型);
COMPLEX (复型);
LOGICAL (逻辑型);
CHARACTER (字符型)。

v 是变量名、数组名、数组说明符、常数符号名、函数名或虚拟过程名。

类型语句的功能是指明 v 具有 typ 类型。例如,

```
INTEGER  X, Y, A1, B2
REAL    NO1, NO2
DOUBLE PRECISION  LONG
COMPLEX  IMAG
LOGICAL  FLAG, EOF, START
CHARACTER * 10  CH1, CH2, CH3 * 15
CHARACTER  CH4, CH5
```

说明 X, Y, A1, B2 是整型变量; NO1, NO2 是实型变量; LONG 是双精度变量; IMAG 是复型变量; FLAG, EOF, START 为逻辑型变量; CH1, CH2, CH3, CH4, CH5 为字符型变量, CH1 和 CH2 的长度均为 10, 它们由 CHARACTER 后面的 * 10 规定, CH3 的长度为 15, 而 CH4, CH5 的长度为 1, 对长度不加规定则默认长度为 1。

应当注意,类型语句应置于程序单位第一个可执行语句之前。类型语句所列变量只在

该类型语句所在的程序单位中有效。而且，在一个程序单位中，一个名字用显式指明其类型不得多于一次。

2. 隐式说明

显式说明必须对各个变量逐个加以说明，隐式说明则可对一批变量进行说明（只要符号名开头的字母相同）。

用 IMPLICIT 语句作隐式说明，其形式为

IMPLICIT typ(a[, a]...)[, typ(a[, a]...)]...

其中：typ 是 INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL 和 CHARACTER 之一。

a 是单个字母或是按字母次序的单个字母的范围。一个范围用该范围的首字母和末字母，其间用减号(-)隔开来表示。写出字母范围 a_1-a_2 和写出从 a_1 到 a_2 所包括的单个字母的表有相同的效果。

IMPLICIT 语句的功能：凡不出现在类型语句（显式说明）中的符号名，若其第 1 个字母在隐式说明 typ 所列字母的范围中，则该符号名被隐式说明为具有 typ 类型。例如：

IMPLICIT INTEGER(R-T), REAL(I-N)

IMPLICIT DOUBLE PRECISION(A-H), COMPLEX(O, P, Q)

IMPLICIT CHARACTER * 5(U-W, X)

表示除类型语句中被显式说明的符号名外，凡以 R、S、T 开头的符号名都是整型的；凡以 I、J、K、L、M 或 N 开头的符号名都是实型的；凡以 A、B、C、D、E、F、G 或 H 开头的符号名都是双精度的；凡以 O、P、Q 开头的符号名都是复型的；凡以 U、V、W、X 开头的符号名为字符型的，且长度均为 5。

IMPLICIT REAL(I, J, K, L, M, N) 与

IMPLICIT REAL(I-N)

有同样的效果。而

IMPLICIT REAL(A-Z)

将保证它所在的程序单位中所有的符号名为实型的。

应当注意，一个字母在隐式说明中最多出现一次，无论是以单个字母的形式还是以字母范围的形式。当使用字母范围的形式，其首字母和末字母必须按照英文字母排列的顺序不可倒置。例如

IMPLICIT REAL(K-A)

是错误的，应写成

IMPLICIT REAL(A-K)

IMPLICIT 语句只适用于它所在的程序单位。一个程序单位可以包含若干个 IMPLICIT 语句。在一个程序单位的说明语句中，IMPLICIT 语句必须置于除 PARAMETER 语句（见 2.5 节）外的所有其它说明语句的前面。无论显式说明还是隐式说明均为非执行语句。

3. 预隐含规则(I-N 规则)

在程序中，若一个符号名既不出现在类型语句中，它的首字母也不出现在 IMPLICIT 语句中，那么凡是以 I, J, K, L, M, N 为首字母的符号名都被认为是整型的；以其它字母开头的符号名都被认为是实型的。这就是预隐含规则，又称为 I-N 规则。此规则仅适用于

整型和实型。因此，双精度型、复型、逻辑型和字符型变量在使用前必须用显式或隐式说明方式来定义。

FORTRAN 77 提供了上述三种说明数据类型的方法，但优先级不同，类型说明语句最优先，而 IMPLICIT 语句又优先于 I—N 规则。虽然显式说明的优先级比隐式说明高，但在程序中，隐式说明语句必须出现在所有显式说明语句之前。经隐式说明过的变量，仍可用显式说明来重新定义其类型。

下面是一个程序片段：

```
IMPLICIT REAL(I, M), DOUBLE PRECISION(D)
IMPLICIT INTEGER(X—Z), COMPLEX(C)
IMPLICIT LOGICAL(L), CHARACTER * 10(E—H)
REAL X1, X2, X3
INTEGER MAX, D1
DOUBLE PRECISION ERROR
LOGICAL FIND
COMPLEX ITEM
CHARACTER CHAR * 10, ZR * 15
```

这个程序片段说明的结论如下：

① 以 I、M 开头的符号名均为实型(隐式说明)，但其中符号名 MAX 为整型，ITEM 为复型(显式说明)。

② 以 D 开头的符号名均为双精度型(隐式说明)，但符号名 D1 为整型(显式说明)；

③ 以 X、Y、Z 开头的符号名均为整型(隐式说明)，但符号名 X1、X2、X3 为实型，ZR 为长度是 15 的字符型(显式说明)；

④ 以 C 开头的符号名均为复型(隐式说明)，但符号名 CHAR 为长度是 10 的字符型(显式说明)；

⑤ 以 L 开头的符号名均为逻辑型(隐式说明)；

⑥ 以 E、F、G、H 开头的符号名均为字符型且长度为 10(隐式说明)，但符号名 ERROR 为双精度型，FIND 为逻辑型(显式说明)；

⑦ 以 J、K、N 开头的符号名均为整型，以 A、B 及 O~W 开头的符号名均为实型(预隐含规则)。

2.3 赋值语句

赋值语句的作用是将一个表达式的值赋给一个变量。而表达式是用运算符和括号把操作数按一定规则连结起来的式子。FORTRAN 77 有以下 4 种表达式：

- ① 算术表达式；
- ② 关系表达式；
- ③ 逻辑表达式；
- ④ 字符表达式。

赋值语句把算术表达式的值赋给一个数值变量(整型、实型、双精度型和复型)，把逻辑表达式的值赋给一个逻辑变量，把字符表达式的值赋给一个字符变量。计算机的计算功

能主要是通过赋值语句实现的。本节只介绍前面三种表达式，有关字符表达式将在 7.3 节中介绍。

一、算术表达式与算术赋值语句

算术表达式中的操作数都是算术量，使用的运算符只能是算术运算符。算术表达式的求值产生一个数值。

1. 算术运算符

FORTRAN 77 可以使用的五个算术运算符见表 2.3。

表 2.3

运 算 符	表 示
**	指数
/	除
*	乘
-	减或负
+	加或正

其中，**，/和*运算符都是对一对操作数进行运算，而且都写在两个操作数之间。运算符+或-中的每一个，或是对一对操作数进行运算，并且写在两个操作数之间；或是对一个操作数进行运算，并且写在该操作数的前面。

2. 算术表达式的形式

算术表达式涉及算术运算符和括号的用法。算术运算符的用法和解释见表 2.4。

表 2.4

运算符的用法	解 释
$X ** Y$	X 的 Y 次幂
X/Y	X 被 Y 除
$X * Y$	X 乘 Y
$X - Y$	X 减 Y
$-Y$	负的 Y
$X + Y$	X 加 Y
$+Y$	与 Y 的值相同

注：X 表示运算符左边的操作数；

Y 表示运算符右边的操作数。

仅涉及一个操作数的运算符称为单目运算符，而涉及两个操作数的运算符称为双目运算符。

算术运算符的优先级见表 2.5。

算术运算符中存在优先级，除非用括号改变操作数的结合次序，否则算术运算符的优先级决定操作数的结合次序。

读者应注意以下几点规定：

① 连续的乘幂运算是从右到左进行的，例如 $X * * Y * * Z$ 应理解为 $X * * (Y * * Z)$ ，而其它连续的同级运算是从左到右进行的。例如 $X/Y/Z$ 应理解为 $(X/Y)/Z$ 。

② 从 FORTRAN 字符集可以看出，只允许用圆括号，且必须成对出现，不能用方括号。若算术表达式中有括号，则优先计算括号内的表达式，内层更优先。由内到外逐层计算算术表达式的值，在同一层括号内，函数引用优先于其它运算符。

③ X/Y 时， Y 不得为零，否则会发生溢出错误。

④ $X * * Y$ 时，当 $X=0$ 时， Y 不能小于等于零；当 $X<0$ 时， Y 不能为实型数或双精度型数。

⑤ 不允许表达式包含两个相邻的算术运算符。例如 $X * * -Y$ 或 $X -- Y$ 是不允许的。然而， $X * * (-Y)$ 和 $X + (-Y)$ 是允许的。

表 2.6 列出一些数学表达式及其对应的 FORTRAN 77 算术表达式。

表 2.5

运算符	优先级
* *	最高
* 和 /	中间
+ 和 -	最低

表 2.6

数学表达式	FORTRAN 77 算术表达式
78	78
$\frac{A}{-B}$	$A/(-B)$
$[(ax+b)x+c]x+d$	$((A * X + B) * X + C) * X + D$
$\left(\frac{h}{z}\right)^{\frac{3}{2}}$	$(H/Z) * * 1.5$
πr^2	$3.14159 * R * * 2$
$\sin 37^\circ + \cos x$	$SIN(37. * 3.14159/180.) + COS(X)$

3. 算术表达式的结果类型

前面讲过，FORTRAN 中的常量和变量是分类型的，能否在不同类型的操作数之间进行算术运算呢？如果可以进行运算，其结果类型又是什么呢？这方面 FORTRAN 77 有详细的规定，我们在这里主要介绍最常用的整型和实型的情况（其它数据类型详见 7.2 节）。

当运算符 + 或 - 对单个操作数进行运算时，结果表达式的数据类型同操作数的数据类型一样。例如，+5 仍为整型数据。

当运算符对两个相同类型的操作数进行运算时，其结果类型与操作数的类型相同。例如， $2 * 4$ 的值为整数 8，而 $2.5 * 2.0$ 的值为实数 5.0（请注意：5.0 是实数而不是整数）。 $2 * * 3$ 的值为整数 8，而 $2.0 * * 3.0$ 的值为实数 8.0。

应当特别注意：两个整数相除的商也是整数。例如， $5/2$ 的值是 2，而不是 2.5（因为 2.5 不是整数，只取其整数部分得 2）。 $-13/11$ 的值是 -1。尤其需要注意的是 $1/2$ 的值为 0。因此，如果要计算 $(\sin x + \cos x)/2$ 不能写成 $1/2 * (SIN(X) + COS(X))$ ，因为它的值是 0，而应写成 $0.5 * (SIN(X) + COS(X))$ 或 $1.0/2.0 * (SIN(X) + COS(X))$ 或 $(SIN(X) + COS(X))/2.0$ 。同样 $4^{0.5}$ 不能写成 $4 * * (1/2)$ 而应写成 $4 * * 0.5$ 或 $4 * * (1.0/2.0)$ 。读者不难理解 $4 * * (-1)$ 和 $16 * * (-4)$ 的值均为零。正因为对整数的除法有这种规定，所以如果表达式中包含整数的除法，应注意运算的顺序。例如，数学上 $2 \times 8/5$ 的值为 3.2，写成

FORTRAN 表达式 $(2 * 8) / 5$ 其值为 3; 若写成 $2 / 5 * 8$ 其值为零。如果改用实数相除就不会发生这种问题, $2.0 * 8.0 / 5.0$ 和 $2.0 / 5.0 * 8.0$ 的值都是 3.2。所以, 在实际应用时, 若算术表达式中有除法运算, 一般均把整数改为实数, 以免出错。

当运算符对两个不同类型操作数进行运算时, 譬如最常用的情况: 当运算符对一个整型操作数和一个实型操作数进行运算, 那么编译系统会自动将整型数转换为实型数, 然后进行运算, 其结果为实型数。例如, $7 * 8.5$, 则系统先将 7 转换为 7.0, 然后乘以 8.5 得 59.5, 为实型数。同样 7×8.0 的值为 56.0。

对整型数和实型数的四则运算和乘方运算的规律见表 2.7。例如, $3 * * 3$ 的值为整数 27, 而 $3.0 * * 3$ 和 $3.0 * * 3.0$ 以及 $3 * * 3.0$ 的值都是实数 27.0。

表 2.7

<div>操作数 2</div> <div>操作数 1</div> <div>运算结果</div>		整 型	实 型
		整 型	实 型
整 型	整 型	整 型	实 型
实 型	实 型	实 型	实 型

类型的转换是从左至右进行的, 在遇到不同类型的操作数时才进行转换。例如, $1/2 * 40.0$, 并不是一开始就同时将 1 和 2 转换成实数 1.0 和 2.0, 然后进行实数运算(得 20.0), 而是先进行整数运算 $1/2$ 得 0, 最后结果也为 0。

有关其它不同类型的操作数的运算情况详见 7.2 节。

4. 运算的误差和溢出

在整数范围内, 整型量的运算是准确的, 没有任何误差。而实常数在计算机内的表示通常受到硬件字长的制约, 而只能表示成具有一定有效位数的近似值。例如, 数学上实数 $1/3$ 的值为 $0.333\ 333\ldots$, 假定所用计算机允许的实数有效位数为 7 位, 那么实常数 $1.0/3.0$ 只能表示成 $0.333\ 333\ 3$, 而不是一个无限循环小数。如果这个数再乘以 3 的结果就不是我们期望的结果 1 而是 $0.999\ 999\ 9$ 。又如 $11\ 111.1 \times 1\ 111.11$ 本应为 $12\ 345\ 654.321$, 但由于受实常数有效位数的限制, 只能得到 $0.1\ 234\ 565 \times 10^8$ 。又如

$$0.003 + 2\ 765\ 737.0 - 2\ 765\ 730.0$$

本应得到 7.003, 但由于有效位数为 7 位, 在进行前两项相加时不可能得到 $2\ 765\ 737.003$, 而只能得到 $2\ 765\ 737.0$, 再进行减法得到 7.0, 这个误差也来源于有效位数。如果把表达式改写为

$$2\ 765\ 737.0 - 2\ 765\ 730.0 + 0.003$$

结果为 7.003, 就没有误差了。这是因为每次运算所得结果的数字位数都不超过有效位数。因此, 在写表达式时应尽量使每一次运算结果都在有效位数范围之内。要防止丢失有效数字, 就要在表达式运算中尽量避免两个数值相差很大的实型数进行加、减运算。

由于实型量的运算会出现一些误差, 所以我们在判断两个实数 X 和 Y 相等时, 一般不用判断式“ $X - Y = 0?$ ”, 因为本应相等的实数 X 和 Y, 可是由于误差, $X - Y$ 的结果可能不等于 0。最好改用判断式“ $|X - Y| \leq \epsilon?$ ”, ϵ 是一个很小的正数, 譬如 10^{-6} , 若此判断式成

立, 则认为 X 和 Y 相等。

整型量的运算速度比实型量快而且不出现误差, 但整数的使用范围受到较大的限制。实型量使用范围大得多, 但运算速度慢且有误差。如果数的表示范围要大又有较高的准确度, 应增加有效位数, 那么可采用双精度型数, 其有效位数可达 16~17 位以上。

任何计算机上, 实数的范围都是有限的。例如 IBM-PC FORTRAN 的实数绝对值的范围大体上为 $10^{-38} \sim 10^{38}$ 。IBM 360/370 则为 $10^{-75} \sim 10^{75}$ 。如果一个数的绝对值超过此范围就会溢出, 绝对值比 10^{38} (或 10^{75}) 大的称为“上溢”, 系统对此按数据出错处理, 发出出错信息, 并终止程序的执行。比 10^{-38} (或 10^{-75}) 还小的称为“下溢”, 大多数计算机系统将该数据按零处理。

如果计算本身不应产生溢出, 而是由于算术表达式书写不当使中间计算产生溢出, 我们就要避免。我们如果在 IBM-PC 上计算 xy/z , 若 $x=3 \cdot 10^{30}$, $y=4 \cdot 10^{10}$, $z=2 \cdot 10^{20}$, 如果用表达式 $(X * Y)/Z$, 则 $3E30 * 4E10/2E20$ 上溢; 若用表达式 $X * (Y/Z)$, 则 $3E30 * (4E10/2E20) = 6E20$ 。若再设 $x=3 \cdot 10^{-30}$, $y=4 \cdot 10^{-10}$, $z=2 \cdot 10^{-20}$, 如果用表达式 $(X * Y)/Z$, 则 $3E-30 * 4E-10/2E-20 = 0.0$ 下溢; 若用表达式 $X * (Y/Z)$, 则 $3E-30 * (4E-10/2E-20) = 6E-20$ 。因此, 要防止上溢就要避免两个大数相乘, 要防止下溢就要避免两个小数相乘。

5. 算术赋值语句

算术赋值语句的形式是

$$V=e$$

其中: V (V 是 Variable 的字头) 是整型、实型、双精度型、复型变量名或数组元素名;

e (e 是 Express 的字头) 是算术表达式。

算术赋值语句的执行是先计算 e 的值, 然后将 e 的值转换成 V 具有的类型, 并将此结果值赋给 V 。

例 2.1 下面是一些算术赋值语句

$$L=2.0 * 3.14159 * R$$

$$S=3.14159 * R * * 2$$

$$ZETA=1.0/(X * * 2 + Y * * 2)$$

$$D=B * * 2 - 4.0 * A * C$$

赋值语句中的“=”是“赋值”的意思, 它与数学上的“相等”是不同的概念。例如 $N=N+5$ 在数学上讲这个方程是无意义的, 而 $N=N+5$ 却是合法的赋值语句, 它表示把 N 的当前值加上 5 后再赋给 N 。赋值号左边必须是变量名或数组元素名, 不允许使用数组名、常数符号名和常数。一句赋值语句只能用一个赋值号, 在 FORTRAN 77 中 $X=Y=Z=15$ 是错误的。

例 2.2 下列赋值语句中的变量类型由 I-N 规则确定, 按给定的值, 指出赋值语句执行后赋给左端变量的值。

$$(1) Y=(L+M)/(N-5)$$

设 $L=7$, $M=16$, $N=9$, 则

$$(L+M)/(N-5)=(7+16)/(9-5)=23/4=5$$

由于左端变量 Y 为实型, 所以要把整常数 5 转换成实型值 5.0 赋给 Y 。

$$(2) L = A + B / C$$

设 $A = 7.8$, $B = 1.0$, $C = 3.0$, 则

$$A + B / C = 7.8 + 1.0 / 3.0 = 7.8 + 0.333\ 333\ 3 = 8.133\ 333\ 3$$

由于左端变量 L 为整型, 则把实型值 $8.133\ 333\ 3$ 取整数部分, 即转换为整型值 8 , 然后赋给变量 L 。

$$(3) I = (J + K) / (L - 8)$$

设 $J = 13$, $K = 9$, $L = 8$, 则

$$(J + K) / (L - 8) = (13 + 9) / (8 - 8) = 22 / 0$$

因为分母为零, 表达式不能求值。

$$(4) X = Y + Z * * (M / N)$$

设 $Y = 7.0$, $Z = -3.0$, $M = 1$, $N = 3$, 则

$$\begin{aligned} Y + Z * * (M / N) &= 7.0 + (-3.0) * * (1 / 3) \\ &= 7.0 + (-3.0) * * 0 \\ &= 7.0 + 1.0 = 8.0 \end{aligned}$$

于是, 把 8.0 赋给 X 。

二、算术关系表达式

算术关系表达式用来比较两个算术表达式的值。

1. 关系运算符

关系运算符见表 2.8。

表 2.8

运算符	所代表的数学符号	英语含义
.LT.	< (小于)	Less Than
.LE.	≤ (小于或等于)	Less than or Equal to
.EQ.	= (等于)	Equal to
.NE.	≠ (不等于)	Not Equal to
.GT.	> (大于)	Greater Than
.GE.	≥ (大于或等于)	Greater than or Equal to

关系运算符中作为分界符的圆点是必须的, 而且所有关系运算符的优先级相同。

2. 算术关系表达式

算术关系表达式的形式是

$$e_1 \text{ relop } e_2$$

其中: e_1 和 e_2 是整型、实型、双精度型或复型表达式;

relop (relational operator) 是关系运算符。凡涉及复型表达式时, 关系运算符只允许用 .EQ. 或 .NE.。

例如:

$$B * B - 4.0 * A * C .LT. 0.0 \quad \text{表示 } B^2 - 4AC < 0$$

$\text{SIN}(X) . \text{GE. } 0.5$	表示 $\sin x \geq 0.5$
$X1 . \text{LE. } X2$	表示 $x1 \leq x2$
$I . \text{NE. } 100$	表示 $I \neq 100$

以下几点应当注意:

① 若一个关系表达式中包括算术运算符,那么先进行算术运算,然后进行关系运算。例如“ $A+B . \text{LT. } X+Y$ ”这个关系表达式在执行时,先分别计算出 $A+B$ 和 $X+Y$ 的值,然后再将它们进行比较,判断是否满足“小于”这个条件。这个表达式相当于“ $(A+B) . \text{LT. } (X+Y)$ ”,括号可以不加,但加上括号,更清晰些。

② 关系表达式是将两个数值量相比较,这两个数值量可以是不同类型的量。例如,常常会遇到一个整型量和实型量相比较,系统会先将整型量转换成实型量,然后再比较,例如

$A . \text{LE. } 8$

先将整数 8 转换为实数 8.0,然后与实型量 A 比较。

③ 我们已知实数的存贮与运算会产生一些微小的误差,因此用 EQ. 和 NE. 这两个关系运算符要特别注意,可能在数学上相等的量用关系运算符进行比较时,结果却是不相等。例如

$(1.0/3.0+1.0/3.0+1.0/3.0) . \text{EQ. } 1.0$

从数学上看,这个关系表达式应当成立,但在实际上却不成立,这是由实数存贮的误差引起的。左边的算术表达式的值为 0.999 999 9,而不等于 1.0。那么,如何来避免这种问题的产生呢?例如,对

$X . \text{EQ. } Y$

可改用

$\text{ABS}(X-Y) . \text{LT. } 1\text{E}-6$

其中,ABS 为内部函数(见 2.4 节)表示取绝对值,即当 X 与 Y 之差的绝对值小于 10^{-6} ,则认为 X 与 Y 相等。同样,对以下关系表达式

$U . \text{NE. } V$

可改用

$\text{ABS}(U-V) . \text{GT. } 1\text{E}-6$

即当 U 与 V 之差的绝对值大于 10^{-6} ,则认为 U 与 V 不相等,只要不大于 10^{-6} 就认为是相等。

④ 关系表达式的值不是一个数值,而是一个逻辑量。逻辑量的值只有 TRUE. (真)或 FALSE. (假)之一。例如

$X . \text{LT. } 0.0$

当 $X < 0$ 时,此关系表达式的值为 TRUE. ,也就是说“ $X < 0$ ”这个条件成立。如果 $X \geq 0$,则此关系表达式的值为 FALSE. ,也就是说“ $X < 0$ ”这个条件不成立。

⑤ 不允许连续进行关系运算。因此不等式 $0 \leq x \leq 10$ 不能写成 $0.0 . \text{LE. } X . \text{LE. } 10.0$ 。

三、逻辑表达式与逻辑赋值语句

逻辑表达式能把几个简单的判别条件组合成一个复杂的判别条件。例如，判别条件 $0 \leq x \leq 10$ ，就是判别 $0 \leq x$ 和 $x \leq 10$ 是否同时成立。这就需要用逻辑表达式来描述。

1. 逻辑运算符

FORTRAN 77 有 5 种逻辑运算符，见表 2.9。

表 2.9

运算符	表 示	英 语 含 义
.NOT.	逻辑非(单目运算符)	not
.AND.	逻辑与(双目运算符)	and
.OR.	逻辑或(双目运算符)	or
.EQV.	逻辑等值(双目运算符)	equivalence
.NEQV.	逻辑不等值(双目运算符)	non - equivalence

逻辑运算符的优先级见表 2.10。

表 2.10

运 算 符	优 先 级
.NOT.	最 高
.AND.	↓
.OR.	
.EQV. 或 .NEQV.	
	最 低

我们知道逻辑量只有两个值，TRUE. 或 FALSE.。我们用 T 代表 TRUE.，用 F 代表 FALSE.，并设 X 和 Y 是逻辑变量。那么，表 2.11 是逻辑运算的真值表。

表 2.11

X	Y	.NOT. X	.NOT. Y	X.AND.Y	X.OR.Y	X.EQV.Y	X.NEQV.Y
F	F	T	T	F	F	T	F
F	T	T	F	F	T	F	T
T	F	F	T	F	T	F	T
T	T	F	F	T	T	T	F

2. 逻辑表达式

逻辑表达式用来表示逻辑计算。逻辑表达式的求值产生一个逻辑型的结果，其值为 TRUE. 或 FALSE.。一个逻辑表达式可以包括多个逻辑运算符，即多个逻辑量经过逻辑运算后可以得到一个逻辑值。

以下是一些逻辑表达式：

```
(A.GT.3.0.AND..NOT.C.EQ.D).OR.B.LT.C
Q.EQV.R.AND.S.OR.T
```

$$(A+B).GT.C.AND.(B+C).GT.A.AND.(C+A).GT.B$$

从这些式子中可看出逻辑表达式中不仅有逻辑运算符,也有关系运算符和算术运算符。那么它们的运算次序是如何规定的呢? FORTRAN 77 规定按以下次序进行运算:

- ① 先求算术表达式的值;
- ② 再求关系表达式的值;
- ③ 最后求逻辑表达式的值。

在各种运算中,如果有括号,则先进行括号内运算。我们把各种运算符的优先级别归纳在表 2.12 中。

表 2.12

运算类别	运 算 符	优先级
算术运算	* *	1
	*, /	2
	+, -	3
关系运算	.LT., .LE., .EQ., .NE., .GT., .GE.	4
逻辑运算	.NOT.	5
	.AND.	6
	.OR.	7
	.EQV., .NEQV.	8

对下面列出的逻辑表达式,其运算次序如下所示:(设 $X=-1.2$, $Y=5.7$, $Z=1.0$)

$$X.GE.0.1.AND.X+Z.GT.Y*Z.OR..NOT..FALSE.$$

① -0.2 ① 5.7

② F
② F
③ T

④ F

⑤ T

即

- ① 求出 $X+Z=-0.2$, $Y*Z=5.7$;
- ② 求出“ $X.GE.0.1$ ”的值为.FALSE.; “ $X+Z.GT.Y*Z$ ”的值为.FALSE.;
- ③ 进行.NOT.运算。“NOT.FALSE.”的值为.TRUE.;
- ④ 进行.AND.运算。“FALSE.AND.FALSE.”的值为.FALSE.;
- ⑤ 最后进行.OR.运算。“FALSE.OR.TRUE.”的值为.TRUE.。

前面提到不允许连续进行关系运算,那么如何表示 $0 \leq x \leq 10$ 呢? 运用逻辑运算符就不难表示了,只要写成

$$(0.0.LE.X).AND.(X.LE.10.0)$$

掌握了逻辑运算，我们很容易用逻辑表达式表示一个开关电路。如果一个开关闭合取值 TRUE，而打开取值 FALSE，那么一个电路导通取值 TRUE，不导通取值 FALSE。对图 2.1 所示两个电路可写出对应的逻辑表达式。

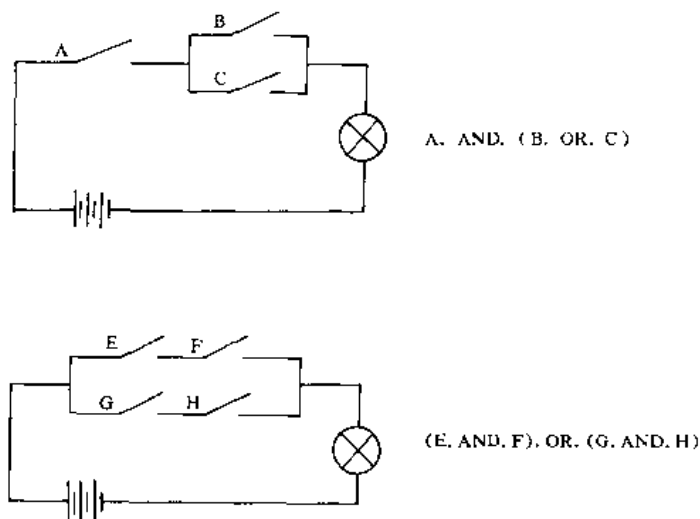


图 2.1

3. 逻辑赋值语句

逻辑赋值语句的形式是

$$V = e$$

其中：V 是逻辑变量名或逻辑数组元素名；

e 是逻辑表达式。

逻辑赋值语句的执行是先求逻辑表达式的值，然后将 e 的值赋给 V，使 V 有定义。注意 e 必须有值 TRUE 或 FALSE。

假定 X、Y 已被定义为逻辑型变量，则可用逻辑赋值语句对它们赋值：

X = .TRUE.

Y = .FALSE.

赋值后 X 的值为 TRUE，Y 的值为 FALSE。

2.4 内部函数

在数学中，表达式不仅可以包含简单的操作数像 x 或 y ，而且也涉及到像正弦和余弦这样的函数。由于这些常用函数在数学中用得频繁，因而在 FORTRAN 程序中也经常用到。为此 FORTRAN 77 编译系统为用户提供了一些常用函数的计算程序，称为内部函数。编译系统定义了这些函数的函数名、函数值的数据类型，规定了参加运算的自变量的个数、次序与类型。程序可直接使用这些函数，使用时要符合系统定义的使用方法。

如果在程序中要计算变量 x 的平方根、正弦值或余弦值，那么就可以用 SQRT(X)，SIN(X) 或 COS(X) 来实现。这里 SQRT，SIN 和 COS 是函数名，而 X 是自变量。FORTRAN 77

提供的全部内部函数见附录 A。一些常用的内部函数见表 2.13。

表 2.13

函数名	含 义	应 用 例 子	相当数学上的运算
ABS	绝对值	ABS(X)	$ x $
EXP	指数	EXP(X)	e^x
SQRT	平方根	SQRT(X)	$x^{\frac{1}{2}}$
SIN	正弦	SIN(X)	$\sin x$
COS	余弦	COS(X)	$\cos x$
ASIN	反正弦	ASIN(X)	$\arcsin x$
ACOS	反余弦	ACOS(X)	$\arccos x$
TAN	正切	TAN(X)	$\tan x$
ATAN	反正切	ATAN(X)	$\arctan x$
LOG	自然对数	LOG(X)	$\ln x$ 或 $\log_e x$
LOG10	常用对数	LOG10(X)	$\log_{10}(x)$
MOD	求余	MOD(X1, X2)	$x_1 - \text{int}(x_1/x_2) * x_2$
SIGN	符号传送	SIGN(X1, X2)	$ x_1 $ 当 $x_2 \geq 0$; $- x_1 $ 当 $x_2 < 0$
INT	转换为整数	INT(X)	$\text{int}(x)$, 取 x 的整数部分
REAL	转换为实数	REAL(X)	
MAX	选最大值	MAX(X1, X2, X3)	$\max(x_1, x_2, x_3)$
MIN	选最小值	MIN(X1, X2, X3)	$\min(x_1, x_2, x_3)$

现在来看一下对内部函数求值的例子:

$$\text{SQRT}(7.0) = 2.645\ 751$$

$$\text{SIGN}(-3.0, -2.0) = -3.0$$

$$\text{SIGN}(-3.0, 2.0) = 3.0$$

$$\text{LOG}(3.0) = 1.098\ 612$$

$$\text{INT}(17.3) = 17$$

$$\text{INT}(-17.3) = -17$$

$$\text{REAL}(17) = 17.0$$

$$\text{MOD}(17, 8) = 1$$

$$\text{MAX}(7, 3, 12, -1) = 12$$

$$\text{MIN}(7, 3, 12, -1) = -1$$

$$\text{SIN}(1.0) = \sin(57.295\ 78^\circ) = 0.841\ 470\ 9$$

$$\text{COS}(1.0) = \cos(57.295\ 78^\circ) = 0.540\ 302\ 3$$

$$\text{TAN}(1.0) = \tan(57.295\ 78^\circ) = 1.557\ 408$$

$$\text{ATAN}(1.557\ 408) = 1.0(\text{弧度})$$

几点说明:

① 一个内部函数要求一个或多个自变量。从附录 A 可以查出各个内部函数的自变量个数。在表 2.13 中已可看出一些常用内部函数的自变量个数。其中 MOD(X1, X2) 与 SIGN(X1, X2) 均为两个自变量, 而且这两个自变量的顺序不能任意颠倒。而 MAX 函数与 MIN 函数的自变量个数 ≥ 2 , 且函数值与自变量的顺序无关。

② 函数的自变量是有类型的, 函数值也是有类型的。例如, MOD(17, 8) 中自变量为整型, 函数 MOD(17, 8) 的值“1”也是整型; 如果写成 MOD(17.0, 8.0), 自变量是实型的, 函数值也是实型的, 其值为 1.0

③ 三角函数中角度的单位是“弧度”而不是“度”。上面例子已指出 SIN(1.0) 表示的不是 $\sin 1^\circ$, 而是 $\sin 57.29578^\circ$ 。 $\sin 43^\circ$ 应写成 $\text{SIN}(43 * 3.14159/180)$ 。

④ 自变量可以是常量、变量或表达式。例如 SQRT(7.8), SQRT(B*B-4.0*A*C) 均为合法。但类型应符合要求, 如果程序中对变量 I 没有做是非整型的说明, 那么 SQRT(I) 就不合法, 因为 I 为整型变量, 而 SQRT 函数的自变量不能是整型量。

⑤ 转换为整数函数 INT 的功能是简单地截去小数部分, 如 $\text{INT}(33.4) = 33$, $\text{INT}(-33.4) = -33$ 。

⑥ SIGN(X1, X2) 的功能是将 X2 的符号传送给 |X1|。例如, $\text{SIGN}(17.3, -4.5) = -17.3$, $\text{SIGN}(-17.3, -4.5) = -17.3$, 而 $\text{SIGN}(-17.3, 4.5) = 17.3$ 。利用这个函数可以检查 X1 和 X2 是否同符号。如果 $\text{SIGN}(X1, X2) = X1$, 则 X1 和 X2 同符号, 而如果 $\text{SIGN}(X1, X2) \neq X1$, 则 X1 和 X2 符号相异。

2.5 DATA 语句和 PARAMETER 语句

一、DATA 语句

DATA 语句用于给变量、数组、数组元素和子串提供初值。DATA 语句是非执行语句。若在程序单位里有 DATA 语句, 它可出现在说明语句后的任何地方。

DATA 语句的形式是

DATA nlist/clist/[[,]nlist/clist/]...

其中: 逗号是任选的, 但是为了增加易读性, 一般不省略逗号;

nlist 是由变量名、数组名、数组元素名、子串名和隐含 DO 表(见 5.4 节)构成的表。

clist 是下述形式的表:

$a[,a] \dots$

这里 a 是 c 或 $r * c$ 的形式, 而 c 是常数或常数符号名; r 是非零无符号整常数或是这样的常数符号名。 $r * c$ 形式等价于常数 c 连续出现 r 次, 这里的“*”不是乘法运算。

DATA 语句的功能是使程序在开始执行前, 把 clist 中给出的值依次赋给 nlist 中的每个元素。

例 2.3 DATA I, J, K, L/4 * 1/

这个语句表示 I, J, K, L 的初值分别为 1。

DATA A, B, C,/67.87, 54.72, 5.0/

等价于

```
DATA A/67.87/, B/54.72/, C/5.0/
```

这两个语句均表示 A, B, C 的初值分别为 67.87, 54.72, 5.0。

例 2.4 LOGICAL L

```
DATA X, Y, I, J, L/4.8, -7.2, 2*4, .FALSE. /
```

等价于

```
LOGICAL L
```

```
DATA X, Y, I, J/4.8, -7.2, 2*4/L/.FALSE. /
```

等价于

```
LOGICAL L
```

```
DATA X, Y/4.8, -7.2/
```

```
DATA I, J/2*4/
```

```
DATA L/.FALSE. /
```

这些语句表示 L 为逻辑变量名, 其初值为逻辑常数 .FALSE., X, Y, I, J 的初值分别为 4.8, -7.2, 4, 4。

使用 DATA 语句应当注意以下几点:

① clist 中只允许出现常数或常数符号名, 不允许出现任何其它形式的表达式。nlist 中的变量个数与对应 clist 表中的常数个数必须相同, 类型按从左到右的顺序一一对应相同。nlist 中变量之间, clist 中初值之间都用逗号隔开。若给出

```
DATA A, J, K/3.2, 4/
```

是错误的, 因为只有两个常数, 而变量是 3 个。

② 程序在编译期间给变量赋以初值, 因此在同一程序单位中一个变量只可能有一个初值。如果在一个程序单位中有多个 DATA 语句给同一个变量赋初值, 则以最后一条 DATA 语句所赋初值为准。例如, 在程序中有

```
DATA K, L/8, 7/
```

```
DATA K/18/
```

这两个语句意味着 K, L 的初值为 18, 7。

③ 当 nlist 中的元素是整型、实型、双精度型或复型时, clist 相对应的常数也必须是整型、实型、双精度型或复型这四种类型之一, 但当两者类型不一致时, 按算术赋值语句的规定进行类型转换, 然后再赋初值。

二、PARAMETER 语句

PARAMETER 语句用于给常数一个符号名。

PARAMETER 语句的形式是

```
PARAMETER(p=e[,p=e]...)
```

其中: p 是符号名;

e 是常数表达式。

在 PARAMETER 语句中, 每个 p 只能标识该程序单位中的相应常数, 故称为常数符号名, 这个符号名根据赋值语句的规则用等号右边的表达式 e 所确定的值定义。例如

```
PARAMETER(PI=3.14159)
```


那么在程序中要多次用到 $\pi=3.141\ 59$ 时,就不必每次重复写 3.141 59,而用 PI 来代表 3.141 59 即可。

PARAMETER 语句与赋值语句不同,它是非执行语句。在同一个程序单位中,一个常数符号名只能定义一次,即在一个程序中定义一个常数符号名后,不能再改写它的值。因此,在一个程序单位中有了

```
PARAMETER(PI=3.141 59)
```

又出现

```
PI=36.48
```

是错误的。

例 2.5 在 FORTRAN 77 程序单位中有下列语句:

```
PARAMETER(I=300)
```

```
PARAMETER(PI=3.141 59)
```

```
PARAMETER(X=-9.2)
```

这些语句命名了:整常数 300 的符号名为 I,实常数 3.141 59 的符号名为 PI,实常数 -9.2 的符号名为 X。这三个 PARAMETER 语句可以用下面一个 PARAMETER 语句代替:

```
PARAMETER(I=300, PI=3.141 59, X=-9.2)
```

使用 PARAMETER 语句必须注意以下几点:

① 若符号名 p 是整、实、双精度或复型,则相应的表达式 e 必须是算术表达式。若符号名 p 是字符型或逻辑型时,则相应的表达式 e 必须分别是字符常数表达式或是逻辑常数表达式。

例 2.6 符号名的类型遵循 I-N 规则,那么

(a) PARAMETER(R=48)

因为符号名 R 为实型,这里按赋值语句规则,R 的值应为 48.0。

(b) PARAMETER(L=9.4)

因为符号名 L 为整型,那么按赋值语句规则,L=9。

② 若一个常数符号名的类型不遵循 I-N 规则时,应在 PARAMETER 语句之前由类型语句或 IMPLICIT 语句来说明。例如,把例 2.6 之(a)改为

```
INTEGER R
```

```
PARAMETER(R=48)
```

即符号名 R 为整型的,它的值为 48。

③ 在同一个程序单位内,出现在表达式 e 中的任一常数符号名,必须是在同一 PARAMETER 语句中先前定义过的或不同的 PARAMETER 语句中已经定义过的。

例 2.7 下而的(a)和(b)作用是相同的:

(a) PARAMETER(PI=3.141 59, A=2 * PI)

(b) PARAMETER(PI=3.141 59)

```
PARAMETER(A=2 * PI)
```

④ 常数符号名不得用来形成另一个常数的一部分,例如,形成一个复常数的任一部分。

⑤ 常数符号名不能作为语句标号,也不能出现在 FORMAT 语句(见 3.2 节)中代替

常数。

常数符号名与变量名有以下几个区别：

① DATA 语句中 clist 表中的某一常数的重复系数 r 可以使用常数符号名，但不允许用变量名。

② 定义方式上，常数符号名仅仅能用 PARAMETER 语句定义，而变量名可由赋值语句、DATA 语句等多种方式来定义。

③ 常数符号名定义之后其值不可改变，而我们可以随意对变量名赋以新值。

例 2.8 在一个 FORTRAN 77 程序单位中，下列语句序列是正确的：

```
INTEGER S
```

```
COMPLEX W
```

```
DOUBLE PRECISION D
```

```
PARAMETER(I=179, S=286, W=(4.8, 7.2), D=7.345 678 931 234D+2)
```

例 2.9 变量类型遵循 I-N 规则，指出下面的 PARAMETER 语句中存在的错误。

```
PARAMETER(L=4, V=64.L, U=7.8, A=E * * 2, U=56.7, E=2.718 28)
```

错 误	理 由
V=64.L	常数符号名不能是另一常数的一部分
A=E * * 2	常数符号名 E 在前面尚未定义
U=56.7	U 重复定义

常数可以使用符号名来命名，给编写程序带来了许多方便。例如，某一常数在程序中要多处使用，就可以统一使用符号名代替该常数。若该常数需要修改，仅需在 PARAMETER 语句中修改该常数符号名的定义，而不必在程序中出现该常数的地方都进行修改。

2.6 STOP 语句和 PAUSE 语句

STOP 语句和 PAUSE 语句可以停止正在执行的程序，这也是对程序的一种控制。

一、STOP 语句

STOP 语句的形式是

```
STOP[n]
```

其中：n 是一个不多于 5 位的数字串或者是一个字符常数。

执行 STOP 语句导致停止程序的执行，且把 STOP 后面出现的数字串或字符常数显示在屏幕上。例如

```
STOP 3456
```

或

```
STOP 'END OF PROGRAM'
```

则在程序停止执行时，在屏幕显示“3456”或“END OF PROGRAM”。

一个程序可以在不同的地方出现 STOP 语句，它们可用 STOP 后面的 n 来区分。可以用 n 来辨别导致停止的原因和辨别程序的流向。子程序中如果有 STOP 语句，也是使整个程

序停止执行而不是使控制返回主程序。

当然, 如果仅写

STOP

则在程序停止执行时, 屏幕上不显示任何内容。

二、PAUSE 语句

PAUSE 语句的形式是

PAUSE[n]

其中: n 是一个不多于 5 位的数字串或者是一个字符常数。

执行 PAUSE 语句导致暂停程序的执行, 且把 PAUSE 后面出现的数字串或字符常数显示在屏幕上。执行是可以恢复的。在打入“恢复命令”后再从 PAUSE 语句的下一语句开始继续程序的执行。而“恢复命令”是随计算机处理系统的不同而不同。例如, 在 IBM-PC 机上, 只要打入回车键即可恢复执行。“恢复命令”不属于 FORTRAN 语句范畴。例如

PAUSE 'END OF FIRST ITERATION'

或

PAUSE 100

则在程序暂停执行时, 在屏幕上显示 END OF FIRST ITERATION 或 100。

当然, 如果仅写

PAUSE

则在程序暂停执行时, 屏幕上不显示任何内容。

在一个程序中, 可出现多个 PAUSE 语句, 它们用 PAUSE 后面的 n 来区分。可用 PAUSE 语句对一个程序进行分段执行, 每段执行后, 暂停一下, 以检查执行的情况或结果, 也可进行人工干预。

习 题 二

本习题中若无特别说明, 变量的类型遵循 I-N 规则。

1. 给出

2225	.137	2E-7	-25
1.562	E-10	-97612	101.2E+2.
-.137	10000	0	+0162.2E-02
005326	+258.	3.6E-22	2,360
23.34.	111-333		

其中, 哪些是合法的整常数? 哪些是合法的实常数? 哪些两者皆不是(并说明理由)?

2. 给出

PI	2X12	NUMBER	M63-2	BSQUARED
AMPS	FRED	I123	ZETA	X+Y
ITER	K(2)	Q	IP	COUNTS
INC*	NUMBER1	L3.6	POWER	L1369P

JIM'S

其中, 哪些是合法的整型变量名? 哪些是合法的实型变量名? 哪些两者皆不是(并说明理由)?

3. 用 FORTRAN 77 算术赋值语句表示下述代数方程:

$$(1) x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$(2) a = \frac{3xy^2(z+1) + \frac{1}{2}yz}{1+x+x^3}$$

$$(3) b = \frac{3}{2}x(x-1)[7y - \ln(\cos x)]^{\frac{1}{2}}$$

$$(4) c = \frac{5x^3[\cos^3(x^2-y^2) + \tan^{-1}(x \cos x)]^{\frac{1}{2}}}{(e^{x+1} + e^{x+1} + 1)}$$

$$(5) d = a \left(x + \frac{d}{2a} \right)^2 + \frac{4ac - b^2}{4a^2}$$

$$(6) e = \sin^2 x - \cos^2 x$$

$$(7) f = e^x - e^{-x}$$

$$(8) g = \sqrt{|p-q|}$$

4. 设 A=100., B=-2.4, C=81., D=0.0 试求下列函数值:

$$(1) \text{SQRT}(C)$$

$$(2) \text{SQRT}(\text{SQRT}(C))$$

$$(3) \text{INT}(B)$$

$$(4) \text{SIGN}(A, B)$$

$$(5) \text{SIGN}(B, A)$$

$$(6) \text{MIN}(A, B, C, D)$$

$$(7) \text{MAX}(A, 2 * B, 4. * B, C, 3 * D)$$

$$(8) \text{ABS}(B) + \text{ABS}(C)$$

$$(9) \text{LOG}_{10}(A)$$

$$(10) \text{SIN}(D) + \text{COS}(D)$$

5. 用 FORTRAN 77 的算术赋值语句表示下面的代数方程。

$$i = \frac{k}{j} \left[10^2 x - \frac{1}{3.0} \frac{(k)^2}{j} \right]$$

当 J=3, K=7, X=0.5 时, 求 I 的值。

6. 如果 I=1, J=2, K=3, L=4 和 M=5, 计算下列表达式的真值:

$$(1) J .GT. K .OR. L .EQ. M - 1$$

$$(2) I + J .NE. K * L .AND. .NOT. M .GT. 3 .OR. K .LT. 2$$

$$(3) .NOT. (J - 3 .EQ. (-1)) .OR. .NOT. (M .LT. 50 .AND. K .EQ. I + 1)$$

7. 计算下列表达式的值:

$$3/4 \quad 8/3 \quad 3./4.$$

$$8./3. \quad 1/2 + 1/2 \quad 1./2. + 1./2.$$

8. 计算下列表达式之值:

$$3/2. \quad 3./2 \quad 4. + 3/2 \quad 4. * 3/2$$

$$4. * (3/2) \quad 4 + 3/2. \quad -1 + 7.2$$

9. 给出

'ALL'S WELL THAT ENDS WELL'

'TODAY'S DATE IS 8 : 3 : 94'

'THE VALUE OF X=

2.35675D2

.TRUE.

(3.65, 2E5)

(5.3 2.1)

.F.

其中, 哪些是合法的 FORTRAN 77 常数? 哪些是不合法的? 并说明理由。

10. 设 $A=3.$, $B=-2.$, $I=6$, $J=0$, 计算下列表达式的值:

- | | | |
|----------------------|----------------|-------------------|
| (1) $A ** 2 + B$ | (2) $I + 2/3$ | (3) $A * * B$ |
| (4) $A * 3. + B * 4$ | (5) A/B | (6) $A/B * 3 + A$ |
| (7) $A/B/2$ | (8) $A/B + 2$ | (9) J/I |
| (10) I/J | (11) $A * * I$ | (12) $(A + I)/B$ |
| (13) $A * * 2 * * 3$ | (14) $B * * B$ | (15) $J * * B$ |

11. 设 $A=3.2$, $B=-2.$, $I=6$, $J=0$, 由下列的赋值语句, 什么值将存于 X 或 IX 中?

- | | | |
|---------------|-----------------|---------------------|
| (1) $X=I$ | (2) $IX=A$ | (3) $X=(I+3)/2$ |
| (4) $IX=-A+B$ | (5) $X=I * * B$ | (6) $X=J * I/.1$ |
| (7) $X=J$ | (8) $X=B * * J$ | (9) $IX=J * \wedge$ |

12. 通过下面一系列赋值语句的运算, 什么值将存于 S、J 和 JK 中。

$I=4.$	$S=(3/I) * 3$
$A=1$	$J=(3./9) * 4$
$B=2$	$JK=(A+2./B)/2$

13. 下面两个不等式是否可能成立:

- (1) $\frac{i}{j}C \neq \frac{i \cdot C}{j}$
 (2) $C \cdot i/j \neq C \cdot (i/j)$

14. 将下列条件写成 FORTRAN 关系表达式:

- (1) $A \geq B$
 (2) $C + D < E + F$
 (3) $U + V \neq X + Y$
 (4) $(32 + S)(63 + t) + \sin 37^\circ > e^x \cdot 36 \ln x$
 (5) $3x + 6y \leq 7x^2 + 8y^2 + 9$
 (6) $7 \cdot \sin(x + y)^2 = \tan^{-1} \frac{\sqrt{a^2 + b^2}}{|c|}$

15. 将下面的条件写成 FORTRAN 逻辑表达式:

- (1) $x \geq 0$ 和 $x \leq 15$
 (2) $x < 0$ 或 $x > 7$
 (3) $2 \leq x \leq 9$ 和 $1 \leq y \leq 10$
 (4) 有三个坐标点: $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ 。BC 之间距离大于 AB 之间距离。
 (5) x 和 y 都是负的或都是正的。

(6) x 和 y 之一为零, 但不能都为零。

16. 设 a, b, c, d, e 为逻辑型变量, 判断下述命题的真假:

(1) $A .AND. B .AND. C$ 与

$(A .AND. B) .AND. C$ 等价

(2) $.NOT. A .OR. B .AND. C$ 与

$((.NOT. A) .OR. B) .AND. C$ 等价

(3) $.NOT. A .EQV. B .OR. C .NEQV. D .AND. E$ 与

$(.NOT. A) .EQV. (B .OR. C) .NEQV. (D .AND. E)$ 等价

17. 确定图 2.2 所示流程图的输出结果:

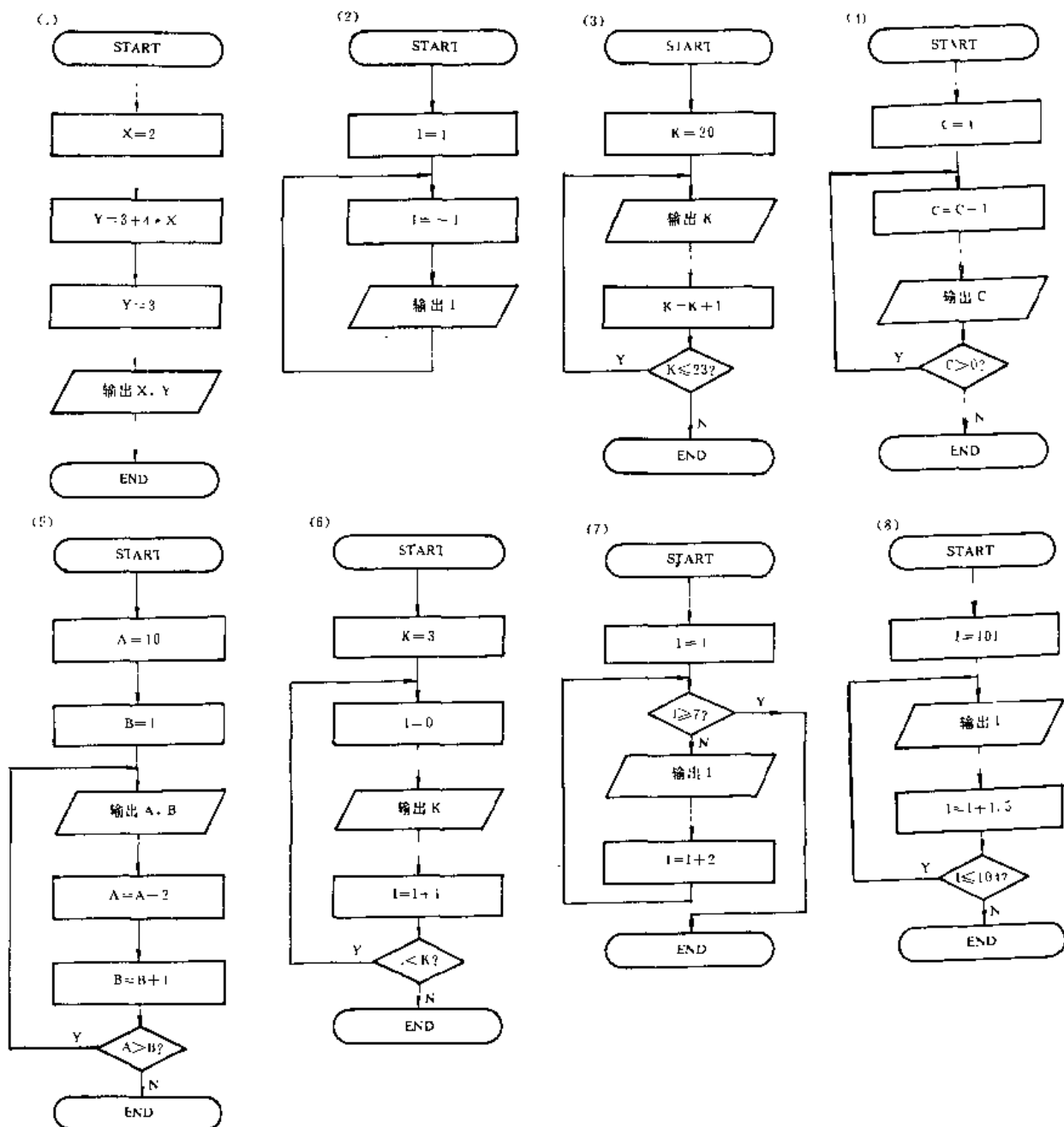


图 2.2

18. 指出显式说明、隐式说明和预隐含规则之差异及使用规则。
19. DATA 语句使用的目的? 在 FORTRAN 程序中它是必不可少的吗? 为什么?
20. IMPLICIT 语句使用的目的? 在 FORTRAN 程序中它是必不可少的吗? 为什么?
21. 判断下述命题的真假:
 - (1) 说明语句的错误在执行时才被发现。
 - (2) DATA 语句可以放在 FORTRAN 程序中任何地方。
 - (3) 语句 INTEGER IWAS, WAS, K32 是合法的。
 - (4) IMPLICIT INTEGER P-S 是不合法的说明语句。
 - (5) DATA A, B, C, L, M, N/3*-1, 3, 4, 10/ 是合法的说明语句。
 - (6) DATA A, B, C, D, E/5*(-1)/ 是合法的说明语句。
22. 给出
DATA A, B, C/34.8, 72.6, 31.5/, D, E/10.2, 7.5/, I, J/8, 9/
其中哪些逗号是必不可少的? 哪些逗号可以省略?
23. 试用逻辑表达式表示图 2.3 所示的电路。

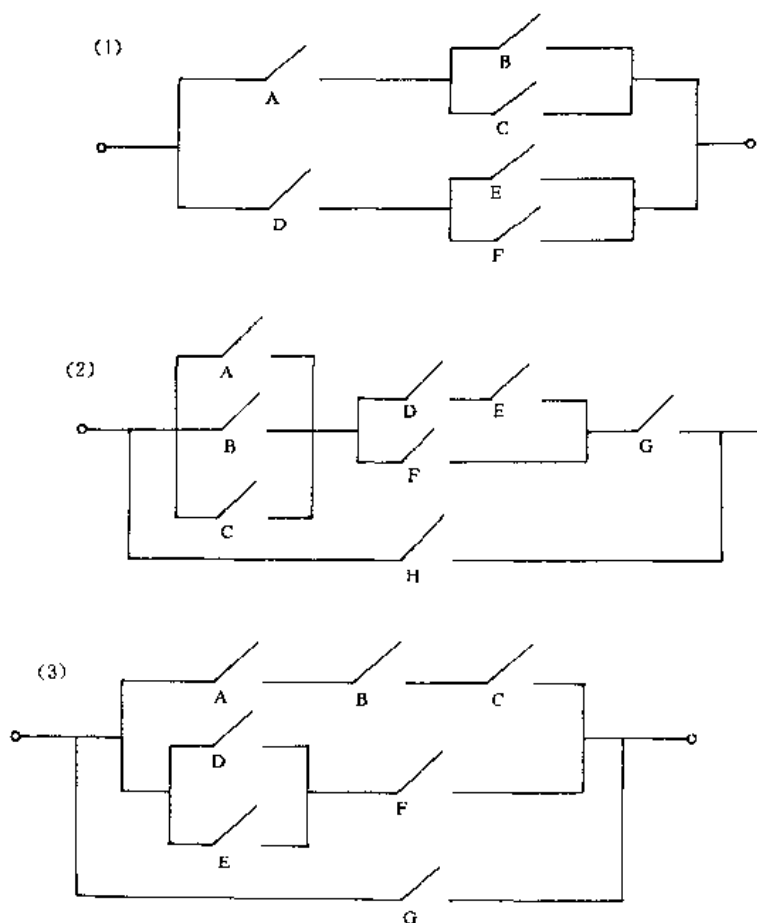


图 2.3

3 输入与输出初步

在这一章，我们主要介绍 FORTRAN 77 语言中有关输入/输出的基本概念，并讨论 FORTRAN 77 语言中相应的输入/输出命令。通过这一章的学习，读者应掌握用 FORTRAN 77 语句处理简单的输入、输出问题。有关文件的输入/输出的详细讨论，将在 9.3 节中进行。

3.1 输入、输出的概念

数据的输入/输出是计算机运行中一项经常性的工作。程序处理的对象和处理的结果都是数据。因此，计算机的使用离不开数据的输入/输出。要处理好输入/输出问题，我们首先要正确理解输入与输出的概念。

输入过程指的是数据从外部介质传输到计算机内存的过程。

这里所说的外部介质包括终端键盘、磁盘、磁带等。例如，从磁盘上将数据文件中的数据读入计算机内存的过程就是一种输入过程。而输出过程则相反，它是指将内存中的数据传送到外部介质的过程。这里所说的外部介质包括终端显示器、打印机、磁盘、磁带等。例如，将内存中的数据送到打印机上打印就是一种输出过程。

现代计算机科学对于数据的输入/输出要求做到准确、多样以及高速。即要求具有迅速而大量的数据吞吐能力。因此，FORTRAN 77 在文件系统的基础上具备了输入/输出的强大功能。它要求被传输的数据组成文件的形式，在操作系统的控制下，完成各种输入/输出语句所规定的功能。

3.2 基本的输入/输出语句

一、READ 语句

READ 语句是 FORTRAN 77 实现输入的手段。其最简单的形式为表控输入，其格式为

```
READ *, iolist
```

其中：“*”是外部部件标识符的一种，它标识所用的部件是计算机系统规定的预连接部件。通常，在输入语句中“*”是指终端键盘。

iolist 是变量列表，它由一个或多个用逗号分隔的变量名(或者是数组名、数组元素名、字符子串名等)所组成。

所谓“表控输入”是指由计算机系统所指定的设备将数据输入到变量列表所指定的变量中去。这里所说的“计算机系统所指定的设备”其含义是指计算机系统所规定的某些外部设备，这些设备专门用于数据的表控输入，如键盘等。在程序设计时，程序员不必专门指定这些设备就可直接实现数据的输入。本书约定终端键盘作为由系统指定的输入设备。

例 3.1 表控输入语句

```
READ *, X, M, N
```

这个输入语句的输入变量列表由 X, M, N 组成。这条语句的执行，就是从系统所指定的输入设备上，读入三个数据值，分别赋给变量列表中所列的变量。显然，这时我们从键盘键入三个数值并回车，就可以完成这条语句的执行。

例如，我们在键盘上键入

```
256.3, 10, 5↵(回车)
```

这时，我们就将 256.3 赋给了 X，10 赋给了 M，5 赋给了 N。也就是说当这条语句执行完后，实型变量 X 具有一实数值 256.3，整型变量 M, N 分别具有整数值 10 和 5。

例 3.2 有下面语句序列：

```
CHARACTER NAME * 10
```

```
READ *, NAME, AGE, ID
```

这时我们从键盘键入：

```
'CHANG—LI', 20.0, 931002 ↵
```

执行结果，字符变量 NAME 被赋值 CHANG—LI—，实变量 AGE 被赋值 20.0，整变量 ID 被赋值 931002。本例需要指出的是，字符变量 NAME 的长度为 10，而我们实际输入的字符串 CHANG—LI 的长度为 8。对于字符型数据赋值来说，应采取左对齐原则，NAME 中右边空余字符位置应用空格(空白符)填满。

如果我们使用终端键盘输入数据时，是以回车(↵)做为一次输入的结束标志，即每打一个“↵”，则代表一次输入已完毕。而我们又知道，组成输入的数据值都各有其一定的数据格式(如类型等)。而我们在表控输入时，我们并没有对输入中的数据值的组成进行描述。这时，输入中数据值格式的安排则全靠表控输入语句中的变量列表与输入中的数据值的对应关系来安排，即由输入变量列表来控制输入中的数据值排列来实现格式安排，这也是为什么称这种输入方式为表控输入的主要原因之一。显然，输入变量列表与输入中数

据值的对应关系应有严格的要求。下面我们将分别加以讨论。

(1) 输入由表示数据的值和值的分隔符组成

这里值可以是常数。值的分隔符可以是一个或多个连续的空格符；分隔符也可以是逗号，逗号的前后可以有一个或多个连续的空格符。

例 3.3 表控输入语句

```
READ *, M, N, X
```

下列的输入方式是正确的而且是等价的。

```
90, 20, 150.5 ✓
```

```
90  20  150.5 ✓
```

```
90, 20  150.5 ✓
```

例 3.4 表控输入语句

```
READ *, I, J, K
```

以下输入是正确的：

```
3 * 2 ✓
```

这里采用的是 $r * c$ 输入方式。其中， c 是常数， r 为无符号非零整数。相当于常数 c 连续出现 r 次。因此上例中相当于输入中是连续的 3 个 2；即 I 、 J 、 K 分别都被赋值为 2。

例 3.5 下列语句序列：

```
I=3
```

```
READ *, X, I, Y
```

输入为

```
5.5/ ✓
```

这时仅对 X 赋值 5.5 后， $READ$ 语句便终止执行。“/”为终止执行符，在“/”出现后还未被赋值的变量则保持有原来的值。例如， I 仍为 3，而 Y 则属于未定义状态，即 Y 中的值不确定。（一般来讲，系统默认其值为 0。）

(2) 输入中值的类型必须和变量列表中变量的类型对应一致

例 3.6 下列输入是正确的：

```
READ *, K, M, X, Y
```

输入：10, 20, 15.5, 35.6 ✓

例 3.7 下列输入与变量类型不匹配，是错误的。

```
READ *, I, X, S
```

输入：10.3, 15.5, 20 ✓

例 3.8 下列输入是正确的：

```
CHARACTER NAME * 10
```

```
COMPLEX COM1
```

```
READ *, NAME, COM1
```

输入：

```
'CHANG LI', (9.8, -2.3) ✓
```

执行结果 $NAME$ 中值为 $CHANG LI$ ，而 $COM1$ 中具有值 $9.8 - 2.3i$ 。

(3) 输入过程中无终止符

一条输入语句，若输入中没有出现终止符“/”，则必须等输入语句变量列表中所有变量都被赋值后，这条输入语句才能正常执行完毕。否则程序将在这里暂停执行。这时，一条输入语句便可读入多组输入数据，直到所有的变量都被赋值为止。

例 3.9 下列输入是正确的：

```
READ *, X, Y, Z
```

输入：

32.2 ✓

28.9 ✓

50.1 ✓

这时如果只输入两个项目，则程序暂停执行，直到输入第三个项目为止。

(4) 输入的数据个数多于变量个数。在语句执行时，若输入的数据个数多于变量个数，则数据中多余的数据值将被忽略。

例 3.10 有下面的输入：

```
READ *, I, J, K
```

输入：

10, 15, 20, 30, 500 ✓

执行后，10 赋给 I，15 赋给 J，20 赋给 K，而 30、500 无变量名与之对应，属多余数据值，从而被忽略。

(5) 每一条输入语句，必须对应一个新的数据输入

例 3.11 有下面的语句序列及输入：

```
READ *, I, J, K, X
```

```
READ *, Y, Z
```

输入：

10, 20, 30 ✓

15.5, 25.5, 11.6 ✓

32.5, 50.2 ✓

这里相当于有 3 组输入数据(有 3 个✓)。由上面讨论可知，第一条 READ 语句执行后，则将 10 赋给 I，20 赋给 J，30 赋给 K，15.5 赋给 X。这时，第二组数据输入中还剩下数据值 25.5 和 11.6，而第二条 READ 语句开始执行时，并不理会这两个数值，却是从第三组数据输入开始读入数据。即 Y 被赋值 32.5，Z 被赋值 50.2。

从以上讨论可以看到，FORTRAN 77 中表控输入是方便而灵活的。这也是一种简单而实用的一种数据输入手段。有关格式输入我们将在下一节中进行讨论。

二、PRINT 语句和 WRITE 语句

FORTRAN 77 中，PRINT 语句可实现一种最简单的输出过程，即表控输出。其形式为

```
PRINT *, iolist
```

其中：“*”前已说明是外部部件标识符的一种，它标识所用的部件是计算机系统规定的预连接部件。而在输出语句中，通常“*”是指终端显示器。

iolist 是输出列表，它由一个或多个用逗号分隔开的变量名(或者是数组名、数组元

素、字符子串名等)、常数、表达式组成。

同表控输入一样,表控输出无需程序员指定输出设备。输出设备由计算机系统指定,通常为终端显示器或打印机。本书约定为终端显示器。

例 3.12 PRINT *, 'UNIT IS', A, 'PRICES ARE', B

若 $A=151.00$, $B=96.8$, 执行后将输出一行

UNIT IS 151.000000 PRICES ARE 96.800000

例 3.13 PRINT *, 1. + INT(X * 2.0)

这个输出表为一表达式,若 $X=2.4$, 则执行结果输出为 5.0。

在表控输出中,输出数据的格式也是由计算机系统设定的,与程序员的安排无关。下面简要说明表控输出语句使用中应该注意的若干问题。

1. 数据值的输出格式

(1) 整型值的输出

如果输出值为整型时,即整型变量或表达式,则按整型输出方式安排数据值的格式,输出的每个数据值在输出时所占用的字段宽度都是统一的。其具体字段宽度(字段中的字符位数)均由计算机系统来设定,本书约定其宽度为 11 位,即包括符号位在内共有 11 个字符位置。数据的排放原则为右对齐放置,即数据值的最低位靠每个字段的最右一个字符位置放置,若不满 11 位,则左端填补空格。

例 3.14 PRINT *, I, J, K

若 $I=256$, $J=-180$, $K=881268$, 则输出结果为

256 -180 881268

(2) 实型值输出

如果输出是实型(包括双精度型)值,则输出时,通常按带小数点的普通十进制浮点数形式或按带有字母 E 阶码表示的十进制数形式输出。需要注意的是:计算机系统的不同对实型值的输出格式会产生影响。本书约定,在本书的例子中均采用普通带小数点的十进制定点数形式,即输出字段宽度为 15 位,其中小数点部分为 6 位。

例 3.15 PRINT *, X1, X2, X3

若 $X1=-29.0$, $X2=8935.667$, $X3=3015.5384423$, 则执行结果如下:

-29.000000 8935.667000 3015.538000

(3) 字符型值输出

如果输出是字符型值时,那么一对撇号内的字符原样照印。

例 3.16 PRINT *, 'NUMBER', X

若 $X=6890.889$, 则执行结果为

NUMBER 6890.889000

2. 每条 PRINT 语句的输出总是在新的一行开始

例 3.17 PRINT *, X1

PRINT *, X2

PRINT *, X3

其中, $X1=1057.776$, $X2=-30.20$, $X3=120.02$ 。

执行结果如下:

```

      1057.766000
      30.200000
      120.020000

```

3. 在 PRINT 语句中, 若无输出数据, 则表示输出一空行

例 3.18 PRINT *, A

```
PRINT *
```

```
PRINT *, B
```

若 A=20.5, B=320.1, 则执行结果如下:

```
      20.500000
```

```
      320.100000
```

其中, PRINT * 功能为输出一个空白行。

再看以下例举:

```

DOUBLE PRECISION X
COMPLEX Y
INTEGER Z
REAL U
DATA U,X,Y,Z/7,8.57,9.485,7.4/
PRINT *,U,X
PRINT *,Y,Z
END

```

其执行结果如下:

```

      7.000000      8.569999694824219
      (9.485000, 0.000000E+00)      7

```

从这个例子可以看到: 每条 PRINT 语句的输出总是在新的一行开始; 同时, 若在 DATA 语句中, 当 nlist 中的元素与 clist 相应的常数两者类型不一致时, 应按算术赋值语句的规定进行类型转换, 然后再赋初值。

而 PRINT * 与 WRITE(*, *) 则具有同样的功能。区别之处在于, WRITE(*, *) 后面紧跟输出列表, 没有分隔符“,”。我们可以把例 3.18 写成:

```

WRITE(*, *)A
WRITE(*, *)
WRITE(*, *)B

```

由以上讨论和例题可以看出, 利用 PRINT * 语句或 WRITE(*, *) 语句来实现 FORTRAN 77 程序的输出是非常方便的。但也可以看出, 由于输出格式是由计算机系统自己规定的, 这给应用程序的编制带来了不便性, 程序员无法按照自己的意愿来安排输出格式。特别是在大量数据输出的情况下, 不能有效地控制输出数据的排列格式。因此表控输出一般只适用于少量的、格式要求不严的情况。而对于大量而又较复杂的输出则应尽量采用格式输出形式。对于输入, FORTRAN 77 同样有丰富的格式输入形式。

三、格式输入/输出

FORTRAN 77 中格式输入/输出语句有下列形式：

```
READ(cilist) [iolist]
```

```
READ f [,iolist]
```

```
WRITE(cilist) [iolist]
```

```
PRINT f [,iolist]
```

其中：cilist 称之为控制信息表，其最简形式为(cilist)=(u, f)。u 称之为部件号，f 称之为格式标识符。

iolist 为输入列表与输出列表，其说明同表控输入/输出语句中对输入/输出列表的说明一致。

1. 部件号

部件号是用以指明输入/输出操作是在哪个部件上进行的。所谓部件，可以理解为外部设备以及这些设备与计算机的连接，是引用存放在设备上数据的一种手段。任何数据在外部设备上的存取操作，都必须通过部件来实现。如磁盘驱动器就可认为是一个部件，它不仅包括构成磁盘驱动器的硬件，也包括操作系统中对磁盘驱动器的管理及驱动软件。任何对磁盘上数据的存取操作，都必须通过磁盘驱动器部件(包括硬件及软件)来实现。

在计算机系统中，系统对所有的输入/输出部件进行了编号(正整数值)。程序员可通过在格式输入/输出语句中指明部件号来控制输入/输出操作在系统的哪个输入/输出部件上进行。例如：整数 6 作为部件号对应于打印机，那么 WRITE(6, f)X 将在打印机上输出 X 的值。至于每个部件的部件号具体是何值，则根据使用的计算机系统不同而异。因此，使用前应查阅有关资料。

这里要指出的是，部件号 u 可以为“*”号，此时说明输入/输出操作是在系统指定的部件上进行。

2. 格式标识符

格式标识符 f 标识一个格式。格式标识符必须是下列情况之一：

① 格式标识符所在的同一程序单位内的一个 FORMAT 语句的语句标号。

例如：

```
      READ(6, 100)A
100   FORMAT (F10.6)
      PRINT *, A
      END
```

当我们执行这个程序时，给变量 A 输入数据 73.4148，那么此数据按标号 100 的格式语句所指定的格式转换并输出 A 的值为 73.414800。

② 整型变量名，它被赋予一个与格式标识符在同一程序单位内 FORMAT 语句的语句标号。

例如：

```
      ASSIGN 100 TO N
      READ (6, N) A
```

```
100  FORMAT (F10.6)
      PRINT *, A
      END
```

该程序中的第一个语句为语句标号赋值语句(见 4.1 节),它把同一程序单位内 FORMAT 语句的标号 100 赋予 N, 这样这个程序就与上一个程序的功能完全一样了。

③ 字符数组名(见 7.3 节),那么该字符数组名应存放相应的格式说明,下面将详细介绍这个内容。此时输入输出将按字符数组中指明的格式进行转换。

④ 字符表达式(见 7.3 节),那么该字符表达式将直接指明转换格式。

例如:

```
      READ *, A
      WRITE(6, '(1X, F10.3)') A
      END
```

这个程序的功能是在 6 号部件上,按格式(1X, F10.3)输出 A 的值。如果我们输入 A 的值为 73.4148,则输出 A 的值为 73.415。

⑤ 星号(*),则表示按表控格式输入输出。表控格式前面已作了详细介绍。

3. 格式说明及 FORMAT 语句

格式说明一般形式为以一对圆括号括起来的编辑描述符。如:(F10.6, I5)便是一个格式说明,其中有两个编辑描述符 F10.6 和 I5。编辑描述符通常有下列各种:

数值编辑符	I, F, E, D, G
逻辑型值编辑符	L
字符型值编辑符	A
字符常数编辑符	H
撇号编辑符	"
位置编辑符	X
斜杠编辑符	/

其中, I, F, E, D, G, L, A 是可重复的编辑描述符,而后四种是不可重复的编辑描述符。可重复的编辑描述符前面可以用重复系数 r 说明描述符重复出现的次数, r 为非零无符号整常数。

下面介绍这些最常用的编辑符及其应用。

(1) I 编辑符

I 编辑符的功能是用于整型量的输入或输出。它的一般形式有以下两种:

IW

或

IW.m

IW 中的 I 代表 Integer(整数); W 表示字段宽度,是非零无符号整常数。例如, I7 表示该整型数应占的列数为 7 列。

在输入时, IW 指出从外部文件记录上读入一段最多含有 W 位数字的整型数据,并转换成整型的内部形式后存入输入数据对应的整型变量中。在输出时, IW 是将输出项中相应的整型值编辑成 W 个字符的外部形式输出在外部介质上。如果该整数的实际位数(包括

符号)不足 W 位, 则左面补以空格。如果多于 W 位, 则无法正确表示出该数, 于是便以 W 个“*”表示。

```
例 3.19  I=2468
          J=-456
          K=1234567
          WRITE(*, '(1X, I4, I6, I6)')I, J, K
          END
```

这个程序的输出结果为

```
2468  -456 * * * * *
```

其中, 格式标识符中前面写上 1X 的理由将在后面讲述; I 是 4 位对应 I4; J 是 4 位对应 I6, 向右看齐, 前面出现两个空格; 而 K 是 7 位对应后一个 I6, 而 I6 只提供 6 列位置, 无法容纳 7 位数字, 在这 6 列位置全部打印“*”, 是“字符宽度不够”的出错信息。再看例 3.20。

```
例 3.20  READ '(I3, I5, I7)', I, J, K
          WRITE(*, '(1X, I3, I5, I7)')I, J, K
          END
```

在这个程序执行中, 我们输入

```
120 -360 45
```

输出结果为

```
120 -360 45
```

IW.m 形式中 I 和 W 的含义同前; m 表示需要打印的数字的最少位数。

在输入时, IW.m 的功能与 IW 相同。在输出时, IW.m 与 IW 的具体处理规则不完全相同。例如: I9.6 表示字段宽度为 9, 其中至少应当有 6 位数字。m 位数字不包括负号。例如:

```
M=-2769
WRITE(*, '(1X, I9, I9.6)')M, M
END
```

则输出结果为:

```
-2769 -002769
```

很显然, 输出正整数时, 应满足条件 $W \geq m$; 输出负整数时, 应满足条件 $W \geq m+1$ 。所以, 应该对输出数的大小事先估计其位数; 如果是正整数其位数为 b , 则应使 $W \geq b$; 如果是负整数其位数为 b , 则应使 $W \geq b+1$ 。

(2) F 编辑符

F 编辑符的功能是用于实型量的输入或输出。它的一般形式为:

```
FW.d
```

其中, F 是 Float(浮点数)的缩写; W 为字段宽度, 为非零无符号整常数; d 为该数的小数位数, 为无符号整常数。

在输入时, FW.d 指出从外部记录上读入一个最多长为 W 位的字段, 再转换成实型的内部形式存入到输入数据对应的存储单元中。

```
例 3.21  READ '(F6.2, F10.6, F10.7, F9.3)', A, B, C, D, 当我们输入 5202,
```


125368, -81409, 79.34, 其结果是:

A=52.020000

B=1.253680E-01

C=-8.140900E-03

D=79.340000

由此例可以看出,在F编辑符下输入数据时,符号位占字宽中一位,十号在输入时省略。输入时数据可以不加小数点,这时小数点位置由d来决定。若输入时数据带小数点,则采用自带小数点优先原则。

在输出时,FW.d将对应的输出数据的值四舍五入,舍入到d位小数后,以W位长的字段向右对齐传送到外部介质上。例如,F9.3表示其数据输出时共占9列,其中有3位为小数部分。

例 3.22 X=92.04

Y=-1687.56

Z=3240.08

W=645.83

PRINT '(1X, F6.2, F9.3, F9.4, F7.2)', X, Y, Z, W

END

此程序输出结果为

□92.04-1687.5603240.0800□645.83

可以看出,在F编辑符下输出时,数值靠右印出,左边不足部分填以空格。数据值之间系统不留间隔,“+”号在输出时省略,“-”号和“.”各占字段宽度中的1个字符位置。在F编辑符中,我们应做到 $W > b + d + 2$,其中,b为数据的整数位数,d为小数位数。当实际数据值的小数位数大于d时,计算机则先对d后一位数码进行四舍五入处理,然后再将其输出。从例3.22的输出结果看Y与Z的值容易搞错,这是由于我们的F编辑符设定得不好造成的。如果我们把例3.22之PRINT语句改为

PRINT '(1X, F6.2, F10.3, F10.1, F7.2)', X, Y, Z, W

则输出结果为

□92.04□-1687.560□3240.0800□645.83

那么X, Y, Z, W的对应数值就看得很清楚了。

如果输出的内容多于W位,则无法正确表示出该数,于是便以W个“*”表示。

(3) E编辑符

E编辑符的功能是用于实型量的输入或输出。用E编辑符输出的实数是用指数形式表示的。它的一般形式为

EW.d

其中,E是Exponent(指数)的意思;W是整个字段宽度;d是数据的数值部分(即E前面的部分)中小数的位数。输出的字段中指数部分必占4列,其中“E”占1列,符号占1列,指数占2列。

在输入时,E编辑符与F编辑符作用相同,两者可互换。

在输出时,E编辑符输出的是以指数形式表示的实数。例如:

```

X=92.04
Y=-1687.56
PRINT '(1X, E12.4, E15.5)', X, Y
END

```

的输出结果为：

```

      .9204E+02      -1.16876E+04

```

对于指数形式的输出，一律以标准化的指数形式表示（即小数点前无非零整数，小数点后第一位为非零数字）。

W 的值应该大于或等于 $d+6$ 。因为小数点后数字为 d 位，指数部分占 4 列，小数点占 1 列，而习惯上两个数据之间留上一个空格。如果输出的数为负的，则 $W \geq d+7$ ，因为还要印出负号。

用 E 编辑符指定输出数据的格式，可以避免“大数印错，小数印丢”的情况，它能表示范围广泛的数。

(4) D 编辑符

D 编辑符的功能是用于双精度型数据的输出。它的一般形式为：

DW.d

其中，D 代表 Double Precision（双精度）；W 是整个字段宽度；d 是数据的小数位数。例如：

```

DOUBLE PRECISION D1, D2
D1=1.234567890123D+02
D2=-9876.543D-03
PRINT '(1X, D18.11, D16.7)', D1, D2
END

```

其输出结果为：

```

      1.2345678901D+03      -9.876543D+01

```

使用方法与 E 编辑符相仿，只是把字母“E”换成“D”。

(5) G 编辑符

上面介绍过，对于实数的输入和输出，可以用 F 编辑符，也可以用 E 编辑符。用 F 编辑符输出格式虽然直观，但有时会出现“大数印错，小数印丢”的情况；用 E 编辑符输出虽然保险，但又不大直观，尤其是计算金额时，不希望用指数形式表示。

FORTRAN 77 提供了一种“两全其美”的方法，即使用 G 编辑符。它可以根据输出的实数大小决定用 F 编辑符或 E 编辑符。当输出大数值或小数值自动按 E 编辑符，当输出的数不大不小时用 F 编辑符。其一般格式为

GW.d

怎样决定用 F 编辑符或用 E 编辑符呢？其规则如下：

如果输出变量 A 已赋予值，则

$10^d > |A| \geq 0.1$

用 F 编辑符，有效位为 d 位；

$|A| < 0.1$ 或 $|A| \geq 10^d$

用 E 编辑符。

例如：A=123456.7

B=123.456789E+06

```

C=-0.0005
PRINT '(IX, G14.7, G15.6, G15.5)', A, B, C
END

```

则输出结果为

```

123456.71234567E+09-0.50000E-03

```

A=123456.7, 用 G14.7; 由于 $d=7$, $|A| < 10^7$, 故用 F 编辑符输出。

而 B=123.456789E+06, 用 G15.6; 由于 $d=6$, $|B| > 10^6$, 故用 E 编辑符输出。

而 C=-0.0005, 用 G15.5; 由于 $|C| < 0.1$, 故用 E 编辑符输出。

如果输出的内容多于 W 位, 则无法正确表示出该数, 于是便以 W 个“*”表示。上例中如果 C 用 G10.5 格式输出, 我们看到前面的结果输出的内容为 11 列, 所以 G10.5 无法表示, 于是输出 10 个“*”。

G 编辑符虽然可以正确输出数值, 但用 F 编辑符输出时, 不能事先确定小数位数, 而有时人们常要求得到若干位小数, 而且要求每一行的数上下以小数点对齐, 而用 G 编辑符输出 F 格式是做不到这点的。

(6) X 编辑符

X 编辑符的一般形式为:

nX

其功能如下: 在输入时将跳过外部介质上的 n 个字符; 在输出时是输出 n 个空格。

为了避免相邻的两个数据紧连在一起, 可以用 X 编辑符在数据之间插入一些空格。

在本章前面的一些例题中, 在输出语句的格式说明中, 一般第一个描述为 1X, 1X 意指跳过一个空格。这是因为在输出时, 每一输出行的第一个字符位被系统用于行控制(也叫做走纸控制)。因此在输出的格式说明中, 应给系统空出这第一个字符位。否则, 第一位若有数据则会被系统“吃掉”, 从而影响输出结果的正确性。

例 3.23 入=12.3456

```

I=123
J=-4567
PRINT '(1X, I3, 2X, I5, 2X, F8.3)', I, J, A
END

```

输出结果为:

```

123-456712.346

```

X 编辑符多用于调整输出数据的排列, 使之易读。X 编辑符不需要与输入输出表中的元素对应。再看下例:

```

READ(*, '(I5, 3X, I7)') I, J
PRINT '(1X, I5, 2X, I7)', I, J
END

```

如果我们读入

```

123456789123456789

```

则输出结果为

```

12345-9123456

```

从此例可见：在输入时，X 编辑符起跳过若干个字符再读的作用。这对于在一个数据记录中有选择地读入数据非常有用。

例 3.24 有一数据记录格式如下：

	学号	姓名	A 课程	分数	B 课程	分数
字符数	5	10	10	5	10	5

若程序只对 A 课程的分数进行处理，那么只需将每个记录中的 A 课程分数读入，于是有

```
READ '(25X, F5.1, 15X)', A
```

该语句实现了将分数数据项读入变量 A 中。在格式说明中，25X 为跳过 25 个字符，即在读入时跳过学号、姓名、A 课程这 3 个数据项（总字符数为 25），F5.1 则恰好对应于分数数据项，这样便完成了读入分数数据项的功能。

(7) A 编辑符

A 编辑符的功能是用于字符型数据的格式输入或输出。它的一般形式有两种：

AW

或

A

其中，A 是字符型的编辑符；W 指出字段宽度。

当使用 AW，在输入时，如果 W 大于或等于字符变量的长度 $L(W \geq L)$ ，则先从外部字段读入 W 个字符，从中取出最右边的 L 个字符，赋给相应的字符变量。若 $W < L$ ，则输入的 W 个字符后面加上 $(L - W)$ 个空格符；然后，赋给相应的字符变量。在输入时，如果外部字段的实际长度小于 W 时，则用空格符加到外部字段的右端，直到它的长度等于 W。

例 3.25 CHARACTER A * 5

```
READ '(A8)', A
WRITE (*, *) A
END
```

如果我们读入 abcdefgh 8 个字符，那么取最右边的 5 个字符 defgh 赋给字符变量 A。如果我们读入 abcdefg 7 个字符，那么就要在右端加上一个空格符，再把最右边的 5 个字符 defg 赋给字符变量 A。如果把上例改为例 3.26，则有

例 3.26 CHARACTER A * 12

```
READ '(A8)', A
WRITE (*, *) A
END
```

如果我们读入 8 个字符 abcdefgh，那么赋给变量 A 的是 abcdefgh。

在输出时，如果 $W \geq L$ ，则输出字符变量的值（长度为 L），在其左面有 $(W - L)$ 个空格。如果 $W < L$ ，则输出字符变量中左边 W 个字符。

例 3.27 CHARACTER A * 8

```
A = '12345678'
WRITE (*, '(1X, A10)') A
END
```

则输出结果为

```
12345678
```

如果把上面的 WRITE 语句改为

```
WRITE(*, '(1X, A6)')A
```

则输出结果为

```
123456
```

当仅使用 A, 则不指出字段宽度, 系统会按字符变量的长度来截取各字符常数。

例 3.28 CHARACTER A * 4, B * 3

```
READ(*, '(A, A)')A, B
```

```
WRITE(*, '(1X, A, 4X, A)')A, B
```

```
END
```

当我们读入 abcdefghijk, 则输出为

```
abcd1234efg
```

字符型数据输入与输出的示例分别列于表 3.1 和表 3.2 中。

表 3.1 字符型数据输入示例

格式	外部字段	字符变量长度	内部形式
A	abcde	6	'abcde '
A	abcdef	6	'abcdef'
A6	abcdefg	5	'bdefg'
A6	abcdefg	3	'defg'
A4	ab	4	'ab '
A6	abc, def	4	'c, de'
A3	'abc'	3	'"ab'
A4	abcd	6	'abcd '

表 3.2 字符型数据输出示例

格式	内部字段	外部字段
A	'abcdefg '	abcdefg
A7	'abcdefg '	abcdefg
A7	'abcd '	1234abcd
A7	'abcde '	1234abcde
A4	'abcdefg '	abcd
A3	'abcdefg '	abc
A2	' '	"
A2	'a" '	a'

在表 3.2 中, 输出字符变量的长度就是表中内部字段的长度。

读者应当注意: 在输入时, 外部记录内的字符型数据同其它类型数据或另一字符型数

据之间是用字符个数来分隔的,此时不能用逗号来表示一个字符型数据的结束;否则,在输入该字符型数据值的长度小于字符变量的长度时,系统将认为字符型数据后面的“分隔号”逗号,也是字符型数据中的一个字符。

(8) H 编辑符

H 编辑符的功能是用来输出字符常数。其形式为:

$$nHh_1h_2\cdots h_n$$

其中: n 为要输出的字符个数,即字符串宽度;

$h_1h_2\cdots h_n$ 为要输出的具体字符。

同 X 编辑符一样, H 编辑符没有变量与之对应。

在实际应用中,有时除了希望打印出数值外,还需要插入一些文字说明,使输出结果更加易读。因此, H 编辑符被大量应用在输出时的格式安排中。

在使用 H 编辑符时,应特别注意: H 前的 n 的值必须与 H 后的字符数($h_1h_2\cdots h_n$)相等;否则, FORTRAN 77 将认为出错。

例 3.29 $A=49.5$

```
PRINT '(1X, 9HTHIS IS A, 4X, 2HA=, F10.6)', A
```

则输出结果为

```
THIS IS A A= 49.500000
```

H 编辑符不许用于输入。

(9) 撇号编辑符

撇号编辑符与 H 编辑符功能相同,但比 H 编辑符使用更方便些,不必计算出需输出的字符串位数。把要输出的字符串直接用一对撇号括起来就行。其形式为

$$'h_1\cdots h_n'$$

其中, $h_1h_2\cdots h_n$ 为要输出的具体字符,允许为计算机处理系统所允许的任一字符。若要输出一个撇号,则必须用 2 个连续的撇号来代替 1 个撇号,输出时仅作 1 个字符计数。

与 H 编辑符相同,撇号编辑符不需要与输出表元素对应。

如果把例 3.29 的 PRINT 语句改为

```
PRINT '(1X, "THIS IS A", 4X, "A=", F10.6)', A
```

其输出结果与例 3.29 完全一样。

在这里,特别要注意的是:为什么格式说明里面的撇号编辑符用的是双撇号呢?上面已提到,撇号编辑符是为了输出字符用的,由于撇号编辑符现在是作为格式说明的一部分,所以是字符常数里面的字符串,按照字符常数中撇号的书写规则,应该用中间不嵌入空白符的连续两个撇号来表示。如果写成

```
PRINT *, 'THIS IS A A= 49.500000'
```

```
END
```

则输出结果是完全一样的。

撇号编辑符不许用于输入。

(10) L 编辑符

L 编辑符的功能是用于逻辑型数据的输出。其形式为:

LW

其中, L 是 Logical 的首字母; W 是字段宽度。

例 3.30 LOGICAL L1, L2

```
L1=.TRUE.
L2=.FALSE.
WRITE(*, '(1X, L4, L5, L2)')L1, L2, .FALSE.
END
```

则输出结果为:

```
    T    F    F
```

对值为“真”时, 在输出时打印一个字母 T; 为“假”时, 则以一个字母 F 表示。T 和 F 在字段范围内的右端。可以看出, 逻辑常量(.TRUE. 或 .FALSE.) 可以直接输出。

(11) 斜线编辑符

斜线编辑符的形式为:

```
/
```

其功能是指出一个记录的传输结束。在输入时, 结束当前记录的传输, 准备对下一个记录传输。当两个斜线编辑符之间没有其它编辑符时, 则表示跳过外部的一个完整的记录。在输出时, 斜线编辑符表示结束现记录的输出, 且准备输出一个新记录, 使其后输出的信息出现在外部的一个新记录上(即新的一行)。当两个斜线编辑符之间没有其它编辑符时, 则表示产生一个空记录。例如, 我们把例 3.30 的 WRITE 语句改为

```
WRITE(*, '(1X, L4/1X, L5, L2)')L1, L2, .FALSE.
```

则输出 L1 的值后遇斜线, 本行结束, 而下面又有 1X, 表示换到下行再输出 L2 之值和 .FALSE., 于是输出结果为:

```
    T
    F    F
```

如果我们将上面的 WRITE 语句改为

```
WRITE(*, '(1X, L4//1X, L5, L2)')L1, L2, .FALSE.
```

那么输出结果为

```
    T

    F    F
```

在输出完 T 后就空一行, 再输出后面的记录。

(12) 重复系数的用法

前而我们介绍过的 11 种编辑符中有 7 种是可重复的编辑符, 即 I、F、E、D、G、L 和 A。这些编辑符可以在其前加一个重复系数, 以代表该编辑符重复出现若干次。它们的一般形式为:

```
rIW, rIW.m, rFW.d, rEW.d, rDW.d, rGW.d, rAW, rA 和 rLW
```

其中, r 为重复系数, 是无符号整数。

例如:

```
WRITE(*, '(1X, 3A)')A1, B2, C3
```

意指 A1、B2、C3 均按 A 编辑符的格式输出。

可以用括号将若干个编辑符括起来组成一个编辑符组。例如(1X, 3A)就是一个编辑符组。在编辑符组前面可加上重复系数,表示此组重复出现的次数。

例如:

```
WRITE(*, '(2(1X, 3A))') A1, B2, C3, D4, E5, F6
```

意指这六个变量均按 A 编辑符的格式输出,在输出 A1、B2、C3 后出现一个空格,再输出 D4、E5 和 F6。

若再对若干个编辑符组用括号括起来,又组成一个新的编辑符组,它的前面,还可加重复系数。

(13) 格式说明语句 FORMAT

在前面曾提到,格式输入/输出中,格式标识符 f 可以为一种格式语句的语句标号。这个格式语句就是 FORMAT 语句。其形式为

```
FORMAT Fs
```

其中, Fs 就是前面提过的格式说明。此语句必须带标号。

例 3.31 WRITE(*, 100)I, J, K

```
100 FORMAT(1X, 3I5)
```

若 I=10, J=-20, K=30, 则输出结果为

```
10-2030
```

我们再考察一下当 I, J, K 的值同上例时 PRINT '(1X, 3I5)', I, J, K 的输出。我们发现输出结果是一样的。

由此可知,格式标识符为格式说明语句(FORMAT)的标号时,相当于把格式说明放在了相应的格式说明语句中。

在 FORTRAN 77 中涉及数据输出的有两条语句,即 PRINT 语句和 WRITE 语句。从功能上看这两条语句都能完成同一格式要求的数据输出操作。这一点从例 3.31 中可以看出。它们的区别在于 PRINT 语句的输出格式,在 PRINT 语句中给出;而 WRITE 语句的输出格式,可以在 WRITE 语句中给出,也可借助于 FORMAT 语句给出。因此在使用上,一般一次输出的数据量较少,且这次输出格式在程序中重复出现的次数较少的情况下,则可采用 PRINT 语句。而对于输出量较大且该种输出格式在程序中出现次数较多时,为使程序简洁以及调试的方便,采用 WRITE 语句较为合适。

但是,应当注意的是,PRINT 语句和 WRITE 语句都能完成对一定格式要求的数据输出操作。

例 3.32 下列输出语句的结果格式是一样的。

① PRINT '(1X, F6.1, 3X, F6.1)', X, Y

② WRITE(*, 100)X, Y

```
100 FORMAT(1X, F6.1, 3X, F6.1)
```

四、应用程序举例

通过本章的学习,我们就可以编制一些较简单的应用程序。

例 3.33 用 FORTRAN 77 语言编写出实现如下功能的程序:两个并联电阻 R_1 和 R_2 的总电阻 R 由公式

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$

计算。其中, R_1 和 R_2 由终端键盘输入。计算后输出 R_1 、 R_2 和 R 的值。

编制程序如下:

```

      READ(*, 100)R1, R2
100  FORMAT(2F5.1)
      R=R1 * R2/(R1+R2)
      WRITE(*, 200)R1, R2, R
200  FORMAT(1X, 'R1=', F5.1/1X, 'R2=', F5.1/1X, 'R=', F6.2)
      STOP
      END

```

在程序执行时我们输入

30.4, 40.5

则输出结果为:

```

R1=30.4
R2=40.5
R=17.37
Stop-Program terminated

```

例 3.34 读入五个实数, 试编制程序输出其中绝对值最大的和最小的两个数。

```

      READ(*, 10)A, B, C, D, E
10  FORMAT(5F6.1)
      BIG=MAX(ABS(A), ABS(B), ABS(C), ABS(D), ABS(E))
      SMALL=MIN(ABS(A), ABS(B), ABS(C), ABS(D), ABS(E))
      WRITE(*, 20)BIG, SMALL
20  FORMAT(1X, 'MAX=', F10.2/1X, 'MIN=', F10.2)
      STOP
      END

```

我们输入数据

123456-78901 1234 34.7 98.9

输出结果为

```

MAX=12345.60
MIN=34.70
STOP-Program terminated

```

习 题 三

1. 在给变量赋值时, 使用 READ 语句比用赋值语句有什么好处?
2. 对于下面的表控输入语句:

```
READ *, X, I, Y
```

若输入为:

- (1) $\square\square 2048.56, \square\square 120, 804.05$ ✓
 (2) $\square\square 3096.84 \square\square 220 \square\square 44.524$ ✓
 (3) $\square\square 36 \square\square 44.05 \square\square 480 \square\square 301.02$ ✓

执行结果 X、I、Y 的值是多少？

3. 设 $I=30, J=5\ 320, X=-2.5, Y=49.103\ 6, Z=43\ 202.67$, 下列输出语句的执行结果是什么？

- (1) PRINT *, I, J
 (2) PRINT *, X, Y, Z
 (3) PRINT *, I, Y
 (4) PRINT '(1X, 2F10.4, I5)', X, Y, J
 (5) PRINT '(1X, I5, F10.4, I5)', I, X, J
 (6) PRINT *, 'X=', X, 'I=', I

4. 编写一程序打印下列表格：

```
STUDNO: 10101
NAME:  CHANG LI
AGE: 24
AVERAGE: 94.20
```

5. 编写一个程序，完成下列操作：

(1) 读入 A、B、C 的值，计算并打印它们的平均值，两组数据如下：

A	B	C
5.6	8.4	9.6
100.1	256.03	320.00

(2) 计算并打印：

$$\frac{\sin(x+y)}{\cos(x^2+y^2)} \quad (x=4.5^\circ, y=8.5^\circ)$$

6. 有一并联电路如图 3.1, $R_1=20\ \Omega, R_2=30\ \Omega, R_3=40\ \Omega, U=220\ \text{V}$, 求电流 I , 请编程序求之。

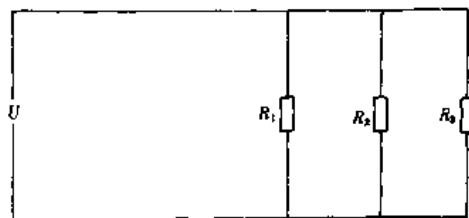


图 3.1

7. 编程计算偿清贷款所需月数，计算公式为

$$M = \frac{\lg P - \lg (P - D \cdot R)}{\lg(1 + R)}$$

其中： D 为贷款数；

P 为每月偿还数；

R 为月利率。

D 、 P 、 R 由键盘输入。

8. 有一输入部件上的记录为

└└4256735.912└└526└└360.4381

那么执行下列输入语句后，各变量中的内容是什么？

(1) READ '(I4, I3, F5.2, I1, I5, F10.4)', I, J, X, K, L, W

(2) READ '(3I3, F4.3, 2I5, F4.3, I1)', M, N, K, F, J, I, H, L

9. 分别写出三个程序，以进行下列计算：

(1) $5^3 - 5^2$ ；

(2) 圆半径 $r=8$ ，求圆面积；

(3) $4 \sin\left(\frac{\pi}{4}\right)$ 。

10. 请写出一个把华氏温度转化为摄氏温度的程序。转化公式为： $C = \left(\frac{5}{9}\right)(F - 32)$ 。

其中， F 是华氏温度值。 C 是摄氏温度值。

11. 自由落体位移公式为： $s = \frac{1}{2}gt^2 + v_0t + s_0$ 。其中， s_0 为初始位移， v_0 为初始速度， g 为重力加速度， t 为经历时间。编写一程序根据时间值求位移 s 。设 $s_0=1$ m， $v_0=2.5$ m/s， $g=9.81$ m/s²。(用输入语句读入 t 值)。

12. 人造卫星轨道近似一圆周。卫星在轨道上的运行速度 v_c 的计算公式为

$$v_c = \frac{7\,900 \sqrt{R}}{\sqrt{R+h}}$$

其中， R 为地球半径， $R \approx 6.371\,54 \times 10^6$ m。 h 为距地球表面高度。在某一高度上，卫星脱离轨道的速度为 $v_e = v_c \cdot \sqrt{2}$ 。编写一程序，求在给定的高度 h 上。计算卫星的运行速度和脱离轨道的速度(单位用 km/h)以及环绕地球一周所需的时间。

4 分支结构

FORTRAN 程序一般是按语句的书写顺序逐一执行的。但在针对实际问题编制程序时，往往根据具体情况要做一些判断处理。这就要在程序中加上控制语句来实现这些判断的完成。本章将对 FORTRAN 77 中条件判断及转移类语句进行讨论。

4.1 GO TO 语句

GO TO 语句有三种：无条件 GO TO 语句，计算 GO TO 语句，赋值 GO TO 语句。在学习 GO TO 语句之前，我们需要指出的是，按照结构化程序设计的原则，应尽量减少 GO TO 语句的使用，滥用 GO TO 语句会使程序结构混乱甚至引起错误。但也需指出，GO TO 语句，特别是无条件 GO TO 语句，使用非常方便，只要使用得当，便可简化程序，提高程序效率。同时，本书考虑到，早期 FORTRAN 语言中计算 GO TO 语句和赋值 GO TO 语句较常用，而且本书使用的 FORTRAN 77 5.10 版的编译系统中也保留这两条 GO TO 语句，因此，在本节也对计算 GO TO 语句和赋值 GO TO 语句给予简单介绍。

一、无条件 GO TO 语句

无条件 GO TO 语句的形式为

GO TO S

其中，S 是与该无条件 GO TO 语句在同一程序单位中的可执行语句的语句标号。

GO TO 语句用来改变程序的执行顺序。当程序执行到 GO TO 语句时，无条件地转去执行语句标号 S 所指定的语句，并从那条语句开始向下运行。

例 4.1 求平均数，程序如下：

```
S=0.0  
N=0  
10 READ '(F6.2)',X
```

```

S=S+X
N=N+1
PRINT *,S/N
GOTO 10
END

```

这个程序比较简单，从键盘上按 F 6.2 格式描述读入 X，由 $S=S+X$ 对输入的数据值进行累加，由 $N=N+1$ 累计数据个数，用 PRINT 语句将平均数 S/N 输出。每输出一个平均值，由 GO TO 语句又返回到 READ 语句，读入下一个数据。如此周而复始地反复执行。当然，细心的读者会发现，这个程序一投入运行，便无法正常结束（即无法执行到 END 语句）。这时若想终止程序的执行，一般可在键盘上同时按下 CTRL 键和 C 键（即 **CTRL** + **C**）来终止程序。

使用 GO TO 语句要注意的是，GO TO 后的语句标号 S，必须是与该 GO TO 语句在同一程序单位中的执行语句的标号。

二、计算 GO TO 语句

计算 GO TO 语句的形式为

GO TO(S1, S2, ...) , I

其中：Si 是与该计算 GO TO 语句在同一程序单位中的可执行语句的语句标号。同一个语句标号可以在同一个计算 GO TO 语句中出现多次；

I 为整型表达式。

该条语句在执行中，首先计算整型表达式 I 的值，然后根据这个值，进行控制转移。下一步执行语句标号表中第 I 个标号标识的语句，规定 $1 \leq I \leq n$ 。其中，n 是语句标号表中语句标号的个数，若 $I < 1$ 或 $I > n$ ，则执行下一个语句。

计算 GO TO 语句相当于具有分线开关的功能，根据 I 值的不同，来闭合不同的开关，接通不同的程序执行线路。需要注意的是，当计算 GO TO 语句执行之前，整型表达式中的变量必须预先被赋值。

例 4.2 写程序完成下面函数的计算。

$$y = \begin{cases} 2x^2 + 3x & 1 \leq x < 2 \\ x^2 - 3x + 2 & 2 \leq x < 3 \\ x + 5 & 3 \leq x < 4 \\ x & 4 \leq x < 5 \end{cases}$$

程序如下：

```

10  READ *,X
    I=INT(X)
    GOTO(20,30,40,50), I
    GOTO 10
20  Y=(2*X+3)*X
    GOTO 60
30  Y=(X-3)*X+2

```

```

        GOTO 60
40      Y=X+5
        GOTO 60
50      Y=X
60      PRINT *,Y
        END

```

该程序首先读入 x ，对 x 取整后赋给 I ，当 $1 \leq x < 2$ 时， I 值为 1，由计算 GO TO 语句转向语句标号为 20 的语句，求出 y 值后转向 PRINT 语句输出 y 值；当 $2 \leq x < 3$ 时， I 为 2，转向语句标号为 30 的语句，计算 y 值后打印，余类推。当 $x < 1$ 或 $x > 4$ 时，执行计算 GO TO 语句的下一条语句，即 GO TO 10，重新读入一个 x 值。

三、赋值 GO TO 语句和语句标号赋值 (ASSIGN) 语句

(1) 赋值 GO TO 语句

赋值 GO TO 语句的形式为

GO TO I [[,] (S [, S] ...)]

其中： I 是整型变量名；

S 是与该赋值 GO TO 语句在同一程序单位中的一个语句标号。同一个语句标号可以在同一个赋值 GO TO 语句中出现多次。

(2) 语句标号赋值 (ASSIGN) 语句

语句标号赋值语句的形式为

ASSIGN S TO I

其中： S 为语句标号；

I 为整型变量。

ASSIGN 语句和赋值 GO TO 语句必须在同一程序单位内配合使用。两条语句中的整型变量必须为同一变量名，而且这个变量名若被这两条语句使用后，不得再以其它方式被引用。注意，这里 I 只能为变量名而不得为表达式，这是与计算 GO TO 语句所不同的。

ASSIGN 语句和赋值 GO TO 语句的执行过程是这样的。由 ASSIGN 语句将一语句标号 S 赋给 I 后，在 GO TO I [[,] (S [, S] ...)] 语句中，根据 I 的值，选择与 I 的值相等的语句标号 S 值，下一步执行由哪个语句标号标识的语句。

例 4.3 计算当 $x=3.2, 5.6, 8.7$ 时，求： $y=\sqrt{x^2+1}-3$ 的值。

```

        X=3.2
        ASSIGN 10 TO I
60      Y=SQRT(X**2+1)-3
        PRINT *,Y
40      GOTO I (10,20,30)
10      X=5.6
        ASSIGN 20 TO I
        GOTO 60
20      X=8.7

```

```

        ASSIGN 30 TO I
        GOTO 60
30    STOP
    END

```

该程序使用了三个 ASSIGN 语句和一个赋值 GO TO 语句。这个程序简单，请读者自己分析一下程序的执行过程。这里要提醒读者的是，赋值 GO TO 语句中，整型变量 I 的值，必须用 ASSIGN 语句来对其赋值，而不能用赋值语句来赋值。如 ASSIGN 30 TO I。I 中 30 为一语句标号，而用 I=30，则 I 中 30 为一整数值，这两者是不同的。

4.2 算术 IF 语句与逻辑 IF 语句

一、算术 IF 语句

算术 IF 语句的形式为

IF(e) S1, S2, S3

其中：e 为算术表达式；

S1, S2, S3 为同一程序单位中的可执行语句的语句标号。

算术 IF 语句在执行时，首先计算算术表达式 e 的值，根据 e 的值作如下操作：

若 $e < 0$ 时，转向 S1 所标识的语句执行；

若 $e = 0$ 时，转向 S2 所标识的语句执行；

若 $e > 0$ 时，转向 S3 所标识的语句执行。

图 4.1 描述了算术 IF 语句的功能，从图中可以看到，算术 IF 语句有一个入口，三个出口。

当然，算术 IF 语句的三个出口可以相互合并，即 S1, S2, S3 这三个语句标号可以不同也可以相同。如果 $S1=S2=S3$ ，则相当于无条件 GO TO 语句。当然这是无意义的。若 $S1=S2$ ，则相当于只有两个出口，即当 $e \leq 0$ 时和 $e > 0$ 时两种情况的判定并转移。

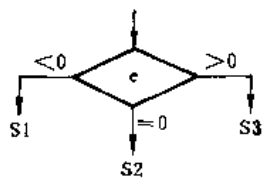


图 4.1

例 4.4 计算下面函数：

$$f = \begin{cases} -5.0 & x < 0.0 \\ 0 & x = 0.0 \\ 5.0 & x > 0.0 \end{cases}$$

程序如下：

```

        READ *,X
        IF (X) 10,20,30
10    F=-5.0
        GOTO 40
20    F=0.0
        GOTO 40

```

```

30  F=5.0
40  PRINT *, F
    END

```

例 4.5 求一元二次方程的根(相同实根, 不同实根, 共轭复根)。
在求解一元二次方程时, 有如下判断:

$$B^2 - 4AC > 0 \quad x = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

$$B^2 - 4AC = 0 \quad x = \frac{-B}{2A}$$

$$B^2 - 4AC < 0 \quad x = \frac{-B \pm i \sqrt{-(B^2 - 4AC)}}{2A}$$

程序中, 令 $D = B^2 - 4AC$, 程序框图如图 4.2。

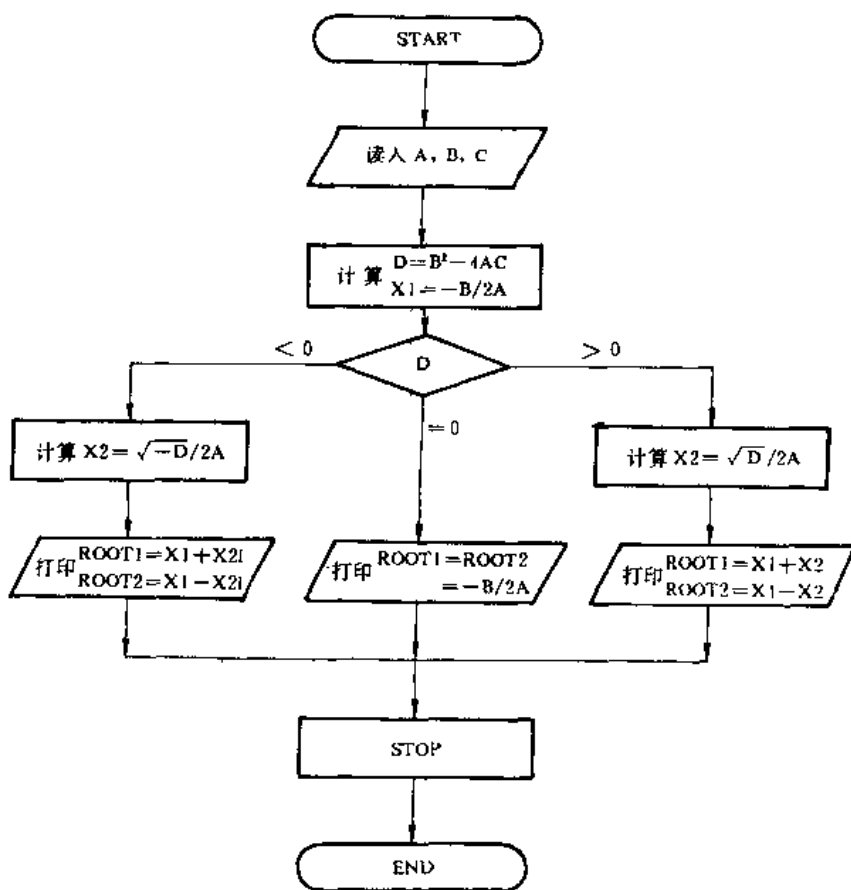


图 4.2

程序如下:

```

READ '(3F6.2)', A, B, C
X1 = -B / (2 * A)
D = B * B - 4 * A * C
IF (D) 10, 20, 30

```



```

10  X2=SQRT(-D)/(2*A)
    PRINT '(1X,6HROOT1=,F6.2,1H+,F6.2,1H1)',X1,X2
    PRINT '(1X,6HROOT2=,F6.2,1H-,F6.2,1H1)',X1,X2
    GOTO 100
20  PRINT '(1X,12HROOT1=ROOT2=,F6.2)',X1
    GOTO 100
30  X2=SQRT(D)/(2*A)
    PRINT '(1X,6HROOT1=,F6.2)',X1+X2
    PRINT '(1X,6HROOT2=,F6.2)',X1-X2
100  STOP
    END

```

二、逻辑 IF 语句

逻辑 IF 语句的形式为

IF (e) S

其中：e 为逻辑表达式；

S 为除 DO 语句、块 IF 语句、ELSEIF 语句、ELSE 语句、ENDIF 语句、END 语句或另一个逻辑 IF 语句之外的任何可执行语句。

在执行逻辑 IF 语句时，首先判断逻辑表达式的值。若 e 为真(e=.TRUE.)，则执行 S 语句；若 e 为假(e=.FALSE.)，则不执行 S 语句，而直接执行逻辑 IF 语句的下一条语句。图 4.3 描述了逻辑 IF 语句的功能。

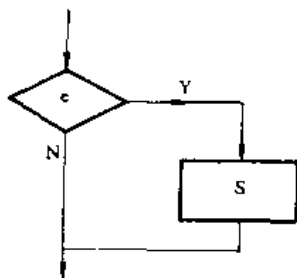


图 4.3

例 4.6 用近似公式

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{N!}$$

求自然对数底 e 的值，其中 N 的值由键盘输入，程序如下：（图 4.4 为框图）

```

      E=1.0
      P=1.0
      M=1
      READ *,N
10    P=P*M

```

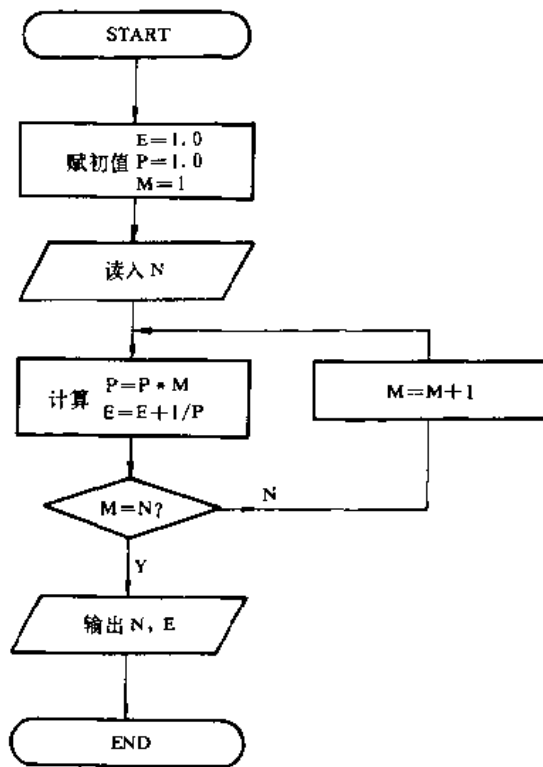


图 4.4

```

      E=E+1.0/P
      IF (M.EQ.N) GOTO 20
      M=M+1
      GOTO 10
20    PRINT *,E,N
      STOP
      END

```

程序中 $P=P * M$ 用来计算阶乘 ($M!$)， $E=E+1/P$ 用来完成对 $M!$ 的倒数的累加。当 $M=N$ ，即做够 N 次后，就去执行输出。

例 4.7 利用牛顿迭代法求三次代数方程：

$$2x^3 - 4x^2 + 3x - 6 = 0$$

在 $x=2.5$ 附近的实根，直到满足

$$|x_{n+1} - x_n| \leq 10^{-5}$$

牛顿迭代公式为

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

牛顿迭代公式的几何意义如图 4.5 所示。

$f'(x_1)$ 是 $f(x)$ 曲线在 x_1 点处切线的斜率。

$$f'(x_1) = \frac{f(x_1)}{x_1 - x_2}$$

于是可得出

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

x_2 为切线与 x 轴的交点。求出 x_2 后，再找出 $f'(x_2)$ ，对应该斜率 $f'(x_2)$ 的切线 ($f(x)$ 曲线在 $(x_2, f(x_2))$ 处的切线) 在 x 轴的交点为 x_3 ，如此反复迭代，逼近 x 的真实根。当前后两次求出的 x 值之差 $\leq \varepsilon$ 时，就认为找到了近似根。

$$f(x) = 2x^3 - 4x^2 + 3x - 6 = x(x(2x - 4) + 3) - 6$$

$$f'(x) = 6x^2 - 8x + 3 = x(6x - 8) + 3$$

令 $x_1=3, \varepsilon=10^{-5}$

程序如下：

```

      PRINT '(1X, 4X, 1HN, 10X, 2HX1, 10X, 1HX)'
      X=3.0
      N=1
10    X1=X
      F=X1*(X1*(2.0*X1-4.0)+3.0)-6.0
      F1=X1*(6.0*X1-8.0)+3.0
      X=X1-F/F1
      WRITE(*,20)N,X1,X

```

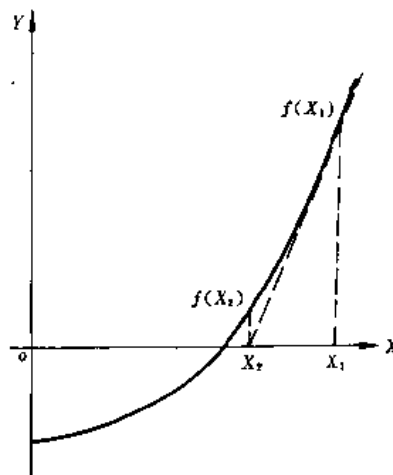


图 4.5

```

      IF (ABS(X-X1).LE.1.0E-5) STOP
      N=N+1
      GOTO 10
20   FORMAT(1X,I5,2F15.5)
      END

```

在程序中, N 为迭代次数。 x_1 为迭代开始时 x 的值(即 x_1), x 为迭代后求出 x 的新值(即 x_2)。IF 语句进行 $|x_2 - x_1| \leq \varepsilon = 10^{-5}$ 的判断。若条件为真, 则结束; 条件为假, 则 N 加 1, 表示再做一次迭代(GO TO 10)。这时, 将刚求出的 x 值(x_2)做为迭代开始的 x 值($x_1 = x$), 再求出一个新的 x 。直到满足 $|x_{n+1} - x_n| \leq \varepsilon$ 为止。

下面是运行结果:

n	x_1	x
1	3.000 00	2.363 64
2	2.363 64	2.070 99
3	2.070 99	2.003 43
4	2.003 43	2.000 01
5	2.000 01	2.000 00

Stop - Program terminated.

4.3 块 IF 结构

结构化程序设计方法的一个基本出发点是使编制出来的程序模块化。即从整体问题出发, 然后将整个问题划分成几个独立的逻辑部分, 对每一个独立的逻辑部分编制成一个程序模块。在模块中涉及判定的逻辑结构, FORTRAN 77 5.10 主要是用块 IF 结构来实现结构化。块 IF 结构在结构化程序设计中占有极为重要的地位, 块 IF 结构的正确使用, 可以使程序结构清晰、合理、易于阅读、便于维护。因此, 通过下面的学习, 我们必须掌握好块 IF 结构, 认识其重要性。

一、基本块 IF 结构

基本块 IF 结构的形式如下:

```

      IF (e) THEN
          BLOCK1
      ENDIF

```

其中: e 为逻辑表达式;

BLOCK1 为一语句序列, 即为一或多条语句;

ENDIF 标识块 IF 结构的结束。

图 4.6 描述了基本块 IF 结构的执行情况。若 e 为真($e = .TRUE.$), 则首先执行 BLOCK1 中的语句, 然后执行 ENDIF 的下一条语句; 若 e 为假($e = .FALSE.$), 则不执行 BLOCK1 中的语句, 而直接执行 ENDIF 的下一条语句。

例 4.8 求阶乘 $P = N!$ ($N \geq 2$)。

程序框图如图 4.7。程序如下:

```

      INTEGER P
      READ *, N
      IF (N. GE. 2) THEN
        P=1
        M=N
10    P=P * N
        N=N-1
        IF (N. GE. 2) GOTO 10
      ENDIF
      PRINT *, M, P
      STOP
      END

```

在程序中, IF 语句(IF(e)THEN)到 ENDIF 语句之间就是 BLOCK1, 程序执行时, 若 (N. GE. 2) 为真, 则做 BLOCK1(求 N 的阶乘); 若 (N. GE. 2) 为假时, 则直接打印 M, P。

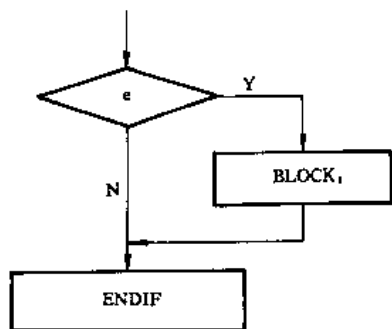


图 4.6

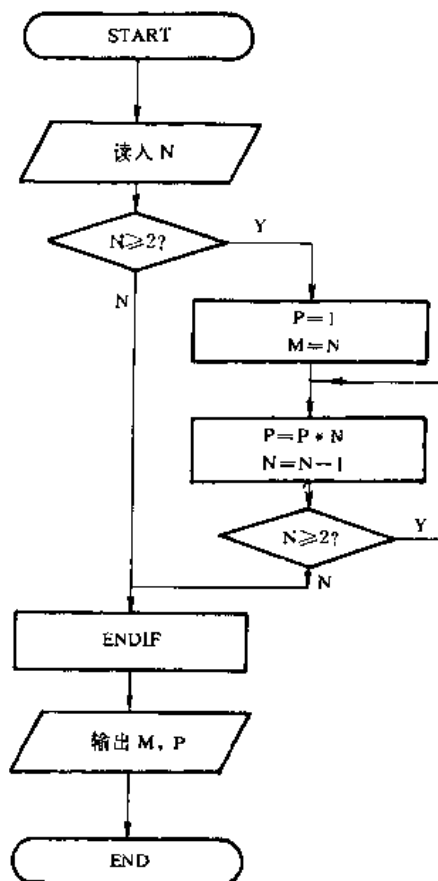


图 4.7

二、ELSE 语句

在块 IF 结构中, 当表达式 e 的值为假时, 我们需要做另一部分的程序处理工作, 这时就需要有 ELSE 语句。ELSE 语句的形式为

```
ELSE
```

块 IF 结构包含有 ELSE 语句时, 呈如下形式:

```
IF (e) THEN
```

```
  BLOCK1
```

```
ELSE
```

```
  BLOCK2
```

```
ENDIF
```

其中, BLOCK2 也是一个语句序列(即一条或多条语句)。

这时, 块 IF 结构的执行过程是: 若 e 为真, 则执行 BLOCK1 中的语句, 然后执行 EN-

ENDIF 后的语句；若 e 为假，则执行 BLOCK2 中的语句，然后执行 ENDIF 后的语句。图 4.8 描述了这种情况。

例 4.9 实现下面函数的计算：

$$F = \begin{cases} x - 5.0 & x \geq 5.0 \\ 5.0 - x & x < 5.0 \end{cases}$$

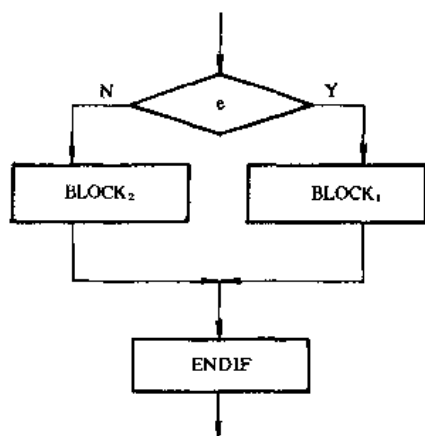


图 4.8

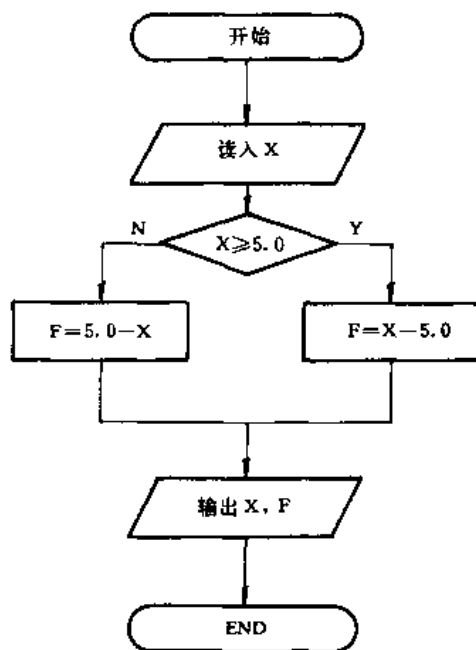


图 4.9

程序框图如图 4.9。程序如下：

```

READ *, X
IF (X.GE. 5.0) THEN
    F=X-5.0
ELSE
    F=5.0-X
ENDIF
PRINT *, X, F
STOP
END

```

程序中 THEN 后 $F=X-5.0$ 相当于 BLOCK1，ELSE 后 $F=5.0-X$ 相当于 BLOCK2。若 $X \geq 5.0$ 时，首先执行 BLOCK1，即 $F=X-5.0$ ，然后执行 ENDIF 后 PRINT 语句；若 $X < 5.0$ ，则执行 ELSE 后 BLOCK2，即 $F=5.0-X$ ，然后执行 ENDIF 后 PRINT 语句。

三、块 IF 小结

块 IF 结构在结构化程序设计中占有重要地位。在结构化程序设计中，应尽量减少使用 GO TO 语句，对于条件判定则最好使用块 IF 结构，这样使程序结构清楚、易读、易调试。

在使用块 IF 结构时, 应注意以下问题:

① 块 IF 结构具有很强的整体结构性

```

      IF (e) THEN
        BLOCK1
      ELSE
        BLOCK2
      ENDIF

```

在这样一种结构中, BLOCK1 或 BLOCK2 以及 ELSE 是可省略的。但表明块 IF 结构的 IF、THEN、ENDIF 是不能省略的。特别是, 一个 IF 必有一个 ENDIF 相对应, 否则块 IF 结构是不完整的。这在 FORTRAN 77 中是不允许的。

② 由于块 IF 结构的完整性, BLOCK1 和 BLOCK2 必须被看作结构中的两个整体, 这两个整体对于进入来说是封闭的。也就是说, 在块 IF 结构中, BLOCK1(或 BLOCK2)块不允许从 BLOCK 块外向块内转移。但可以向外转移。

例如有下面情况:

```

a.      :
      IF (X. GT. 0. 0) THEN
        Y=X+SQRT(X)
10      PRINT *,X,Y
      ELSE
        Y=X-SQRT(-X)
        PRINT *,X,Y
      ENDIF
      IF (Y. LT. 0. 0) GOTO 10
      :
b.      :
      IF (A. GT. B) THEN
        W=A-B
        GOTO 10
      ELSE
        W=B-A
10      PRINT *,A,B,W
      ENDIF
      :

```

a. 是错误的。由于块 IF 结构的整体性, 在语句 IF(Y. LT. 0. 0)GO TO 10 的条件为真时, 会产生一个向标号为 10 的语句转移, 而标号为 10 的语句在块 IF 结构 BLOCK1 中, 这样的转移在执行时是不允许的。

b. 也是错误的。块 IF 结构中 BLOCK1 和 BLOCK2 是相对独立的。程序执行 BLOCK1 还是执行 BLOCK2, 完全依赖于 IF 语句进行对逻辑表达式的值的判断, 而不是由 BLOCK 块外的语句来决定。在 b 中, THEN 后(BLOCK1)中语句 GO TO 10 产生一无条件转移, 而标号为 10 的语句在 BLOCK2 中, 即 ELSE 后。这在 FORTRAN 77 中是不允许的。

4.4 多路判定与块 IF 的嵌套

一、多路判定及 ELSEIF 语句

在处理许多实际问题时，常常需要对多个条件进行判定，以决定程序的执行。这种对三个或更多的条件判定问题，就称之为多路判定问题。

解决多路判定问题，FORTRAN 77 提供了一条效率较高的语句，ELSEIF 语句。ELSEIF 语句处在 IF (e) THEN 语句和 ELSE 语句之间，其形式为

```

IF (e1) THEN
    BLOCK1
ELSEIF (e2) THEN
    BLOCK2
ELSEIF (e3) THEN
    :
ELSEIF (en) THEN
    BLOCKn
ELSE
    BLOCK n+1
ENDIF

```

其中：e1, e2, ..., en 均为逻辑表达式；

BLOCK1, BLOCK2, ..., BLOCKn, BLOCK n+1 均为可执行语句序列。

执行过程是，当 e1 为真时，执行 BLOCK1，然后执行 ENDIF 语句的下一条语句；若 e1 为假时，执行 ELSEIF 语句，判定 e2 的真假。若 e2 的值为真，则执行 BLOCK2，然后执行 ENDIF 的下一条；若 e2 为假，则执行下一个 ELSEIF 语句，…。当 e1, e2, ..., en 均为假时，执行 BLOCK n+1，然后执行 ENDIF 的下一条语句。

由以上讨论可以看出，只要当 e1, e2, ..., en 中的某一个 ei (1 ≤ i ≤ n) 为真值，程序执行相应的 BLOCKi，然后立即跳出块 IF 结构。这显然比对于所有的 ei 都用 IF - THEN - ELSE - ENDIF 结构的情况效率高得多。

例 4.10 计算运费。若 50 t 以下，每吨运费 0.35 元；若 50 t 以上 100 t 以下，则每吨 0.33 元；若大于 100 t 且在 200 t 以下，每吨 0.30 元；若大于 200 t 且在 500 t 以下，每吨 0.26 元；500 t 以上，每吨 0.20 元。

设货物吨位为 X，运费为 TOTAL，于是有：

$0 < X \leq 50$	$TOTAL = 0.35 * X$
$50 < X \leq 100$	$TOTAL = 0.33 * X$
$100 < X \leq 200$	$TOTAL = 0.30 * X$
$200 < X \leq 500$	$TOTAL = 0.26 * X$
$500 < X$	$TOTAL = 0.20 * X$

这就是一个多路判定问题。程序框图如图 4.10 所示。

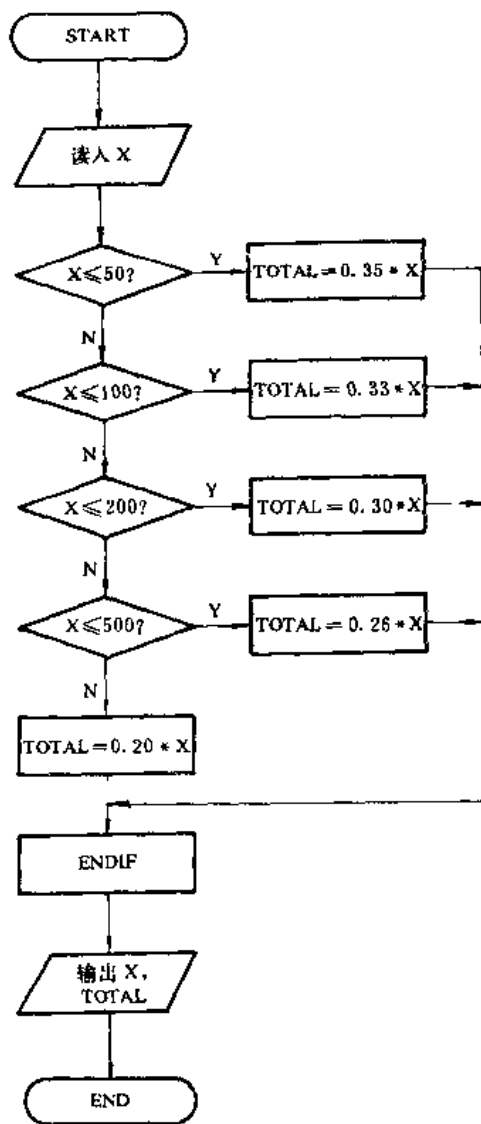


图 4.10

此问题的程序如下：

```

READ *, X
IF (X. LE. 50. 0) THEN
    TOTAL = 0.35 * X
ELSEIF (X. LE. 100. 0) THEN
    TOTAL = 0.33 * X
ELSEIF (X. LE. 200. 0) THEN
    TOTAL = 0.30 * X
ELSEIF (X. LE. 500. 0) THEN
    TOTAL = 0.26 * X
ELSE
    TOTAL = 0.20 * X
ENDIF

```



```

PRINT *,X,TOTAL
END

```

二、块 IF 结构的嵌套

在处理实际问题中，不仅会遇到多路判定问题，而且还会遇到多层判定问题。FORTRAN 77中可用块 IF 的嵌套来解决多层判定问题。

实际上，在块 IF 结构中，THEN 后的 BLOCK 块和 ELSE 后的 BLOCK 块均为可执行语句序列，当这两个块包含有另外的块 IF 结构，这时也就形成了所谓块 IF 结构的嵌套，这种嵌套的一般形式可描述为：

```

① IF (e1) THEN
    ... ..
    ② IF (e2) THEN
        ... ..
        ③ IF (e3) THEN
            ... ..
            ELSE
                ... ..
            ENDIF
        ELSE
            ... ..
            ④ IF (e4) THEN
                ... ..
                ELSE
                    ... ..
                ENDIF
            ... ..
        ENDIF
    ... ..
    ELSE
        ... ..
    ENDIF

```

这里我们只给出了三层嵌套的块 IF 结构。其中：①为最外层；②为第二层；③、④均为第三层。在处理实际问题时，这种嵌套产生多少层次，应根据实际情况由程序员自己实现。

在实现块 IF 的嵌套时，应特别注意：

- ① 每一层的块 IF 结构必须是完整的。
- ② 不允许从块外直接转移到块内。因此在使用 GO TO 语句时要特别小心。
- ③ 使用 ENDIF 时应仔细判明嵌套的层次结构，以防止层次划分不当，导致程序逻辑过程的错误。

例 4.11 对前面的求解一元二次方程的问题, 试用块 IF 结构编制其程序。

设: 一元二次方程为: $ax^2 + bx + c = 0$

(1) 若 $a=0$, 则有

$$bx + c = 0, \quad x = -\frac{c}{b};$$

(2) 若 $a \neq 0$, 则方程的根为

$$\text{root}_1, \text{root}_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a};$$

(a) 若 $b^2 - 4ac = 0$,

$$\text{root}_1 = \text{root}_2 = -\frac{b}{2a};$$

(b) 若 $b^2 - 4ac > 0$,

$$\text{root}_1 = -\frac{b}{2a} + \frac{\sqrt{b^2 - 4ac}}{2a}, \quad \text{root}_2 = -\frac{b}{2a} - \frac{\sqrt{b^2 - 4ac}}{2a};$$

(c) 若 $b^2 - 4ac < 0$,

$$\text{root}_1 = -\frac{b}{2a} + \frac{\sqrt{-(b^2 - 4ac)}}{2a}i, \quad \text{root}_2 = -\frac{b}{2a} - \frac{\sqrt{-(b^2 - 4ac)}}{2a}i.$$

程序框图如图 4.11。程序如下:

```

READ *, A, B, C
IF (ABS(A).LE. 1.0E-30) THEN
    X = -B/C
    PRINT '(1X,12HROOT1=ROOT2=,F15.5)', X
ELSE
    X1 = -B/(2.0 * A)
    D = B * B - 4 * A * C
    IF (ABS(D).LE. 1.0E-30) THEN
        PRINT '(1X,12HROOT1=ROOT2=,F15.5)', X1
    ELSE
        IF (D.GT. 0.0) THEN
            X2 = SQRT(D)/(2.0 * A)
            PRINT '(1X,6HROOT1=,F15.5)', X1 + X2
            PRINT '(1X,6HROOT2=,F15.5)', X1 - X2
        ELSE
            X2 = SQRT(-D)/(2.0 * A)
            PRINT '(1X,6HROOT1=,F15.5,1H-,F15.5,1HI)', X1, X2
            PRINT '(1X,6HROOT2=,F15.5,1H-,F15.5,1HI)', X1, X2
        ENDIF
    ENDIF
ENDIF
STOP
END

```

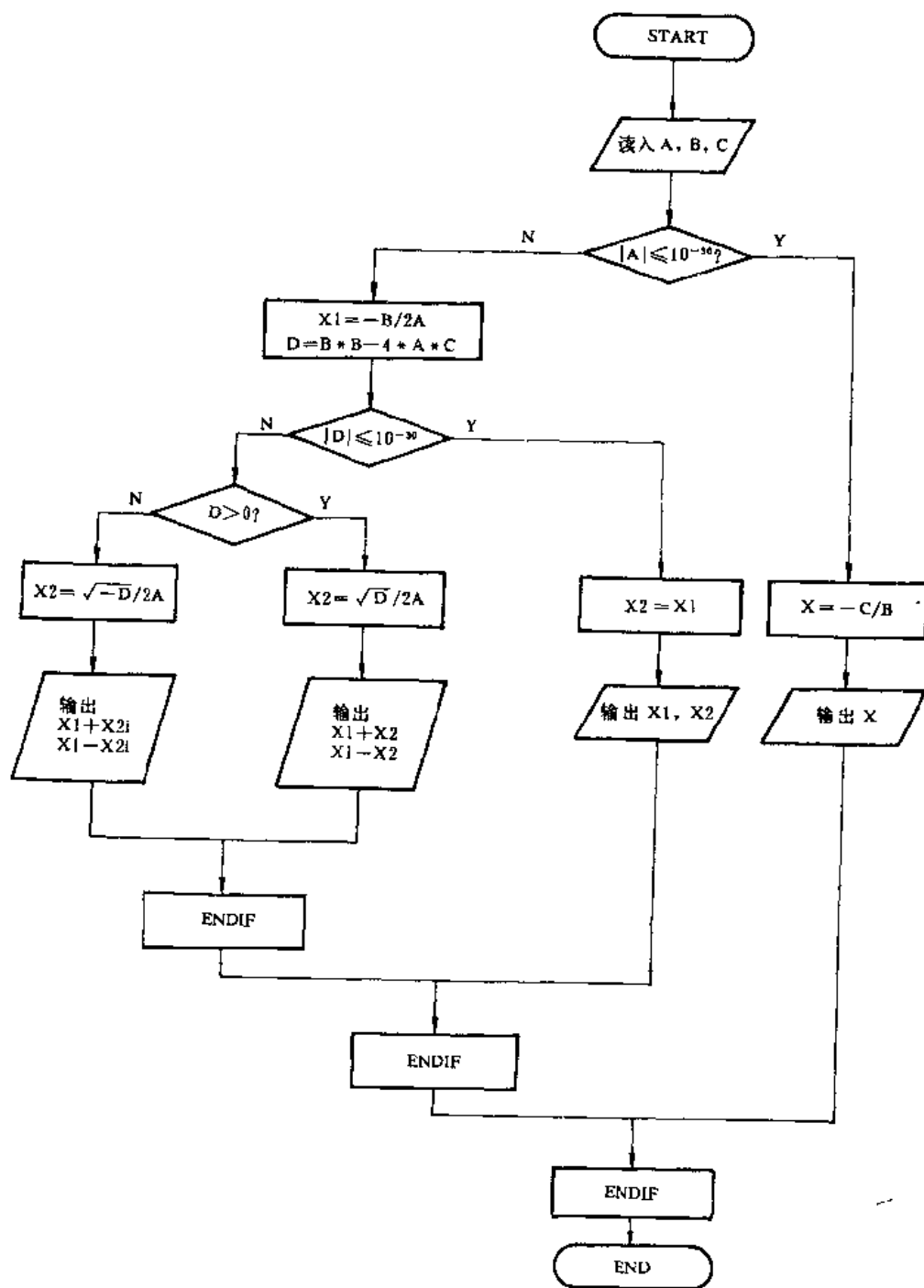


图 4.11

4.5 应用举例

通过这一章的学习，我们就可以编写较复杂的应用程序了。

例 4.12 某旅行社，组织桂林、漓江、阳朔七日游，其收费标准是：个别票每位 1 265.4 元，50 人至 100 人的团体票享受九五折优惠，101 人至 150 人的团体票享受九折优惠，151 人至 200 人的团体票享受八五折优惠，201 至 250 人的团体票享受八折优惠，251 人以上的团体票享受七五折优惠。要求编制程序，根据一次的购票数计算出应收的金额。

解 设一次的购票数为 n ，应收的总金额为 TOTAL 元。此问题求解的流程图见图 4.12。

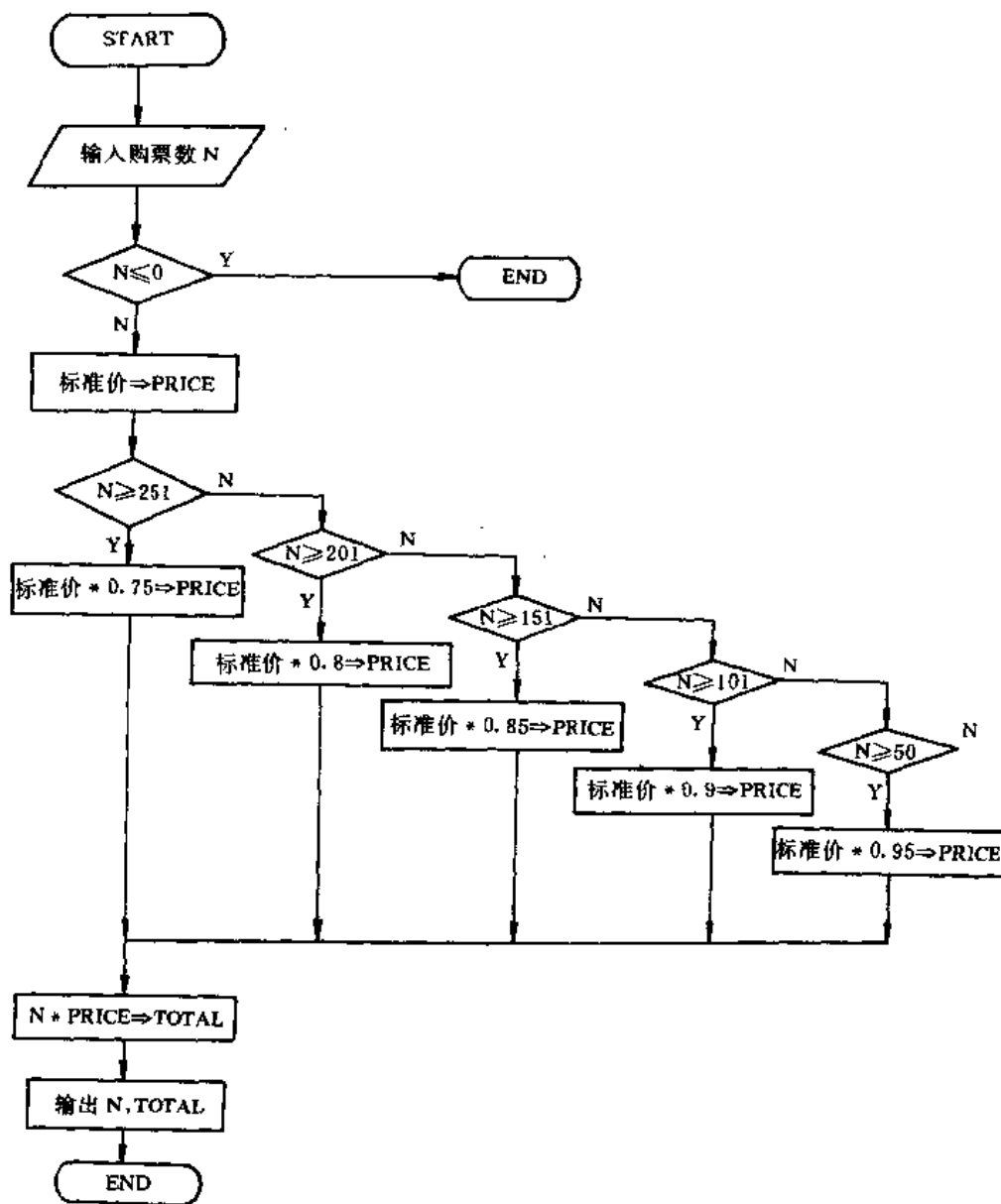


图 4.12

其程序如下：

```

WRITE(*,*) 'ENTER POPULATION OF TRAVELLERS: '
READ *, N
IF(N.LE.0) STOP 'ERROR DATA: N <= 0'
PRICE=1265.4
IF(N.GE.251) THEN
    PRICE=PRICE*0.75
ELSEIF(N.GE.201) THEN
    PRICE=PRICE*0.8
ELSEIF(N.GE.151) THEN
    PRICE=PRICE*0.85
ELSEIF(N.GE.101) THEN
    PRICE=PRICE*0.9
ELSEIF(N.GE.50) THEN
    PRICE=PRICE*0.95
END IF
TOTAL=N*PRICE
PRINT *, N, TOTAL

```

```

PRICE=PRICE*0.8
ELSEIF(N.GE.151) THEN
    PRICE=PRICE*0.85
ELSEIF(N.GE.101) THEN
    PRICE=PRICE*0.9
ELSEIF(N.GE.50) THEN
    PRICE=PRICE*0.95
ENDIF
TOTAL=N*PRICE
WRITE(*,100) N, TOTAL
100 FORMAT(1X,' N= ',I3,'    TOTAL= ',F10.2)
STOP
END

```

例 4.13 输入 A、B、C 三个数，按从大到小的次序将它们输出。

解 求解此问题的程序流程图如图 4.13。

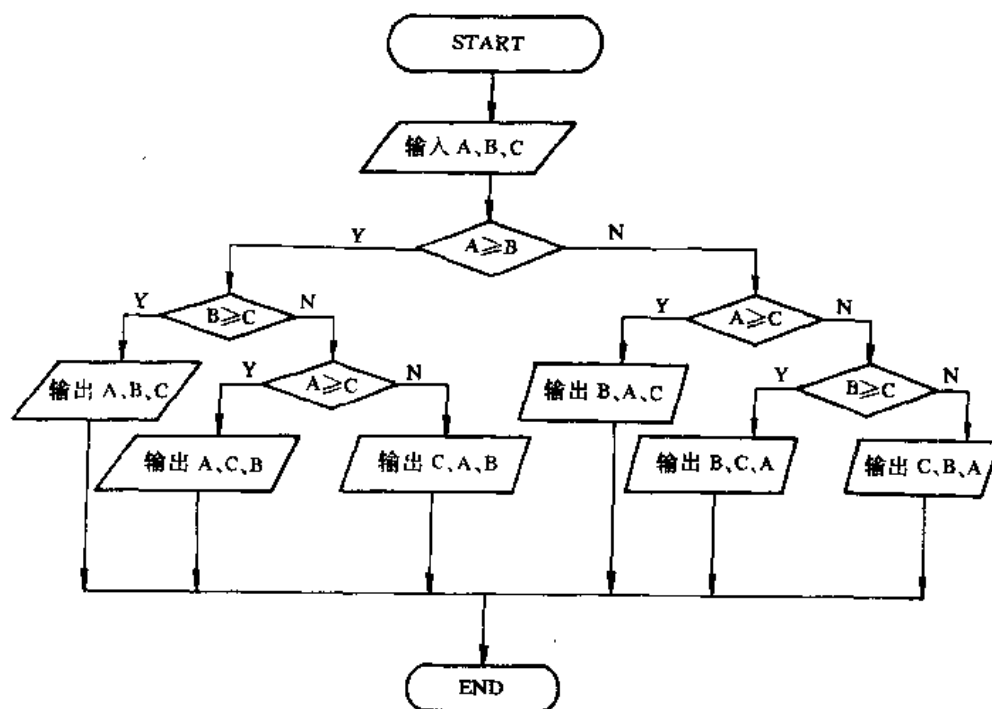


图 4.13

其程序如下：

```

WRITE(*,*) '    ENTER A, B, C:'
READ *, A, B, C
IF(A.GE.B) THEN
    IF(B.GE.C) THEN
        WRITE(*,100) A, B, C
100  FORMAT(1X,3F8.2)
    ELSEIF(A.GE.C) THEN

```

```

      WRITE(*,100) A, C, B
    ELSE
      WRITE(*,100) C, A, B
    ENDIF
  ELSE
    IF(A.GE.C) THEN
      WRITE(*,100) B, A, C
    ELSEIF(B.GE.C) THEN
      WRITE(*,100) B, C, A
    ELSE
      WRITE(*,100) C, B, A
    ENDIF
  ENDIF
STOP
END

```

例 4.14 港口每月进行入港船只统计，排水量不超过 4 000 t 的称为小型船，排水量超过 10 000 t 的称为巨型船，排水量在这两者之间的称为大型船。试编制一个程序分别统计小型船、大型船和巨型船的船数及相应的总排水量，还要给出入港船只的总数及入港船只的总排水量。

解 对每条入港的船只都要按其吨位数判定是小型船、大型船或巨型船，然后按要求进行统计。假定每条船吨位数用输入给定，最后以一个负值结束程序的执行。判定结构可用流程图 4.14 表示。

编制程序如下：

```

      NSMALL=0
      NLARGE=0
      NSUPER=0
      NTOTAL=0
      WSMALL=0.
      WLARGE=0.
      WSUPER=0.
      WTOTAL=0.
100  READ *, TONS
      IF(TONS.GT.0) THEN
        IF(TONS.GT.10000) THEN
          NSUPER=NSUPER+1
          WSUPER=WSUPER+TONS
        ELSEIF(TONS.GT.4000) THEN
          NLARGE=NLARGE+1
          WLARGE=WLARGE+TONS
        ELSE
          NSMALL=NSMALL+1
          WSMALL=WSMALL+TONS

```

```

ENDIF
GOTO 100
ELSE
    NTOTAL=NSMALL+NLARGE+NSUPER
    WTOTAL=WSMALL+WLARGE+WSUPER
    PRINT *, ' NSMALL= ', NSMALL, '      WSMALL= ', WSMALL
    PRINT *, ' NLARGE= ', NLARGE, '      WLARGE= ', WLARGE
    PRINT *, ' NSUPER= ', NSUPER, '      WSUPER= ', WSUPER
    PRINT *, ' NTOTAL= ', NTOTAL, '      WTOTAL= ', WTOTAL
ENDIF
STOP
END

```

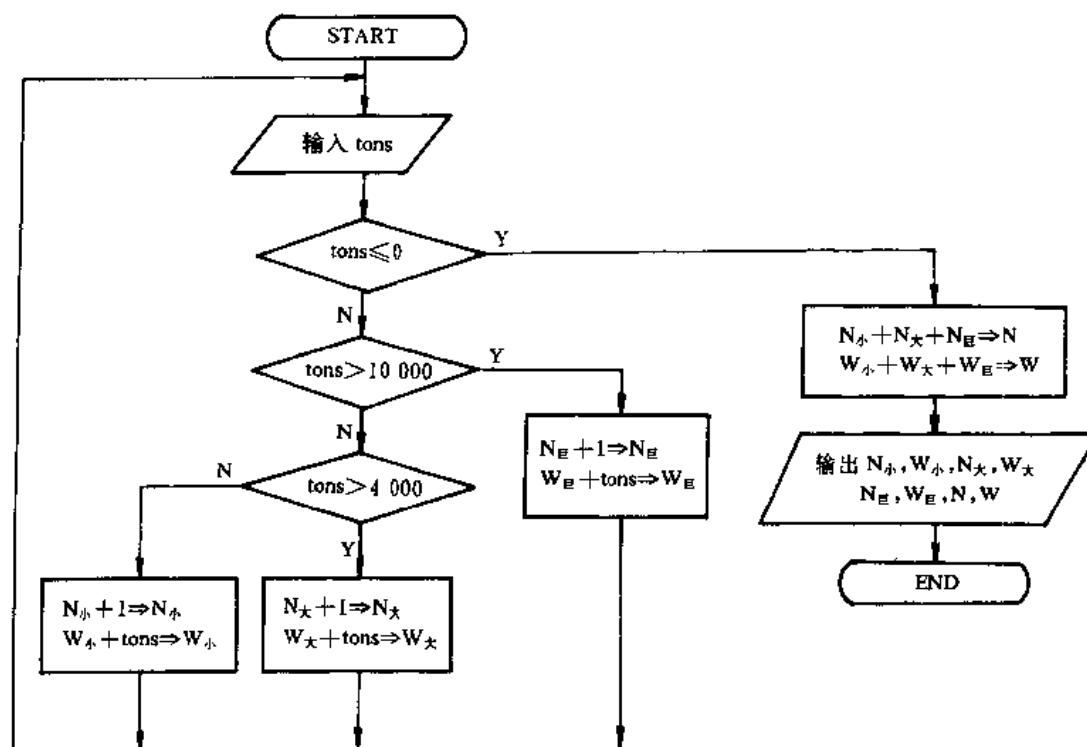


图 4.14

例 4.15 编写一个程序：改变 I 的值并输出结果。若 I 是 1 改为 2；I 是 2 改为 1；I 是 3 改为 0，若 I 是其它值打印出错误信息。

解 可用嵌套判定结构编制程序如下：

```

READ *, I
IF ((I.EQ.1).OR.(I.EQ.2).OR.(I.EQ.3)) THEN
    IF(I.EQ.1) THEN
        I=2
    ELSEIF(I.EQ.2) THEN

```

```

      I=1
    ELSE
      I=0
    ENDIF
    PRINT *, I
  ELSE
    PRINT *, 'THE VALUE OF I IS INCORRECT'
  ENDIF
END

```

例 4.16 在例 4.7 我们用牛顿迭代法求解非线性方程, 在此例我们用二分法求解非线性方程。二分法的示意图见图 4.15。

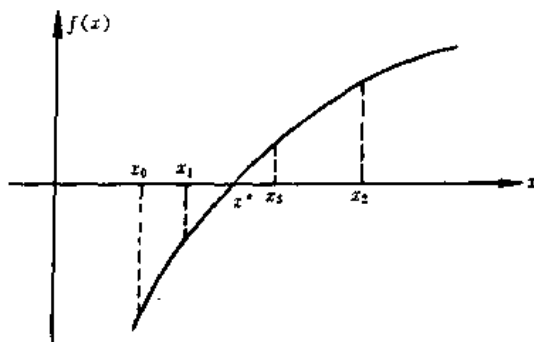


图 4.15

从几何上看, $f(x)=0$ 的根实际上是曲线 $f(x)$ 与 x 轴的交点 x^* 。如果 $f(x)$ 连续, 则 x^* 左右两边的函数值 $f(x^*-\varepsilon)$ 和 $f(x^*+\varepsilon)$ 必定异号。二分法就是根据这一特性, 先从某个 x 初始值开始, 按等距间隔逐点计算 $f(x)$ 的值, 直到相邻的两个函数值 $f(x_n)$ 和 $f(x_{n+1})$ 具有相反的符号为止。这就找到了 x^* 所在的区间 $[x_n, x_{n+1}]$, 然后逐次将这个区间对分, 即用下式求出 x_n 和 x_{n+1} 的中点 x_{mid} 。

$$x_{mid} = \frac{x_n + x_{n+1}}{2}$$

计算 x_{mid} 的函数值 $f(x_{mid})$ 。如果 $f(x_{mid})$ 足够小, x_{mid} 即为所求的近似根, 求解过程结束。否则, 检查 $f(x_{mid})$ 与 $f(x_n)$ 的符号, 若 $f(x_{mid})$ 与 $f(x_n)$ 同号, 则 $f(x_{mid})$ 必定与 $f(x_{n+1})$ 异号, 用 x_{mid} 代替 x_n , 与 x_{n+1} 组成新的搜索区间; 若 $f(x_{mid})$ 与 $f(x_n)$ 异号, 则用 x_{mid} 代替 x_{n+1} , 与 x_n 组成新的搜索区间。从而使搜索区间缩小一半, 经过 N 次迭代, 搜索区间将会缩小到最初的区间长度的 2^{-N} , 最后必定能收敛到所要求的精确度。

二分法的程序框图见图 4.16。

现在要求用二分法求解例 4.7。给出的三次代数方程为

$$2x^3 - 4x^2 + 3x - 6 = 0$$

解 我们从 $x_0=0$ 开始搜索, 步长取 1。由图 4.16 可以看出, 在程序中将多次用到函数

$$f(x) = 2x^3 - 4x^2 + 3x - 6$$

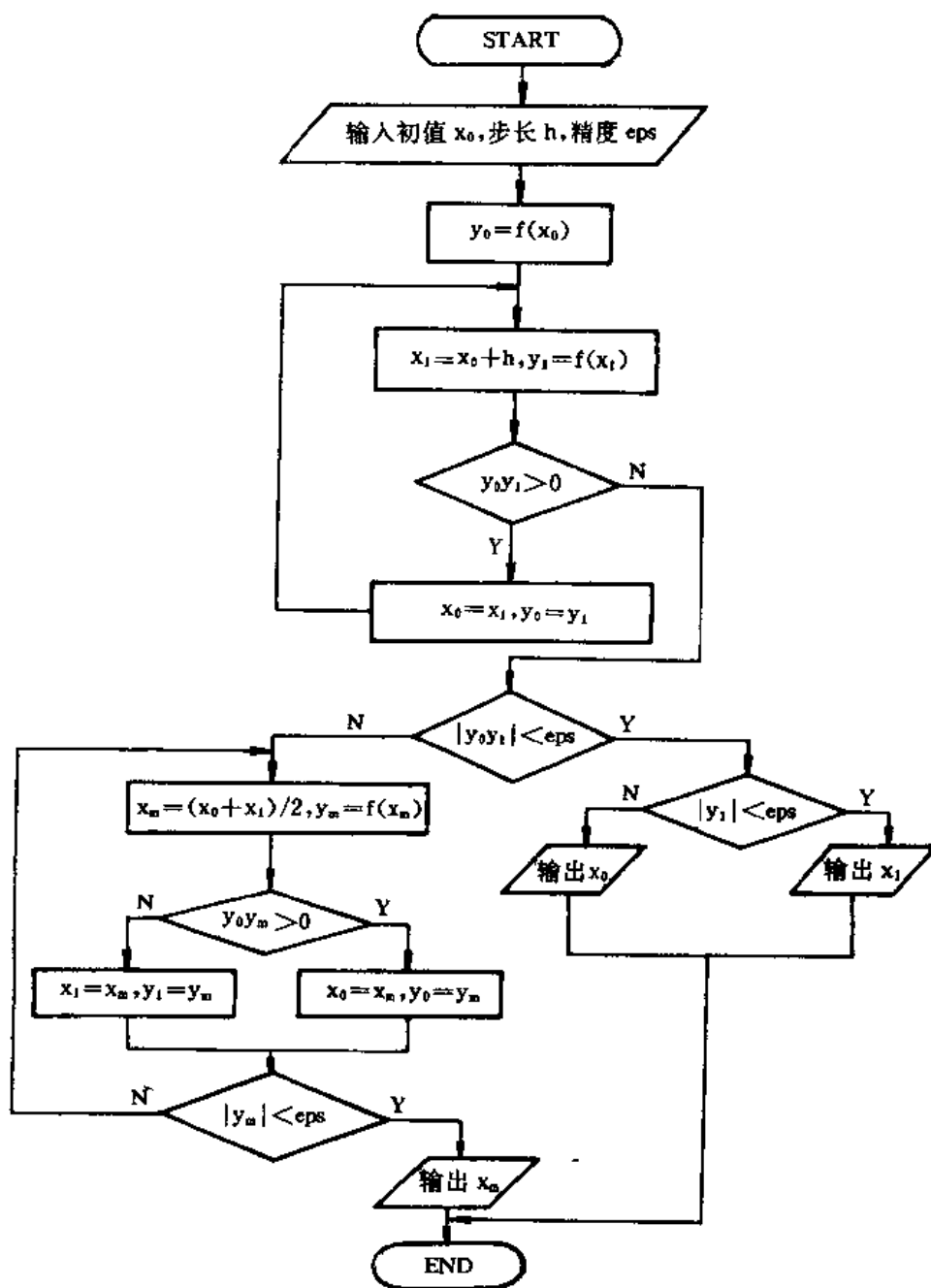


图 4.16

为了能像内部函数一样很方便地使用这个函数，我们自定义这个函数，称为语句函数。欲求这个函数当 $x=a$ 时的值，就如内部函数一样，正是 $f(a)$ 。（这部分的内容将在第6章详细介绍。）

我们编制程序如下：

```
F(X)=X*(X*(2.0*X-4.0)+3.0)-6.0
```

```
H=1.0
```

```
X0=-3.0
```

```
EPS=1E-20
```

```
Y0=F(X0)
```

```
10 X1=X0+H
```

```

      Y1=F(X1)
      IF(Y0 * Y1.GT. 0. 0) THEN
        X0=X1
        Y0=Y1
        GOTO 10
      ENDIF
      IF(ABS(Y0 * Y1).LT. EPS) THEN
        IF(ABS(Y1).LT. EPS) THEN
          WRITE( *, 100) X1
        ELSE
          WRITE( *, 100) X0
        ENDIF
      ELSE
20      XM=(X0 +X1)/2. 0
        YM=F(XM)
        IF(Y0 * YM.GT. 0. 0) THEN
          X0=XM
          Y0=YM
        ELSE
          X1=XM
          Y1=YM
        ENDIF
        IF(ABS(YM).LT. EPS) THEN
          WRITE( *, '00) XM
        ELSE
          GOTO 20
        ENDIF
      ENDIF
100  FORMAT(1X, ' X= ', F13.6)
      END

```

该程序执行结果：

X= 2.000 000

用二分法求解非线性方程时，应当注意初始值 x_0 的取值，由于在这个程序中取步长为正值，若 x_0 取在方程解的右侧，则会出现死循环而得不到解。

例 4.17 某实验需按如下规律计算函数值：

$$f(x) = \begin{cases} 17.8x^{1/2} - 12.5x^{3/2} + 22.8x^{5/2} + x^{7/2} & 0 \leq x < 1 \\ 12.7x^{1/2} - 67.8x^{3/2} + 123.5x^{5/2} & 1 \leq x < 2 \\ 1.5x^{1/2} - 8.7x^{3/2} & 2 \leq x < 3 \\ x^{1/2} & 3 \leq x < 4 \end{cases}$$

试编出计算函数的程序， x 值由输入给定。

解 先把函数表达式整理成如下便于计算的形式：

$$f(x) = \begin{cases} (17.8 - x(12.5 - x(22.8 + x)))x^{1/2} & 0 \leq x < 1 \\ (12.7 - x(67.8 - 123.5x))x^{1/2} & 1 \leq x < 2 \\ (1.5 - 8.7x)x^{1/2} & 2 \leq x < 3 \\ x^{1/2} & 3 \leq x < 4 \end{cases}$$

为了使程序简练,我们引入整型变量 IX,它与输入的 x 值有如下关系:

$$IX = \begin{cases} 1 & 0 \leq x < 1 \\ 2 & 1 \leq x < 2 \\ 3 & 2 \leq x < 3 \\ 4 & 3 \leq x < 4 \end{cases}$$

于是可编制程序如下:

```

      READ *, X
      IX=X+1.0
      F=1.0
      IF=(IX.LT.1.OR.IX.GT.4)THEN
        WRITE(*,100)IX,X
100    FORMAT(IX,I4,F10.4,5X,'INVALID VALUE FOR IX')
        STOP
      ELSEIF(IX.EQ.1) THEN
        F=17.8-X*(12.5-X*(22.8+X))
      ELSEIF(IX.EQ.2) THEN
        F=12.7-X*(67.8-123.5*X)
      ELSEIF(IX.EQ.3) THEN
        F=1.5-8.7*X
      ENDIF
      F=F*SQRT(X)
      WRITE(*,200)X,IX,F
200    FORMAT(IX,F10.4,I8,5X,F10.4)
      END

```

习 题 四

1. 用判定结构表示下列运算:

(1) 若 $x \geq 5.6$, 则 $F = X - 5$, 否则 $F = 5 - X$;

(2) 若 $x + y > A + B$, 则打印 $x + y$, 否则打印 $x - y$, 不论哪种情况都要打印 $A + B$ 。

2. 按图 4.17~图 4.20 的框图编写 IF-THEN-ELSE 判定结构。

3. 试编程序求 y 的值:

$$y = \begin{cases} x & x > 0 \\ 0 & x = 0 \\ -x & x < 0 \end{cases}$$

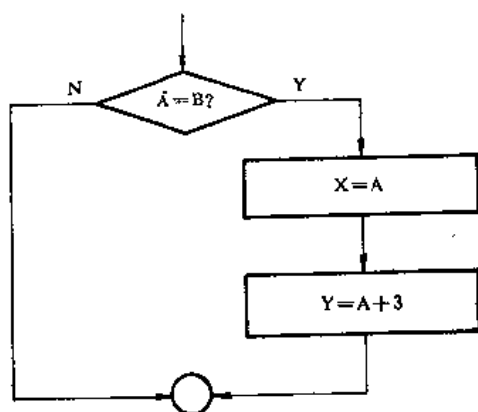


图 4.17

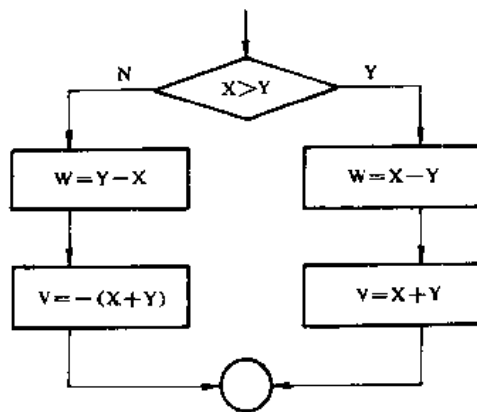


图 4.18

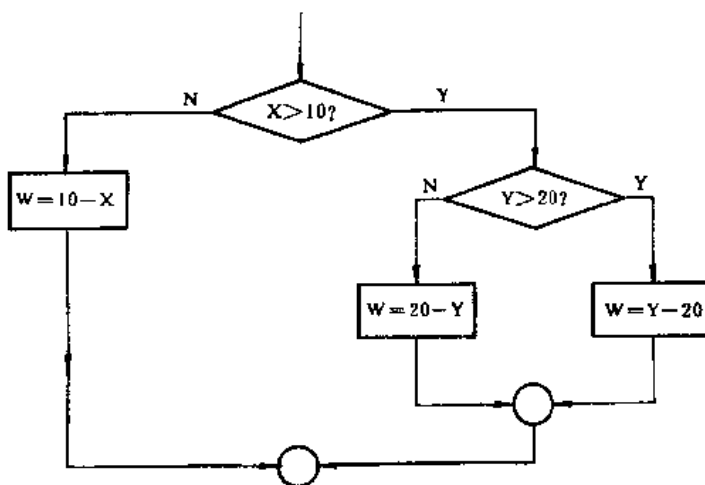


图 4.19

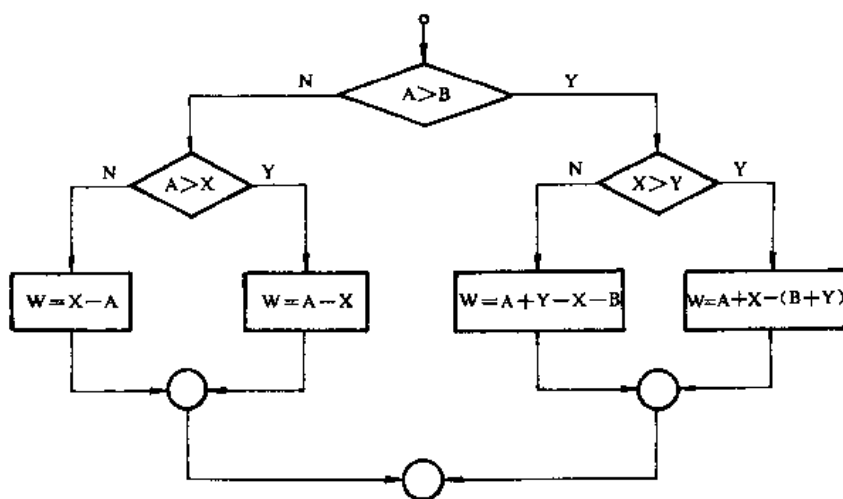


图 4.20

4. 当收入在 800 元以上时, 收取 20% 的税金; 在 800 元以下时免税。试编程序计算之。

5. 三角形的三边长分别为 a, b, c , 构成三角形的条件为任意两边之和大于第三边。试编一程序, 首先判定三角形是否成立, 然后计算其面积。

6. 编写一个过桥计费统计程序, 汽车载重量 1 t 以下收费 0.20 元, 1 t 以上 5 t 以下收费 0.25 元, 5 t 以上收费 0.30 元, 程序要求统计各吨位范围内过桥的车辆数及收费数。

7. 输入三个数 A, B, C , 按由小到大顺序打印出来, 试编写一程序。

8. 编写一程序, 统计学生考试成绩分布情况。统计成绩在 90 分以上的人数, 成绩在 75~90 分之间的人数, 成绩在 60~75 分之间的人数, 成绩在 60 分以下的人数, 并求出所有人的平均成绩。

9. 假定 x 由键盘输入, 计算 y 值

$$y = \begin{cases} x & 0 \leq x < 1 \\ x - 1 & 1 \leq x < 2 \\ x^2 - 1 & 2 \leq x < 3 \\ x^2 - 2x + 1 & 3 \leq x < 4 \\ x^3 - 4x + 2x - 1 & 4 \leq x < 5 \end{cases}$$

若 $x < 0$ 或 $x \geq 5$, 打印“out of range”, 并重新读入 x 。试编写一程序。

10. 编写一程序, 求函数 F 的值(x, y 的值由键盘输入)。

$$F = \begin{cases} \frac{2x^2 + 3x + 1}{y^5 + 1} & \text{当 } x \geq 0, y > 0 \\ \frac{4x^2 + 4x + 2}{y^5 + 1} & \text{当 } x \geq 0, y \leq 0 \\ 3 \sin(x + y) & \text{当 } x < 0 \end{cases}$$

11. 当一年代(如 1992)能被 4 整除但不能被 100 整除, 则该年为闰年; 若能被 4 整除又能被 400 整除, 则该年也为闰年; 其它为非闰年。编写一程序, 输入一个年份, 判定其是否闰年, 并打出相应的信息。

12. 输入一个整数, 判定它能否被 3, 5, 7 整除, 要求程序有以下输出信息之一:

- (1) 能同时被 3, 5, 7 整除。
- (2) 能被 3, 5, 7 中的两个数整除(指明是哪两个数)。
- (3) 能被 3, 5, 7 中的一个数整除(指明是哪个数)。
- (4) 不能被 3, 5, 7 整除。

13. 有一张足够大的纸, 厚 0.09 mm, 问将它对折多少次后, 可以达到珠穆朗玛峰的高度(8 848 m)?

5

循环结构与数组

5.1 循环结构概念

一、循环问题的提出

在编制程序来解决一个实际问题时，常常遇到如下情况，即程序中某一段语句序列，按一定的规则而多次重复执行。类似这样的程序结构常被称之为循环结构。

例 5.1 求 $s=1+2+3+\cdots+N$ 。

在求累加时用到 $s=s+n$ ，而 n 是从 1 直到 N 连续变化的。显然语句 $s=s+n$ 要被连续执行多次(N 次)。

程序如下：

```
INTEGER S
READ *,N
S=0
10 IF (N.GT.0) THEN
    S=S+N
    N=N-1
    GOTO 10
ENDIF
PRINT *,S
STOP
END
```

在上例程序中，可以看出 $s=s+n$ 语句到 GO TO 10 语句在 $n>0$ 时被反复执行，直到 $n=0$

时, 则转向 ENDIF 后向下执行。这样的程序结构称之为循环结构。循环结构由两部分组成:

(1) 循环体

即被重复执行的语句序列。如上例中 $s=s+n$ 到 GO TO 10 语句。

(2) 循环控制机构

即一般用以决定循环是否产生并判定何时结束的机制, 如上例中的 IF(N.GT.0) 语句。

在处理循环问题时, 循环体的循环次数有时是已知的, 有时是未知的。本节先讨论循环次数未知的情况。对于循环次数未知的情况, 通常分为两类来讨论, 即“当…做循环”型 (Loop-while 型) 和“做循环…直到”型 (Loop-until 型)。

二、Loop-while 型循环

这类循环结构的特点是先判断后执行, 即首先判定循环条件, 当条件满足时, 执行循环体。图 5.1 描述了这样的情况。

例 5.2 求 $e=1+\frac{1}{1!}+\frac{1}{2!}+\cdots+\frac{1}{n!}$ 。当满足 $|e_{n+1}-e_n|\leq 10^{-7}$ 时结束。

这里重复计算求阶乘的倒数 $s_n=\frac{1}{(n-1)!}\cdot\frac{1}{n}$, 以及每次将 s_n 进行累加 $E_n=E_{n-1}+s_n$ 。循环判定条件为 $|E_{n+1}-E_n|\leq 10^{-7}$, 将其化简

$$E_{n+1}-E_n=E_n+s_{n+1}-E_n=s_{n+1}$$

因此 $|E_{n+1}-E_n|=|s_{n+1}|\leq 10^{-7}$ 为判定条件。

程序如下:

```

REAL N
E=0.0
N=0.0
S=1.0
10  IF (S.GT. 1.0E-7) THEN
      E=E+S
      N=N+1
      S=S*(1.0/N)
      GOTO 10
    ENDIF
PRINT *,E
STOP
END

```

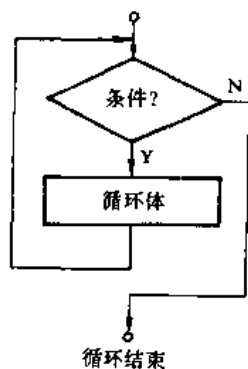


图 5.1

例 5.3 求如下的无穷级数之和, 要求的精度为 10^{-5} :

$$y = \sum_{n=1}^{\infty} (-1)^{n-1} \cdot \frac{x^{2n-1}}{(2n-1)!}$$

即若存在某个 m , 使

$$\left| (-1)^{m-1} \cdot \frac{x^{2m-1}}{(2m-1)!} \right| < 10^{-5}$$

```

PROCESS(a,b)
    CONSTANT max_limit; INTEGER:=255;
BEGIN
    FOR i IN 0 TO max_limit LOOP
        IF (done(i)=TRUE)THEN
            NEXT;
        ELSE
            done(i):=TRUE;
        END IF
        q(i)<=a(i) AND b(i);
    END LOOP;
END PROCESS;

```

当 LOOP 语句嵌套时,通常 NEXT 语句应标有“标号”和“WHEN 条件”。例如,有一个 LOOP 嵌套的程序如下所示:

```

L1: WHILE i<10 LOOP
L2: WHILE j<20 LOOP
    :
    NEXT L1 WHEN i=j;
    :
END LOOP L2;
END LOOP L1;

```

在上例中,当 $i=j$ 时, NEXT 语句被执行,程序将从内循环跳出,而再从下一次外循环开始执行。

由此可知, NEXT 语句实际上是用于 LOOP 语句的内部循环控制。

5.1.9 EXIT 语句

EXIT 语句也是 LOOP 语句中使用的循环控制语句,与 NEXT 语句不同的是,执行 EXIT 语句将结束循环状态,而从 LOOP 语句中跳出,结束 LOOP 语句的正常执行。EXIT 语句的书写格式为:

```
EXIT [标号] [WHEN 条件]
```

如果 EXIT 后面没有跟“标号”和“WHEN 条件”,则程序执行到该语句时就无条件地从 LOOP 语句中跳出,结束循环状态,继续执行 LOOP 语句后继的语句,如例 5-16 所示。

【例 5-16】

```

PROCESS(a)
    VARIABLE int_a; INTEGER;
BEGIN
    int_a:=a;
    FOR i IN 0 TO max_limit LOOP
        IF(int_a<=0) THEN
            EXIT;
        ELSE

```



```

int_a:=int_a-1;
q(i)<=3.1416/REAL(a*i);
END IF
END LOOP;
y<=q;
END PROCESS;

```

在该例中 int_a 通常代入大于 0 的正数值。如果 int_a 的取值为负值或零将出现错误状态, 算式就不能计算。也就是说 int_a 小于或等于 0 时, IF 语句将返回“真”值, EXIT 语句得到执行, LOOP 语句执行结束。程序将向下执行 LOOP 语句后继的语句。

EXIT 语句具有 3 种基本的书写格式。第一种书写格式是 EXIT 语句没有“循环标号”或“WHEN 条件”。当条件为“真”执行 EXIT 语句时, 程序将按如下顺序执行: 执行 EXIT, 程序将仅仅从当前所属的 LOOP 语句中退出。如果 EXIT 语句位于一个内循环 LOOP 语句中, 即该 LOOP 语句嵌在任何其它的一个 LOOP 语句中。那么执行 EXIT, 程序仅仅退出内循环, 而仍然留在外循环的 LOOP 语句中。

第二种书写格式是 EXIT 语句后跟 LOOP 语句的标号。此时, 执行 EXIT 语句时, 程序将跳至所说明的标号。

第三种书写格式是 EXIT 语句后跟“WHEN 条件”语句。当程序执行到该语句时, 只有所说明的条件为“真”的情况下, 才跳出循环的 LOOP 语句。此时, 不管 EXIT 语句是否有标号说明, 都将执行下一条语句。如果有标号说明, 下一条要执行的语句将是标号所说明的语句。如果无标号说明, 下一条要执行的语句是循环外的下一条语句。

EXIT 语句是一条很有用的控制语句。当程序需要处理保护、出错和警告状态时, 它能提供一个快捷、简便的方法。

5.2 并发描述语句

在 VHDL 语言中能进行并发处理的语句有: 进程(PROCESS)语句, 并发信号代入(Concurrent Signal Assignment)语句, 条件信号代入(Conditional Signal Assignment)语句, 选择信号代入(Selective Signal Assignment)语句, 并发过程调用(Concurrent Procedure Call)语句和块(BLOCK)语句。由于硬件描述语言所描述的实际系统, 其许多操作是并发的, 所以在对系统进行仿真时, 这些系统中的元件在定义的仿真时刻应该是并行工作的。并发语句就是用来表示这种并发行为的。并发描述可以是结构性的也可以是行为性的。在并发语句中最关键的语句是进程。下面介绍一下各种并发语句的使用。

5.2.1 进程(PROCESS)语句

PROCESS 语句在前面已多次提到, 并在众多实例中得到了广泛的使用。进程语句是一种并发处理语句, 在一个构造体中多个 PROCESS 语句可以同时并行运行。因此, PROCESS 语句是 VHDL 语言中描述硬件系统并发行为的最基本的语句。

PROCESS 语句归纳起来有如下几个特点:

算的精度。

(2) 控制循环的条件不能是 $T \geq 1E-5$, 因为当 $T < 0$ 时该条件就不成立, 但并未达到计算的精度, 所以必须写成

$$\text{ABS}(T) \geq 1E-5$$

(3) 在本题中, 当利用级数的前项 a_n 来计算后项 a_{n+1} 后, a_n 已无其它用处了, 所以在程序中把级数的前项和后项用同一个变量 T 来表示。在语句

$$T = -X * X * T / (2 * N) / (2 * N + 1)$$

中, 等号右边的 T 表示前项, 等号左边的 T 表示后项。这种技巧在累加、累乘时是常用的, 在例 5.2 中也是这样处理的。

(4) 显然, 在计算无穷级数求和时, 级数必须收敛, 否则会出现死循环。

我们用逻辑 IF 语句也可以实现 Loop-while 型循环。对例 5.3 用逻辑 IF 语句可写成如下程序:

```

      WRITE(*, 100)
100  FORMAT(1X, 'Enter X: ')
      READ(*, 200)X
200  FORMAT(F10.5)
      N=1
      Y=0.0
      T=X
10   IF(ABS(T).LT.1E-5) GOTO 20
      Y=Y+T
      N=N+1
      T=-X*X*T/(2*N)/(2*N+1)
      GOTO 10
20   WRITE(*, 300)Y
300  FORMAT(1X, F10.5)
      STOP
      END

```

例 5.4 对于一个大于或等于 3 的正整数, 判断它是不是一个素数。

所谓素数, 是指除 1 和该数本身之外, 不能被其它任何整数整除的数。例如, 11 是素数。因为它不能被 2, 3, 4, ..., 10 整除。

判断一个数 $N(N \geq 3)$ 是否为素数的方法很简单: 将 N 作为被除数, 将 2 到 $(N-1)$ 各个整数轮流作为除数, 如果都不能被整除, 则 N 为素数。还可以有更简单的方法: 将 N 作为被除数, 将 2 到 \sqrt{N} 之间各个整数轮流作为除数, 如果都不能被整除, 则 N 为素数。例如, 判断 17 是否为素数, 只需将 17 被 2, 3, 4 除即可, 如都除不尽, 17 必为素数。请读者分析为什么这两种方法可用来判断一个正整数 $N(N \geq 3)$ 是否为素数。

解 我们用上述第二种方法来判断一个正整数 $N(N \geq 3)$ 是否为素数。其流程图见图 5.4。

其程序如下:

```

      READ *, N
      J=SQRT(N)

```

```

I=2
10  IF((I.LE.J).AND.(MOD(N,I).NE.0)) THEN
      I=I+1
      GOTO 10
    ENDIF
    IF(I.GT.J) THEN
      PRINT *, N, ' IS A PRIME NUMBER. '
    ELSE
      PRINT *, N, ' IS NOT A PRIME NUMBER. '
    ENDIF
  END

```

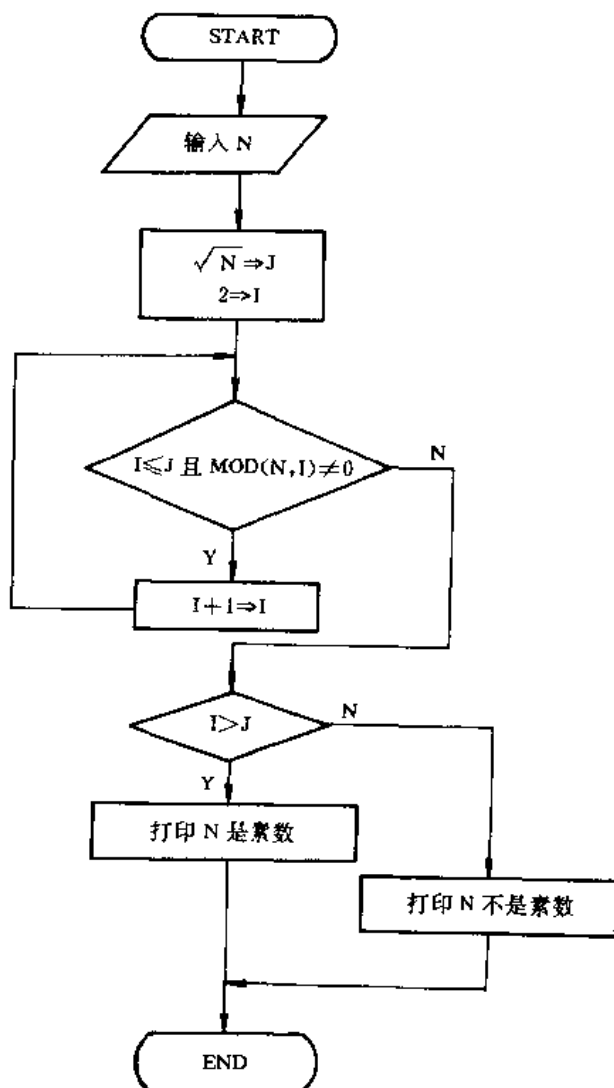


图 5.4

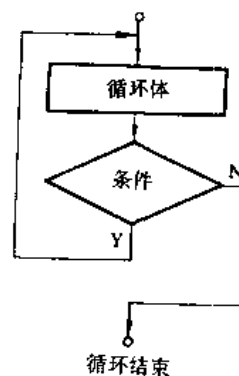


图 5.5

三、Loop - until 型循环

这类型的循环结构特点是先执行，后判定，即先执行循环体，再对循环条件进行判定。这种类型的循环结构与“Loop - while”型结构不同之处在于，在“Loop - until”型结构下，当判

5.2.4 选择信号代入(Selective Signal Assignment)语句

选择信号代入语句类似于 CASE 语句,它对表达式进行测试,当表达式取值不同时,将使不同的值代入目的信号量。选择信号代入语句的书写格式如下:

```
WITH 表达式 SELECT
    目的信号量 <= 表达式1 WHEN 条件1
    表达式2 WHEN 条件2
    :
    表达式 n WHEN 条件 n;
```

下面仍以四选一电路为例说明一下该语句的使用方法,具体如例 5-18 所示。

【例 5-18】

```
ENTITY mux IS
    PORT(i0,i1,i2,i3,a,b: IN STD_LOGIC;
          q: OUT STD_LOGIC);
END mux;
ARCHITECTURE behav OF mux IS
    SIGNAL sel: INTEGER;
BEGIN
    WITH sel SELECT
        q <= i0 WHEN 0,
            i1 WHEN 1,
            i2 WHEN 2,
            i3 WHEN 3,
            'X' WHEN OTHERS;
    sel <= 0 WHEN a='0' AND b='0' ELSE
        1 WHEN a='1' AND b='0' ELSE
        2 WHEN a='0' AND b='1' ELSE
        3 WHEN a='1' AND b='1' ELSE
        4;
END behav;
```

上例中的选择信号代入语句,根据 sel 的当前不同值来完成 i0, i1, i2, i3 及剩余情况的选择功能。选择信号代入语句在进程外使用。当被选择的信号(例如 sel)发生变化时,该语句就会启动执行。由此可见,选择信号的并发代入,可以在进程外实现 CASE 语句进程的功能。例如,四选一电路用 CASE 语句进程所描述的程序如例 5-19 所示。

【例 5-19】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY mux4 IS
    PORT(input: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
          i0,i1,i2,i3: IN STD_LOGIC;
          q: STD_LOGIC);
```

```

END mux4;
ARCHITECTURE rtl OF mux4 IF
BEGIN
    PROCESS(input)
    BEGIN
        CASE input IS
            WHEN "00" => q<=i0;
            WHEN "01" => q<=i1;
            WHEN "10" => q<=i2;
            WHEN "11" => q<=i3;
            WHEN OTHERS=> q<='X';
        END CASE;
    END PROCESS;
END rtl;

```

对照例 5-18 和例 5-19 可以看到, 两者功能是完全一样的, 所不同的仅仅是描述方法有区别而言。

5.2.5 并发过程调用(Concurrent Procedure Call)语句

并发过程调用语句可以出现在构造体中, 而且是一种可以在进程之外执行的过程调用语句。有关过程的结构及书写方法在 2.2 节中已详述, 这里仅就调用时应注意的几个问题作一说明。

- 并发过程调用语句是一个完整的语句, 在它的前面可以加标号;
- 并发过程调用语句应带有 IN, OUT 或者 INOUT 的参数, 它们应列于过程名后跟的括号内;
- 并发过程调用可以有多个返回值, 但这些返回值必须通过过程中所定义的输出参数带回。

在构造体中采用并发过程调用语句的实例如下所示:

```

ARCHITECTURE ...
BEGIN
    vector_to_int (z,x_flag,q);
    ;
END;

```

例中的 vector_to_int 并发过程调用是对位矢量 z 进行数制转换, 使之变成十进制的整数 q。x_flag 是标志位, 当标志位为“真”表明转换失败, 为“假”表明转换成功。

这种并发过程调用语句实际上是一个过程调用进程的简写。如 2.2 节所述, 过程调用语句可以出现在进程语句中, 如果该进程的作用就是进行过程调用, 完成该过程的操作功能, 那么两者是完全等效的。由此可知, 上例的并发过程调用语句和下面的过程调用进程是完全等效的, 因为两者都是为了完成位矢量至整数的转换。

```

ARCHITECTURE ...
BEGIN
    PROCESS(z,q)

```

(2) REAL A(-2 : 3), N(2, 3)

两种写法是等价的。

三、数组元素的排列顺序

数组元素在计算机内存中是按一定规则顺序排列的。如有以下数组：

A ($n_1 : n_2$)

B ($n_1 : n_2, m_1 : m_2$)

C ($n_1 : n_2, m_1 : m_2, k_1 : k_2$)

用图 5.6 描述了 A, B, C 数组的排列顺序情况。

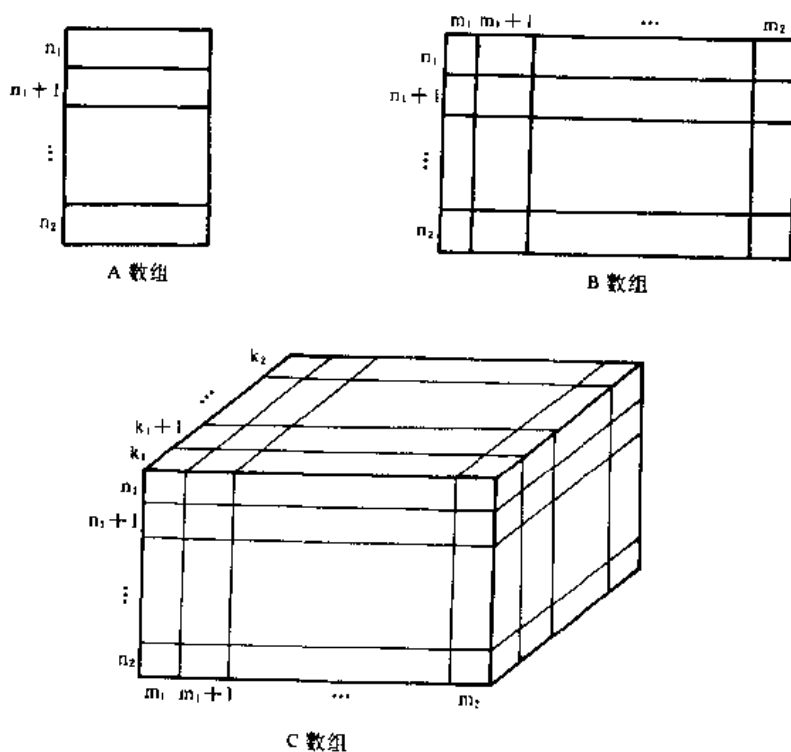


图 5.6

例 5.9 给出矩阵

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

要存入数组 A(3, 3)。

A(3, 3)在计算机内存中排列顺序为 A(1, 1)、A(2, 1)、A(3, 1)、A(1, 2)、A(2, 2)、A(3, 2)、A(1, 3)、A(2, 3)、A(3, 3)。A 中元素对应的值应为

$$\begin{array}{lll} A(1, 1)=1 & A(1, 2)=2 & A(1, 3)=3 \\ A(2, 1)=4 & A(2, 2)=5 & A(2, 3)=6 \\ A(3, 1)=7 & A(3, 2)=8 & A(3, 3)=9 \end{array}$$

所以我们给数组 A, 赋值的顺序应为 1, 4, 7, 2, 5, 8, 3, 6, 9。FORTRAN 中, 二维数组元素在内存中总是一列接一列存放的, 列号小的放在前面, 列号大的放在后面。这种存放方式, 称为按列存放。数组元素的排列方式对数组输入输出影响很大, 读者务必注意。

对于三维数组 C(I, J, K), 我们将 I 叫做行, J 叫做列, K 叫做面。三维数组在内存中也是占一串连续的存储单元。就以 C(2, 3, 4) 为例, 它的元素在内存中的排列次序如图 5.7 所示, 先存放第 1 面中的元素, 再存放第 2 面中的元素, 然后存放第 3 面中的元素, 最后存放第 4 面中的元素。而对每面中的元素就如同二维数组一样按列存放。

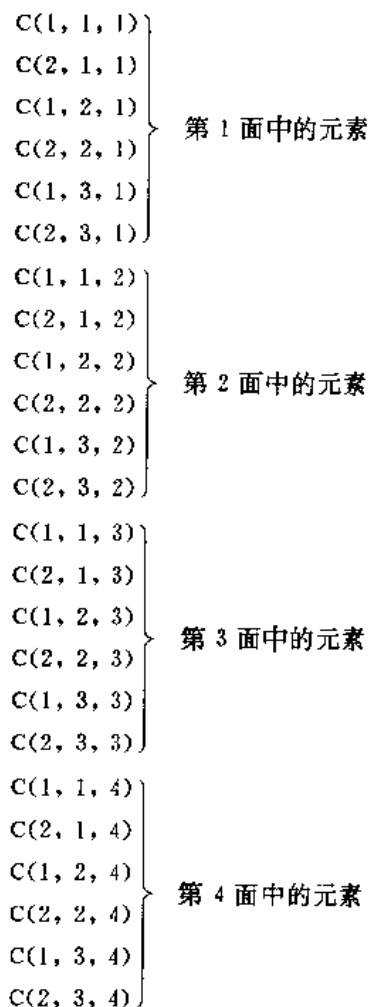


图 5.7

四、利用 DATA 语句给数组赋初值

FORTRAN 77 中, 可以利用 DATA 语句方便地给一个数组赋初值。

例 5.10 `DIMENSION A(10, 20)`
 `DATA A/200*0.0/`

该例中, 将 A 中所有的元素赋零。

例 5.11 `DIMENSION X(3, 5)`
 `DATA X(1, 1), X(2, 1), X(3, 1)/3*1.0/`

该例中, 将 X 数组的第一列均赋值 1.0。

例如, 例 5-20 中的 qbus 信号, 只在 ALU 块中说明。因此, 它是 ALU 块的局部信号量, 所有 ALU 块中的语句都可以使用 qbus 信号, 但是在 ALU 块之外则不能使用该信号。如 REG8 块就不能使用 qbus 信号。再譬如, zbus 信号是 REG8 块所说明的局部信号量, REG1 块是嵌套在 REG8 块中的内层块, 所以 REG1 块可以使用 zbus 信号。

在 REG1 块的信号说明项中也有一个称为 qbus 的信号, 该信号与 BLOCK 块中所说明的信号具有相同名字。那么, 这样做会不会引起问题呢? 事实上, 编译器将分别对这两个信号进行处理。在语法上虽然是合法的, 但是容易引起混淆。两个信号分别在各自的说明区域中加以说明, 同样也仅仅在这些范围中有效。因此, 可以这样认为, 它们是具有相同信号名的各自独立的信号。每个 qbus 只能在所说明的 BLOCK 块区域中使用。

还有一种应该注意的情况如下所示:

```
BLK1: BLOCK
    SIGNAL qbus: tw32;
BEGIN
    BLK2: BLOCK
        SIGNAL qbus: tw32;
    BEGIN
        -- BLK2 语句
    END BLOCK BLK2;
    -- BLK1 语句
END BLOCK BLK1;
```

在上述的实例中, 信号 qbus 在两个块中都作了说明。应注意的是, 这两个块是嵌套关系, 一个块包含另一个块。现在先来看一下 BLK2 块对信号 qbus 的操作。第一种类型的操作是 BLK2 中的语句对 BLK2 中所说明的 BLK2 局部信号量 qbus 进行的操作; 第二种类型的操作是 BLK2 中的语句对 BLK1 中所说明的局部信号量 qbus 进行的操作(由于 BLK1 包含 BLK2, 因此这种操作是允许的)。由此可见, BLK1 块所说明的信号可以看作是 BLK2 块内部说明的信号。如果名字相同, 可以在信号名字前面加块名字前缀。例如, 在本例中 BLK1 块的 qbus 可以用 BLK1 qbus 来表示。

通常, 这种同名信号会使编程发生混乱。出现这个问题的起因是, 在有限时间内, 如果不对信号说明进行详细分析, 就不能保证正确地使用 qbus 信号。

以上所提到的只是块本身范围所涉及的问题。但是, 块是一个独立的子结构, 它可以包含 PORT 和 GENERIC 语句。这样就允许设计者通过这两个语句将块内的信号变化传递给块的外部信号, 同样也可以将块外部的信号变化传递给块的内部。

PORT 和 GENERIC 语句的这种性能, 将允许在一个新的设计中可重复使用 BLOCK 块。例如, 在上例的 CPU 的设计中, 如果需要扩展 ALU 部分的功能, 设计一个新的 ALU 模块, 使其完成新的所需要的功能。在新的 CPU 模块中 PORT 名和 GENERIC 名与原来的不一致, 此时, 如果在块中采用 PORT 和 GENERIC 映射就可以顺利解决这个问题。也就是说, 在上例的基础上, 在设计中映射信号名和产生参数, 就可以建立一个新的 ALU 模块, 如例 5-21 所示。

【例 5-21】

```
PACKAGE math IS
```



```

TYPE tw32 IS ARRAY(31 DOWNTO 0)
    OF STD_LOGIC;
FUNCTION tw_add(a,b: tw32)
    RETURN tw32;
FUNCTION tw_sub(a,b: tw32)
    RETURN tw32;
END math;
USE WORK.math.ALL;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY cpu IS
    PORT(clk, interrupt: IN STD_LOGIC;
        add: OUT tw32; comt: IN INTEGER;
        data: INOUT tw32);
ARCHITECTURE cpu_blk OF cpu IS
    SIGNAL ibus, dbus: tw32;
BEGIN
    ALU: BLOCK
        PORT(abus, bbus: IN tw32;
            d_out: OUT tw32;
            ctbus: IN INTEGER);
        PORT MAP(abus => ibus, bbus => dbus,
            d_out => data, ctbus => comt);
        SIGNAL qbus: tw32;
    BEGIN
        d_out <= tw_add(abus, bbus) WHEN
            ctbus = 0 ELSE
            tw_sub(abus, bbus) WHEN
            ctbus = 1 ELSE
            abus;
    END BLOCK ALU;
END;

```

从例 5-21 可以看出,除了端口和端口映射语句之外,ALU 的说明部分和前面例子中所述的是一样的。端口语句说明了端口号和方向,并且还说明了端口的数据类型。端口映射语句映射了带有信号的新的端口,或者映射了 BLOCK 块外部的端口。例如,本例中端口 abus 被映射到 cpu_blk 构造体内说明的局部信号 ibus;端口 bbus 被映射到 dbus;端口 d_out 和 ctbus 被映射到实体外部的端口。

映射实现了端口和外部信号之间的连接,使连接到端口的信号值发生变化,由原来的值变成一个新的值。如果这种变化发生在 ibus 上,则 ibus 上出现的新的值,将被传送到 ALU 块内,使得 abus 端口得到新的值。当然,其它有映射关系的端口也应如此。

5.3 其它语句和有关规定的说明

在 VHDL 语言中除了顺序描述语句和并发描述语句之外,还有一些说明语句,定义语

```

        M=I+1
100  CONTINUE
而下面用法是不允许的:
        DO 100 I=1, 5
            :
        I=M*N
            :
100  CONTINUE
    
```

③ 循环体内的判定结构必须是完整的, 但要注意转移时不得以 DO 语句本身作为转移目标。

例如, 下面的用法是不正确的:

```

        X=8
20  DO 100 I=1, 10
            :
        IF (X.GT.0.0) GO TO 20
            :
100  CONTINUE
    
```

这种情况会无休止的循环下去。

④ 在循环体外不得用转移语句不经过 DO 语句而进入循环体。但由循环体内可转移到循环体外。

四、DO 循环例子

例 5.13 利用 DO 循环给数组赋值。

```

        DIMENSION X(5)
        DO 10 I=1,5
            READ *,X(I)
10  CONTINUE
            :
    
```

本例中首先定义数组 X(5), 这时若要从键盘上给 X 数组送入数据, 就利用了 DO 循环结构。当循环开始时 I=1, 执行循环体 READ *, X(I), 相当于读入了 X(1); 第二次循环时, I=2, 执行循环体读入了 X(2); 如此下去, 当 I=5 时, 读入 X(5); 下一次 I=6, 这时循环次数已为零, 循环结束。到这时, 已实际读入了 X(1)、X(2)、X(3)、X(4)、X(5), 完成了对 X 数组的数据输入。

在上例中, 若将 READ 语句换为 PRINT 语句或 WRITE 语句, 则就可实现对数组的输出处理。请读者自己分析一下。

例 5.14 求由键盘输入的 20 个数据的平均值。

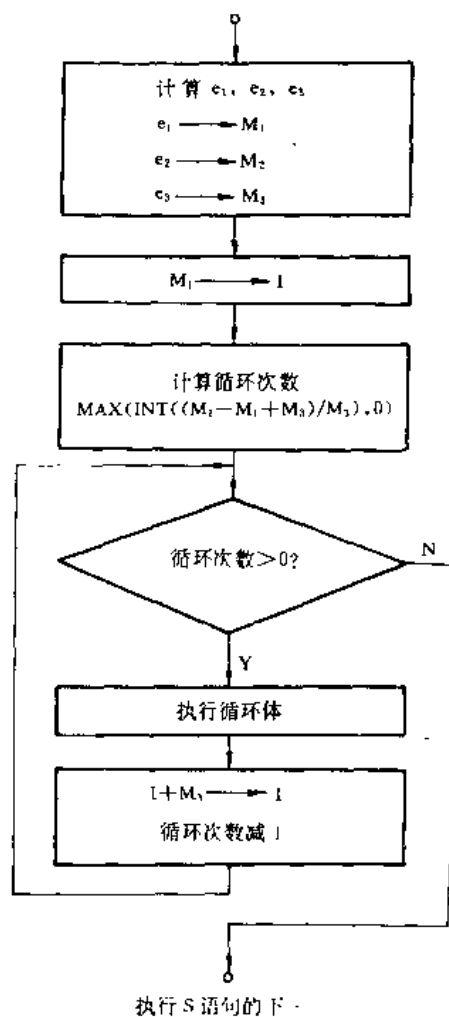


图 5.8

框图见图 5.10。程序如下：

```
SUM=0.0
N=0
DO 10 I=1,20
  READ *,X
  SUM=SUM+X
  N=N+1
10 CONTINUE
AVE=SUM/N
PRINT *,AVE
STOP
END
```

要注意的是每读入一个数据要打回车键。

五、DO WHILE 语句

在 FORTRAN 77 5.10 版本中，处理循环结构除 DO 语句之外，还提供了一条 DO WHILE 语句。其基本形式为

DO S WHILE (e)

其中：S 为循环终止语句标号；

e 为逻辑表达式。

同 DO 语句一样，S 为一条可执行语句的语句标号，表明循环体的终止。通常这条语句为 CONTINUE 语句。

DO WHILE 语句的执行是这样的：首先判定逻辑表达式 e 的值；若 e 的值为真，则执行循环体；若 e 的值为假，则表示循环结束，跳出循环，执行 S 语句后的语句。图 5.11 说明了该语句的执行过程。

例 5.15 仿例 5.14 用 DO WHILE 语句完成程序。框图如图 5.12 所示。程序如下：

```
SUM=0.0
N=1
DO 100 WHILE(N. LE. 20)
  READ *,X
  SUM=SUM+X
  N=N+1
100 CONTINUE
AVE=SUM/(N-1)
PRINT *,AVE
STOP
END
```

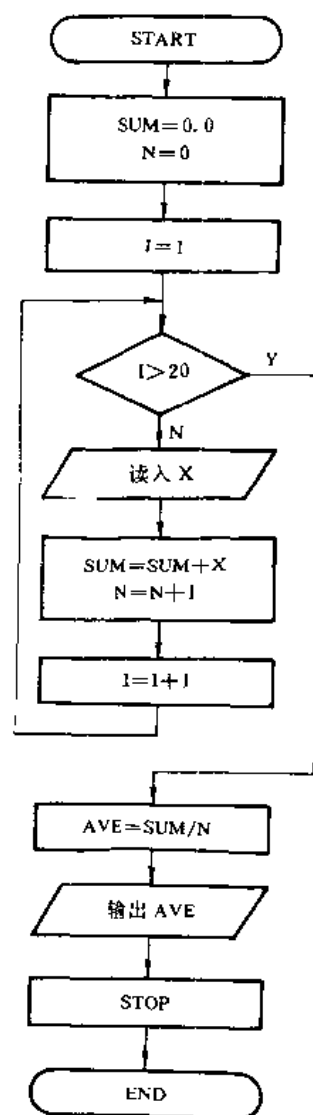


图 5.10

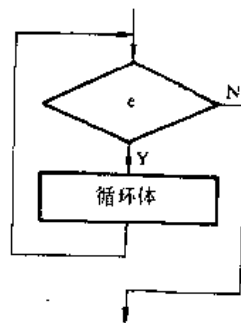


图 5.11

```

uprange, lowrange, INTEGER
BEGIN
    left_range := bit_range'LEFT; -- 得到 31
    right_range := bit_range'RIGHT; -- 得到 0
    uprange := bit_range'HIGH; -- 得到 31
    lowrange := bit_range'LOW; -- 得到 0
END PROCESS;

```

从例 5-22 可以看出, 不同的属性可以得到不同的关于数据类的信息。例如, 当数据类的区间用(a TO b)来定义时, 那么 $b > a$, 此时'LEFT 属性的值通常等于'LOW 属性的值。相反, 如果数据类的区间用(b DOWNT0 a)来定义时, 那么 $b > a$, 此时'LEFT 属性的值则与'HIGH 的属性值相对应。

数值类属性不光适用于数字类型, 而且该属性还可以适用于任何标量类型。用于枚举类型的情况如例 5-23 所示。

【例 5-23】

```

ARCHITECTURE time1 OF time IS
    TYPE tim IS(sec,min,hous,day,month,year);
    SUBTYPE reverse_tim IS
        tim RANGE month DOWNT0 min;
    SIGNAL tim1,tim2,tim3,tim4,tim5,tim6,tim7,tim8:TIME
BEGIN
    tim1<=tim'LEFT; -- 得到 sec
    tim2<=tim'RIGHT; -- 得到 year
    tim3<=tim'HIGH; -- 得到 year
    tim4<=tim'LOW; -- 得到 sec
    tim5<=reverse_tim'LEFT; -- 得到 min
    tim6<=reverse_tim'RIGHT; -- 得到 month
    tim7<=reverse_tim'HIGH; -- 得到 month
    tim8<=reverse_tim'LOW; -- 得到 min
END time1;

```

在例 5-23 中, 信号 tim1 和 tim2 代入的是 sec 和 year, 分别是区间的左端值和右端值。这一点很容易在类型说明中得到验证。但是, 如何来说明, 用'HIGH 和'LOW 属性来得到枚举类数据的数值属性呢? 实际上, 这里的'HIGH 和'LOW 表示的是数据类的位置序号值的大小。对于整数和实数来说, 数值的位置序号值与数本身的值相等, 而对于枚举类型的数据来说, 在说明中较早出现的数据, 其位置序号值低于较后说明的数据。例如, 在例 5-23 中 sec 的位置序号为 0, 因为它最先在类型说明中说明; 同样, min 位置序号为 1, hous 为 2 等等。这样, 位置序号大的, 其属性为'HIGH; 而位置序号小的, 其属性为'LOW。

信号 tim5 到 tim8 代入的是 reverse_tim 类数据的属性值。该类数据的区间用 DOWNT0 来加以说明。此时, 用属性'HIGH 和'RIGHT 得到的将不是同一个值(在用 TO 来说明区间时, 两者的属性值是相同的), 其原因就在于区间内的数据说明颠倒了。在例 5-23 中, 对 reverse_tim 数据类型来说, month 的位置序号大于 min 的位置序号。

2) 数组的数值属性

数组的数值属性只有一个, 即'LENGTH。在给定数组类型后, 用该属性将得到一个数组的长度值。该属性可用于任何标量类数组和多维的标量类区间的数组。例 5-24 就是一个简单应用的示例。

【例 5-24】

```
PROCESS(a)
    TYPE bit4 IS ARRAY (0 TO 3) OF BIT;
    TYPE bit_strange IS ARRAY(10 TO 20) OF BIT;
    VARIABLE len1,len2; INTEGER;
BEGIN
    len1 := bit4'LENGTH -- len1 = 4
    len2 := bit_strange'LENGTH; -- len2 = 11
END PROCESS;
```

在例 5-24 中, len1 代入的是数组 bit4 的元素个数; len2 代入的是数组 bit_strange 的元素个数。

该属性同样也可以用于枚举类型的区间, 如例 5-25 所示。

【例 5-25】

```
PACKAGE p_4val IS
    TYPE t_4val IS ('X','0','1','Z');
    TYPE t_4valx1 IS ARRAY(t_4val'LOW TO
        t_4val'HIGH) OF t_4val;
    TYPE t_4valx2 IS ARRAY (t_4val'LOW TO
        t_4val'HIGH) OF t_4valx1;
    TYPE t_4valmd IS ARRAY
        (t_4val'LOW TO t_4val'HIGH,
        t_4val'LOW TO t_4val'HIGH) OF t_4val;
    CONSTANT andsd:t_4valx2 :=
        (('X',      -- XX
         '0',      -- X0
         'X',      -- X1
         'X'),    -- XZ
         ('0',     -- 0X
          '0',     -- 00
          '0',     -- 01
          '0'),   -- 0Z
         ('X',     -- 1X
          '0',     -- 10
          '1',     -- 11
          'X'),   -- 1Z
         ('X',     -- ZX
          '0',     -- Z0
          'X',     -- Z1
          'X')),  -- ZZ
```

一、多重循环使用时的规定

① 循环的嵌套不得产生交叉。

例如下面的用法是不正确的：

```
      DO 10 I=1,20
        DO 20 J=1,5
          :
10     CONTINUE
20     CONTINUE
```

② 多重循环中，各个层的循环变量不得同名。

例如，下面用法是不正确的：

```
      DO 100 I=1,20
        DO 200 I=1,5
          :
200    CONTINUE
100    CONTINUE
```

③ 多重循环根据实际问题可共用一个终端语句。

例如，下面用法是正确的。

```
      DO 10 I=1,5
        DO 10 J=1,3
          :
10     CONTINUE
```

④ 各层的循环体都应是完整的，且均不允许从循环体外直接转向循环体内。

二、例子

例 5.19 利用二重循环给二维数组送数据。

```
      DIMENSION A(3,5)
      DO 10 I=1,3
        DO 20 J=1,5
          READ *,A(I,J)
20     CONTINUE
10     CONTINUE
      :
```

这个例子完成了二维数组的数据输入问题，请读者自己作一下三维数组的输入过程。

读者可以回顾一下，我们曾在例 5.10 和例 5.12 中用数组名来输入输出整个数组，这样做程序执行的效率比较高，因为只需执行一次输入(或输出)语句就可完成整个数组的输入(或输出)。但是其缺点是无法由程序员来控制输入(或输出)元素的个数，也无法由程序

员来指定输入(或输出)元素的顺序,因此也无法由程序员完全控制输入(或输出)的格式。

例 5.20 将 10 个数由大到小排序。

做此题的思路如下:

① 先定义一个数组 A 来存放输入的数据。
② 令 $I=1$, 将 $A(I)$ 与 $A(J)$ ($J=I+1, \dots, 10$) 依次进行比较, 若出现 $A(I) < A(J)$, 则将 $A(I)$ 与 $A(J)$ 交换, 这轮比较结束后 $A(I)$ ($I=1$) 中放着 10 个数中最大的一个。

③ 令 $I=2$, 重复②, 直到 $I=9$ 时, 已将 10 个数由大到小排好。

④ 输出 A 数组。

程序框图如图 5.14, 程序如下:

```

      DIMENSION A(1:10)
      DO 10 I=1,10
        READ *,A(I)
10    CONTINUE
      DO 20 I=1,9
        DO 20 J=I+1,10
          IF (A(I).GE. A(J)) GOTO 20
          X=A(I)
          A(I)=A(J)
          A(J)=X
20    CONTINUE
      DO 30 I=1,10
        PRINT *,A(I)
30    CONTINUE
      STOP
      END
  
```

这种排序方法通常称为比较排序法。

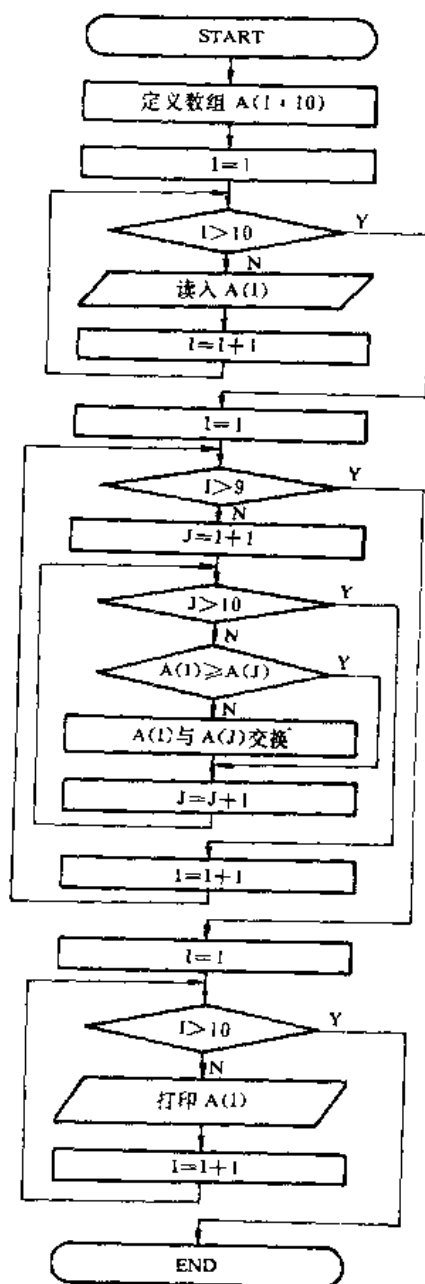


图 5.14

前面讨论数组的输入/输出时,采用的是 DO 循环结构,而 FORTRAN 77,对数组的输入/输出问题,提供了隐含 DO 循环方式。

一、隐 DO 表

隐 DO 表的形式是

(dlist, I=e1, e2 [, e3])

其中: $l, e1, e2, e3$ 同 DO 语句中所做的说明一样;

dlist 是输入/输出表, dlist 还可以包含隐 DO 表。输入/输出表中或是输入表项或是输出表项。输入表项必须是下列之一: 变量名、数组元素名、字符子串名和数组名; 而输出表项必须是下列之一: 变量名、数组元素名、字符子串名、数组名和表达式。

隐含 DO 循环的执行是按照 DO 循环的规则给 dlist 中的内容进行输入/输出处理。

例如 READ *, (A(I), I=1, 3)

相当于给 A(1)、A(2)、A(3) 通过键盘输入数据, 也相当于:

```
DO 10 I=1, 3
  READ *, A(I)
10 CONTINUE
```

当 dlist 中变量名和数组元素同时出现时, 应将它们视为每次循环的一个循环整体。

例如, READ *, (X, A(I), I=1, 3)

这时输入应按 X, A(1), X, A(2), X, A(3) 的顺序进行。

二、隐含 DO 循环的嵌套

隐 DO 表的嵌套中要遵循多重循环对循环嵌套的全部要求。

例如

```
READ *, ((A(I,J), J=1, 2), I=1, 2)
```

其读入顺序为 A(1, 1), A(1, 2), A(2, 1), A(2, 2), 相当于

```
DO 10 I=1, 2
  DO 10 J=1, 2
    READ *, A(I, J)
10 CONTINUE
```

在利用隐含 DO 循环作输入时, 应注意循环变量不能出现在 dlist 中。例如, 下面的用法是不允许的:

```
READ *, (I, A(I), I=1, 5)
```

而且若 $e1, e2, e3$ 为变量名, 则该变量也不允许出现在 dlist 中。例如, 下面的用法是不允许的:

```
READ *, (I, A(N), N=I, 10)
```

三、举例

例 5.21 设有矩阵如下:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \end{bmatrix}$$

求 $C = A * B$

矩阵 C 的构成元素可由公式

$$c_{ij} = \sum_{k=1}^4 a_{ik} * b_{kj}$$

得到, 显然 C 为 3 行 2 列的矩阵。

程序框图见图 5.15。

求解矩阵的程序如下:

```

      REAL A(3,4),B(4,2),C(3,2)
      READ *, ((A(I,J), J=1,4), I=1,3)
      READ *, ((B(I,J), J=1,2), I=1,4)
      DO 10 I=1,3
        DO 10 J=1,2
          C(I,J)=0.0
          DO 10 K=1,4
            C(I,J)=C(I,J)-A(I,K)*B(K,J)
10    CONTINUE
      WRITE(*,100)((A(I,J),J=1,4),I=1,3)
      WRITE(*,200)((B(I,J),J=1,2),I=1,4)
      WRITE(*,200)((C(I,J),J=1,2),I=1,3)
100  FORMAT(1X,4F10.2)
200  FORMAT(1X,2F10.2)
      STOP
      END

```

例 5.22 利用枚举法求解方程

$$x^3 + y^3 + z^3 = 8$$

在 $-2 \leq x \leq 2$, $0 \leq y \leq 3$, $-3 \leq z \leq 4$ 区间上所有的整数根。

由于求解的是整数根, 可用枚举法将 x, y, z 在其区间内的所有整数代入方程进行验证, 求得方程的整数根。程序框图如图 5.16。

```

      INTEGER X,Y,Z
      DO 10 X=-2,2
        DO 10 Y=0,3
          DO 10 Z=-3,4
            M=X**3+Y**3+Z**3
            IF (M.EQ.8) THEN
              PRINT *,X,Y,Z
            ENDIF
10    CONTINUE
      STOP
      END

```

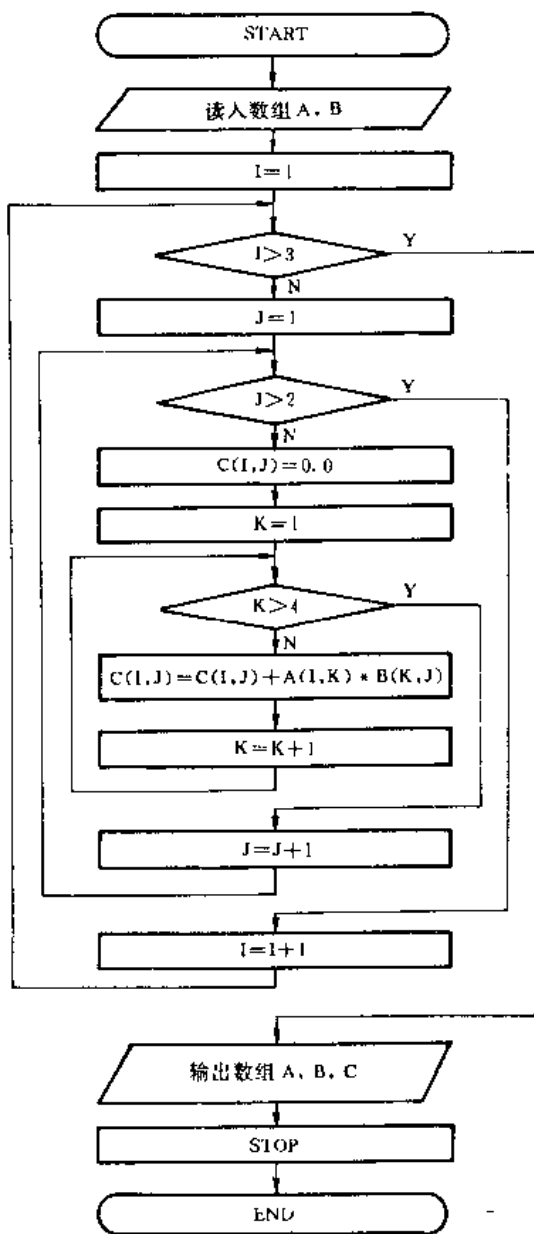


图 5.15

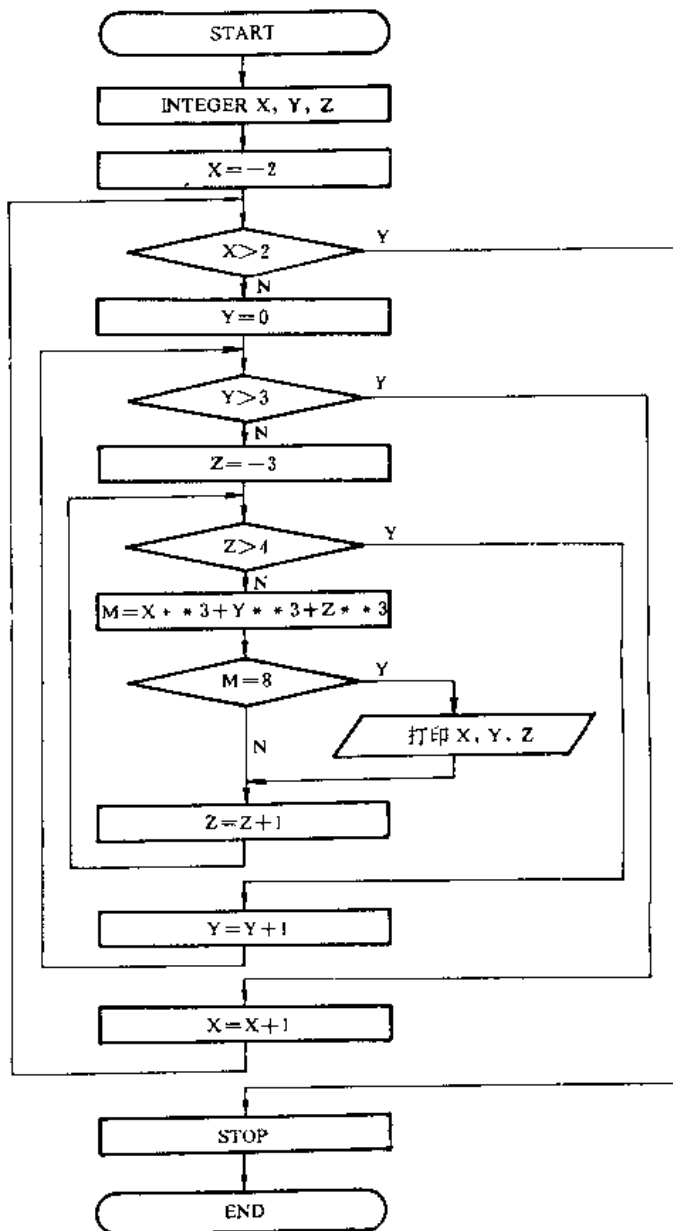


图 5.16

5.6 应用举例

例 5.23 用 100 元钱买 100 只鸡，小鸡每 3 只 1 元，公鸡每只 5 元，母鸡每只 3 元。要求每种鸡至少买一只，且必须是整只的，问各种鸡各买多少只？

解 这是一个组合问题。令 i 、 j 、 k 分别是公鸡、母鸡和小鸡的只数。为了确定 i 、 j 、 k 的范围，可以有不同的方法，我们可以看到，不同的方法，其计算量可能相差甚远。

[方法一]

由于要保证每种鸡至少买一只，因而 i 、 j 、 k 的范围分别为：

i : 1~18 (公鸡至多买 18 只)

j : 1~31 (母鸡至多买 31 只)

$k=100-i-j$ (i 和 j 确定之后，小鸡的数量也就确定了)

用二重循环实现，程序如下：

```

DO 100 I=1,18
DO 200 J=1,31
  K=100-I-J
  IF(5*I+3*J+K/3.NE.100) GOTO 200
  PRINT *, I, J, K
200  CONTINUE
100  CONTINUE
END

```

计算结果为：

3	20	77
4	18	78
7	13	80
8	11	81
11	6	83
12	4	84

读者不难发现，在这六组解中第1组、第3组和第5组并不是解，因为这三组解的花费是100.67元。究其原因在于我们用了整除K/3。如果把程序略改一下：

```

DO 100 I=1,18
DO 200 J=1,31
  A=100-I-J
  IF(5*I+3*J+A/3.NE.100) GOTO 200
  PRINT *, I, J, A
200  CONTINUE
100  CONTINUE
END

```

计算结果为：

4	18	78.000 000
8	11	81.000 000
12	4	84.000 000

在这个程序中，循环体被执行 $18 \times 31 = 558$ 次。

[方法二]

由题意可得到下列方程组：

$$\begin{aligned} i+j+a &= 100 \\ 5i+3j+a/3 &= 100 \end{aligned}$$

为了避免在方法一曾经出现过的问题，我们取小鸡的数量为实型变量。由上述方程组可得

$$14i+8j=200$$

由于 i 和 j 至少为1，实际上 i 最多为13， j 最多为23，这样可得到如下程序：

```

DO 100 I=1,13
DO 200 J=1,23
  A=100-I-J

```

```

        IF(5*I+3*J+A/3.NE.100) GOTO 200
        PRINT *, I, J, A
200    CONTINUE
100    CONTINUE
END

```

在该程序中, 循环体被执行 $13 \times 23 = 299$ 次, 略大于方法一的一半。

[方法三]

在方法二中的方程

$$7i + 4j = 100$$

可得

$$j = (100 - 7i) / 4$$

因此, 可用一重循环编出程序如下:

```

DO 100 I=1, 13
    J=25-7*I/4
    A=100-I-J
    IF(5*I+3*J+A/3.NE.100) GOTO 100
    PRINT *, I, J, A
100    CONTINUE
END

```

在该程序中, 循环体只执行了 13 次, 不足方法二的 5%。

因此, 从此例可以看出, 正确地确定变量的取值范围, 即搜索范围, 对于减少计算量有很大的影响。同时, 恰当地选用整型变量和实型变量也是很重要的。

例 5.24 有一个数字谜, A, B, C, D 各代表 0~9 中的数, 四个数之间有如下关系:

$$\begin{array}{r}
 \text{A B C D} \\
 -) \quad \text{C D C} \\
 \hline
 \text{A B C}
 \end{array}$$

试求 A, B, C, D 各是几?

解 根据题意可知, A, B, C, D 为 0~9 内的数字, 且 $A=1$, $C \neq 0$, 由此可确定搜索范围为:

A=1
 B: 0~9
 C: 1~9
 D: 0~9

我们对 B, C, D 依次增加 1, 从最小可能取值直到最大可能取值, 这种搜索方法称为盲目搜索。采用盲目搜索的程序如下:

```

INTEGER A, B, C, D, X, Y
A=1
DO 100 B=0,9
DO 100 C=1,9
DO 100 D=0,9

```

```

X=1000*A+100*B+10*C+D
Y=100*C+10*D+C
IF(X-Y.EQ.100*A+10*B+C) THEN
  PRINT *, A, B, C, D
ENDIF
100 CONTINUE
END

```

此程序执行结果为

```
1    0    9    8
```

即

A=1, B=0, C=9, D=8

例 5.25 有这样一个数字谜

```

      TEN
      TEN
+ )   FORTY

```

SIXTY

在这道算式中有 10 个字母，它们各代表 0~9 中不同的数字，试求解此数字谜。

解 此题可用逻辑判断推导解之，在此用盲目搜索法解之。为了减少搜索次数，我们可以做些初步判断。不难看出，E 和 N 只能取 0 或 5，所以其它字母就不必从 0 开始搜索，而从 1 开始搜索，S=F+1，于是 F 不可能为 9。编制程序如下：

```

INTEGER T,E,N,F,O,R,Y,S,I,X,U,V
DO 10 T=1,9
DO 10 E=0,5,5
  IF(T.NE.E) THEN
    DO 20 N=0,5,5
      IF(N.NE.E.AND.N.NE.T) THEN
        DO 30 F=1,8
          IF(F.NE.N.AND.F.NE.E.AND.F.NE.T) THEN
            DO 40 O=1,9
              IF(O.NE.F.AND.O.NE.N.AND.O.NE.E.AND.O.NE.T) THEN
                DO 50 R=1,9
                  IF(R.NE.O.AND.R.NE.F.AND.R.NE.N.AND.R.NE.E.AND.R.NE.T) THEN
                    DO 60 Y=1,9
                      IF(Y.NE.R.AND.Y.NE.O.AND.Y.NE.F.AND.Y.NE.N
C                      .AND.Y.NE.E.AND.Y.NE.T) THEN
                        DO 70 I=1,9
                          IF(I.NE.Y.AND.I.NE.R.AND.I.NE.O.AND.I.NE.F
C                          .AND.I.NE.N.AND.I.NE.E.AND.I.NE.L) THEN
                            DO 80 X=1,9
                              IF(X.NE.I.AND.X.NE.Y.AND.X.NE.R.AND.X.NE.O
C                              .AND.X.NE.F.AND.X.NE.N.AND.X.NE.E.AND.X.NE.T) THEN

```

```

      S=F+1
      IF(S. NE. X. AND. S. NE. I. AND. S. NE. Y. AND. S. NE. R. . AND. S.
C          NE. O. AND. S. NE. N. AND. S. NE. E. AND. S. NE. T) THEN
          U=200 * T+20 * E+2 * N+10000 * F+10000 * O+100 * R+10 * T+Y
          V=10000 * S+1000 * I+100 * X+10 * T+Y
      IF(U. EQ. V)THEN
          WRITE(* ,100)T,E,N,F,O,R,Y,S,I,X
100      FORMAT(1X,'T=', I1,2X,'E=',I1,2X,'N=',I1,2X,'F=',I1,2X,'O=', I1,2X,'
C          R=',I1,2X,'Y=',I1,2X,'S=', I1,2X,'I=',I1,2X,'X=',I1)
          ENDIF
          ENDIF
          ENDIF
80      CONTINUE
          ENDIF
70      CONTINUE
          ENDIF
60      CONTINUE
          ENDIF
50      CONTINUE
          ENDIF
40      CONTINUE
          ENDIF
30      CONTINUE
          ENDIF
20      CONTINUE
          ENDIF
10      CONTINUE
      END

```

此程序执行结果为：

T=8, E=5, N=0, F=2, O=9, R=7, Y=6, S=3, I=1, X=4。

例 5.26 水仙花数问题。水仙花数是一个三位数，它的三个数字的立方和正好与这个三位数相等。例如， $153=1^3+5^3+3^3$ ，因此 153 是一个水仙花数。试编程序求出所有的水仙花数。

解 设一个三位数 L 的百位数、十位数和个位数分别为 I、J、K，显然有：

$$\begin{aligned} 1 &\leq I \leq 9 \\ 0 &\leq J \leq 9 \\ 0 &\leq K \leq 9 \end{aligned}$$

可用两种方法求解。

[方法一]

先由 $L(100 \leq L \leq 999)$ 分别求出其各位的数字 I、J、K，再判断是否有 $I^3 + J^3 + K^3 = L$ 。

程序如下：

```

DO 10 L=100, 999
  I=L/100
  J=MOD(L, 100)/10
  K=MOD(L, 10)
  IF (I * 3 + J * 3 + K * 3.EQ. L) THEN
    PRINT *, L
  ENDIF
10 CONTINUE
END

```

[方法二]

用排列的方法将 I、J、K 组成三位数，判断是否有 $I^3 + J^3 + K^3 = L = 100I + 10J + K$ 。程序如下：

```

DO 10 I=1, 9
  I1=I * 3
  I2=100 * I
DO 10 J=0, 9
  J1=J * 3
  J2=10 * J
DO 10 K=0, 9
  L=I2+J2+K
  IF(I1+J1+K * 3.EQ. L) THEN
    PRINT *, L
  ENDIF
10 CONTINUE
END

```

这两种方法执行循环体均为 900 次，但方法二的计算量略小于方法一。程序运行结果为：

```

153
370
371
407

```

例 5.27 假如有一头母牛，它每年年初生一头小母牛，每头小母牛从第 4 年年头起，每年年初也生一头小母牛，问在第 15 年母牛的头数共有多少？（假定年底有一头大母牛，从次年年初开始计算。）

解 根据题意可列下表：

年 数	1	2	3	4	5	6	...
大母牛数	1	1	1	2	3	4	...
小母牛数	1	2	3	4	6	9	...
总 数	2	3	4	6	9	13	...

由上表可得到该问题的数学模型为：

$$L(n) = \begin{cases} 1 & n=0 \\ 2 & n=1 \\ 3 & n=2 \\ 4 & n=3 \\ L(n-1)+L(n-3) & n>3 \end{cases}$$

这实际上是一个求数列公项的问题，其递推公式为：

$$L(n) = L(n-1) + L(n-3) \quad n > 3$$

其中， $L(n-1)$ 为第 n 年的小母牛数， $L(n-3)$ 为第 n 年的大母牛数。所以，当 $n=15$ 时， $L(15)$ 即为所求的结果。显然，反复利用递推公式即可得到 $L(15)$ 。程序如下：

```

      DIMENSION L(15)
      L(1)=2
      L(2)=3
      L(3)=4
      DO 10 I=4, 15
        L(I)=L(I-1)+L(I-3)
10    CONTINUE
      PRINT *, L(15)
      END

```

计算结果为 406。

例 5.28 1982 年，我国总人口为 10 亿 3 000 万，设定人口增长率分别为 3%，2.5%，2%，1.5%，1%，0.5% 时，到 2000 年，我国的总人口将达到多少？按我国国土 960 万平方公里计算，人口密度分别是多少？

解 设 $N=1982$ ， $M=2000$ ， P_N 为 1982 年的人口数， P_M 为 2000 年的人口数， r 为人口增长率，则有

$$P_M = P_N(1+r)^{M-N}$$

为增加程序的实用性，参数便于改变，我们把基准年份 1982 及人口基数 10.3E8 存放在符号常数 N 和 P_N 中， M 由键盘读入，国土面积为 $A(\text{km}^2)$ ，程序依次打印出年份，人口增长率、总人口及人口密度。程序如下：

```

      PARAMETER(N=1982, DN=10.3E8)
      READ *, M
      A=96E5
      R=0.03
      DO 10 I=1, 6
        P=PN
      DO 20 J=1, M-N
20    P=P*(1+R)
      WRITE(*, 100) M, R, P, P/A
100  FORMAT(1X, 'M=', 14, 3X, 'R=', F4.3, 3X, 'P=', F13.2, 3X, 'P/A=', F8.4)
10    R=R-0.005
      END

```


执行结果为:

M=2 000	R=.030	P=1 753 505 000.00	P/A=182.656 8
M=2 000	R=.025	P=1 606 447 000.00	P/A=167.338 3
M=2 000	R=.020	P=1 471 093 000.00	P/A=153.238 9
M=2 000	R=.015	P=1 346 561 000.00	P/A=140.266 8
M=2 000	R=.010	P=1 232 031 000.00	P/A=128.336 6
M=2 000	R=.005	P=1 126 747 000.00	P/A=117.369 5

例 5.29 大数阶乘算法。一个较大的数 N , 其阶乘 $N!$ 的有效数字是很多的(例如 $9! = 362\,880$, $15! = 1\,307\,674\,368\,000$), 但一般计算机所能保留的有效数字是非常有限的。例如, 采用四字节的整型数, 也只能求到 $12! = 479\,001\,600$ 。对于大数阶乘如何得到充分多的有效数字呢? 这里给出一种算法。其办法是将阶乘值按其各位数字单独依次存放, 因此可以作如下考虑:

第一步, 根据阶乘的定义, 有

$$N! = N \cdot (N-1)!$$

假设 $(N-1)!$ 共有 k 位数字, 即 $(N-1)!$ 可以写成

$$L_k L_{k-1} L_{k-2} \cdots L_1$$

其中, 每个 $L_i (i=1, 2, \dots, k)$ 取值均为 $0 \sim 9$, 且每两个相邻的 L_i 之间相差 10 倍。如果把 k 个 L_i 分别存放起来, 在进行求 $N!$ 运算时, 可以将 N 与每个 L_i 分别相乘, 相乘的结果也单独按位存放, 最后将各位数字依次打印出来, 就可得到 $N!$ 的结果。例如:

若已求出 $11! = 39\,916\,800$, 结果共八位, 即

$$L_8 = 3$$

$$L_7 = 9$$

$$L_6 = 9$$

$$L_5 = 1$$

$$L_4 = 6$$

$$L_3 = 8$$

$$L_2 = 0$$

$$L_1 = 0$$

现在可用按位乘法来求 $12!$, 即

$$L_1 = L_1 \times 12 = 0 \times 12 = 0$$

$$L_2 = L_2 \times 12 = 0 \times 12 = 0$$

$$L_3 = L_3 \times 12 = 8 \times 12 = 96, \text{ 取 } L_3 = 6, \text{ 进位为 } 9$$

$$L_4 = L_4 \times 12 + \text{进位} = 6 \times 12 + 9 = 81, \text{ 取 } L_4 = 1, \text{ 进位为 } 8$$

$$L_5 = L_5 \times 12 + \text{进位} = 1 \times 12 + 8 = 20, \text{ 取 } L_5 = 0, \text{ 进位为 } 2$$

$$L_6 = L_6 \times 12 + \text{进位} = 9 \times 12 + 2 = 110, \text{ 取 } L_6 = 0, \text{ 进位为 } 11$$

$$L_7 = L_7 \times 12 + \text{进位} = 9 \times 12 + 11 = 119, \text{ 取 } L_7 = 9, \text{ 进位为 } 11$$

$$L_8 = L_8 \times 12 + \text{进位} = 3 \times 12 + 11 = 47, \text{ 取 } L_8 = 7, \text{ 进位为 } 4$$

最后将进位存放在 L_9 中, 即 $L_9 = 4$ 。

按照顺序将 L_i 打印出来, 有 $L_9 L_8 L_7 L_6 L_5 L_4 L_3 L_2 L_1$, 即

$$12! = 479\,001\,600$$

也就是说,按这种算法来求阶乘,对数的大小没有限制,只要准备足够多的存贮单元来存放 L_i 即可。

第二步,考虑如何求得每个 L_i 及如何存放每个 L_i 。

根据第一步的分析,我们知道,要计算 $N!$ 的各个 L_i ,只要将 $(N-1)!$ 的各个 L_i 乘上 N 即可。但由于每个 L_i 只能是一位数,因此计算中要把低位的进位(可能是 0、一位数或几位数)以及向高位的进位(可能是 0、一位数或几位数)考虑进去,于是得到下列的计算公式:

(1) 计算 $L_i * N$, 并考虑低位来的进位 IC

$$IC = L_i * N + IC$$

(2) 计算新的 L_i

$$L_i = \text{MOD}(IC, 10)$$

(3) 计算向高位的进位

$$IC = IC / 10$$

至于每个 L_i 的存放,由于各个 L_i 有严格的顺序要求,因而我们用数组来存放。

从以上两个步骤,可以看出,要求 N 的阶乘,可从 1 的阶乘开始,向上一步一步地扩展。开始,由于 0 和 1 的阶乘都等于 1,于是,可以设定最低位的初值 $L(1)=1$, 进位初值 $IC=0$, 位数初值 $K=1$ 。在计算 L_i 时,先计算上次阶乘得到的 k 位,当 k 位容纳不下,则通过 $k=k+1$ 来扩充。程序如下:

```

      DIMENSION L(100)
      READ *, N
      L(1)=1
      IC=0
      K=1
      IF(N, GE. 0) THEN
        DO 20 I=1, N
          DO 10 J=1, K
            IC=IC+I*L(J)
            L(J)=MOD(IC, 10)
            IC=IC/10
10          CONTINUE
15          IF(IC, NE. 0) THEN
            K=K+1
            L(K)=MOD(IC, 10)
            IC=IC/10
            GOTO 15
          ENDIF
20          CONTINUE
          ELSE
            PRINT *, 'ERROR'
            STOP
          ENDIF

```

```

        WRITE(*,100)N
100    FORMAT(1X,'FACTORIAL OF',I4,1X,'IS')
        WRITE(*,200)(L(I),I=K,1,-1)
200    FORMAT(1X,30I1)
        END

```

此程序执行中如果我们读入 18, 则结果为:

```

        FACTORIAL OF 18 IS
        6402373705728000

```

如果读入 -6, 则结果为

```

        ERROR
        STOP - Program terminated

```

习 题 五

1. 解释下列名词:

数组, 数组名, 数组元素, 数组元素名, 下标变量。

2. 数组 A 和 B 的内容如下表, 根据下图, 指出下列元素的值。

A 数组

1	2	3	4	5	6	7	8	9	10		60	61	62	63
3.50	4.25	5.67	10.30	-8.4	5.70	13.40	21.50	62.30	34.50		33.40	18.80	20.10	50.30

B 数组

1	2	3	4	5	6	7	8	9	10	11	12
3	6	5	7	12	10	8	1	9	20	13	16

并且 $I=4$

(1) $B(4)$, $A(8)$, $A(60)$, $B(10)$, $A(I+1)$, $B(I-1)$, $A(2*I-3)$, $B(3+1)$

(2) $A(I)$, $A(B(I))$, $A(B(B(I))+1)$

3. 下列循环共执行多少次, 并写出运行结果:

(1)

```

DO 10 I=1, 10
  PRINT *, I

```

100 CONTINUE

(2) 若增值参数改为 3 时:

(3) 若终值参数改为 -10 时。

4. 指出下列 DO 循环有哪些错误:

(1)

```
      DO 10 X=0.5, 3.5, -0.5  
      :  
10    CONTINUE
```

(2)

```
      DO 10 I=1, 10, 0  
      :  
10    CONTINUE
```

(3)

```
      DO 10 I=10, 1  
      :  
10    CONTINUE
```

5. 指出下列输出结果及循环次数:

(1)

```
      DO 10 M=1,3  
      DO 20 N=1,2  
      PRINT *, M,N  
20    CONTINUE  
10    CONTINUE  
      STOP  
      END
```

(2)

```
      DO 10 I=1,2  
      PRINT '(1X,I2)', I  
      DO 20 J=1,3  
      PRINT '(1X,5X,I2)', J  
      DO 30 K=1,4  
      PRINT '(1X,10X,I2)', K  
30    CONTINUE  
20    CONTINUE  
10    CONTINUE  
      STOP  
      END
```

(3)

```
      N=0  
      DO 10 I=1,5  
      L=I  
      DO 20 J=1,4  
      K=J  
      N=N+1
```

```

20      CONTINUE
10      CONTINUE
      PRINT '(1X,2X,2HI=,I3,2X,2HJ=,I3,2X,2HK=,I3,2X,2HL=,I3,2X
      * 2HN=,I3)',I,J,K,L,N
      STOP
      END

```

6. 指出下列循环嵌套有什么错误:

(1)

```

      DO 10 I=1,10
          DO 20 J=1,5
              A(I)=B(J)
10      CONTINUE
20      CONTINUE

```

(2)

```

      DO 20 I=1,5
          DO 10 J=2,6
              DO 30 K=6,2,-1
                  PRINT *,I,J,K
30      CONTINUE
20      CONTINUE
10      CONTINUE

```

7. 编写以下三题的程序:

(1) 求 $\frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \frac{1}{3 \cdot 4} + \cdots + \frac{1}{n(n+1)}$, $n=100$

(2) 求 π 的近似值公式为

$$\frac{\pi}{2} = \frac{2 \cdot 2}{1 \cdot 3} \times \frac{4 \cdot 4}{3 \cdot 5} \times \frac{6 \cdot 6}{5 \cdot 7} \times \cdots \times \frac{(2n)^2}{(2n-1)(2n+1)}$$

当 $n=100$ 时, 求 π 的近似值。

(3) 用台劳多项式求 $\sin X$ 的近似值

$$\sin X \approx \frac{x}{1} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots + (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!}$$

8. 设数组 A 存有数据 10, 20, 30, 40, 数组 B 存有数据 5, 15, 25, 35, 指出下列执行结果是什么?

(1) PRINT *, (A(I), I=1, 4)

PRINT *, (B(I), I=1, 4)

(2) PRINT *, (A(I), I=1, 4), (B(I), I=1, 4)

(3) PRINT *, (A(I), B(I), I=1, 4)

(4)

DO 10 J=1, 2

PRINT *, J, (A(I), I=1, 4)

10 CONTINUE

9. 设数组 A 有 15 个元素, 数组 B 有 23 个元素, 将这两个数组中具有相同值的那些数组元素打印出来。试编制程序。

10. 设有一 5×5 矩阵, 用数组 A(5, 5) 读入该矩阵(数据自定), 求:

(1) 矩阵中的最大元素;

(2) 求矩阵中负值元素之和;

(3) 求对角线元素之积;

(4) 求每行元素之和, 并打印出和值最大的那行的行号。

11. 编一程序, 将两个大小相同的矩阵相加。

12. 编一程序, 完成对正方形矩阵的转置(第一行和第一列的元素一一对调, 第二行和第二列的元素一一对调, ……)

13. 求矩阵中的一个元素, 使其值为各行最大中的最小者。试编一程序。

14. 阿姆斯特朗数(Armstrong number)是符合条件: 一个数等于其每一个数字的立方和的数。例如 $4^3 + 0^3 + 7^3 = 407$, 则 407 为一阿姆斯特朗数, 编写一程序求 1~2 000 之间的阿姆斯特朗数。

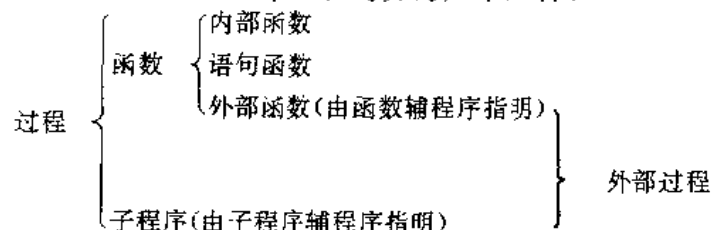
15. 用牛顿迭代法求 $x + e^x = 0$ 在 $x=0$ 附近的一个近似根。

16. 一个数若出现在它的平方数的右端, 则称该数为同构数。例如 5 的平方为 25, 5 是 25 中右端的数字, 则 5 为一同构数。同理可知, 25 也为一同构数。编写一程序。求 1~100 之间的所有同构数。

6 过 程

6.1 过程的概念及分类

在日常工程设计、科学计算以及事务处理中,经常涉及到一些功能相对独立,处理过程大致相同的问题。例如求 $n!$, 线性代数方程组的求解, 矩阵的运算, 非线性方程(组)的求解, 以及数据处理中的查找、排序等。有时即使在某一个问题的求解过程中, 也可能前后多次涉及到某一表达式的计算或多次用到某一算法。例如求 $M! + N! + K!$, 或求某一个圆环的面积(圆环的面积等于外圆的面积减去内圆的面积)。为了使程序的逻辑结构比较清晰, 便于阅读、理解和调试, 便于程序功能的扩充和修改, 减少程序的书写量, 提高编程的效率, 我们把程序中那些逻辑功能比较独立的部分分离出来, 编写成一个个程序模块。上一级程序通过调用相应的功能模块, 以实现其特定的操作。这就使整个程序的结构变成一种积木式结构, 极大地方便了系统功能的维护。当要增加某一功能时只需增加一个相应的功能模块, 而要删除某一功能时, 只需删除相应的功能模块即可。这样编写的模块程序称为过程。所谓过程, 就是求得问题解答的一系列操作。在算法语言中就是完成某一特定功能的语句序列集合。在 FORTRAN77 中过程可分为如下几种:



其中, 内部函数是由编译系统提供的。而语句函数、函数辅程序和子程序辅程序由程序员来定义。

在 FORTRAN 中每个辅程序(函数辅程序, 子程序辅程序, 数据块辅程序)都编制成一个独立的程序单位。它们可分别编制, 独立调试。辅程序(也称外部过程)可以被主程序调

用,也可以被其它外部过程调用,但不允许直接或间接地调用自己。即在 FORTRAN 程序的调用过程中允许嵌套,例如主程序 A 可以调用辅程序 B,而辅程序 B 又可以调用辅程序 C。其结构如图 6.1 所示。

在调用一个程序的过程中又直接或间接地调用了该段程序本身,称为程序的递归调用,如图 6.2 及图 6.3 所示。在 FORTRAN 77 中不允许递归调用,这是 FORTRAN 77 的一个缺陷。正因为如此,在 Fortran 90 中做了改进。Fortran 90 是允许递归调用的。

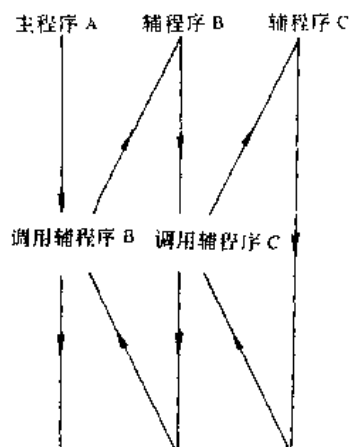


图 6.1

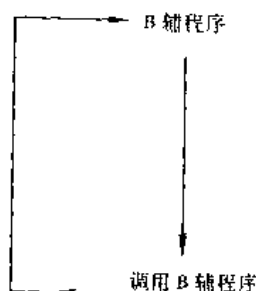


图 6.2

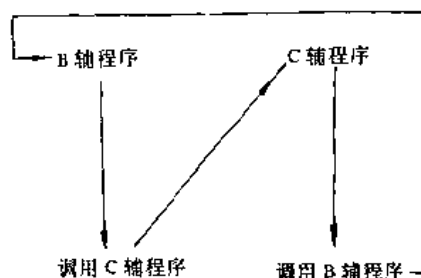


图 6.3

在 FORTRAN 中,各种变量的作用域仅局限于其定义的程序单位,包括辅程序单位中所定义变量的作用域也只在该辅程序中,因而此类变量称为局部变量。而主程序名,函数辅程序名,子程序辅程序名,数据块辅程序名都是全局名。它们的作用域是整个程序,必须互不相同,也不能与任何程序单位中定义的局部名相同。

6.2 语句函数

一、语句函数的定义

当在一个程序中多次用到同一表达式的运算时,我们希望能把这一表达式定义成一个函数,该表达式一经定义后,就能像内部函数那样被引用。为此 FORTRAN 提供了语句函数语句,其格式如下:

$$\text{fun}([d[, d]\cdots]) = e$$

其中: fun 是语句函数的符号名。它的命名方法与变量命名的方法相同,类型也遵循 I-N 规则。若不遵循此规则,可用类型说明语句加以说明。例如要将表达式 $4x^3 - 2x \cdots^2 + 5$ 定义成一个自定义函数,则可在程序的说明部分写出如下语句:

$$P(X) = 4 * X * * 3 - 2 * X * X + 5$$

若程序中未对函数名 P 作说明,则该函数名隐含为实型。若要将此函数名改为整型,则可做如下说明:

INTEGER P

$$P(X) = 4 * X * * 3 - 2 * X * X + 5$$

语句函数的符号名不能与本程序单位中的任何其它名字相同。

d 代表语句函数的自变量,称为语句函数的虚元。虚元表只用来指出函数变元的个数、次序和类型。虚元的类型遵循 I-N 规则,也可用类型说明语句说明。在一个语句函数的虚元表中,虚元不能同名。虚元的作用域仅是本语句函数语句。它可以与同一程序单位中其它地方出现的同类型变量或虚元具有相同的名字。当虚元的个数为 0 时,语句函数的形式为

$$\text{fun}() = e$$

e 表示语句函数所代表的表达式(算术表达式,逻辑表达式或字符串表达式)。即用户期望定义的表达式。在该表达式内可以包含常数、变量、语句函数的虚元以及函数(外部和内部函数),也可以含有已定义过的语句函数。但不允许出现本语句函数的名字。

例如求一元二次方程式 $ax^2 + bx + c = 0$ 的一个实根的表达式 $x_1 = (-b + \sqrt{b^2 - 4ac}) / (2a)$ 可用语句函数语句定义如下:

$$\text{ROOT}(A, B, C) = (-B + \text{SQRT}(B * B - 4.0 * A * C)) / (2.0 * A)$$

其中, A、B、C 为虚元。

二、语句函数的引用

语句函数一旦被定义后,就可以在同一程序单位中如同内部函数一样被引用。在希望得到函数值的地方写出函数名及其相应的实元,则计算机就会通过虚实结合把实元的值传递给虚元,再按照函数语句所定义的表达式计算函数值,并将计算结果通过函数名返回。

例如,在前边定义了对一元二次方程求根的函数,当要求方程 $1.2x^2 + 3.5x - 4 = 0$ 的一个根时,就可在程序中直接调用已定义好的求根函数 ROOT:

$$A = 1.2$$

$$B = 3.5$$

$$C = -4$$

$$x1 = \text{ROOT}(A, B, C)$$

在这里 $\text{ROOT}(A, B, C)$ 就是调用语句。A、B、C 称为实元。在自定义函数调用语句中,实元可以是已赋过值的变量、常量,或具有确定值的表达式。即上述调用语句也允许是

$$x1 = \text{ROOT}(1.2, 3.5, -4)$$

或者 $x1 = \text{ROOT}(2.4/2, 5 - 1.5, -2 * 2)$

在使用语句函数时要注意:

① 语句函数语句属于非执行语句,它在程序单位中的位置必须位于该程序单位中所有的说明语句之后,且位于第一条可执行语句之前。

② 虚元的主要作用是标明所定义函数参数的类型,个数和位置关系等。其作用域仅限于本语句。因而在本程序单位中,其它地方的标识符可以与之同名,但两者的类型是一致的。

③ 表达式的类型应与函数的类型一致。当两者都是算术型时,如果表达式计算的结果与函数名的类型不一致,计算机会自动先把计算结果转化为函数名的类型再赋给它。

④ 在引用时虚元与实元是按位置一一对应结合的。因而实元的个数、类型、次序应与虚元对应一致。当无虚元时调用语句形式为

fun()

三、例子

例 6.1 求方程 $a_1x^2 + a_2x + a_1 + a_2 = 0$ 的两个根, 其中, a_1, a_2 为方程 $1.2x^2 + 3.5x - 4 = 0$ 的两个根。

解 根据题意, 在计算方程根中要反复用到一元二次方程的求根公式。因此可以把求根公式 $x_1 = (-b + \sqrt{b^2 - 4ac})/2a$, $x_2 = (-b - \sqrt{b^2 - 4ac})/2a$ 分别定义成二个语句函数。

程序及运行结果如下:

```

C          EXAMPLE 6.1
          ROOT1(A,B,C)=(-B+SQRT(B*B-4*A*C))/2/A
          ROOT2(A,B,C)=(-B-SQRT(B*B-4*A*C))/2/A
          A=1.2
          B=3.5
          C=-4
          A1=ROOT1(A,B,C)
          A2=ROOT2(A,B,C)
          D=A1+A2
          X1=ROOT1(A1,A2,D)
          X2=ROOT2(A1,A2,D)
          WRITE(6,100)X1,X2
100      FORMAT(1X,2F10.2)
          END
  
```

运行结果为

4.99 - .67

在上述程序中, A_1, A_2 表示原方程的两个根。由于实元允许是表达式, 故上述程序还可改写成:

```

C          EXAMPLE 6.2
          ROOT1(A,B,C)=(-B+SQRT(B*B-4*A*C))/2/A
          ROOT2(A,B,C)=(-B-SQRT(B*B-4*A*C))/2/A
          A=1.2
          B=3.5
          C=-4
          X1=ROOT1(ROOT1(A,B,C),ROOT2(A,B,C),ROOT1(A,B,C)
$              +ROOT2(A,B,C))
          X2=ROOT2(ROOT1(A,B,C),ROOT2(A,B,C),ROOT1(A,B,C)
              +ROOT2(A,B,C))
          WRITE(*,100)X1,X2
100      FORMAT(1X,2F10.2)
          END
  
```

在上例中假定要求解的方程具有实根。

例 6.2 计算:

$$F(X, Y, Z) = x^2 + \sqrt{1 + 2x^2} + \frac{y}{y^4 + \sqrt{1 + 2y^4}} + \frac{1}{\sin^2 Z + \sqrt{1 + 2 \sin^2 Z}}$$

解 我们发现这三个算式都具有 $x^2 + \sqrt{1 + 2x^2}$ 形式的计算。若我们把这一算式定义为

$$S(x) = x^2 + \sqrt{1 + 2x^2}$$

则

$$F(x, y, z) = S(x) + \frac{y}{S(Y^2)} + \frac{1}{S(\sin Z)}$$

程序及运行结果如下:

```

C      EXAMPLE 6.2
      S(X)=X*X+SQRT(1.0+2.0*X*X)
      F(X,Y,Z)=S(X)+Y/S(Y*Y)+1./S(SIN(Z))
      WRITE(*,100)
100    FORMAT(1X,'ENTER X, Y AND Z:')
      READ(*,110)X,Y,Z
110    FORMAT(3F8.2)
      P=F(X, Y, Z)
      WRITE(*,200)X,Y,Z,P
200    FORMAT(1X,'X=', F8.2,2X,'Y=', F8.2,2X,'Z=', F8.2,2X,'P=', F8.2)
      END

```

运行结果如下:

```

ENTER X, Y AND Z:
      320      410      201
X=      3.20  Y=      4.10  Z=      2.01  P=      15.30

```

6.3 外部函数

上一节介绍的语句函数,在实际使用时受到一定限制。例如,在求多项式 $P(x) = x^3 - 3x^2 + 2x + 1$ 的值时用语句函数是合适的。但对于下面的函数关系式

$$y(x) = \begin{cases} x^2 + 1 & x > 0 \\ 2 & x = 0 \\ x^2 - 1 & x < 0 \end{cases}$$

就无法用语句函数来表达。因为语句函数必须在一个逻辑行上写完。这时我们就可以用前面 6.1 节中提到的函数辅程序来定义以上函数。

```

FUNCTION Y(X)
  IF (X.GT.0.0) THEN
    Y=X*X+1.0
  ELSE IF (X.EQ.0.0) THEN
    Y=2
  ELSE
    Y=X*X-1.0
  END IF
END

```

一、外部函数的定义

FORTRAN 允许用户定义一些外部函数。外部函数是以 FUNCTION 语句开头，以 END 语句结束的程序单位。它的一般形式是

```
[typ] FUNCTION fun ([d[, d]...])
...
...
:      函数辅程序体
...
END
```

其中：

① [typ] FUNCTION fun([d[, d]...])称为 FUNCTION 语句。

fun 是外部函数名，其取名规则与变量名相同，typ 是任选的函数类型说明，可以是 INTEGER, REAL, LOGICAL, CHARACTER, DOUBLE PRECISION, COMPLEX 之一；当无类型说明时，fun 遵循 I-N 规则；d 是虚元，可以是变量名、数组名或虚拟过程名。

该语句的作用是：表示函数辅程序单位的开始，并指出函数的名字、类型，虚元的名字，个数及顺序。

② FUNCTION 语句之后，END 语句之前组成函数辅程序体。函数辅程序体内语句排列顺序与主程序相同，即说明语句在前，执行语句在后。

在说明语句部分中，可以对函数名、虚元以及函数中所用变量、数组和符号常量进行说明。如果函数辅程序中未对上述变量进行说明，则这些变量的类型遵循 I-N 规则。

函数辅程序体中的执行语句是用来完成求函数值的运算的。在函数辅程序体中至少应包含一条如下形式的赋值语句：

函数名 = 表达式

以便于函数辅程序通过函数名把计算结果传送给调用程序。

注意：该函数名后不应该带虚元表。

③ 函数辅程序的最后一条语句也必须是 END 语句，且一个函数辅程序中也只允许有一个 END 语句。该 END 语句在编译时作为标志通知编译程序“本程序单元到此结束”。在执行时，计算机执行该语句后立即返回到调用单位并由函数名把所执行函数的值带回。

在辅程序体中可以使用 RETURN 语句，当计算机执行 RETURN 语句时也立即返回到调用单位。它与 END 的区别是在一个程序单位中 RETURN 可以有多个，也可以没有。而 END 在一个程序单位中必须有，且只能有一个。

例 6.3 编写一个求阶乘的函数辅程序，并用它求 C_{10}^0 的值。

程序及运行结果如下：

```
C      EXAMPLE 6.3
      FUNCTION FACTORY(N)
      INTEGER FACTORY, N, P, I
      P=1
      DO 10 I=1, N
      P=P*I
```

```

10      CONTINUE
        FACTORY=P
        END

C      BEGIN MAIN PROGRAM
        REAL S
        INTEGER FACTORY
        READ(*,*)M,N
        S=FACTORY(M)/FACTORY(N)/FACTORY(M-N)
        WRITE(*,100)M,N,S
100     FORMAT(1X,'M=' ,I3,2X,'N=' ,I3,1X,'S=' ,F8.2)
        END

```

运行结果如下：

```

10, 4
M= 10 N= 4 S= 210.00

```

二、函数辅程序的引用

函数辅程序一经定义之后，便可如同例 6.3 那样引用。函数辅程序的引用形式如下：

函数名([d[, d]...])

外部函数总是出现在表达式中，作为表达式的操作数而被引用。不过，引用外部函数的程序单位不应是定义该函数的函数辅程序，而是可以执行的主程序或其它辅程序。且在调用程序单位中必要时应对函数名的类型加以说明，以使得函数名在定义和调用程序单位中类型一致。

d 是实元。实元可以是表达式，数组名，内部函数名，外部过程名。

外部函数引用的执行过程如下：

① 计算实元表达式的值。

② 虚元与实元按对应位置相结合。若实元是常数值或由表达式求值得到的结果，则实元将其值传递给虚元。若实元是变量或数组的符号名，则虚元与实元共享同一存储单元。有关虚元与实元结合的更进一步讨论见后 6.5 节。

③ 当实元的值传递给对应位置上的虚元之后，程序的流程接着转去执行函数体内的执行语句，直到遇到 RETURN 或 END 语句为止。此时把函数名的当前值作为函数值，返回到调用程序。

④ 控制返回后，将函数值参与表达式的运算。

例 6.4 求由 6 个电阻组成的梯形网络的输入电阻。

电路图如图 6.4。

根据电路知识可知，该电路输入电阻应为 R ，其计算过程如下：

$$\begin{aligned}
 \frac{1}{R_{\#1}} &= \frac{1}{R_4} + \frac{1}{R_5 + R_6} \\
 \frac{1}{R_{\#2}} &= \frac{1}{R_2} + \frac{1}{R_3 + R_{\#1}} \\
 R &= R_1 + R_{\#2}
 \end{aligned}$$

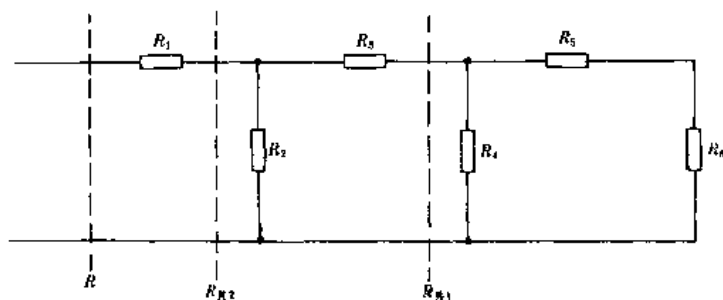


图 6.4

当该电路有 $2n$ 个电阻时(构成 n 级网络)其等效电阻计算公式为

$$\begin{aligned} \frac{1}{R_{\#1}} &= \frac{1}{R_{2n-2}} + \frac{1}{R_{2n-1} + R_{2n}} \\ \frac{1}{R_{\#2}} &= \frac{1}{R_{2n-4}} + \frac{1}{R_{2n-3} + R_{\#1}} \\ &\vdots \\ \frac{1}{R_{\#n-1}} &= \frac{1}{R_2} + \frac{1}{R_3 + R_{\#n-1}} \\ R &= R_1 + R_{\#n-1} \end{aligned}$$

根据上述公式, 求 n 级梯形网络的输入电阻的程序如下:

```

C      EXAMPLE 6.4
      REAL INR
      DIMENSION B(100)
      PRINT *, 'ENTER N'
10     READ(*, 15) N
15     FORMAT(I3)
      IF (N.EQ. 0) STOP
      PRINT *, 'ENTER RESISTANCE VALUE'
      READ(*, 20) (B(I), I=1, N)
20     FORMAT(8F10.4)
      DO 30 I=2, N, 2
30     B(I)=1.0/B(I)
      R=INR(B,N)
      PRINT *, 'INR=', R
      GOTO 10
      END

      FUNCTION INR(B,N)    子程序用于完成 n 级网络输入电阻的计算
      REAL INR
      DIMENSION B(100)
      B2=1.0/B(N)
      N1=N-1
      DO 40 I=1, N1
  
```

```

NN=N-1
B1=B(NN)+B2
IF (NN.EQ.1) GOTO 40
B2=1.0/B1
40 CONTINUE
INR=B1
RETURN
END

```

例 6.5 已知一个班的数学课程的考试成绩，试求该班数学课的平均成绩和及格率。

解 该题要求分别计算平均成绩和及格率两个值，若编两个函数辅程序来分别计算它们的值时，将会降低程序的效率。我们可以将计算平均成绩和及格率的计算编成一个函数辅程序，把平均成绩作为函数值返回给引用程序，及格率作为虚元通过虚实结合回送给引用程序。

框图见图 6.5 及图 6.6。

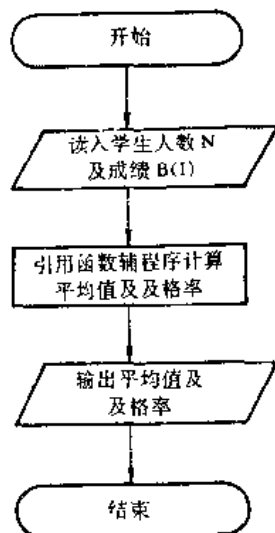


图 6.5

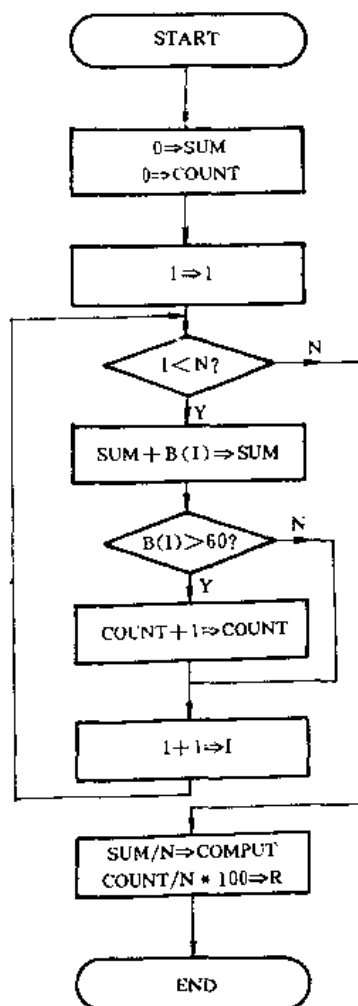


图 6.6

程序及运行结果如下:

```
C      EXAMPLE 6.5
      DIMENSION B(100)
      WRITE(*,10)
10     FORMAT(1X,'ENTER NUMBER OF STUDENTS: ')
      READ(*,*)N
      PRINT *,'ENTER MARK OF STUDENTS:'
      DO 20 I=1,N
20     READ(*,*)B(I)
      AVERAG=COMPUT(B,N,R)
      WRITE(*,30)AVERAG,R
30     FORMAT(1X,'AVERAGE=',F6.2,'    R=',F6.2,'%')
      STOP
      END

      FUNCTION COMPUT(B,N,R)
      DIMENSION B(N)
      REAL SUM, COUNT
      SUM=0.0
      COUNT=0
      DO 10 I=1, N
      SUM=SUM+B(I)
      IF (B(I).GE.80) COUNT=COUNT+1
10     CONTINUE
      COMPUT=SUM/N
      R=COUNT/N*100.
      RETURN
      END
```

运行结果:

```
ENTER NUMBER OF STUDENTS;
8
ENTER MARK OF STUDENTS;
70.
80.
78.
92.
65.
50.
100.
69.
AVERAGE= 75.50    R= 87.50 %
```

6.4 子程序

子程序和外部函数都是外部过程,它们有许多共同之处,但也有各自的特点。在功能

上尽管外部函数也可以通过定义某些虚元,返回时带回一些处理结果,但是最主要的还是通过函数名返回时传递函数值,且其功能以计算为主。在实际编程工作中,我们往往根据需要把一个较大的程序按其功能划分为若干个“子块”,分别进行编译、调试,然后再连接在一起完成整个程序的运算。这样不仅便于调试、查错,还便于进行模块化程序设计,提高编程的质量。各模块之间需要传递的数据(可能没有,也可能有若干个)可通过变量的虚实结合来传递。在这种情况下用子程序辅程序编程比用函数辅程序编程更为优越。

一、子程序辅程序的定义

子程序辅程序必须以 SUBROUTINE 语句开头,以 END 语句结束。它的一般形式是

```
SUBROUTINE sub [(d[, d]...)]
...
...
:
...
END
```

} 子程序辅程序体

几点说明:

① SUBROUTINE 语句是子程序辅程序的开始语句。sub 是 SUBROUTINE 语句所在的子程序辅程序的符号名,称为子程序名。d 是虚元,它可以是变量名、数组名、虚拟过程名或是星号。

② SUBROUTINE 语句之后,END 语句之前的语句构成子程序辅程序体。该辅程序的说明部分应包括对虚元和子程序辅程序中所用变量、数组等的说明;其执行部分完成子程序辅程序的运算和操作功能。

③ 子程序辅程序中的 END 或 RETURN 使程序执行流程返回到引用单位去继续执行引用语句后的语句。RETURN 语句与 END 语句的区别如 6.3 节所述。

④ 子程序的符号名仅标识子程序,没有值的意义,所以没有数据类型之分。也不能在本子程序辅程序体内出现,而且它是全局名,不能与程序中任何其它项目的符号同名。

⑤ 子程序辅程序也是一个独立的编译单位,所以在标号、变量名等的使用方面与函数辅程序完全相同。

例 6.6 设 A, B 是两个 5 阶的方阵,编写求 A, B 乘积的子程序

解 设 $C=A \times B$, 相应子程序辅程序如下:

```
C      EXAMPLE 6.6
      SUBROUTINE MUT(A,B,C)
      PARAMETER(N=5)
      REAL A(N,N), B(N,N), C(N,N)
      DO 10 I=1, N
      DO 10 J=1, N
      S=0.0
      DO 20 K=1, N
      S=S+A(I,K)*B(K,J)
20      CONTINUE
      C(I,J)=S
```

```

10      CONTINUE
      END

```

例 6.7 编写一个求 $\sum_{i=1}^n i$ 的子程序辅程序

解 我们可以将 n 和 $S(S = \sum_{i=1}^n i)$ 作为虚元，其程序如下：

```

C      EXAMPLE 6.7
      SUBROUTINE SUM (N,S)
      INTEGER S, N
      S=0
      DO 10 I=1, N
      S=S+I
10      CONTINUE
      END

```

从以上两例可以看出，在子程序辅程序中，子程序名在程序体内没有出现，这与函数辅程序名必须在函数辅程序体内作为变量被赋值有很大区别。

子程序也可以没有虚元，例如在程序中多次使用一组打印‘* * * * *’操作，可以写成下面的子程序辅程序：

```

      SUBROUTINE PRIN( )
      WRITE( *,10)
10      FORMAT(IX,'* * * * *')
      END
      CALL PRIN( )
      END

```

二、子程序辅程序的引用

同函数辅程序的引用一样，一个子程序辅程序一旦被定义之后，便可在其它程序单位中对之进行引用。子程序辅程序不能由定义它的子程序辅程序引用。但不同于外部函数，子程序的引用不能作为操作数而出现在表达式中，必须用专门的 CALL 语句引用。

CALL 语句的形式是

```
CALL sub [(a[, a]...)]
```

其中：sub 为子程序名；

a 为实元。实元可以是表达式，数组名，外部过程名，内部函数名，虚拟过程名，交错返回说明符等(见 6.5 节)。

引用子程序的实元必须在次序、个数、类型上与被引用的子程序的虚元相一致。

子程序的引用过程：

CALL 语句的执行触发子程序辅程序 sub 的执行，其过程是：

- ① 若实元是表达式，则先求该表达式的值；
- ② 实元与对应的虚元相结合；
- ③ 执行子程序辅程序体，直到遇到 RETURN 语句或 END 语句后再返回到引用它的 CALL 语句的下一条语句，继续执行原来的程序。

例 6.8 编写程序, 求 C_m^n 的值。

解 由求组合公式 $C_m^n = \frac{m!}{n! (m-n)!}$ 知, 计算 C_m^n 要多次用到计算阶乘过程, 在例 6.3 节中我们曾将该过程编为一段函数辅程序。在此, 我们可以把求阶乘过程编为一段子程序辅程序。当主程序中需要求阶乘时对之进行引用, 框图见图 6.7 及图 6.8。

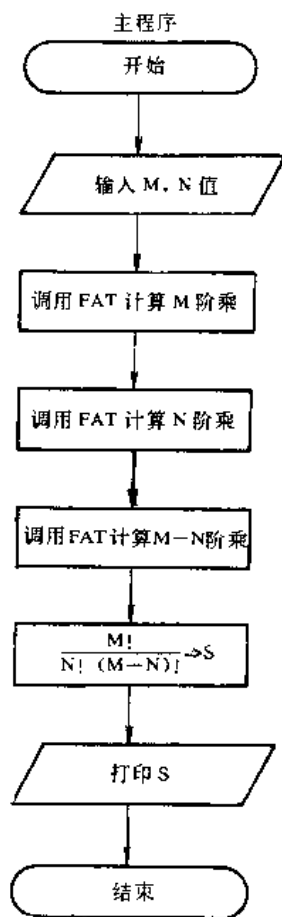


图 6.7

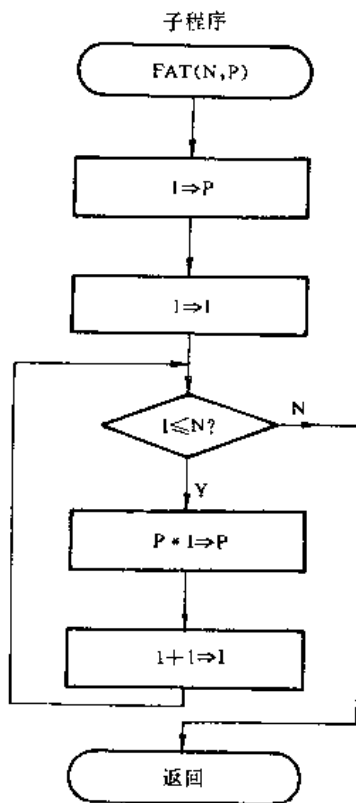


图 6.8

程序及运行结果如下:

```

C      EXAMPLE 6.8
      SUBROUTINE FAT (N, P)
      INTEGER N, P
      P = 1
      DO 10 I = 1, N
      P = P * I
10     CONTINUE
      END

      PROGRAM MAIN
      REAL S
      WRITE(*,10)
10     FORMAT (1X,'ENTER M, N')
  
```

```

      READ (*,*) M, N
      CALL FAT (M,K)
      S= K
      CALL FAT (N, K)
      S=S/K
      CALL FAT (M-N, K)
      S=S/K
      WRITE(*, 20) S
20    FORMAT (1X,'S=', F10. 2)
      END

```

运行结果为

```

      ENTER M, N
      10. 4
      S=    210. 00

```

例 6.9 求函数 $f(x)=e^x-\sin x$ 在区间 $[0, 1]$ 上的定积分。

解 根据高等数学的知识可知, 函数 $f(x)$ 在区间 $[a, b]$ 上的定积分的几何意义, 是求曲线 $f(x)$ 和直线 $x=a, x=b, y=0$ 所围成的图形面积。为了近似求出此面积, 可将 $[a, b]$ 区间分成若干个小区间, 每个小区间的宽度为 $(b-a)/n$, n 为小区间个数。近似求出每个小区间的面积, 然后将之累加起来, 就近似得到了总面积, 即定积分的近似值。区间分得越细, 近似程度就越高。求每个小区间的面积一般可以用矩形法、梯形法及抛物线法。下边我们给出用梯形法所编写的程序。

用梯形法计算定积分的公式为

$$S = \int_a^b f(x) dx \approx \frac{h}{2} [f(a) + f(b) + 2 \sum_{i=1}^{n-1} f(a + ih)]$$

$$h = \frac{b-a}{n}$$

框图见图 6.9 及图 6.10。

程序及运行结果如下:

```

C      EXAMPLE 6.9
      REAL S
      PRINT *, 'A,B,N='
      READ(*,*) A,B,N
      CALL SUM (N,A,B,S)
      PRINT *, 'S=', S
      END

      SUBROUTINE SUM (N,A,B,S)
      REAL A,B,S
      INTEGER N
      F(X)=EXP(X)-SIN(X)
      X=A
      H=(B-A)/N
      S=0.0

```

```

DO 10 I=1, N
Y=X+H
S=S+(F(X)+F(Y))*H/2
X=Y
10 CONTINUE
END

```

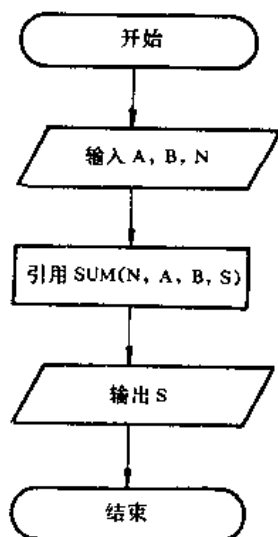


图 6.9

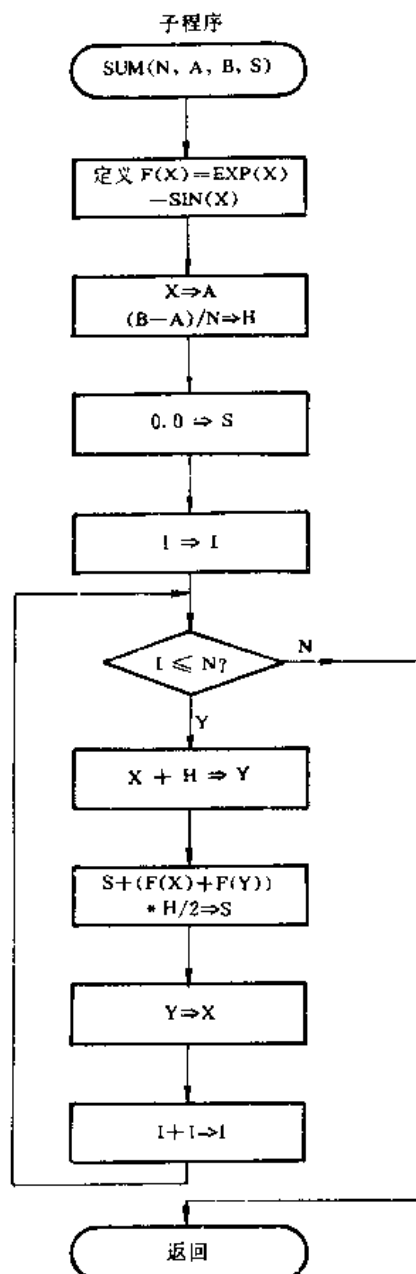


图 6.10

结果为

```

A, B, N=
0., 1., 10
S=      1.260399

```

三、FORTRAN 77 四种过程比较

如上所述, FORTRAN 77 中有四种过程。它们的定义及引用有所不同。为了便于比较, 把它们归纳在表 6.1 中。

表 6.1 FORTRAN 77 四种过程比较

名 称	内部函数	语句函数	外部函数	子程序
符号名	由处理系统给出	用字母开头 1 至 6 个字母数字串组成		
类 型	按处理系统规定	对符号名用类型语句或 IMPLICIT 语句或 I-N 规则加以说明	函数名的类型说明同左	子程序名没有类型意义
定 义	处理系统已定义过	在程序单位中说明语句之后, 可执行语句之前, 用一个语句函数语句定义	以 FUNCTION 语句开头, END 语句结尾的函数辅程序单位定义	以 SUBROUTINE 语句开头, END 语句结尾的子程序辅程序单位定义
虚 元	可有一个或多个	可有零个或任意个虚元变量名	可有零个或任意个虚元变量名、数组名或虚拟过程名; 当没有虚元时, 括号仍要保留	可有零个或任意个虚元变量名, 数组名, 虚拟过程名或尾号; 当没有虚元时, 括号可任选
引用方式	把函数名连同实元表直接写在表达式中, 作为一个操作数来引用			用 CALL 语句引用
产生值的个数	一个	一个	通过函数名产生一个函数值, 通过实元产生零个或任意个值	通过实元产生零个或任意个值
引用范围	局部于引用程序单位	只能在定义该语句函数的程序单位	可执行程序中, 除定义本过程的辅程序单位外的任何程序单位	

四、在子程序中的 SAVE 语句和 DATA 语句

1. SAVE 语句

在函数辅程序或子程序辅程序中所使用的所有局部变量, 在被引用前通常没有被分配确定的存贮单元, 只有在该辅程序单位被引用时, 系统才临时分配给存贮单元, 而且在退出子程序时, 这些存贮单元可能又都被释放掉了。相应局部变量的值在退出辅程序时也即丢失。如果想要保存辅程序中某些局部变量的值, 也即当下次再进入该辅程序单位时, 这些局部变量所分配的存贮单元不变, 其值也能保留下来。在 FORTRAN 77 中可以用 SAVE 语句达到此目的。

SAVE 语句的形式是

SAVE [a[, a]...]

其中: a 是要被保留的局部变量名或数组名。

如果 SAVE 语句中不带变量表, 则表示本程序单位中所有局部变量在程序执行期间所占有的确定存贮单元, 不因退出辅程序而被释放。

注意:

- ① 虚元不允许出现在 SAVE 后的变量表中。
- ② 对于在主程序中定义的变量, 没有必要用 SAVE 语句来保存其存贮单元。

2. DATA 语句

在辅程序中也可以利用 DATA 语句给局部变量赋初值, 其用法与在主程序中的用法相同。值得说明的是 DATA 语句是说明语句。程序在编译期间给变量赋定初值。当用 DATA 语句给子程序中的变量或数组赋初值时, 这些变量和数组元素只是在子程序第一次调用时才有初值, 如果没有 SAVE 语句说明, DATA 语句中的变量和数组在第一次调用退出后就可能会变成无定义的了。(与具体所使用的编译系统有关, 在 Microsoft FORTRAN 5.00 中, 当系统所提供的内存空间比程序运行所需要的空间大时, 这些局部变量退出该程序单位时并不被释放。)

例 6.10 分析下述程序及其运行结果:

```

C      EXAMPLE 6.10
      CALL EXAP(3)
      CALL EXAP(4)
      CALL EXAP(5)
      END

      SUBROUTINE EXAP(M)
      INTEGER P,S
      SAVE P, S
      DATA P/1/,S/0/
      IF (M.GT.0) THEN
      DO 10 I = 1, M
      P=P*I
      S=S+1
10     CONTINUE
      WRITE(*,20) M, P, S
20     FORMAT(IX,'M=',I2,' P=',I10,' S=',I3)
      END IF
      END
  
```

结果如下:

M= 3	P=	6	S= 6
M= 4	P=	144	S= 16
M= 5	P=	17280	S= 31

当在辅程序中同时使用 SAVE 语句和 DATA 语句时, SAVE 语句应放在 DATA 语句之前。

读者可在自己所使用的系统上将上例中的 SAVE 语句去掉, 看运行结果如何。

6.5 虚元和实元的结合

从前面几节的讨论中可以清楚地看到：FORTRAN 77 程序是由主程序和辅程序单位组成的。在执行函数引用和子程序引用时，就要进行虚元和实元的结合。虚元可分为四类：变量名、数组名、虚拟过程名或星号。表 6.2 给出了可以与各类虚元相结合的实元类别。

表 6.2

虚 元		实 元
变 量 名		同类型的 { 变量名 数组元素名 子串名 其它表达式
数 组 名		同类型的 { 数组名 数组元素名 数组元素子串名
虚拟过程名	虚拟函数名	同类型的 { 内部函数名 外部函数名 另一个虚拟函数名
	虚拟子程序名	{ 子程序名 另一个虚拟子程序名
星 号		交错返回说明符

当虚元与实元结合时，虚元的体积(元素个数及字符总数)不应超过与之对应的实元体积。

一、变量作为虚元

当虚元是变量时，对应的实元可以是同一类型的常量、变量、数组元素和表达式。如果实元是变量或数组元素，在引用子程序时，对应的虚元实际上将与之共享同一个存储单元。实元的值就是虚元的初值。虚元的值改变时对应实元的值也随之改变。虚元与实元之间值的这种传送方式称作按地址传送。

若实元是常量或表达式，则在辅程序中不能对与之相结合的虚元赋值。

如果虚元是字符变量，则它的长度不应超过对应实元的长度。若虚元字符的长度用(*)来定义，则表示其长度不定，当引用子程序时，具有不定长度的虚元变量将自动定义成为与对应实元具有同样的长度。请看例 6.11。

例 6.11

```

C   EXAMPLE 6.11
      CHARACTER CH1 * 5, CH2 * 20
      CH1 = 'ABCD'
      CH2 = 'ABCDEFGHIL'
```



```

PRINT *, 'First call'
CALL STRL(CH1)
PRINT *, 'Second call'
CALL STRL (CH2)
END

SUBROUTINE STRL(CH4)
CHARACTER * ( * ) CH4
N=LEN (CH4)  LEN 为测字符串长度函数, 详见 7.3 节
PRINT *, 'THE STRING IS: ', CH4
PRINT *, 'LENGTH OF THE STRING IS', N
END

```

运行结果如下:

```

First call
THE STRING IS: ABCD
LENGTH OF THE STRING IS      5
Second call
THE STRING IS: ABCDEFGHIJ
LENGTH OF THE STRING IS     20

```

二、数组作为虚元

FORTRAN 中有三种类型的数组说明符: 当维说明符中每一维的上下界均为常数时, 称为固定数组; 当维说明符中包含有变量时, 称为可调数组; 当维说明符中最后一维上界用 * 说明时, 称为假定大小的数组。可调数组和假定大小的数组只能在辅程序中使用。

当虚元是数组名时, 相结合的实元可以是同类型的数组名或数组元素名。实元数组的维数与虚元数组的维数, 各维的上下界可以不同。虚元数组与实元数组按照存贮次序从首地址单元起一一对应结合。所有的虚元元素必须有相应的实元元素与之对应。

1. 实元为数组名

当实元为数组名时, 实元与虚元根据数组元素在内存中的排列次序, 按对应首地址依次结合。其过程如例 6.12 所示。

例 6.12

```

C      EXAMPLE 6.12
      PROGRAM MAIN
      DIMENSION A(5), B(2,4)
      CHARACTER * 4 C(2,3)
      DATA A /1.,2.,3.,4.,5./,B/1.,2.,3.,4.,5.,6.,7.,8./
      DATA C/'1111','2222','3333','4444','5555','6666'/
      CALL SUB1(A)
      CALL SUB2(B)
      CALL SUB3(C)
      PRINT *, A(3),A(5)
      PRINT *, B(1,3),B(2,4)

```

```

PRINT *,C(1,1),C(2,2)
END

SUBROUTINE SUB1 (X)
  DIMENSION X (2,2)
  DO 10 I=1,2
  DO 10 J=1,2
    X(I,J)=I * J
10  CONTINUE
  END

SUBROUTINE SUB2 (Y)
  DIMENSION Y (6)
  Y(4)=0.
  Y(5)=0.
  Y(6)=0.
  END

SUBROUTINE SUB3 (Z)
  CHARACTER * 2 Z(2,2)
  Z(1,1)='AA'
  Z(1,2)='BB'
  Z(2,1)='CC'
  Z(2,2)='DD'
  END

```

运行结果如下：

```

2.000000      5.000000
0.000000E+00  8.000000
AACC4444

```

执行 CALL SUB1(A)时，实元 A 与虚元 X 的结合过程如表 6.3 所示。

表 6.3

实元	A(1)	A(2)	A(3)	A(4)	A(5)
虚元	X(1, 1)	X(2, 1)	X(1, 2)	X(2, 2)	

执行 CALL SUB2(B)时，实元 B 与虚元 Y 的结合过程如表 6.4 所示。

表 6.4

实元	B(1, 1)	B(2, 1)	B(1, 2)	B(2, 2)	B(1, 3)	B(2, 3)	B(1, 4)	B(2, 4)		
虚元	Y(1)	Y(2)	Y(3)	Y(4)	Y(5)	Y(6)				

执行 CALL SUB3(C)时，实元 C 与虚元 Z 的结合过程如表 6.5 所示。

表 6.5

实元	C(1, 1)		C(1, 2)		C(1, 3)		C(2, 1)		C(2, 2)		C(2, 3)	
虚元	Z(1, 1)	Z(2, 1)	Z(1, 2)	Z(2, 2)								

当数组是字符型时,虚元和实元数组不是按数组元素的顺序,而是按字符位置一一对应结合的。

2. 实元为数组元素名

当虚元为数组时,实元可以为数组元素。虚元与实元的结合过程是这样的,实元从该数组元素开始,虚元从虚数组的第一个元素开始,按照数组元素的排列次序依次结合。但虚元数组的最后一个元素必须落在对应的实元数组范围内,而且与数组维数无关。

当虚元为数组,实元为数组元素时,虚实元素结合过程请看例 6.13。

例 6.13

```

C      EXAMPLE 6.13
      DIMENSION A(8), B(3,3)
      DATA A/8*0.0/, B/9*1.0/
      CALL SUB1(A(3))
      PRINT *, 'ARRAY OF A'
      WRITE(*,10)A
10     FORMAT(1X, 8F8.2)
      CALL SUB2(B(1,2))
      PRINT *, 'ARRAY OF B'
      WRITE(*,20)((B(1,J), J=1,3), I=1,3)
20     FORMAT(1X,3F8.2)
      END

      SUBROUTINE SUB1(X)
      DIMENSION X(4)
      DO 10 I=1,4
      X(I)=I
10     CONTINUE
      END

      SUBROUTINE SUB2(Y)
      DIMENSION Y(4)
      DO 10 I=1,4
      Y(I)=I
10     CONTINUE
      RETURN
      END
  
```

其运行结果如下:

```

      ARRAY OF A
      .00    .00    1.00    2.00    3.00    4.00    .00    .00

      ARRAY OF B
      1.00    1.00    4.00
      1.00    2.00    1.00
      1.00    3.00    1.00
  
```

虚元与实元的对应关系见表 6.6 及表 6.7。

执行 CALL SUB1(A(3))时实元与虚元的对应关系见表 6.6。

表 6.6

实元	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)
虚元			X(1)	X(2)	X(3)	X(4)		

执行 CALL SUB2(B(1, 2))时实元与虚元的对应关系见表 6.7。

表 6.7

实元	B(1, 1)	B(2, 1)	B(3, 1)	B(1, 2)	B(2, 2)	B(3, 2)	B(1, 3)	B(2, 3)	B(3, 3)
虚元				Y(1)	Y(2)	Y(3)	Y(4)		

3. 可调数组

在本章例 6.5 中, 计算学生考试平均成绩的函数辅程序 COMPUT 中包含了 DIMENSION B(N)的数组说明语句, 此处 N 是虚元。如果数组说明符的维界表达式中包含了整型变量名, 则这个数组称为可调数组。包含变量名的维说明符称为可调维。

FORTRAN 规定, 可调数组只能在函数辅程序单位和子程序辅程序单位中使用, 而不能在主程序单位中使用。而且可调数组的名字必须是辅程序的虚元, 且可调维界表达式中出现的变量名也必须出现在虚元表中, 或出现在该辅程序的公用语句中(见 8.2 节)。但可调数组名不得出现在公用语句中。

例 6.14 求 $S = \sum_{i=1}^n a_i b_i$ 。

解 将 a_i, b_i 的值分别放入一维数组 A、B 中, 并将求 $\sum_{i=1}^n a_i * b_i$ 的过程编成一个包含可调数组的通用程序。引用该程序计算 S 的值。

程序及运行结果如下:

```

C      EXAMPLE 6.14
      DIMENSION A(10), B(10)
      READ (*, 10) A, B
10     FORMAT (10F8.3)
      S=0.0
      CALL SCAL (A,B,10,S)
      WRITE(*,20)S
20     FORMAT (1X,'S=',F10.3)
      END

      SUBROUTINE SCAL (X,Y,N,Z)
      DIMENSION X (N),Y (N)
      DO 10 I=1, N
      Z=Z+X(I)*Y(I)
10     CONTINUE
      RETURN
      END

```

运行结果如下:

```

1., 2.1, 3.5, 4.0, 1., 4, 6.7, 7.8, 9.1, 12.3, 3.45
2.43, 4.58, 8.75, 5.67, 7.43, 2.12, 1.2, 4.7, 9.8, 10.2

```

S= 297.819

对于假定大小的数组在有些版本中不能使用,故在此不再详述。

三、过程名作为虚元

当过程名作为虚元时与它相结合的实元必须是内部函数名,外部过程名或另一个虚拟过程名。

1. 实元是内部函数名

若实元是内部函数名时,则在该实元所在的程序单位中必须用内部语句(INTRINSIC 语句)说明这个用作实元的内部函数名。因为 FORTRAN 中没有保留字,用 INTRINSIC 语句对内部函数名进行说明,以便于和其它变量名区分。

INTRINSIC 语句的形式为

INTRINSIC fun [, fun]...

其中, fun 为内部函数名;

INTRINSIC 语句的功能是指明 fun 是内部函数名。

例 6.15 编写一个函数子程序,通过函数名的传递,使之即能求 $e^{2x} + e^x/\sin x + 5x^2 - 1$, 又能求 $\sin^2 x + \sin x/e^x + 5x^2 - 1$ 。

解 e^x 和 $\sin x$ 都是 FORTRAN 的内部函数。若用 F1(x)和 F2(x)表示 FORTRAN 内部函数,则以上两式可统一写成:

$$F1(x) * * 2 + F1(x)/F2(x) + 5 * x * * 2 - 1$$

据此我们可以写出如下程序:

```
C      EXAMPLE 6.15
      PROGRAM MAIN
      INTRINSIC SIN,EXP
      READ (*,*)X
      Y1=COMPT (EXP,SIN,X)
      Y2=COMPT (SIN,EXP,X)
      WRITE (*,10) Y1,Y2
10    FORMAT (1X,'Y1=',F8.3,' Y2=',F8.3) .
      END

      FUNCTION COMPT (F1,F2,X)
      COMPT=F1(X) * * 2 + F1(X)/F2(X) + 5 * X * * 2 - 1
      END
```

其运行结果如下:

1.

Y1= 14.619 Y2= 5.018

2. 实元是外部过程名

若实元是外部函数、子程序或虚拟过程名时,必须在调用程序中用外部语句(EXTERNAL)来指明用作实元的外部函数名、子程序名或虚拟过程名。

EXTERNAL 语句的形式为

EXTERNAL proc [, proc]...

其中: proc 是外部过程名或虚拟过程名;

EXTERNAL 语句的功能是指明 proc 是外部过程名或虚拟过程名, 以区别于虚实结合时的简单变量名。

例 6.16 设有 $f_1(x) = \frac{\sin x}{1+x}$, $f_2(x) = e^x \cos x$ 求这两个函数在区间 $[0, 1]$ 上的定积分

解 若选用辛普生公式来求定积分, 将 $[a, b]$ 区间分成 $2n$ 等分, 令 $h = \frac{b-a}{2n}$, 则 $\int_a^b f(x) \cdot dx$ 的辛普生公式为

$$\begin{aligned} \int_a^b f(x) dx &\approx \frac{h}{3} \{f(a) + f(b) + 2 \sum_{i=1}^n f(a + 2ih) + 4 \sum_{i=1}^n f(a + (2i-1)h)\} \\ &= \frac{h}{3} \{f(a) + f(b) + \sum [4f(a + jh) + 2f(a + (j+1)h)]\} \end{aligned}$$

其中, 最后一个式子的 \sum 是对 $j = 1, 3, 5, \dots, 2n-1$ 求和。

我们把辛普生公式计算积分编制成一个通用积分程序。把 $f_1(x)$, $f_2(x)$ 分别编成外部函数, 其程序框图如图 6.11~6.14 所示。

```

C      EXAMPLE 6.16
      PROGRAM MAIN
      EXTERNAL F1, F2
      PRINT *, 'Enter A,B,N:'
      READ (*, *) A,B,N
      CALL SIMPSON(A,B,N,F1,S1)
      WRITE (*,10) S1
10     FORMAT (1X,'S1=', F10.5)
      CALL SIMPSON(A,B,N,F2,S2)
      WRITE (*,20) S2
20     FORMAT (1X,'S2=', F10.5)
      END

      SUBROUTINE SIMPSON(A,B,N,F,S)
      H=(B-A)/(2*N)
      S=F(A)-F(B)
      DO 10 I=1,2*N-1,2
      X1=A+I*H
      X2=X1+H
      S=S-4*F(X1)+2*F(X2)
10     CONTINUE
      S=S*H/3.0
      END

      FUNCTION F1(X)
      F1=SIN(X)/(1+X)
      END

      FUNCTION F2(X)
      F2=EXP(X)*COS(X)
      END
  
```

运行结果如下:

Enter A, B, N:

0., 1., 10

S1 = .28423

S2 = 1.37802

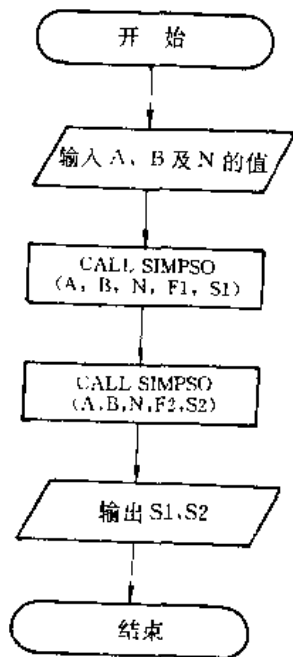


图 6.11

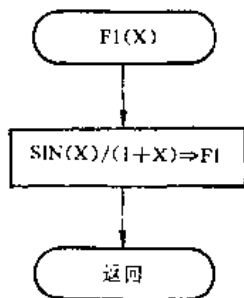


图 6.13

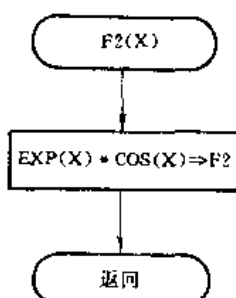


图 6.14

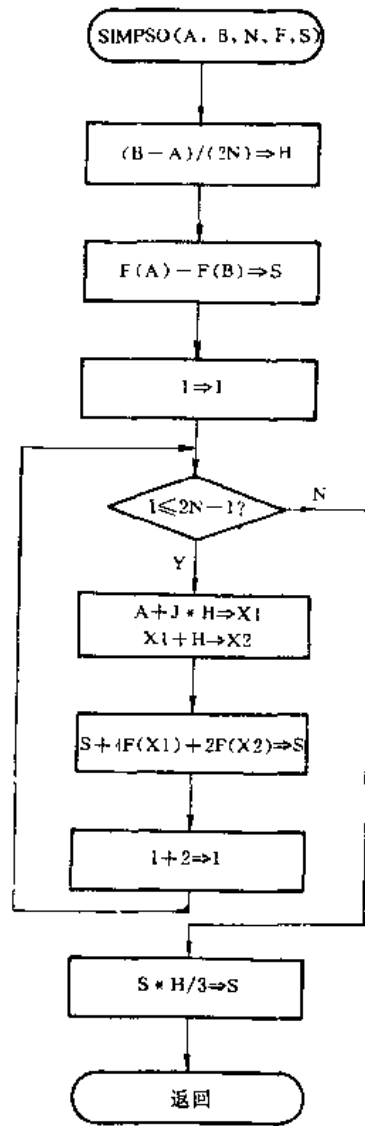


图 6.12

注意: 在上例中 F1, F2 不能在主程序中被定义成语句函数来引用 Simpson(辛普生)过程。因为引用时对应于虚元 F 的实元应是外部过程名。

四、星号作为虚元

当子程序辅程序的虚元表中出现一个 * 号时, 对应的实元应该是一个冠有 * 号的语句标号(称为交错返回说明符)。且此语句标号是与该 CALL 语句出现在同一程序单位中的可执行语句标号。为了搞清楚交错返回过程, 请看下例:

例 6.17 从键盘读入 X, Y, N 三个值, 并做以下计算

$$S = \begin{cases} X+Y & N=1 \\ X-Y & N=2 \\ X*Y & N=3 \\ X/Y & N=4 \end{cases}$$

解 我们将上述计算过程编成一个子程序。在主程序中读入 X, Y, N 的值, 调用上述子程序并输出计算结果。框图见图 6.15 及图 6.16。

程序如下:

```

C      EXAMPLE 6.17
      PRINT *, 'Enter X,Y,N,'
      READ (*, *) X,Y,N
      IF ((N. GE. 1). AND. (N. LE. 4)) THEN
        CALL AT (X,Y,N,S, * 10, * 20, * 30, * 40)
        STOP
10     PRINT *, 'X+Y=', S
        STOP
20     PRINT *, 'X-Y=', S
        STOP
30     PRINT *, 'X*Y=', S
        STOP
40     PRINT *, 'X/Y=', S
        STOP
      ELSE
        PRINT *, 'DATA ERROR'
      END IF
      END

      SUBROUTINE AT (X,Y,N,S, *, *, *, *)
      IF (N. EQ. 1) THEN
        S=X+Y
        RETURN 1
      ELSE IF (N. EQ. 2) THEN
        S=X-Y
        RETURN 2
      ELSE IF (N. EQ. 3) THEN
        S=X*Y
        RETURN 3
      ELSE
        S=X/Y
        RETURN 4
      END IF
      END
  
```

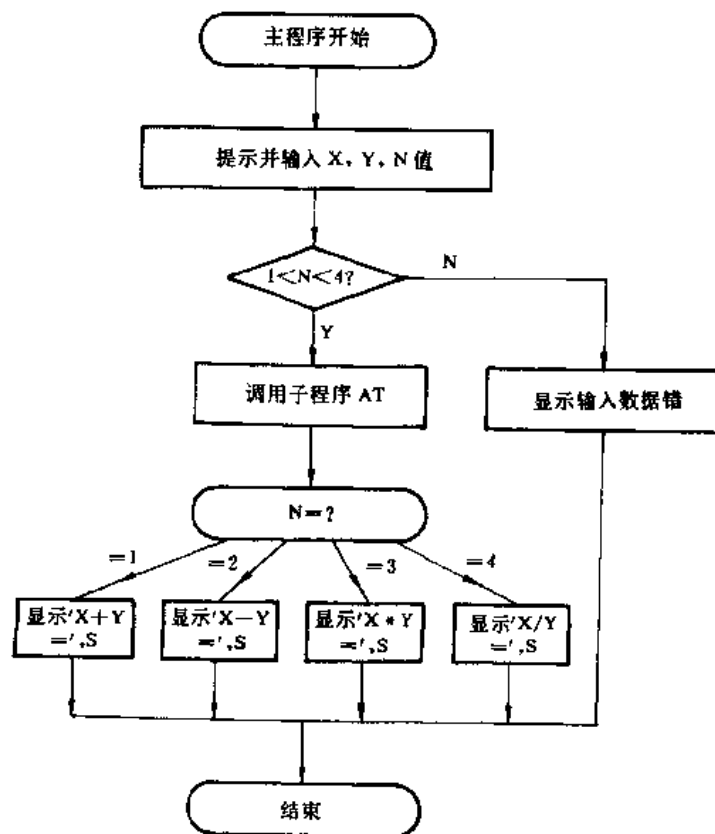



图 6.15

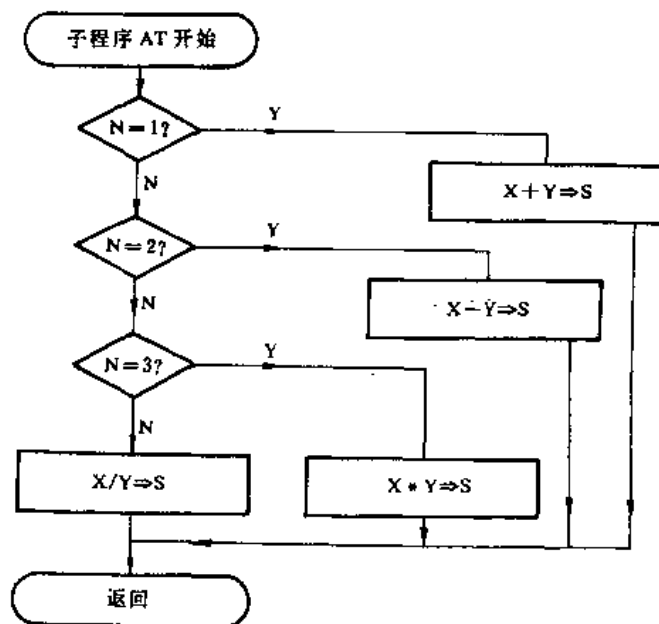


图 6.16

请对照程序，分析以下运行结果：

Enter X, Y, N;

1.7, 2.6, 1

X+Y= 4.300000

Stop — Program terminated.

第二次执行:

```
Enter X, Y, N;  
3, 5, 1.5, 2  
X - Y =      2.000000  
Stop - Program terminated.
```

第三次执行:

```
Enter X, Y, N;  
2.5, 4.0, 03.  
X * Y =     10.000000  
Stop - Program terminated.
```

在 CALL AT(X, Y, N, S, * 10, * 20, * 30, * 40) 语句中, 与虚参第一个 * 号对应的语句标号为 10, 第二个 * 号对应的语句标号为 20, 第三个 * 号对应的语句标号为 30, 第四个 * 号对应的语句标号为 40。在执行 AT 过程中, 当遇到 RETURN 1 语句时, 执行的流程返回主程序并跳到与第一个 * 号对应的 10 语句去继续执行。当遇到 RETURN 4 时, 执行的流程返回到主程序并跳到与第 4 个 * 号对应的 40 语句去继续执行。当遇到 RETURN 时将返回到主程序中 CALL 语句的下一条语句去执行。

利用 * 号作虚元, 可使执行流程从子程序中返回到主程序的不同地方。

6.6 利用过程实现结构化程序设计

FORTRAN 中的辅程序为我们提供了结构化程序设计的有力工具。我们可以将要做的工作按功能划分成一个个模块。在主程序中, 只引用相应的辅程序模块即可。整个程序结构如图 6.17 所示。

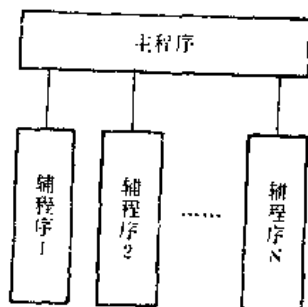


图 6.17

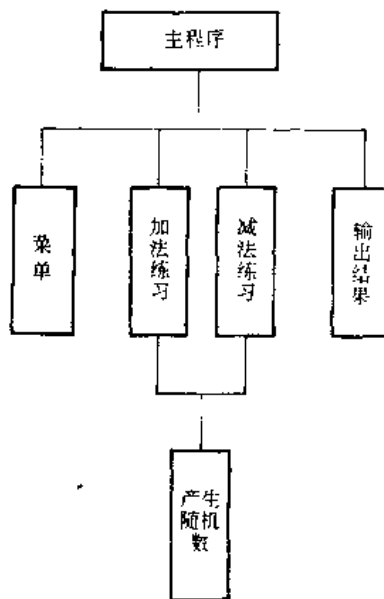


图 6.18

例 6.18 编写一个应用程序,用来给小学生做一位数的加减运算练习,并能按答题情况给出成绩。

解 按照结构化程序设计的思想,我们将该程序划分为主程序(MAIN),菜单(MENU),加法(ADD),减法(SUB),随机数产生(RAND)及结果输出(OUT)等六个模块。利用随机数产生过程产生两个随机操作数。在加法及减法过程中,给学生在屏幕上出题,并让用户从键盘上输入答案。根据用户(学生)回答的正确与否,对成功或失败的次数累加。输出过程中根据答对题数的百分比给出本次做题的成绩。菜单过程给用户提示可做的工作,并让用户选择。主程序负责引用和协调上述各过程的工作,以完成给定的任务。

整个程序的功能模块图见图 6.18。

主要模块流程图见图 6.19 及图 6.20。

减法练习过程基本上与加法相同。

程序如下:

```

C   EXAMPLE 6.18
      PROGRAM MAIN
      INTEGER FAIL, SUCC
      CHARACTER CH * 1
      FAIL = 0
      SUCC = 0
      CH = ' '
      DO 10 WHILE (CH.NE.'3')
      CALL MENU (CH)
      IF (CH.EQ.'1') THEN
          CALL ADD (SUCC,FAIL)
      ELSE IF (CH.EQ.'2') THEN
          CALL SUB (SUCC,FAIL)
      ELSE IF (CH.EQ.'3') THEN
          CALL OUT (SUCC,FAIL)
          STOP
      END IF
10  CONTINUE
      END

      SUBROUTINE MENU (C)
      CHARACTER * 1 C
      WRITE (*,100)'    1>Add'
      WRITE (*,100)'    2>Subtract'
      WRITE (*,100)'    3>EXIT'
      WRITE (*,100)' '
      WRITE (*,100)'    SELECT 1,2 OR 3'
      READ (*,'(A)') C
100  FORMAT (1X,A)'
      END
  
```

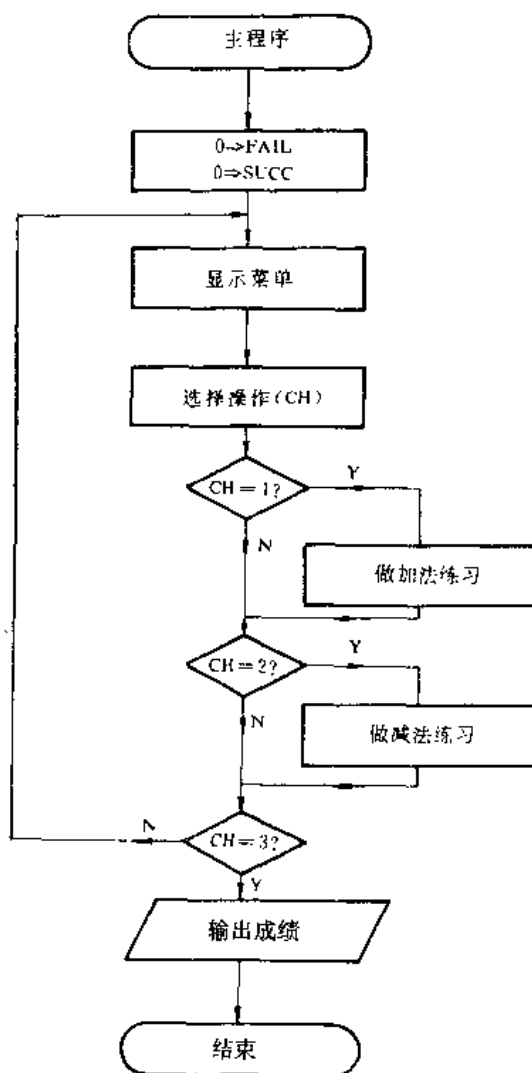


图 6.19

```

SUBROUTINE ADD (M,N)
  INTEGER OPR1,OPR2,L
  OPR1=10 * RAND( )
  OPR2=10 * RAND( )
  WRITE ( * ,100)OPR1,'+',OPR2,'='
100  FORMAT(1X,5X,I2,A,I2,A, $ )
  READ ( * ,200) L
200  FORMAT (I2)
  IF (OPR1+OPR2. EQ. L) THEN
    M=M+1
    WRITE ( * ,300)' YOU ARE RIGHT'
300  FORMAT(1X,9(' * '),A,9(' * '))
  ELSE
    N=N+1
    WRITE ( * ,300)'YOU ARE WRONG'
  END IF
END
SUBROUTINE SUB (M,N)
  INTEGER OPR1,OPR2
  OPR1=10 * RAND( )
  OPR2=10 * RAND( )
  IF (OPR1. LE. OPR2) THEN
    L1=OPR1
    OPR1=OPR2
    OPR2=L1
  END IF
  WRITE ( * ,100) OPR1,'-',OPR2,'='
100  FORMAT (1X,I2,A,I2,A, $ )
  READ ( * ,200) L
200  FORMAT (I2)
  IF (OPR1-OPR2. EQ. L) THEN
    M=M+1
    WRITE ( * ,300)'YOU ARE RIGHT'
  ELSE
    N=N+1
    WRITE ( * ,300)'YOU ARE WRONG'
  END IF
300  FORMAT (1X,9(' * '),A,9(' * '))
END
SUBROUTINE OUT (M,N)
  L=M+N
  IF (L. EQ. 0) THEN

```

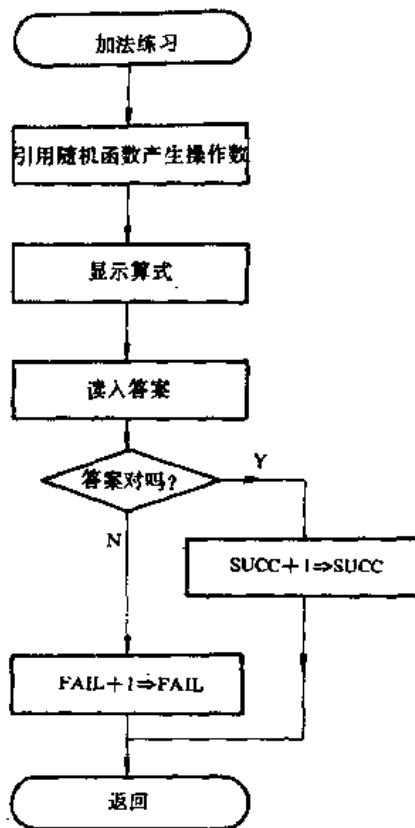


图 6.20

```

      T=0.0
    ELSE
      T=100.0/L*M
    END IF
    WRITE (*,100)' YOUR MARK IS:',T
100  FORMAT (1X,A,F5.1)
    END

    FUNCTION RAND( )
    INTEOER SEED,L,C,M
    PARAMETER (L=29,C=217,M=1024)
    DATA SEED/0/
    IF (SEED.EQ.0) THEN
      WRITE (*,'(1X,A,$)')'ENTER RAND SEED (0--32767):'
      READ (*,*) SEED
    END IF
    SEED=MOD (SEED*L+C,M)
    RAND=REAL (SEED)/M
    END

```

运行结果如下:

```

      1>Add
      2>Subtract
      3>EXIT

    SELECT 1, 2 OR 3
1
ENTER RAND SEED (0 - 32767) : 5
      3+ 4=7
***** YOU ARE RIGHT *****
      1>Add
      2>Subtract
      3>EXIT

    SELECT 1, 2 OR 3
2
      6- 4=3
***** YOU ARE WRONG *****
      1>Add
      2>Subtract
      3>EXIT

    SELECT 1, 2 OR 3
3
YOUR MARK IS: 50.0
Stop — Program terminated.

```

在上述程序中, WRITE 语句格式串中引用了 '\$' 符号。其作用是使紧随其后的 READ

语句在读入数据时,数据与该 WRITE 语句输出的数据显示在同一行上。

读者可以发现,在上述程序中加法过程与减法过程基本相同。有兴趣的读者可以将这两个过程中相同的部分再编成一个子程序,以使得整个程序结构更加严谨。

例 6.19 用全选主元高斯(Gauss)消去法求线性代数方程组的解。

$$\begin{cases} 0.2368x_1 + 0.2471x_2 + 0.2568x_3 + 1.2671x_4 = 1.8471 \\ 0.1968x_1 + 0.2071x_2 + 1.2168x_3 + 0.2271x_4 = 1.7471 \\ 0.1582x_1 + 1.1675x_2 + 0.1768x_3 + 0.1871x_4 = 1.6471 \\ 1.1161x_1 + 0.1254x_2 + 0.1397x_3 + 0.1490x_4 = 1.5471 \end{cases}$$

解 线性代数方程组的一般形式为

$$AX=B$$

其中:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \quad \text{为系数矩阵}$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{为变量向量}$$

$$B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad \text{为常数向量}$$

根据线性代数的知识可知,高斯消去法分两步进行。

第一步:消去过程。

在这一过程中,为了保证数值计算的稳定性,本程序采用了全选主元。这一过程的步骤为:

对于 k 从 1 开始一直到 $n-1$ 作以下三步:

① 全选主元。即从系数矩阵的第 k 行,第 k 列开始的子阵中选取绝对值最大的元素,并将它交换到主元素的位置上。

② 归一化。即

$$a_{kj}/a_{kk} \Rightarrow a_{kj} \quad j=k+1, \cdots, n$$

$$b_k/a_{kk} \Rightarrow b_k$$

③ 消去。即

$$a_{ij} - a_{ik}a_{kj} \Rightarrow a_{ij} \quad i, j=k+1, \cdots, n$$

$$b_i - a_{ik}b_k \Rightarrow b_i \quad i=k+1, \cdots, n$$

第二步:回代过程。

① $b_n/a_{nn} \Rightarrow x_n$

② $b_i - \sum_{j=i+1}^n a_{ij}x_j \Rightarrow x_i \quad i=n-1, \cdots, 2, 1$

相应程序如下:

```

C      EXAMPLE 6.19
      PROGRAM MAIN
      DIMENSION A (4,4), B(4), X(4), JS(4)
      DOUBLE PRECISION A,B,X
      DATA A/0.2368,0.1968,0.1582,1.1161,0.2471,0.2071,1.1675,
$      0.1254,0.2568,1.2168,0.1768,0.1397,1.2671,0.2271,0.1871,
$      0.1490/
      DATA B/1.8471,1.7471,1.6471,1.5471/
      N=4
      CALL GAUSS (A,B,N,X,L,JS)
      IF (L.NE.0) THEN
        WRITE (*,10)(I,X(I), I=1,4)
      END IF
10     FORMAT (1X,'X(',I2,')=' ,D15.6)
      END

      SUBROUTINE GAUSS (A,B,N,X,L,JS)
      DIMENSION A (N,N), X(N), B(N), JS(N)
      DOUBLE PRECISION A,B,X,T
      L=1
      DO 50 K=1, N-1
        D=0.0
        DO 210 I=K, N
          DO 210 J=K, N
            IF (ABS(A(I,J)).GT.D) THEN
              D=ABS (A(I,J))
              JS (K)=J
              IS=1
            END IF
          210 CONTINUE
            IF (D+1.0.EQ.1.0) THEN
              L=0
            ELSE
              IF (JS(K).NE.K) THEN
                DO 220 I=K,N
                  T=A (I,K)
                  A (I,K)=A (I,JS(K))
                  A (I,JS(K))=T
                220 CONTINUE
              END IF
              IF (IS.NE.K) THEN
                DO 230 J=K,N
                  T=A (K,J)

```

```

      A (K,J)=A (IS,J)
      A (IS,J)=T
230    CONTINUE
      T=B(K)
      B (K)=B(IS)
      B (IS)=T
      END IF
    END IF
    IF (L. EQ. 0) THEN
      WRITE( * ,100)
      RETURN
    END IF
    DO 10 J=K+1,N
      A (K,J)=A (K,J)/A (K,K)
10    CONTINUE
      B(K)=B (K)/A (K,K)
      DO 30 I=K+1,N
        DO 20 J=K+1,N
          A (I,J)=A(I,J)-A (I,K) * A (K,J)
20    CONTINUE
          B (I)=B (I)-A (I,K) * B (K)
30    CONTINUE
50    CONTINUE
      IF (ABS(A(N,N))+1. 0. EQ. 1. 0) THEN
        L=0
        WRITE( * ,100)
        RETURN
      END IF
      X (N)=B(N)/A(N,N)
      DO 70 I=N-1,1,-1
        T=0. 0
        DO 60 J=I+1,N
          T=T+A(I,J) * X(J)
60    CONTINUE
          X (I)=B(I)-T
70    CONTINUE
100   FORMAT(1X,'FAIL')
      JS (N)=N
      DO 150 K=N,1,-1
        IF (JS(K). NE. K) THEN
          T=X (K)
          X (K)=X(JS(K))
          X (JS(K))=T

```



```

        END IF
150     CONTINUE
        RETURN
    END

```

其运行结果为

```

X( 1)= .104058D+01
X( 2)= .986956D+00
X( 3)= .935053D+00
X( 4)= .880900D+00

```

在子程序 GAUSS(A, B, N, X, L, JS) 中, 虚元 A 存放方程式系数矩阵, 体积为 $N \times N$ 。B 存放方程组右端的常数, 体积为 $1 \times N$ 。X 为一维数组, 存放方程组的解。N 为整型变量, 存放方程式阶数。L 为整型变量, 输出参数。若 $L=0$, 则表示原方程组的系数矩阵奇异, 并在子程序中印出“FAIL”; 若 $L \neq 0$, 则方程组的解由实型一维数组 X 给出。

习 题 六

1. 下列语句函数语句哪些是错误的? 说明理由。

- (1) $SN(A(I), B) = A(I) * 2$
- (2) $SM(1, 2, 3) = A + I + 2 + 3$
- (3) $SQRT(X, Y) = SQRT(X) + SQRT(Y)$
- (4) $T(Y) = Y * * 2 + 2$
 $F(X) = T(X) + 1$
- (5) $SON(I, J, K) = I * J * K$
- (6) $MON(I, J, K) = I * J * K$
 $Z = MON(1, 2, 3) + I * J + 1.2$
- (7) $ADD(X, Y, Z) = X + Y + Z$
 $T(u, v, w) = ADD(u + v, u - v, u + w) + W$
- (8) $RST(A, A) = SQRT(A * 2.0) + A$

2. 编程计算下列各式的值。

- (1) $W = \frac{X^2}{Y+Z-X} + \frac{Y^2}{X-Y+Z} + \frac{Z^2}{Y-Z+X}$
- (2) $AV = \sqrt{1.2Y^2 + 3.5Y} + \sqrt{4.5Y^2 + \pi Y + 21}$

3. 判断下列提法是否正确:

- (1) 有时 END 语句在辅程序里是不需要的。
- (2) 函数辅程序只能返回一个值到调用程序。
- (3) 函数调用 $I = STAN(X, Y)$ 是错误的, 因为 I 和 STAN 的类型不同。
- (4) 在有些情况下, 函数子程序可以和调用程序同时编译。
- (5) 在一个函数辅程序的说明里, 虚元不能是数组元素。
- (6) 在一个子程序里, 数组名允许作为自变量。
- (7) 在对语句函数的调用里所使用的实元可以是数组元素。

(8) 子程序辅程序可能没有自变量。

(9) 在辅程序中, 至少有一个 RETURN 语句放在 END 语句的前面。

4. 将下列函数 $f(x)$ 编写成函数辅程序:

$$(1) f(x) = \begin{cases} 2x^2 + \sqrt{x} - 1 & x > 0 \\ 0 & x = 0 \\ 2x^2 + \sin x & x < 0 \end{cases}$$

$$(2) f(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \frac{x^8}{8!}$$

5. 编写一个求 $n!$ 的函数辅程序, 并用它来求

$$5! + 7! + 10!$$

6. 定义一个函数辅程序, 用以求两个整数 I, J 的最大公约数 SCM ; $SCM(I, J)$

提示: 求两个整数的最大公约数的方法有多种, 其中一种方法如下:

$$M = \text{MAX}(I, J)$$

$$N = \text{MIN}(I, J)$$

$$I = \text{MOD}(M, N)$$

若 $I < > 0$ 则

$$M = N$$

$$N = I$$

$$I = \text{MOD}(M, N)$$

若 $I = 0$, 则 N 为最大公约数, 否则重复上述过程。

7. 用梯形法计算下列函数的定积分(要求编写一个通用求积分函数, 调用此函数求给定积分的值)

$$(1) \int_0^1 \sin x \, dx$$

$$(2) \int_1^4 e^{\sqrt{x}} \cos x \, dx$$

8. 用牛顿迭代法求方程 $x^3 - 2x^2 + 4x + 1 = 0$ 在 $x = 0$ 附近的一个实根, 要求其精度 $E = 10^{-6}$ 。

9. 将矩阵 $A(M, N) * B(N, P)$ 编写成一个通用子程序, 并用该程序求

$$\begin{bmatrix} 1 & 2 & 10 \\ 3.1 & 5.0 & 4 \\ 2.8 & 9.3 & 6 \end{bmatrix} \times \begin{bmatrix} 3.4 & 7.6 & 8.1 \\ 1.2 & 6.5 & 4.7 \\ 2.8 & 9.1 & 10 \end{bmatrix}$$

10. 已知一个 $N(N \geq 1)$ 个元素的递增排序的整数表, 存放在数组 $(\text{LIST}(1:M))$ 中 ($M \geq N$)。现给定一个整数 K , 若它不在表中, 则把它插入表中, 插入后的表仍按递增排序。要求把它编制成过程。

11. 下述三题语句都不齐全, 请在指定的空格处填入适当语句, 以使程序能完成相应题目的要求。

(1) 本程序是用弦切法求方程 $x^3 - 2x^2 + 7x + 4 = 0$ 的根。设 $f(x) = x^3 - 2x^2 + 7x + 4$ 。该方法是: 任给两点 x_1, x_2 , 若 $f(x_1)$ 与 $f(x_2)$ 异号, 取曲线上 $f(x_1)$ 与 $f(x_2)$ 的连线与 x 轴的交点 x , $x = x_2 - (x_2 - x_1) / (F(x_2) - F(x_1)) * F(x_2)$, 找出具有相反符号函数值的区间 (x_1, x) , (x, x_2) 之一继续作弦, 求与 x 轴的交点直到 $|f(x)| < 10^{-6}$ 或 $|x_1 - x_2| < 10^{-5}$ 为止, 取 x 为近

似解。

[程序]

```

LOGICAL S
  ①
S(A, B)=SIGN(A, B).EQ. A
10 READ(*, *)X1, X2
   F1=F(X1)
   F2=F(X2)
   IF (S(F1, F2)) GO TO 10
   FO=1.0
   DO 20, WHILE (ABS(X1-X2).GT.1E-5.AND.FO.GT.1E-6)
     ②
     FO=F(X)
     IF (S(FO, F1)) THEN
       X1=X
       ③
     ENDIF
     IF (S(FO, F2)) THEN
       ④
       ⑤
     ENDIF
20  CONTINUE
   IF (FO.GT.1E-6) X=(X1+X2)/2.0
   WRITE(*, *)X
END
  
```

(2) 本程序是输入若干个学生的学号和三门课的成绩, 求出每个学生的平均成绩, 并按平均成绩由高到低的顺序输出每个学生的学号和平均成绩(学生人数不超过 50 人)

```

PARAMETER (NR=50, NC=3)
REAL S (NR, NC), AVER (NR)
CHARACTER * 6 NUM (NR)
N=1
WRITE(*, *)'ENTER STUDENT' IS'
$   'NO. & SCORES OF 3 TESTS:'
   READ(*, 100) NUM(1), (S(N, T), J=1, NC)
   DO 10 WHILE(NUM(N).NE.' ' .AND. N.LT. NR)
     ①
     READ(*, 100) NUM(N), (S(N, J), J=1, NC)
10  CONTINUE
   N=N+1
   DO 20 I=1, N
     AVE(I)= ②
20  CONTINUE
  
```

```

      CALL SORT (AVER, NUM, N)
      DO 30 I=1, N
      WRITE(*, 200) NUM (I), AVER (I)
30    CONTINUE
100   FORMAT (A, 3F10.6)
200   FORMAT (1X, A, F10.6)
      END
      FUNCTION SUM (A, I, J, NK)
      ③
      S=0
      DO 15, K=1, J
15    S=S+A (I, J)
      ④
      END
      SUBROUTINE SORT (A, NUM, K)
      DIMENSION A(K)
      ⑤
      DO 25, I=1, K
      DO 25, J=I+1, K
      IF A (I), LT. A (J) THEN
      ⑥
      NUM (I)=NUM (J)
      ⑦
      T=A (I)
      ⑧
      A (J)=T
25    ENDIF
      END

```

(3) 本程序用来求方程 $xe^x - 1 = 0$ 的近似解, 精度 ε 为 10^{-6} 。子程序 ROOT 是用牛顿迭代法求方程的根。求方程 $f(x) = 0$ 在 x_0 附近的近似解的公式是 $x_{n+1} = x_n - f(x_n)/f'(x_n)$ 。

本题的迭代公式如下:

$$x_0 = 0.5$$

$$x_{n+1} = x_n - (x_n \cdot \exp(x_n) - 1) / [\exp(x_n) \cdot (x_{n+1})]$$

迭代终止条件为: $|x_{n+1} - x_n| < \varepsilon$ 或迭代次数达到 100。本程序在 $|x_{n+1} - x_n| < \varepsilon$ 时, 终止迭代, 并输出 x_{n+1} , 作为方程的近似解; 当迭代次数达到 100 次, 但仍未达到精度要求时, 终止迭代, 且打印 'NOT CONVERGENT'。

```

      ①
      ②
      IF (M. GE. 100) THEN
      WRITE(*, 200) M, X

```

```
200      FORMAT (1X, I4, 'NOT CONVERGENT', F10.6)
      ELSE
        WRITE( *, 300)X
300      FORMAT(1X, 'X=', F10.6)
      ENDIF
      END
      SUBROUTINE ROOT (X0, X1, F, EPS, N)
        N=0
        X1=X0
5        ③
        X1=F(X)
        N=N+1
        IF ④ GO TO 5
      END
      FUNCTION FUN(X)
        ⑤ =X - (X * EXP(X) - 1.0)/(EXP(X) * (X+1.0))
      END
```

7

FORTRAN 数据结构

7.1 概述

电子数字计算机常被称为数据处理机，就是因为它实质上是一种处理数据的装置。数据是计算机加工的对象，是一些可以输入到计算机中去的描述客观事物的符号。这些符号可以是数、字符、语言、图像等。计算机可以对这些数据进行存贮、加工和按一定形式输出。例如，1、3.5、ABC 分别是整数、实数和字符串。

数据的单位用数据元素表示。一般情况下，一组数据元素并不是杂乱无章的，而是具有某种结构形式。这里，结构形式是指数据元素间的相互关系，这种相互关系是以对数据元素的一些运算表示的。数据结构是描述一组数据元素及元素间相互关系的。在计算机高级语言中提供了实现各种数据结构的功能，用“数据类型”来表示不同的数据结构。每种高级语言都规定了它可以使用的数据类型。一般有以下三类数据结构：

① 基本类型(又称简单类型)。它们是最基本的、不可再分的数据项。例如，FORTRAN 语言中的整型、实型、双精度型、复型、逻辑型和字符型就是基本类型。

② 构造类型。它们是由其它种数据类型按一定规则构造而成的复合类型数据。例如，在 FORTRAN 语言中的数组、记录和文件等。

③ 指针类型。它们主要用于构造各种形式的动态数据结构。例如，PASCAL 和 C 语言中的链表结构等。FORTRAN 中不提供这种数据类型。

对于同一组数据，可以用不同的数据结构进行描述，数据结构选择的是否合适，直接影响算法的复杂程度。

在第二章中，已经对 FORTRAN 基本数据类型(整型、实型、双精度型、复型、逻辑型和字符型)的说明、存贮方式、取值范围及其赋值规则等给予解释。在第三章中对这些类型数据的输入/输出进行了较详细的讨论。本章主要讨论 FORTRAN 中数值型数据的类型转

换与运算以及字符型数据处理问题。

7.2 数值型数据之间的类型转换与运算

对于不同类型的数值型数据，它们的存贮方式、运算速度、取值范围和有效位数等方面都有所不同，它们之间可以按一定的规则进行运算和转换。

一、不同类型数据之间运算的规则

四种不同类型的数值型数据，在运算时，总是先将级别低的类型转换成级别高的类型，然后同一类型之间进行运算。类型的等级次序规定为：复型(1级)、双精度型(2级)、实型(3级)、整型(4级)。复型最高，整型最低。在算术表达式的运算过程中，这种类型转换是从左到右，当类型不同时才转换。

设 $C = A * B$ ，对于此乘方运算在 Microsoft FORTRAN 5.10 中变量及其运算结果的类型列在表 7.1。

表 7.1

B 的类型 C 的类型 A 的类型		整 型	实 型	双精度型	复 型
整 型		整	实	双	复
实 型		实	实	双	复
双精度型		双	双	双	复
复 型		复	复	复	复

这里我们要着重谈一下，当 A 为复型，B 为整型应当注意的问题。由于这种情况读者用得较多，所以要特别当心。如果我们要计算 $(3+4i)^4$ ，则正确的程序书写格式如下：

```
COMPLEX C
M=4
C=(3, 4) * * M
PRINT *, 'C= ', C
END
```

二、不同类型数据的赋值规则

在第二章中，已经对不同类型数据的赋值规则作了简要说明，在本章中把它们归纳为表 7.2。表中 V 为变量，e 为表达式。

表 7.2

e 的类型 v 的类型	整 型	实 型	双精度型	复 型
整 型	直接赋值	取整赋值	取整赋值	实部取整赋值, 舍虚部
实 型	转为实型赋值	直接赋值	舍多余有效数字, 转为实型赋值	实部赋值, 舍虚部
双精度型	转为双精度型赋值	转为双精度型赋值	直接赋值	实部转为双精度型赋值, 舍虚部
复 型	转为实型, 虚部取 0, 然后赋值	实部赋值, 虚部赋 0	转为实型赋实部, 虚部赋 0	直接赋值

三、不同类型数据的比较规则

在关系表达式中, 可以对两个数值型数据进行比较。若两个数值型数据的类型不同, 则首先将级别低的, 转换为级别高的再进行比较。表 7.3 给出了不同类型数据的比较规则。

表 7.3

操作数 2 比较 规则 操作数 1	整 型	实 型	双精度型	复 型
整 型	✓	✓	✓	☆
实 型	✓	✓	✓	☆
双精度型	✓	✓	✓	×
复 型	☆	☆	×	☆

在上表中✓表示允许比较, ×表示不允许比较, ☆表示只允许进行 .EQ. 或 .NE. 的比较。例如 3.0.EQ.(3.0, 5.0) 是合法的关系表达式, 而 4.0.GT.(3.0, 5.0) 是不允许的。在 Microsoft FORTRAN 5.10 中, 允许复型数与整型数, 实型数与复型数进行相等或不等的比较, 请看例 7.1。

例 7.1 阅读下列程序并写出运行结果。

```

C      EXAMPLE 7.1
      REAL P
      COMPLEX W1,W2, W3
      LOGICAL L1,L2, L3
      PRINT *, 'ENTER P'
      READ(*,10)P
      PRINT *, 'ENTER W1'
      READ(*,20)W1
      PRINT *, 'ENTER W3'
      REWIND(*,*)W3
10     FORMAT(F5.1)

```



```

20    FORMAT(2F5.1)
      L1=P.EQ.W1
      WRITE(*,30)W1
30    FORMAT(1X,'W1=',2F5.1)
      W2=(2.0,3.0)
      L2=W2.NE.W1
      L3=W2.EQ.W3
      WRITE(*,40)L1,L2
40    FORMAT(1X,'L1=',L1,3X,'L2=',L2)
      PRINT *, 'L3=', L3
      END

```

运行结果如下：

```

ENTER P
2.
ENTER W1
2., 2.
ENTER W3
(2.0, 3.0)
W1=  2.0  2.0
L1=F  L2=T
L3=T

```

四、类型转换函数

FORTAN 77 中提供的数值型数据类型转换函数见表 7.4。在表中 a 为自变量。

表 7.4

自变量 转换 结果 函 数	整 型	实 型	双精度型	复 型
INT(a)	仍为 a	对 a 取整	对 a 取整	对实部取整, 去掉虚部
REAL(a)	a 转换为实型	仍为 a	舍去多余的有效数字	取实部, 去掉虚部
DBLE(a)	a 转换为双精度型	a 转换为双精度型	仍为 a	实部转换为双精度型, 去掉虚部
CMPLX(a)	a 转换为实型 做实部, 虚部为零	a 为实部, 虚部为零	舍去 a 多余的有效数字做实部, 虚部为零	仍为 a
CMPLX(a ₁ , a ₂)	将 a ₁ 、a ₂ 转换为实型, 分别做实部和虚部	a ₁ 为实部, a ₂ 为虚部	将 a ₁ 、a ₂ 舍去多余有效数字, 分别做实部和虚部	实部为 REAL(a ₁) 虚部为 REAL(a ₂)

五、复型数据应用举例

复型数据在数学、自然科学和工程技术中是很有用的。复变函数的理论和方法是解决诸如流体力学、电磁学、热学、弹性力学中一些问题的有力工具。我们用下面的例 7.2 的电路计算为例,说明在 FORTRAN 程序中,如何利用复型变量来解决实际问题。

例 7.2 有一交流电路如图 7.1 所示。求

- (1) 等效阻抗 Z ;
- (2) 流过 R_0 、 R_1 的电流 I 、 I_1 。
- (3) 电流 I 的模 A 及相位角 P 。

假设 $U = 220 \text{ V}$, $r_0 = 10 \text{ } \Omega$, $L_0 = 10^{-3} \text{ H}$, $r_1 = 100 \text{ } \Omega$, $C_1 = 100 \text{ } \mu\text{F}$, $r_2 = 50 \text{ } \Omega$, $L_2 = 10^{-2} \text{ H}$, $C_2 = 200 \text{ } \mu\text{F}$, 电压频率 $F = 50 \text{ s}^{-1}$ 。

解 由电工学知:

$$Z_0 = r_0 + j\omega L_0$$

$$Z_1 = r_1 + \frac{1}{j\omega C_1} = r_1 - \frac{j}{\omega C_1}$$

$$Z_2 = r_2 + j\omega L_2 + \frac{1}{j\omega C_2}$$

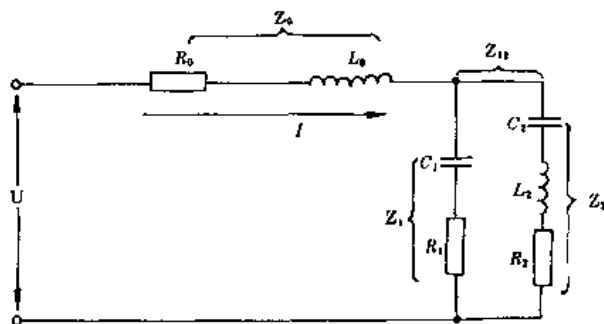


图 7.1

其中: 角频率 $\omega = 2\pi F = 50 \times 2 \times 3.14159 = 314.159$

总阻抗 $Z = Z_0 + Z_{12} = Z_0 + \frac{Z_1 Z_2}{Z_1 + Z_2}$

电流 $I = \frac{U}{Z}$, $I_1 = I \times \frac{Z_{12}}{Z_1}$

电流模 $AI = \text{ABS}(I)$

电流相位角 $PI = \text{ATAN2}(\text{AIMAG}(I), \text{REAL}(I))$

程序如下:

```
C      EXAMPLE 7.2
      IMPLICIT REAL(L)
      IMPLICIT COMPLEX(U,I,Z)
      PARAMETER (OMEGA=314.159)
      U=(220.,0.)
      WRITE(*,*)'ENTER R0,L0,R1,C1,R2,L2,C2'
      READ(*,*)R0,L0,R1,C1,R2,L2,C2
      Z0=CMPLX(R0,OMEGA*L0)
      Z1=CMPLX(R1,-1.0/(OMEGA*C1))
      Z2=CMPLX(R2,OMEGA*L2-1.0/(OMEGA*C2))
      Z12=(Z1*Z2)/(Z1+Z2)
      Z=Z0+Z12
      I=U/Z
      I1=I*Z12/Z1
      AI=ABS(I)
```

```

PI=57.295 * ATAN2(AIMAG(I),REAL(I))
PRINT * , 'Z=' , Z
PRINT * , 'I=' , I
PRINT * , 'I1=' , I1
PRINT * , 'A1=' , A1
PRINT * , 'PI=' , PI
END

```

运行结果如下：

```

ENTER R0, L0, R1, C1, R2, L2, C2
10. , 0.001, 100. , 100E-6, 50. , 0.01, 200E-6
Z=          (43.360200, -8.891928)
I=          (4.869015, 9.984947E-01)
I1=         (1.591622, 3.914836E-01)
A1=         4.970342
PI=         11.588880

```

7.3 字符型数据

事务管理是计算应用的一个重要领域。例如，图书管理系统、人事管理系统、合同管理系统、学校的学籍管理和教务管理系统等。这些管理系统都涉及字符串的排队、检索、修改和删除等操作。在这些非数值计算领域中，主要是对字符型数据进行加工、处理。为此目的，FORTRAN 语言提供了字符型数据。

对于字符型数据的概念、类型说明及其输入/输出，已经分别在第2章、第3章叙述过了，本节主要讨论字符表达式、字符子串及用于字符处理的内部函数。

一、字符子串

字符串的任一相邻接的部分叫字符子串(简称子串)。例如，字符串'XYZ'中的X，Y，Z，XY，YZ 和 XYZ 都是字符子串。用子串名标识字符子串，字符子串可以被赋值和被引用。

子串名的形式是

$$V([e_1]:[e_2])$$

$$a(s[,s]\cdots)([e_1]:[e_2])$$

其中：V 是字符变量名；

a(s[,s]...)是字符数组元素名；

e_1 和 e_2 都是整型表达式。 e_1 的值表示子串在原字符串中最左边的字符位置， e_2 的值表示子串在原字符串中最右边的字符位置。例如，A(3:8)指明了字符变量A的第3到第8列位置上的字符，而B(1,2)(3:5)指明了字符数组元素B(1,2)的第3到第5列位置上的字符。

e_1 和 e_2 的值必须满足

$$1 \leq e_1 \leq e_2 \leq \text{len}$$

其中, len 是字符变量或字符数组元素的长度。若省去 e_1 , 则隐含 e_1 的值为 1。若省去 e_2 , 则隐含 e_2 的值为 len 。 e_1 和 e_2 两者均可省去, 那么 $V(:)$ 等价于 V , 而 $a(s[,s] \cdots)(:)$ 等价于 $a(s[,s] \cdots)$ 。

例 7.3 阅读下列程序并写出运行结果。

```
C      EXAMPLE 7.3
      CHARACTER * 10 X
      DATA X/'I LOVE YOU'/
      PRINT *,X(1:1)
      PRINT *,X(3:6)
      PRINT *,X(8:10)
      PRINT *,X(:,6)
      PRINT *,X(3,:)
      PRINT *,X(:,)
```

运行结果如下:

```
I
LOVE
YOU
I LOVE
LOVE YOU
I LOVE YOU
```

二、字符表达式和字符变量的赋值

1. 字符表达式可以是字符型的常数、变量或函数, 也可以是将它们用字符运算符连接起来所构成的式子。

字符运算符只有一个“//”, 叫做并置运算符, 它的作用是将两个字符数据并接起来。例如, 'ABC'// 'XYZ' 的值是字符串 ABCXYZ。并置后的字符长度为两者之和。设 $X = 'AB \square'$, $Y = 'CD'$, 则下列都是合法的字符表达式:

```
'AB \square' // 'CD'
X // 'AB'
X // Y
```

2. 字符赋值语句

字符赋值语句的形式是

$$V = e$$

其中: V 是字符变量名, 字符数组元素名或字符子串名;

e 是字符表达式。

此语句的功能是将字符表达式 e 的值赋给 V 。若 V 的长度与 e 的长度相等, 则将 e 直接赋给 V ; 若 V 的长度大于 e 的长度, 则在 V 的右端添加空格, 补足 V 的长度; 若 V 的长度小于 e 的长度, 则将 e 中右端大于 V 的长度的那部分舍掉。例如, 设 X 是长度为 5 的字符型变量, Y 是长度为 3 的字符型变量, 执行下列语句:

```
X='12345'
```

```
Y='ABCDEF'
```

其结果为: Z 的值是'12345', Y 的值是'ABC'。

若执行下列语句:

```
X='12345'
```

```
Y='ABCDEF'
```

```
X=Y
```

则 X 的值是'ABC', Y 的值是'ABC'。若执行下列语句:

```
X='12345'
```

```
Y='ABC'
```

```
X(2:4)=Y
```

则 X 的值为'1ABC5'。若执行下列语句:

```
X='12345'
```

```
Y='ABC'
```

```
X=X(2:4)//Y
```

则 X 的值为'234AB'。

三、字符关系表达式

字符关系表达式是用关系运算符将两个字符表达式连接起来进行比较, 结果是一个逻辑值。关系运算符在 2.3 节已经介绍, 它们是 .LT.、.LE.、.GT.、.GE.、.EQ.、.NE.。

设 CH 为字符型变量, 下面是一些 FORTRAN 77 中合法的字符关系表达式:

```
'B' .GT. 'A'
```

```
'BBA' .LE. 'BBC'
```

```
CH .NE. 'ZHANG'
```

```
(CH(2:5)//'GOOD') .EQ. 'LIHUAGOOD'
```

比较两个算术表达式, 就是比较它们值的大小, 那么对于两个字符表达式是比较什么呢?

我们知道, FORTRAN 中使用的字符集在计算机中对应着一个代码系统。不同类型机器所用代码系统可能不同。目前国际上常用的代码有两种: ASCII 码(American Standard Code for Information Interchange, 即美国信息交换标准码)和 EBCDIC 码(Extended Binary-Coded Decimal Interchange Code, 即扩充的二—十进制交换码)。(见附录 I)在任一种代码系统中, 字符都按编码有一定的排列顺序。例如, 在 ASCII 代码中, 编码由小到大的字符顺序是:

```
空 格 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ? @
A B C D E F G H I J K L M N O P Q
R S T U V W X Y Z [ \ ] ^ _ a b c
d e f g h i j k l m n o p q r s t
u v w x y z { | } ~
```

字符数据的比较是按照字符的代码来进行的。两个单个字符进行比较时，代码大者为大。本节中我们使用 ASCII 代码。例如 'A' 的代码为 65，'B' 的代码为 66，则关系表达式 'A' .LT. 'B' 为真，'A' .GT. 'B' 为假。

对于两个字符串的比较，是从两者的左边起逐位逐个字符进行比较的。如果两个字符串的字符内容与排序全部相同，则认为两者相等，否则第一个出现的不同字符的比较结果，为两个字符串最终的比较结果。例如：

'ABCDE' .EQ. 'ABCDE'

'ABFFF' .LT. 'ACB'

这两个关系表达式均取真值。

读者应注意以下几点：

- ① 在 FORTRAN 中，字母的代码顺序与英文字典序一致。
- ② 空格也算做一个字符，在所有字符中它的代码最小。
- ③ 若两个字符串长度不同，且短字符串中的字符与长字符串中从左边开始的字符一相同，则长字符串的值为大。例如 'ABC' .GT. 'ABC' 为真。
- ④ 字符串中英文大小写字母代码是不同的。

四、有关字符处理的内部函数

在 FORTRAN 中，为了便于对字符型数据进行处理，系统提供了一些字符处理的内部函数。常用到的有：

1. 类型转换函数 ICHAR 和 CHAR

(1) “转换为整数”函数 ICHAR(C)

此函数的功能是得到字符串 C 中第一个字符的代码。具体代码的值取决于所用的代码系统。这里全以 ASCII 代码为例，所以 ICHAR('A') 将给出 'A' 的 ASCII 代码为 65。

(2) “转换为字符”函数 CHAR(i)

此函数是 ICHAR 的反函数，其作用是给出代码 i 所对应的字符。于是 CHAR(65) 的值为 'A'。这一函数在显示或打印某些不可见字符时尤为重要。

2. 字符比较函数

这类函数共有四个，它们的实元都是字符型表达式。通过比较字符型实元的大小来确定函数值，其值为逻辑常数。这四个函数是：

(1) “按字序大于等于”函数 LGE(e_1, e_2)

该函数的取值同字符关系表达式 e_1 .GE. e_2 。

(2) “按字序大于”函数 LGT(e_1, e_2)

该函数的取值同字符关系表达式 e_1 .GT. e_2 。

(3) “按字序小于等于”函数 LLE(e_1, e_2)

该函数的取值同字符关系表达式 e_1 .LE. e_2 。

(4) “按字序小于”函数 LLT(e_1, e_2)

该函数的取值同字符关系表达式 e_1 .LT. e_2 。

3. “长度”函数 LEN(e)

该函数实元为字符表达式，该函数的作用是给出实元中所含字符的个数。例如 C

= 'FORTRAN', 则 LEN(C) 的值为 7。

4. “子串下标”函数 INDEX(e_1, e_2)

其中, e_1, e_2 均为字符型实元。INDEX 函数的作用是指出 e_2 在 e_1 中全相同的第一个子串的开始位置(整数), 若 e_1 中不包含 e_2 , 则此函数的返回值为 0。例如:

INDEX('XIAN □ OF □ CHINA ', ' OF ') 的值为 6;

INDEX('XIAN □ OF □ CHINA ', ' A ') 的值为 3;

INDEX('XIAN □ OF □ CHINA ', 'BEIJING') 的值为 0。

五、应用举例

例 7.4 从一篇文章中查出其中所包含某个关键词组的个数。假设该词组的最大长度为 10 个字符, 这篇文章每行最多 80 个字符。框图见图 7.2(有关 OPEN 及 CLOSE 语句可参阅 9.2 节)。程序如下:

```

C      EXAMPLE 7.4
      CHARACTER CH1 * 80, CH2 * 10
      PRINT *, 'ENTER KEY WORD'
      READ(*, 4) CH2
4      FORMAT(A10)
      L = LEN(CH2)
      N = 0
      OPEN(12, FILE='F1.DAT', STATUS='OLD')
5      READ(12, 10, END=200) CH1
10     FORMAT(A80)
      I = 1
      DO 100 WHILE(I.LT. 80 - L)
          J = INDEX(CH1(I,:), CH2)
          IF (J.NE. 0) THEN
              N = N + 1
              I = J + L
          ELSE
              I = I + 1
          END IF
100    CONTINUE
      GOTO 5
200    PRINT *, 'TOTAL = ', N
      CLOSE(12)
      END
  
```

例 7.5 打印由“*”构成的 n 行金字塔图形, 该图形如图 7.3。

分析: 在这一图形中, 第 1 行要打印 1 个“*”, 第 2 行要打印 3 个“*”, 第 3 行要打印 5 个“*”, ……第 i 行要打印 $2i-1$ 个“*”。而且每一行的起始位置都比上一行往回退一个字符位置。即第 i 行的第 1 个“*”的位置应在 $(M+1-i)$ 位置上, M 为第 1 行第 1 个“*”打印的位置。

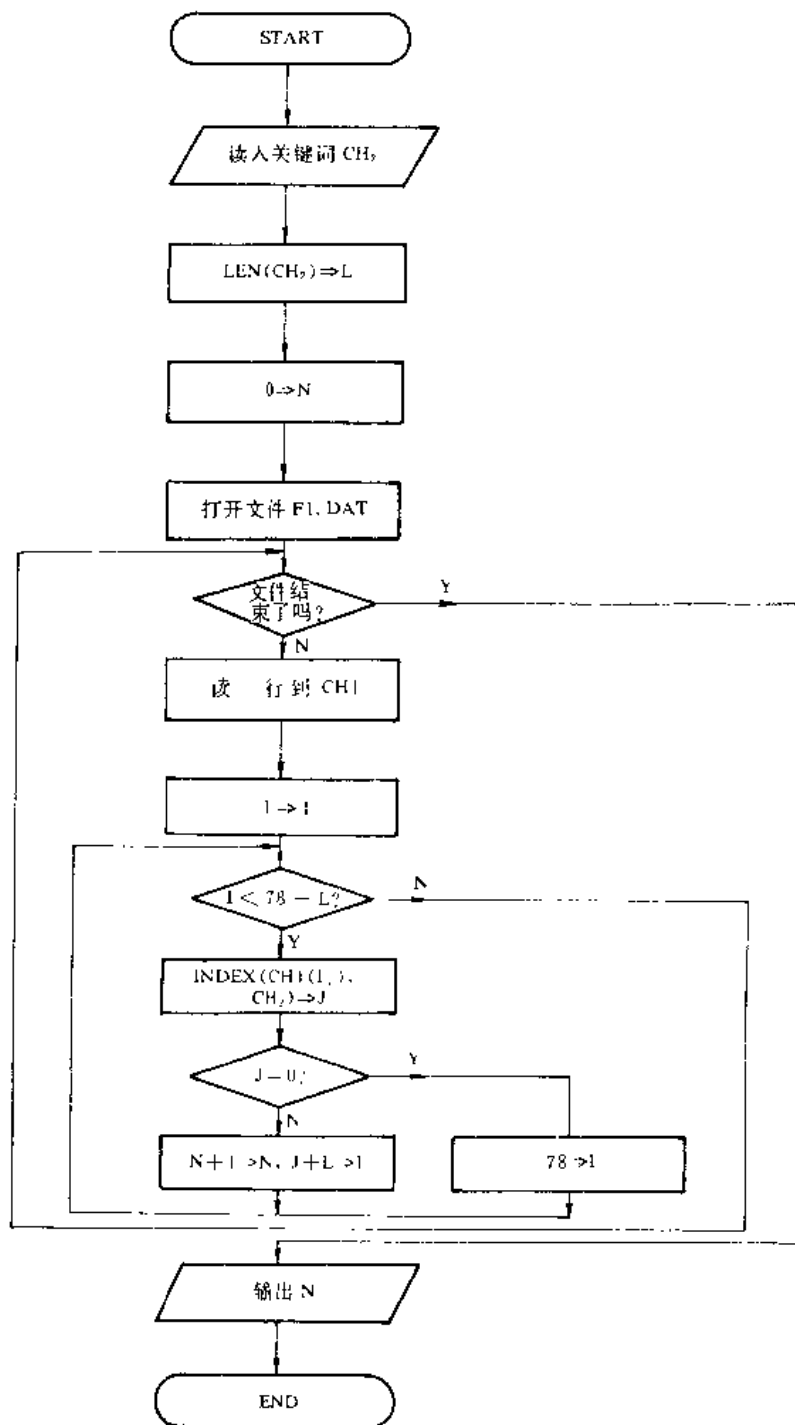


图 7.2

程序如下:

```

C      EXAMPLE 7.5
      CHARACTER CH * 78, CH1 * 40
      CH1 = ' * * * * * * * * * * * * * * * * * * * * '

```



```

PRINT *, 'ENTER N'
READ(*, *) N
M = 40
DO 10 I = 1, N
  CH = ' '
  CH(M+1-I:M+1-I) = CH
  PRINT *, CH
10 CONTINUE
END

```

图 7.3

例 7.6 设有一批国家名字，请按字典序对其进行排序。并使用对半查找方法，查找其中是否有某个国家。

解 按照题目的要求，我们将此任务分解为三个模块。

主控模块(MAIN)：完成对字符型数组 A 的说明，将国家名字存入 A 数组并调用排序及检索子模块。框图见图 7.4。

排序(SORT)：完成给定字符数组元素从小到大的排序。其排序采用希尔排序法。此方法是由 D. L. Shell 发明的。它比在 5.4 节中提到的比较排序法要快。其排序步骤如下：

假定要将 A 数组中的数据按由小到大的顺序排列。首先任意选定进行比较的两个元素的距离 $H(1 \leq H \leq N)$ ，把 $A(I)$ 与 $A(I+H)$ 比较，若 $A(I)$ 大于 $A(I+H)$ ，则把这两个元素中的数据进行对调，小的放入 $A(I)$ ，大的放入 $A(I+H)$ 。该操作步骤称为“比较对调”。对于给定的 N 个元素， I 从 1 变化到 $N-H$ ， I 每取定一个值时进行一次“比较对调”操作。 I 从 1 变化到 $N-H$ 这一全部比较对调操作称为“一趟”。如果在一趟中有对调发生（哪怕只有一次），则保持 H 值不变，重复进行比较对调工作（ I 从 1 变化到 $N-H$ ）。直到在一趟操作中没有任何对调发生为止。接着改变 H 值，使新的 H 值小于老的 H 值，用新的 H 值重复上述过程，直到 H 等于 1 且以此为距离进行一趟操作过程中没有对调发生为止，至此排序完成。其算法见框图 7.5 描述。

查找(SEARCH)：在已排好序的数组 A 中查找是否有某个国家（设国家名已存入变量 X 中）。对半查找的思想是：设三个位置指针 TOP, BOT 及 MID。BOT 指向查找范围的底部，TOP 指向查找范围的顶部， $MID = (TOP + BOT) / 2$ 指向查找范围的中间位置。

① 若 $X = A(MID)$ ，则已找到，否则进行下面步骤；

② 若 $X < A(MID)$ ，则 X 应在范围 TOP 与 MID-1 之间，故新的查找范围应为 TOP（原来位置不动）， $BOT = MID - 1$ ；

③ 若 $X > A(MID)$ ，则 X 应在范围 MID+1 与 BOT 之间，故新的查找范围应为 $TOP = MID + 1$ ，BOT（原来位置不动）。

在确定了新的查找范围之后，重复上述过程，直到找到或 $BOT < TOP$ 时没找到为止。其算法见框图 7.6 描述。

程序如下：

```

C      EXAMPLE 7.6
      PROGRAM MAIN
      CHARACTER * 8 A(1:8), X
      LOGICAL FIND

```

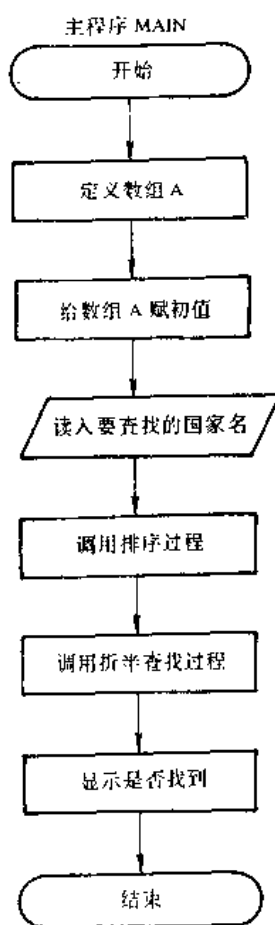


图 7.4

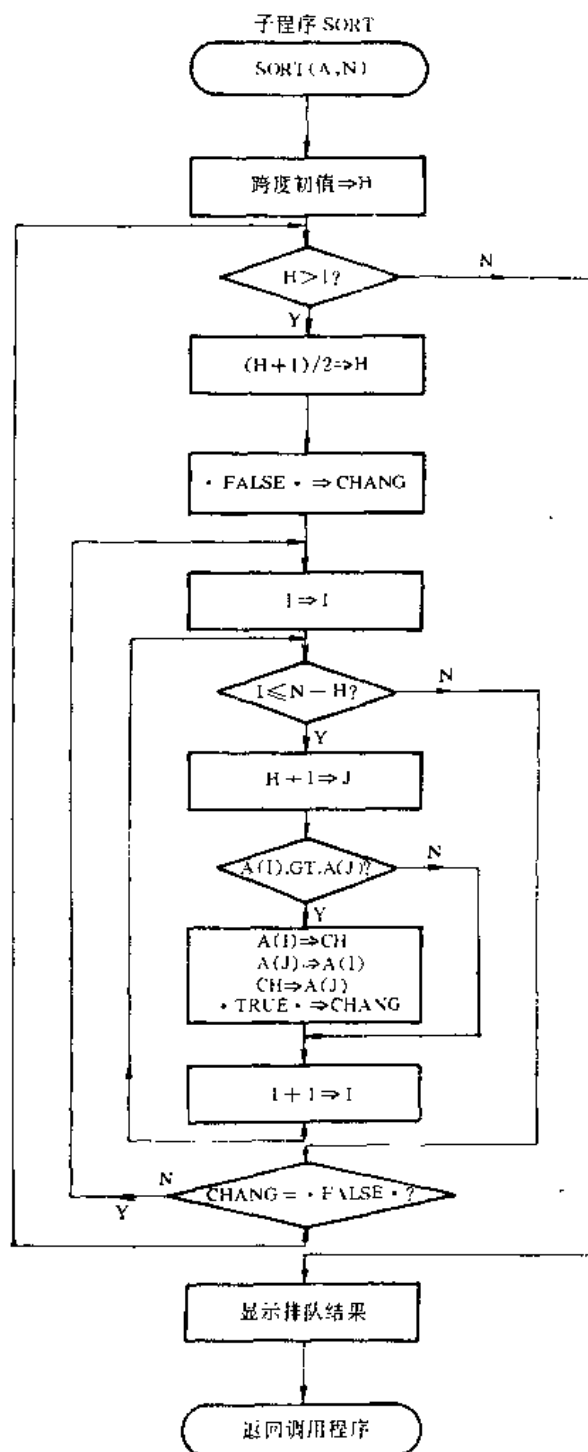


图 7.5

DATA A/'CHINA', 'JAPAN', 'CANADA', 'KOREA', 'ENGLAND'

*, 'FRANCE', 'AMERICA', 'INDIA' /

PRINT *, 'ENTER A COUNTRY NAME FOR SEARCH'

READ(*, 10)X

10 FORMAT(A8)

CALL SORT(A, 8)

CALL SERCH(A, 1, 8, X, FIND)

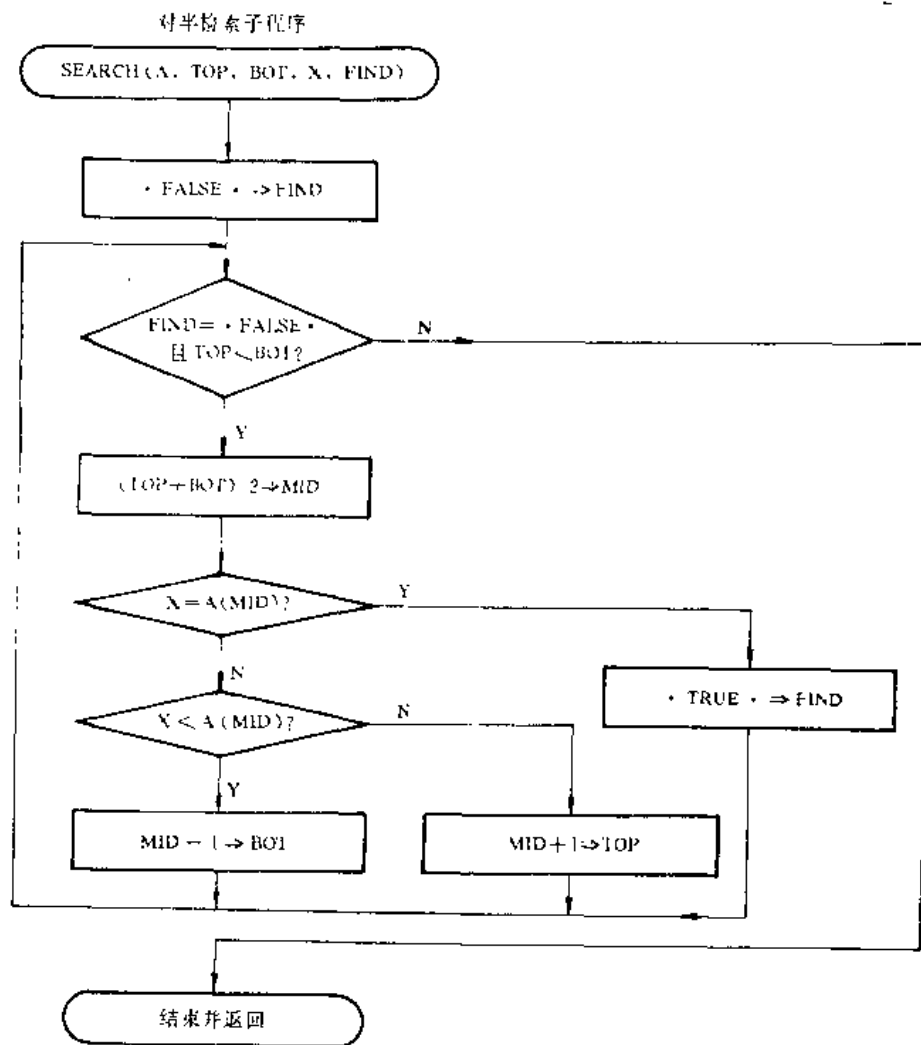


图 7.6

```

IF (FIND) THEN
  PRINT *, 'HAVE FOUND'
ELSE
  PRINT *, 'NOT FIND'
END IF
END
SUBROUTINE SORT(A, N)
  CHARACTER * 8 A(N), CH
  INTEGER H, I, J, N
  LOGICAL CHANG
  H = 10
  DO 10 WHILE (H.GT.1)
    H = (H+1)/2
20    CONTINUE
C    REPEAT
  
```

```

        CHANG=.FALSE.
        DO 30 I=1, N-H, 1
            J=H+1
            IF (A(I).GT. A(J)) THEN
                CH=A(I)
                A(I)=A(J)
                A(J)=CH
                CHANG=.TRUE.
            END IF
30      CONTINUE
C      UNTIL(.NOT. CHANG)
        IF (.NOT. CHANG) THEN
            ELSE
                GOTO 20
            END IF
10     CONTINUE
        PRINT *, A(1), A(2), A(3), A(4), A(5), A(6), A(7), A(8)
        END
        SUBROUTINE SERCH(A, TOP, BOT, X, FIND)
            INTEGER TOP, BOT, MID
            CHARACTER * 8 A(TOP : BOT), X
            LOGICAL FIND
            FIND=.FALSE.
            DO 10 WHILE(TOP.LE. BOT .AND. .NOT. FIND)
                MID=(TOP+BOT)/2
                IF (X.EQ. A(MID)) THEN
                    FIND=.TRUE.
                ELSE IF (X.LT. A(MID)) THEN
                    BOT=MID-1
                ELSE
                    TOP=MID+1
                END IF
10     CONTINUE
        END

```

运行结果为

```

ENTER A COUNTRY NAME FOR SEARCH
CHINA
AMERICA CANADA CHINA ENGLAND FRANCE INDIA JAPAN KOREA
HAVE FOUND

```

7.4 记录类型数据

记录是由不同类型数据项构成的构造类型。在现实生活中我们经常用几个不同性质的

数据项来描述某一客体。例如，在人事档案数据中经常包含下列各项：编号、姓名、性别、年龄、工龄、学历、职称和工资级别等等。

在这里编号、姓名、性别、学历、职称通常采用字符型，年龄、工龄采用整型，而工资、级别则应该采用实型。为了全面反映某一个职工的情况，常常需要把上述数据做为一个整体来处理。在 MS-FORTRAN 4.0 以上版本系统提供了描述这种数据的数据类型(记录)。

一、记录类型变量的定义

描述记录类型数据的语句(STRUCTURE 语句)一般格式：

```
STRUCTURE/type-name/
    element-declaration(s)
    :
END STRUCTURE
```

其中：

type-name 是新数据类型的名字，它应遵循 FORTRAN 命名约定。它不能与其它变量或内部函数同名。也不能与标准数据类型同名。

element-declaration(s)是一个或多个元素说明语句。用它来说明构成记录类型的每一个元素的类型。一个元素也可以是一个已定义的结构类型。

例如，把上述数据项集合定义为 ZGDA 结构类型的语句如下：

```
STRUCTURE /ZGDA/
    CHARACTER * 8  BH (编号)
    CHARACTER * 8  XM (姓名)
    CHARACTER * 1  XB (性别)
    INTEGER * 2     NL (年龄)
    INTEGER * 2     GL (工龄)
    CHARACTER * 6   XL (学历)
    CHARACTER * 6   ZC (职称)
    REAL            GZ (工资)
END STRUCTURE
```

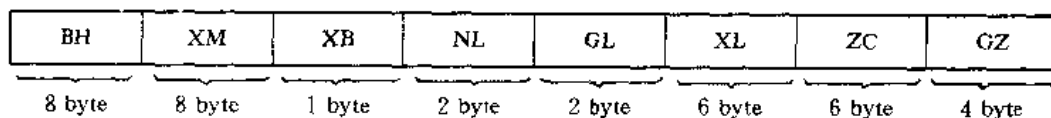
二、记录类型变量的说明

上述 STRUCTURE 语句只是定义了一个名为 ZGDA 的新变量类型，称为结构。它并不是一个特定程序的变量说明。为说明一个特定结构类型的变量，则应使用 RECORD 语句。

例如，要把 A、B 说明成由上述数据项组成的记录型变量，语句如下：

```
RECORD /ZGDA/ A, B
```

则变量在内存中被分配的存贮结构如下：



记录型变量定义的一般格式如下：

```
RECORD/type-name/vname [[attrs]][(dim)][,vname][[attrs]][(dim)]...
```

其中：type-name：用户自定义的结构类型的名字

vname: 一个变量, 数组或数组说明符, 它将被定义为这种结构类型。

attrs: 一个用逗号分隔的属性表。attrs 描述 vname(变量名)。如 EXTERN, FAR, NEAR 等。

dim: 一个维数说明符。dim 把 vname 说明成一个数组。

例如: RECORD/ZGDA/A[NEAR, REFERENCE], B(100)则把 A 说明成具有近地址属性的记录型(RECORD/ZGDA/)变量, 把 B 说明成由上述 100 条记录所构成的数组。

注意:

① STRUCTURE 语句定义一个新的数据类型。RECORD 语句把这个新的数据类型赋给一个变量。

② type-name 必须定义在 RECORD 语句之前。

③ RECORD 语句必须在所有的可执行语句之前。

三、记录类型数据的引用

结构中的每个数据项被称为“结构元素”。结构元素是通过指定元素所需序列来引用的。元素间用圆点分隔。如对上述结构变量 A 中的元素 XM 可通过如下语句来赋值:

A. XM='LI MING'

对数组 B 中第 10 个人 XM 项的引用将是 B(10). XM。

考虑到工资项又可能是由若干项组合而成的, 则上述工资档案记录又可定义如下:

```
STRUCT /GZX/
  INTEGER * 2 JBGZ      (基本工资)
  REAL JJ              (奖金)
  REAL KC              (扣除)
  REAL SF              (实发)
END STRUCTURE
```

```
STRUCTURE      /ZGDA/
  CHARACTER      * 8 BH
  CHARACTER      * 8 XM
  CHARACTER      * 1 XB
  INTEGER        * 2 NL
  INTEGER        * 2 GL
  CHARACTER      * 6 XL
  CHARACTER      * 6 ZC
  RECORD         /GZX/ GZ
END STRUCTURE
```

若有如下定义:

```
RECORD/GZDA/    C, D
RECORD/GZDA/    C1(100), D1(100)
```

则对 C1 中第 10 个人实发工资项的引用将是:

C1(10). GZ. SF

如果要把职工李明的情况存入变量 C 中, 可写出如下程序片断:

```

C. BH='19800001'
C. XM='LI MING'
C. XB='M'
C. NL=35
C. GL=17
C. XL='DAXUE'
C. ZC='GAOGONG'
C. GZ. JBGZ=275
C. GZ. JJ=128.5
C. GZ. KC=53.2
C. GZ. SF=C. GZ. JBGZ+C. GZ. JJ-C. GZ. KC

```

当然也可以用 READ 语句来分别对上述各项赋值。要输出李明的姓名、学历、职称和基本工资等项时可采用如下输出语句：

```

WRITE(*,100) C. XM, C. XL, C. GZ. JBGZ
100 FORMAT(1X, A8, 2X, A6, 2X, A6, 2X, I5)

```

从上例可以看出，结构体元素可像简单变量一样被引用。

在 MS-FORTRAN 5.10 中允许相同的结构体变量之间相互赋值。例如我们可以用语句

```
D=C
```

把结构体变量 C 的各个元素值赋到 D 的对应元素中去。但应注意不能把结构体变量当做一个整体来进行输入或输出

例如：READ(*,*) C

或 PRINT *, C

都是错误的。

四、应用举例

例 7.7 设某班级有 30 名同学。每个学生的基本情况包括姓名、性别、年龄和成绩共四项。编写程序从键盘输入每个学生的基本情况，然后按学习成绩从高到低排列并输出其结果。

我们采用记录型数组来存放每个学生的基本情况。采用交换算法按成绩进行排序。其程序如下：

```

C      EXAMPLE 7.7
      PROGRAM MAIN
      PARAMETER (N=30)
      STRUCTURE /STDD/
        CHARACTER * 10 NAME
        CHARACTER * 2 SEX
        INTEGER * 1 AGE
        REAL SCORE
      END STRUCTURE
      RECORD /STDD/ STU(N), TMP

```

```

      INTEGER I, J
C      读入数据
      DO 10 I=1, N
        WRITE( *, '(1X, A, \)') 'ENTER NAME:'
        READ( *, '(A10)') STU(I). NAME
        WRITE( *, '(1X, A, \)') 'ENTER SEX:'
        READ( *, '(A2)') STU(I). SEX
        WRITE( *, '(1X, A, \)') 'ENTER AGE'
        READ( *, *) STU(I). AGE
        WRITE( *, '(1X, A, \)') 'ENTER SCORE'
        READ( *, *) STU(I). SCORE
10     CONTINUE
C      排序过程
      DO 20 I=1, N-1
        DO 30 J=I+1, N
          IF (STU(I). SCORE. LT. STU(J). SCORE) THEN
            TMP=STU(I)
            STU(I)=STU(J)
            STU(J)=TMP
          ENDIF
30      CONTINUE
20      CONTINUE
C      输出过程
      DO 40 I=1, N
        WRITE( *, 100) STU(I). NAME, STU(I). SEX, STU(I). AGE, STU(I). SCORE
100     FORMAT(1X, A10, 2X, A2, 2X, I3, 2X, F6.2)
40      CONTINUE
      END

```

在上例中把 TMP 定义成了与数组 STU 性质相同的结构体变量，方便了数组元素的交换，当然也可以借用简单变量来把 STU(I) 和 STU(J) 中的元素分别逐个交换。

从上例可以看出，由于记录类型的数据变量的引入使得 FORTRAN 语言不仅能解决数值计算问题，而且大大增强了 FORTRAN 语言的数据处理能力。在 MS-FORTRAN 5.1 编译系统中还增加了混合编程与图形处理功能，使得 FORTRAN 语言改变了以往“只适合于数值计算”的形象。

习 题 七

1. 指出下列程序的运行结果：

```

COMPLEX X1, X2
A=1
B=2
C=5

```



```

D=B*B-4.0*A*C
PART1=-B/(2.0*A)
PART2=SQRT(ABS(D))/(2.0*A)
IF(D.GE.0.0) THEN
    X1=CMPLX(PART1+PART2,0.0)
    X2=CMPLX(PART1-PART2,0.0)
ELSE
    X1=CMPLX(PART1,PART2)
    X2=CMPLX(PART1,-PART2)
END IF
WRITE(*,*)'X1=',X1
WRITE(*,*)'X2=',X2
END

```

2. 用双精度型数据计算：

$$(1) \sum_{i=1}^n \frac{1}{i} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

直到 $\frac{1}{n} \leq 10^{-5}$ 为止。

$$(2) \frac{\pi}{2} = \frac{2 \cdot 2}{1 \cdot 3} \times \frac{4 \cdot 4}{3 \cdot 5} \times \frac{6 \cdot 6}{5 \cdot 7} \times \cdots \times \frac{(2n)^2}{(2n-1)(2n+1)}$$

求 π 的近似值(设 $n=100$)。

3. 有一交流电路(见图 7.7), 其中 $L=2 \times 10^{-2} \text{ H}$, $R=100 \Omega$, $C=10 \mu\text{F}$, 电源电压 $U=220 \text{ V}$, 求电路电流 I , 要求打印出 I 的实部、虚部、模和相位。

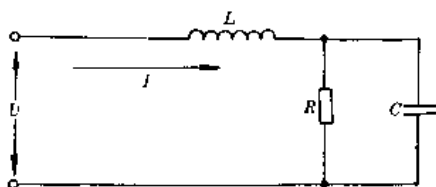


图 7.7

4. 已知三角形三个顶点的坐标分别为 $A(1.5, 2.5)$, $B(-2.5, 1)$, $C(1, -1)$ 。求三角形的面积和重心。[提示: 重心坐标 $= \frac{A+B+C}{3}$ 。]

5. 指出下列程序打印出来的结果：

```

CHARACTER*5 FI, SE, TH, FO*15
DATA FI, SE, TH/'SEE', 'JANE', 'RUN'/
PRINT *, FI//SE//TH
PRINT *, LEN(FI)
PRINT *, INDEX(FI//SE//TH, 'JANE')
FO=SE(:2) //'CKB' //FI(:3) //FI(:1) //'B' //SE(:4) //'.'
PRINT *, FO
END

```

6. 下列程序是在一篇文章中找出所有的 OLD 并用 NEW 替换之, 并给出替换的次数。

请在空格处填入合适的语句，以完成给定功能。

```

      CHARACTER LINE * 80, WORD * 3
5      REAF( *, * )N
      WORD='OLD'
      (1)
      DD 20, 1=1, N
      READ( *, * )LINE
      L=1
10     L= (2)
      IF (L.EQ.0) THEN
          L=81
      ELSE
          LINE= (3)
          NUM=NUM+1
          L= (4)
      END IF
      IF (L.LE.80) (5)
      PRINT *, LINE
20     CONTINUE
      WRITE( *, 100) NUM
100    FORMAT(1X, 'THE NUMBER OF REPLACE IS', I6)
      END
  
```

7. 已知 $f(x) = \frac{1}{x+1}$, $g(x) = \frac{1}{3x+2}$, 试将这两个函数表示的图形用曲线打印出来, 设 x 的取值范围为 0 到 10, 步长为 0.1。

8. 打印出如图 7.8 所示的图案。

```

      *
      * * *
      * * * * *
      * * * * * *
      * * * * * * *
      * * * * * * * *
      * * * * * * *
      * * * * *
      * * *
      *
  
```

图 7.8

8 数据联系

FORTRAN 中不同程序单位之间数据通讯的方法有多种。在 6.5 节中, 我们已经知道可以通过变量之间的虚实结合来传递数据, 另一种是通过公用语句来建立不同程序单位内的数据联系。对于同一程序单位中的数据, 则可通过等价语句建立起对应关系。本章主要讨论等价语句, 公用语句和数据块子程序。

8.1 EQUIVALENCE 语句(等价语句)

一、EQUIVALENCE 语句的形式

EQUIVALENCE 语句的形式为

EQUIVALENCE (nlist) [, (nlist)]...

每个 nlist 称为等价表, 表元素可以为变量名, 数组元素名, 数组名或字符子串名。每个等价表必须至少包括两个元素。各表元素之间用逗号分隔。

例如 EQUIVALENCE(A, B, C), (X, Y, Z)

EQUIVALENCE 语句的功能, 是使同一等价表中的表元素都具有相同的第一存贮单元。如上述等价语句就使得简单变量 A、B、C 共享一个存贮单元; X、Y、Z 共享一个存贮单元。

二、用 EQUIVALENCE 语句的注意事项

① 等价语句只能使同一程序单元中的变量建立起等价关系, 而不能使不同程序单元中的变量之间建立起联系。

② 在辅程序中, 外部过程的虚元名不得在等价表中出现, 若一个变量名也是一个函数名, 则该名字不得在等价表中出现。

三、EQUIVALENCE 语句的用途

等价语句是一个说明语句，它在程序单位中应位于第一个可执行语句之前。它的第一个用途是节省内存。例如，在某一程序单位中首先用到变量 A，往后不用了；又用到变量 B，往后也不用；接着用到变量 C。那么在该程序单位运行期间，三个不同的变量就得在内存中分配三个存贮单元。由于这三个变量的使用是顺序的，我们就可以使用等价语句 EQUIVALENCE(A, B, C)，使三个变量共享同一存贮单元，从而节省两个存贮单元。如果 A, B, C 都是含有 1 000 个元素的数组，那么就可以节省 2 000 个存贮单元。

等价语句的另一个用途，是允许程序员对同一量定义两个或多个名字。例如，在一个由两人或多人协作进行的大型程序设计中，由于不同程序员对变量命名的风格不同，可能导致对同一个量前后所使用的名字不同。假设前者用 S 表示面积，而后者用 AREA 表示面积。若要求二者之一对表示面积的变量进行修改而达到一致，这未免是一个苛刻的要求。这时完全可以采用 EQUIVALENCE (AREA, S)而使问题方便地得到解决。

四、EQUIVALENCE 语句的使用规则

1. 一个等价语句可使多组变量等价

例如：EQUIVALENCE(M, N, O), (X, Y), (A, B, C, D) 这里的 M, N, O, X, Y, A, B, C, D 均为简单变量。这个语句表示 M, N, O 共享同一存贮单元。X, Y 共享同一存贮单元。A, B, C, D 共享同一存贮单元。

2. 数组的等价

等价表中的表元素可以是数组名或数组元素名。

例如 DIMENSION A(3), B(2, 2), C(2, 2, 2)

EQUIVALENCE (A, B, C)

这一等价语句表明 A, B, C 3 个数组中，数组都从第一个元素起共享存贮单元。其对应方式如表 8.1 所示(每一栏代表一个存贮单元)。

表 8.1

C(1,1,1)	C(2,1,1)	C(1,2,1)	C(2,2,1)	C(1,1,2)	C(2,1,2)	C(1,2,2)	C(2,2,2)
B(1, 1)	B(2, 1)	B(1, 2)	B(2, 2)				
A(1)	A(2)	A(3)					

由于数组存放时占一个连续的存贮单元，因而等价表中两个数组元素的等价，可能会引起这两个数组中的其它元素的依次等价。

例如，上述数组 A, B, C 的等价语句

EQUIVALENCE(A, B, C)

与等价语句

EQUIVALENCE(A(1), B(1, 1), C(1, 1,1))

的作用完全相同。

又如 DIMENSION A(2, 3), B(3)

EQUIVALENCE(A(1, 2), X), (X, B(2))

由于 X 的存在, A、B 间接地建立起了等价关系。它们的等价关系用表 8.2 表示。

表 8.2

A(1, 1)	A(2, 1)	A(1, 2)	A(2, 2)	A(1, 3)	A(2, 3)
		X			
	B(1)	B(2)	B(3)		

3. 字符型实体的等价

字符型实体(包括字符型变量, 字符型数组元素与字符型数组)只能与另一个字符型实体等价。不允许字符型实体与数值型实体等价。

例如 CHARACTER CA * 6, CB * 4, CC(2, 2) * 2

EQUIVALENCE(CA, CB), (CB, CC(2, 1))

该语句建立起的字符型变量 CA, CB 及字符型数组 CC 的对应关系如表 8.3 所示。

表 8.3

CA		✓	✓	✓	✓	✓	✓
CB		✓	✓	✓	✓		
CC	CC(1, 1)	CC(2, 1)	CC(1, 2)	CC(2, 2)			
CA 中字符位置		1 2	3 4	5 6			

4. 不允许通过等价语句而建立矛盾的等价关系

我们知道在 FORTRAN 中, 一个变量在其作用域中只能与唯一的一个存储单元相对应。不能由于等价语句的使用而企图使一个变量与两个或多个存储单元相对应; 也不能使同一数组的不同数组元素分配到同一个存储单元中。

例如 DIMENSION A(5)

EQUIVALENCE (X, A(2) (A(4), X)

第一个等价表中变量 X 与 A(2) 等价, 它们共享同一存储单元, 第二个等价表中 A(4) 又与 X 等价, 它们应当共享同一存储单元。这就通过变量 X 使得 A(2) 与 A(4) 要共享同一单元(或者 X 要同时分配到两个不同的存储单元中)。这显然是不允许的。

5. 不同类型变量的等价

若同一等价表中的两个不同的元素具有相同的类型, 则在包含该等价语句的程序单位中, 这两个元素始终具有相同的值。若两个变元类型不同时, 则它们之间没有数值上的联系。当对其中的一个元素赋值时, 另一个就变成无定义的了(其值无实际含义)。

例 8.0

```

C      EXAMPLE 8.0
      REAL X
      INTEGER I
      EQUIVALENCE (X, I)
      X=21.35
      PRINT *, 'X=', X, 'I=', I
      I=16
      PRINT *, 'X=', X, 'I=', I
      END
  
```

其运行结果为

X= 21.3500001=1101712589

X= 2.242078E-441= 16

可见等价与数学上的等值不同,当同一等价表中的两个变元类型不同时,等价仅仅是为了节省存贮单元。

例 8.1 设有 3 个班的学生(一班有 45 人,二班有 50 人,三班有 70 人)参加了一门课程的考试,要求分别计算每个班的平均成绩,并按平均成绩的高低排出名次。

程序如下:

```
C   EXAMPLE 8.1
      PROGRAM MAIN
      DIMENSION A(45),B(50),C(70)
      EQUIVALENCE (A(1),B(1),C(1))
      PRINT *, 'ENTER ARRAY A(45)'
      READ( *, *) (A(I), I=1,45)
      CALL AVER(45,A,X)
      PRINT *, 'ENTER ARRAY B(50)'
      READ( *, *) (B(I), I=1,50)
      CALL AVER(50,B,Y)
      PRINT *, 'ENTER ARRAY C(70)'
      READ( *, *) (C(I), I=1,70)
      CALL AVER(70,C,Z)
      IF (X. GT. Y) THEN
        IF (Y. GT. Z) THEN
          CALL OUT('ONE', 'TWO', 'THREE', X, Y, Z)
        ELSE
          IF (X. GT. Z) THEN
            CALL OUT('ONE', 'THREE', 'TWO', X, Z, Y)
          ELSE
            CALL OUT('THREE', 'ONE', 'TWO', Z, X, Y)
          END IF
        END IF
      ELSE
        IF (X. GT. Z) THEN
          CALL OUT('TWO', 'ONE', 'THREE', Y, X, Z)
        ELSE
          IF (Y. GT. Z) THEN
            CALL OUT('TWO', 'THREE', 'ONE', Y, Z, X)
          ELSE
            CALL OUT('THREE', 'TWO', 'ONE', Z, Y, X)
          END IF
        END IF
      END IF
      END IF
```

```

END

SUBROUTINE AVER(N,W,U)
DIMENSION W(N)
U=0.0
DO 10 I=1,N
U=U+W(I)
10 CONTINUE
U=U/N
END

SUBROUTINE OUT(N1,N2,N3,X1,X2,X3)
CHARACTER * 5 N1,N2,N3,BL * 10
REAL X1,X2,X3
BL='
PRINT *, 'FIRST ', BL, 'CLASS ', N1, BL, X1
PRINT *, 'SECOND', BL, 'CLASS ', N2, BL, X2
PRINT *, 'THIRED', BL, 'CLASS ', N3, BL, X3
END

```

运行结果为

```

ENTER ARRAY A(45)
78. , 83. , 76. , 98. , 100.
ENTER ARRAY B(50)
67. , 87. , 54. , 48. , 90.
ENTER ARRAY C(70)
90. , 87. , 100. , 98. , 85.
FIRST      CLASS THREE      92.333340
SECOND     CLASS ONE        83.750000
THIRED     CLASS TWO        69.200000

```

上述程序引入了等价语句，使数组 A、B、C 共享一组存贮单元。

8.2 COMMON 语句(公用语句)

通常，一个程序单位中出现的变量或数组只在本程序单位内有意义。不同程序单位中即使名字相同的变量或数组，彼此也没有联系。不同程序单位之间数据建立联系的一种方法是，通过过程单位与引用程序单位之间变量的虚实结合。本节将介绍另一种不同程序单位之间数据联系的方法，即使用 COMMON 语句。

通过 COMMON 语句在内存开辟公共存贮区，这种公共存贮区简称公用块。利用公用块使不同的程序单位的变量或数组占用相同的存贮单元，以实现数据之间的联系。

在 FORTRAN 中，有两种公用块，无名公用块和有名公用块。一个 FORTRAN 程序中最多只能有一个无名公用块。有名公用块可以有若干个，也可以没有。

一、COMMON 语句形式

COMMON 语句的形式为

COMMON [/[cb]] nlist [[,] / [cb] / nlist]...

其中: cb 是公用块名, 每一个省去的 cb 指明无名公用块。若第一个 cb 被省掉, 则开始的两条斜线是任选的;

nlist 是变量名, 数组名或数组说明符的表。表的元素之间用逗号分隔。

二、COMMON 语句的功能

FORTRAN 编译程序在编译过程中, 当遇到 COMMON 语句时就会按照 COMMON 语句的要求在内存中开辟公共存储区, 并将 nlist 中的各元素依次分配到名为 cb 的公用块中。当 cb 省去时, 则分配到无名公用块中。

例如 COMMON u, v/X1/ A, B, C /X2/W(2)//P

这个公用语句经过编译后开辟一个无名公用块, 并将变量 u, v, p 依次分配在该区中。开辟名为 X1 和 X2 的两个有名公用块, 并将变量 A, B, C 依次分配在 X1 块中, 将数组 W 分配在 X2 块中。共占 2 个存储单元。各公用块变量的分配情况如图 8.1 所示。

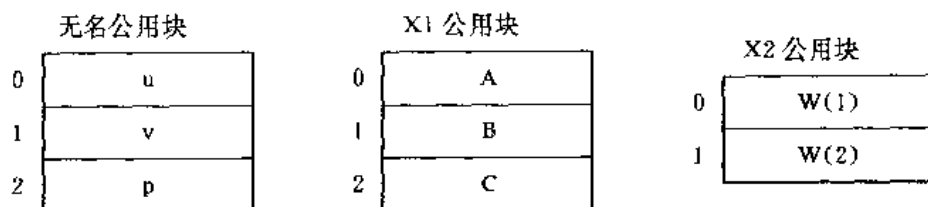


图 8.1

三、使用 COMMON 语句的注意事项

① 公用块块名是全局名, 因此它不能与外部过程名、主程序名、本程序单元中使用的内部函数名及数据块辅程序名相同。但允许公用块名与程序中的变量名、数组名同名。

例如 COMMON /C1/ A, B, C1

是合法的公用语句。

② 由于公用块中元素被分配在实际的存储单元中, 因此虚元与可调数组不能出现在公用语句中。

③ 在不同程序单位中, 同名公用块的元素占用内存单元的个数(即公用块长度)必须相等, 而对于无名公用区允许长度不相等。

④ 在一个程序单位中, 一个变量名, 数组名或数组说明符只能在公用语句中出现一次。

⑤ 同一公用块中不允许既出现数值型数据项又出现字符型数据项。

四、COMMON 语句的使用说明

1. 在不同程序单位中变量的存放次序

在不同程序单位中, 对公用块的元素, 编译程序按它们在同名公用块中出现的先后顺序一一对应地存放在公用块中, 而与名字无关。

例如 在主程序中有语句


```
DIMENSION A(2, 2)
```

```
COMMON I, A, X/W/O, Y, V
```

在辅程序中有语句

```
DIMENSION B(3)
```

```
COMMON J, B, Y/W/I, V, R
```

则公用块中变量的分配情况如图 8.2 所示。

在主辅程序中，无名公用块变量统一分配在内存无名公用区中，W 公用块的变量则统一分配在 W 区中。

无名公用块						
公用块存储单元号	0	1	2	3	4	5
主程序变量	I	A(1, 1)	A(2, 1)	A(1, 2)	A(2, 2)	X
辅程序变量	J	B(1)	B(2)	B(3)	Y	

W 公用块			
公用块存储单元号	0	1	2
主程序变量	0	Y	V
辅程序变量	I	V	R

图 8.2

2. 在同一程序单位中，可以有多个公用语句

在同一程序单位中，多个公用语句的作用相当于一个。FORTRAN 编译程序按 COMMON 语句在同一程序单位中出现的先后次序，把语句中的变量按顺序放在对应公用块的存储单元中。

例如，在主程序中有语句

```
COMMON A, B/Z1/C, D
```

```
COMMON X, I/Z1/J, R
```

在子程序中有语句

```
COMMON A1, B1/Z1/C1, D1
```

```
COMMON X1/Z1/J1
```

```
COMMON I1/Z1/R1
```

则公用块的存储分配如图 8.3 所示。

无名公用块			
主程序变量	A	B	X
子程序变量	A1	B1	X1

Z1 公用块			
主程序变量	C	D	J
子程序变量	C1	D1	J1

图 8.3

3. 利用公用语句可以进行数组说明

例如 INTEGER A

```
COMMON/Z1/A(3, 5)
```

在上述 COMMON 语句中, A(3, 5) 不是代表数组元素, 而是说明 A 是含有 15 个元素, 第一维上界为 3, 第二维上界为 5 的二维数组。COMMON 语句不但对数组 A 进行了定义, 同时将数组 A 中各元素分配到 Z1 公用存储块之中。

4. 公用语句与等价语句可以联合使用

变量或数组可以直接通过公用语句分配到公用块中, 也可以通过等价语句与公用语句联用而间接地分配到公用块中。

例如 `DIMENSION A(5), B(8)`

```
COMMON A, X
```

```
EQUIVALENCE (A(2), B(1))
```

在上述语句中, 首先 COMMON 语句将数组 A 及变量 X 分配到了无名公用块中, 接着 EQUIVALENCE 语句说明 A(2) 和 B(1) 等价。由于数组在内存的存储分配是相邻接的, 因而导致 A(3) 和 B(2), A(4) 和 B(3) 都共用同一存储单元。因此数组 B 也被分配到了无名公用块中。它们在公用块的存储安排如图 8.4 所示。

无名公用块

A(1)	A(2)	A(3)	A(4)	A(5)	X			
	B(1)	B(2)	B(3)	B(4)	B(5)	B(6)	B(7)	B(8)

图 8.4

值得一提的是公用语句与等价语句联用时, 不能导致存储单元分配发生矛盾。

下列情况是不允许的:

① COMMON A, B, C

```
EQUIVALENCE (A, B)
```

因为 COMMON 语句把变量 A, B, C 分配在无名公用块相邻的三个存储单元之中, 而 EQUIVALENCE 语句却要把 A, B 分配在同一个存储单元之中, 这显然是矛盾的。

② DIMENSION S(5)

```
COMMON X, Y, Z
```

```
EQUIVALENCE (S(2), Y), (S(4), Z)
```

由于数组在内存中是按下标顺序连续存放的。在上述语句序列中, COMMON 语句将 X, Y, Z 三个变量依次分配在内存中相邻的三个存储单元中。EQUIVALENCE 语句要求将 S(2) 与 Y 分配在同一单元中, 则 S(3) 自然应与 Z 分配在同一单元中, 但上述语句却要求 S(4) 与 Z 分配在同一单元中。这显然也是矛盾的。同一数组的不同元素不允许分配到同一存储单元中。

③ 在图 8.4 所示的结构中, 数组 A 和变量 X 被分配到了无名公用块中, 由于数组 B 与 A 的元素等价, 因而也被分配(间接分配)到了公用块中, 且使该公用块的元素个数由 6 增加到了 9。这叫做“公用块延伸”。但延伸时只能向公用块末端延伸, 而不允许向顶端延伸。

例如 `DIMENSION S(5)`

```
COMMON X, Y, Z
EQUIVALENCE(Y, S(4))
```

这种情况称为“冒顶”，是不允许的。因为 S 的元素超越了公用区的第一个存贮位置。

五、有名公用块和无名公用块的区别

有名公用块和无名公用块具有相同特性，但也有不同之处。主要表现在：

① 执行 RETURN 语句或 END 语句后，有时会使有名公用块中的元素丧失定义，但决不会使无名公用块中的元素丧失定义。

② 在有名公用块出现的可执行程序的所有程序单位中，相同名字的有名公用块的大小必须相同。而无名公用块的大小可以不同。

③ 有名公用块中的实体可用数据块辅程序中的 DATA 语句来进行初始定义，但无名公用块的实体不得用辅程序中的 DATA 语句进行初始定义。

六、应用举例

例 8.2 设 A, B 均为 5 阶方阵，求 $C=A+B$ 。

解 我们把两个五阶方阵分别存放在数组 A, B 中，并将 A, B 的和存放于数组 C 中。用子程序辅程序来完成求和过程，利用 COMMON 语句来实现主辅程序之间的数据传递。其程序如下：

```
C      EXAMPLE 8.2
      DIMENSION A(5,5),B(5,5),C(5,5)
      COMMON A,B,C
      PRINT *, 'ENTER ARRAY A(5,5)'
      READ(*,100)((A(I,J),J=1,5),I=1,5)
      PRINT *, 'ENTER ARRAY B(5,5)'
      READ(*,100)((B(I,J),J=1,5),I=1,5)
100    FORMAT(5F10.2)
      CALL ADD
      WRITE(*,200)((C(I,J),J=1,5),I=1,5)
200    FORMAT(1X,5F10.2)
      END
      SUBROUTINE ADD
      DIMENSION X(5,5),Y(5,5),Z(5,5)
      COMMON X,Y,Z
      DO 10 I=1,5
      DO 10 J=1,5
      Z(I,J)=X(I,J)+Y(I,J)
10     CONTINUE
      END
```

运行结果为

```
ENTER ARRAY A(5, 5)
1. , 1. , 1. , 1. , 1.
```

```

1., 1., 1., 1., 1.
1., 1., 1., 1., 1.
1., 1., 1., 1., 1.
1., 1., 1., 1., 1.
ENTER ARRAY B(5,5)
2., 2., 2., 2., 2.
2., 2., 2., 2., 2.
2., 2., 2., 2., 2.
2., 2., 2., 2., 2.
2., 2., 2., 2., 2.
      3.00   3.00   3.00   3.00   3.00
      3.00   3.00   3.00   3.00   3.00
      3.00   3.00   3.00   3.00   3.00
      3.00   3.00   3.00   3.00   3.00
      3.00   3.00   3.00   3.00   3.00

```

8.3 数据块辅程序

数据块辅程序和子程序辅程序一样，也是一种程序单位，但它不是可执行的程序单位，主要被用来给有名公用块中的变量和数组元素提供初值。程序单位内不允许包含可执行语句。它的形式是：

```

BLOCK DATA [sub]
      : } 辅程序体
END

```

其中：sub 是数据块辅程序的符号名，是全局变量，因而不允许与其它全局名相同，也不能与该辅程序的任何局部名相同。当省略 sub 时称为无名数据块辅程序。在一个可执行程序单元中，无名数据块辅程序只能有一个。有名数据块辅程序则可有若干个。

辅程序体只能包括注解行，DATA，COMMON，PARAMETER，DIMENSION，SAVE，EQUIVALENCE，IMPLICIT 和类型语句。

例 8.3 给出程序

```

C      EXAMPLE 8.3
      BLOCK DATA Z1
      INTEGER A
      LOGICAL L
      DIMENSION A(2,4)
      COMMON /W1/A,X,Y/W2/C1,C2,L
      DATA A/1,2,3,4,5,6,7,8/,Y/3.0/,L/.TRUE./
      END
      INTEGER Q(8)
      COMMON /W1/Q,U,V
      PRINT *,Q,U,V

```

END

其运行结果为

1	2	3	4	5	6
7	8	0.000000E+00	3.000000		

上述 Z1 数据块辅程序的作用, 就是给 W1 公用块中的整型数组 A 的元素依次赋初值 1, 2, 3, 4, 5, 6, 7, 8。给变量 Y 赋初值 3.0。给 W2 公用区中变量 L 赋初值 .TRUE.。

在使用数据块辅程序时要注意以下几点:

① 不能用此语句给无名公用块中的元素赋初值。

② 当给某个有名公用块中的部分元素赋初值时, 该有名公用块中的全部公用元素必须都在 COMMON 语句中指明。就像例 8.3 中, 只是给 W1 公用区中的 A, Y 赋初值, 但 X 也必须在 COMMON 语句中列出。

③ 在一个数据块辅程序中, 可以为多个有名公用区元素赋初值, 如例 8.3。但一个有名公用区中的元素, 只能在一个数据块辅程序中赋初值, 而且只能赋初值一次。

在用法上, 数据块辅程序可作为一个辅程序单位, 和其它辅程序一样放在主程序前面或后面的任何位置上。

习 题 八

1. 画图表示下列数组和变量的存储分配图:

(1) DIMENSION X(10), Y(6)

EQUIVALENCE (X(5), Y(2))

(2) DIMENSION X(2, 3), Y(4)

EQUIVALENCE (X(2, 1), Y(1))

(3) COMPLEX A

REAL B

INTEGER C(4)

DOUBLE PRECISION D(3)

EQUIVALENCE (D(1), A) (D(2), B), (D(3), C(3))

(4) INTEGER I(3)

CHARACTER * 8 C(5)

REAL W(8)

EQUIVALENCE (I, C), (C(3), W(4))

2. 在同一程序单位中能否出现下列等价语句:

DIMENSION A(10), B(10), C(10)

(1) EQUIVALENCE (A(3), X), (X, Y), (Y, A(5))

(2) EQUIVALENCE (A(3), B(2)), (A(5), X) (X, B(4))

(3) EQUIVALENCE (A(3), B(2)), (A(5), X), (X, C(4))

3. 阅读下面的程序, 指出公用块中存储单元的对应关系, 并写出运行结果。

```

INTEGER A(2), X
COMMON A, X
A(1)=1
A(2)=2
CALL S
PRINT *, A(1), A(2), X
END
SUBROUTINE S
INTEGER X, Y
COMMON X(2), Y
X(1)=2 * X(1)
X(2)=2 * X(2)
Y=X(1)+X(2)
END

```

4. 画出下列语句的存贮分配图:

(1) DIMENSION A(2, 3), B(6), C(10)

COMMON A, X, B

EQUIVALENCE (X, C(1))

(2) DIMENSION A(2, 2), B(3, 2), C(3, 3)

COMMON A

EQUIVALENCE (A, C), (B(1, 1), C(1, 1))

5. 在主程序和子程序中分别有以下公用语句, 指出哪些变量共享存贮单元, 哪些变量没有公用存贮单元。

主程序

(1) COMMON A, B, C, D

(2) COMMON A, B(3, 2)

(3) COMMON /C1/X(3)//Y(2)

(4) COMMON /T/X(2)//Y/T/Z(4)

(5) COMMON //Q, R

COMMON A(4)/S/P

子程序

COMMON X, Y, Z

COMMON X(2, 2), Y(2, 2)

COMMON A(2)/C1/B(3)

COMMON C(2)/T/F(3, 2)

COMMON /S/T(3)//B(4)

6. 设某班(人数不超过 70)学生参加了 3 门课程的考试, 编程计算每个人的平均成绩, 并按平均成绩的高低依次排名。

要求将平均成绩的计算及按平均成绩排队编制成两个子程序, 并采用公用块实现数据传递。

9 文 件

在前面几章的学习中，我们所用到的变量(包括简单变量和数组)统称为内存变量。因为这些变量的存贮是与内存存贮单元相联系的。对内存变量而言，一旦结束程序的运行或停电，其值就无法引用了。如果我们需要保存程序运行的中间结果及最终结果，以便本程序的下次运行或其它程序运行时能直接引用这些值(而不是从键盘再次手工输入)，这时就需要以文件的形式将这些数据保存到其它外部存贮介质上。

目前计算机中常用的外部存贮设备有光盘、磁盘、磁带、磁鼓及纸带等。存贮设备按其存取方式不同，又分为直接存取设备和顺序存取设备。光盘、磁盘、磁鼓等属于直接存取设备，而磁带、纸带等则属于顺序存取设备。

用于存放源程序的文件叫源文件，存放数据的文件叫数据文件。本章主要介绍数据文件。它主要是用 FORTRAN 中的语句来进行操作的。FORTRAN 程序可以借助于对文件进行操作的语句，经过指定的外部设备把数据输出到外部介质上，或者从外部介质上输入数据。

实际上，在 FORTRAN 程序中，数据的输入、输出操作都是以文件的方式进行的。所谓文件就是记录的序列。记录是值的序列或字符的序列。我们从键盘上输入的一行行内容，就可以看成是来自输入文件，而从屏幕或打印机上输出的一行行内容就构成了输出文件。如果存放文件的介质是磁盘，则该文件就称为磁盘文件。

9.1 FORTRAN 文件分类

一、内部文件与外部文件

FORTRAN 中的数据文件当按存贮性质来划分时，可分为外部文件和内部文件两种。

当组成文件的数据存放在外部介质上时，这类文件称为外部文件。当组成文件的数据存放在内存时，这类文件称为内部文件。

内部文件的记录是一个字符变量, 字符数组元素, 或是一个字符子串, 它提供了从内存到内存数据的传输和转换的一种手段。内部文件中记录的划分仅由数据本身而定。不允许用表控格式的格式输入/输出语句来读写内部文件记录。辅助输入输出语句也不允许用于内部文件。

下面的程序就用到了内部文件 INF:

```
CHARACTER INF * 10, Y * 5
X = 21.3
WRITE(INF, 20) X
20  FORMAT(F10.2)
Y = INF(6 : 10)
PRINT *, INF
PRINT *, Y
END
```

其运行结果为:

```
21.30
21.30
```

该程序把 X 的值写到了内部文件 INF 中。由于内部文件使用不多, 本书不再详述。下边所讨论的文件就是指外部文件。

二、顺序存取文件与直接存取文件

外部文件有两种存取方法: 顺序存取和直接存取。一般存放在顺序存取设备(例如磁带机、打印机、卡片机)上的文件只能顺序存取。而存放在直接存取设备(例如磁盘、磁鼓)上的文件既可以直接存取也可以顺序存取。对文件所允许的存取方法依赖于文件和处理系统。

当文件设置成顺序存取方式时, 读/写一个记录就必须严格按顺序进行。为了读得 50 号记录, 就必须首先读出前 49 个记录; 为了写入 80 号记录, 就必须首先写入前 79 条记录。文件记录的次序就是写入记录的次序, 且不允许用直接读写语句对文件进行操作。

当文件被设置成直接存取方式时, 读/写一个记录就可以按任意次序进行。例如, 对于 50 号记录可以直接读出, 而不必先去读前 1~49 号记录。文件的每一个记录由一个称为记录号的正整数唯一标识。当写一个记录时就指定了它的记录号。一个记录的记录号一旦建立了就不能改变, 可重写一条记录, 但不允许删除。记录以它们的记录号的顺序被排序, 但不必按记录号的次序读或写记录。例如, 当还没有写 5~9 号记录时, 却可以直接写 10 号记录。在这种文件中记录的长度必须相同。只能用直接存取输入输出语句对文件进行读/写操作, 且不允许采用表控格式对文件进行操作。

无论是顺序存取文件还是直接存取文件, 构成文件的记录要么全是有格式的, 要么全是无格式的。但顺序存取文件的最后一条记录允许是结束记录。文件一旦建立, 存在的记录就可以读取。

三、有格式文件与无格式文件

FORTRAN 中的记录有三种形式: 格式记录、无格式记录和结束文件记录。

格式记录由字符序列组成，它的长度是以字符个数来计算的。长度可以为零，长度为零的记录称为空记录。格式记录中数据的传输要经过编辑转化，即要由包含在格式说明符中的编辑信息，来实现字符序列与数据的计算机内部二进制表示形式之间的转换，所以传输速度相对较慢。

无格式记录由二进制值的序列组成，其中所包含的数据都是以 FORTRAN 数据的内部表示形式存放的。无格式记录的长度单位是由处理系统来决定的。一般来说长度为该记录所占的存贮单元数。无格式记录在传输过程中只是原样复制，不必进行形式的转换，所以传输速度较快。

结束文件记录不包含任何数据，只不过是给用户提供一种标记文件结束逻辑位置的手段。它只能作为顺序存取文件的最后一个记录，可由 ENDFILE 语句产生。

由格式记录所组成的文件称为格式文件，由无格式记录所组成的文件称为无格式文件。

格式文件在读/写时，需要经过数据的外部表示与计算机内部表示之间的格式转换，而无格式文件则无需转换，因而处理速度快。但无格式文件与各计算机所使用的内部代码有关。因此，无格式数据文件一般只供本计算机系统读取。如果要把输出数据提供给其它计算机系统，则应采用格式文件。无格式文件只能存放在磁盘或磁带中，不能在屏幕上显示，也不能输出到打印机上打印。

9.2 辅助文件操作语句

外部文件的数据是贮存在外部介质上的。在对文件进行任何数据传输操作之前，必须将文件连接到一个部件上。在文件操作的过程中，有时需要将文件指针重新定位。当文件操作完成之后，应该断开文件与部件的连接。

实现上述功能的语句就是辅助文件操作语句。它们是 OPEN 语句，CLOSE 语句，REWIND 语句和 BACKSPACE 语句。下边逐一介绍这几条语句。

一、OPEN 语句

OPEN 语句可用于把一个存在的文件连接到部件上，建立一个预连接文件，或改变文件和部件之间一些连接的说明符。一个 OPEN 语句只能打开一个文件。OPEN 语句形式如下：

OPEN (olist)

其中：olist 是一个用逗号分隔的说明符表。

说明符表中包括的说明符列于表 9.1 中。

表 9.1 OPEN 语句说明符简表

说明符	形 式	说明符取的值
部 件	[UNIT=]u(正整数表达式)	0~99 的整数
文 件	FILE=fin(字符表达式)	合法的 FORTRAN 文件名
文件状态	STATUS=sta(字符表达式)	OLD, NEW, SCRATCH, UNKNOWN

续表

说明符	形 式	说明符取的值
存取方式	ACCESS=acc(字符表达式)	SEQUENTIAL, DIRECT
格式类型	FORM=fm(字符表达式)	FORMATTED, UNFORMATTED
记录长度	RECL=rl(整表达式)	大于零
输入/输出状态	IOSTAT=ios (整变量或整数组元素)	
错 误	ERR=S(语句标号)	标号应与 OPEN 语句在同一单位中
空 格	BLANK=blnk(字符表达式)	ZERO, NULL

Olist 中必须恰好包含一个外部部件说明符, 其它说明符每种至多包含一个。各项的次序是任意的。

下面对各项说明符逐一介绍。

1. 部件说明符

一般形式:

[UNIT=]u

其中: u 为部件号。其取值范围为 0 到 99。程序在每次运行过程中, 一个部件号只能与一个文件对应; 一个文件也只能对应一个部件号。文件一旦与对应的部件号连接起来, 就可以通过部件号而直接引用文件。

当部件说明是 OPEN 语句的首项时, UNIT= 可以省略不写。

2. 文件说明符

一般形式:

FILE=fin

其中: fin 是字符表达式, 此字符表达式的值去掉尾部空格之后, 就是与部件连接的文件名。在 FORTRAN 中, 外部数据文件的名字通常由三部分组成: 存放文件的位置、文件本身的名字、文件的类别。例如, 在 PC 机上, 可对希望在 B 驱中操作、主名为 DA 的数据文件取名为 B:DA.DAT。其中, B 表示磁盘驱动器号, DA 为文件主名, DAT 为文件后缀, 表明为数据文件。

在 OPEN 语句中可以省略文件说明符一项, 若省去这个说明符, 且部件未连接到文件, 则部件连接到由处理系统决定的文件上。

3. 文件状态说明

一般形式:

STATUS=sta

它用来指明文件存在的状态。其中, sta 是字符表达式。去掉尾部空格后其值可以为 OLD, NEW, SCRATCH 或 UNKNOWN 中的任何一个。

NEW: 表示指定的文件尚不存在。一般用于建立一个新文件。

OLD: 表示指定的文件已经存在。若该文件已经存在, 则可进行读操作, 也可以进行重写或添加操作。

SCRATCH: 当文件用于暂时使用或准备在执行相应的 CLOSE 语句或程序运行结束时

删除,则该项应选取 SCRATCH。

注意:此状态不能与 FILE=fin 项共存。

UNKNOWN:这一指定表示由计算机系统来确定文件的状态。若文件存在,则文件状态说明符取值为 OLD,否则为 NEW。

若省略 STATUS=sta 这一说明项时,则意味着指定的是 UNKNOWN。

4. 存取说明符

一般形式:

ACCESS=acc

它用来指明所连接的外部文件的使用方法。acc 是字符表达式,此字符表达式的值去掉尾部空格后是 SEQUENTIAL(说明存取方式是顺序方式)或 DIRECT(说明存取方式是直接方式)。

当省略此项说明时,则表示按顺序方式存取。

5. 格式类型说明符

一般形式:

FORM=f_m

它用来指明所连接的外部文件是按有格式记录存取,还是按无格式记录存取。f_m 是一字符表达式。此字符表达式的值去掉尾部空格后是:

FORMATTED:说明文件记录按有格式的形式存放。

UNFORMATTED:说明文件记录按无格式的形式存放。

若省略此项说明,则对顺序存取方式默认为有格式的;对于直接存取文件默认为无格式的。

6. 记录长度说明

一般形式:

RECL=r_l

它用来指明直接存取文件每一记录的长度。其中,r_l 应是一个值为正整数的算术表达式。记录的长度单位为字节。若文件连接成无格式输入输出,则此项长度依赖于计算机系统数据的单位。

对于直接存取文件必须指明记录长度,对于顺序存取文件该项必须省略。

7. 输入/输出状态说明符

一般形式:

IOSTAT=ios

它用来指明 OPEN 语句执行的状态。其中,ios 是整型变量或整型数组元素。当 OPEN 语句执行时不存在错误,则系统自动给此变量赋于零;若有错误,则系统赋给此变量一个与错误类型有关的正整数值。若省略此项说明,则无法测试 OPEN 语句的执行状态。

8. 错误说明符

一般形式:

ERR=S

它用来规定当 OPEN 语句执行出错时,程序的转向。其中,S 是与 OPEN 语句在同一程序单位中的可执行语句的语句标号。当 OPEN 语句执行有错时,则转去执行该语句。

若省略此项说明符, 则 OPEN 语句执行出错时, 系统将给出错误信息, 并终止程序执行。

9. 空格说明符

一般形式:

BLANK = blk

它用来指出在格式输入/输出时, 数字字段中空格的含意。其中, blk 是一字符表达式。此字符表达式的值去掉尾部空格后是:

NULL: 说明输入/输出字段的全部空格忽略不计。如果全部是空格, 其值为零。

ZERO: 说明输入字段上的全部空格(除了前导空格外)处理成数字零。当省略该项说明时, 默认值是 NULL。此说明符只用于格式输入/输出的文件。

在该说明符表中, 常用的是前面 7 个。

例 9.1

(1) OPEN(2, FILE='F1.DAT', STATUS='OLD')

其作用为打开当前盘上 F1.DAT 文件, 指定其逻辑文件号为 2(即将 F1.DAT 连接到部件 2 上), 默认其存取方式为顺序存取方式, 文件为有格式文件。文件状态为 OLD。

(2) OPEN(10, FILE='AB.DAT', STATUS='NEW', ACCESS='DIRECT', RECL=50)

其作用为将文件 AB 连接到部件 10 上, 指定其文件状态为新文件, 存取方式为直接存取。记录长度 50, 默认为无格式文件。

(3) OPEN (FILE='B:CD.DAT', STATUS='OLD', ACCESS='DIRECT', FORM='FORMATTED', UNIT=19, RECL=30)。

其作用为将文件 B:CD.DAT 连接到部件 19 上, 指定文件状态为 OLD, 存取方式为 DIRECT。文件由有格式记录组成, 记录长度为 30 个字符。

文件一旦被打开, 文件的各种连接特性(存在状态, 存取方式, 记录格式等)就被指定。且文件与一特定的部件随之建立起连接。在其后的文件操作中, 就可以通过部件代替文件名面引用文件。

二、CLOSE 语句

CLOSE 语句用来解除部件与文件的连接, 又称为关闭文件。CLOSE 语句形式如下:

CLOSE(clist)

其中: clist 是用逗号分隔的说明符表, 所包含的说明符列在表 9.2 中。

表 9.2 CLOSE 语句说明符表

说 明 符	形 式
部 件	[UNIT =]u
输入输出状态	IOSTAT = ios
错 误	ERR = S
文件状态	STATUS = sta

在 clist 中必须恰好包含一个外部部件说明符, 其它说明至多包含一个。上表各项的具体含义如下:

部件说明[UNIT=]u; u是取值为正整数的算术表达式,由用户给出,说明所要关闭的部件号。若该项为CLOSE语句的首项时,UNIT=可以省略。

输入/输出状态说明:IOSTAT=ios,其用法与在OPEN语句中相同。

错误说明符:ERR=S,其用法与在OPEN语句中相同。

文件状态说明符:STATUS=Sta。Sta是一个字符表达式,由用户给出。去掉尾部空格其值应该为以下两种之一:

KEEP:说明在关闭操作后,与部件号连接的文件保留下来,不被删除。

DELETE:说明在关闭操作后,与部件号连接的文件不予保留,被删除。

若文件在打开时STATUS='SCRATCH',则在关闭语句中不允许用STATUS='KEEP'。

当CLOSE语句中省略DELETE说明项时,除了在OPEN语句中指定为SCRATCH的文件外,其它全部保留。

例 9.2

(1) CLOSE(2)

该语句的作用是,将例9.1中第一个OPEN语句打开的顺序有格式文件F1.DAT关闭掉。即解除部件2与原文件的连接。原文件依然保存在磁盘上。

(2) 设在某一程序中曾经执行过

```
OPEN(9, STATUS='SCRATCH', FORM='UNFORMATTED')
```

则执行CLOSE(9)语句后,将关闭无格式临时文件,且该文件一经关闭后随即删除。

允许CLOSE语句与OPEN语句不在同一程序单位中。执行CLOSE语句后,文件与逻辑部件的连接即行断开。若该文件还存在,则在同一可执行程序中,此文件还可以用OPEN语句再连接到相同或不同部件上。

三、REWIND 语句

REWIND语句称为反绕语句,它使与部件号相连的文件指针指向文件的开头位置。若文件已定位于文件的开始位置,则此语句对文件定位无影响。语句形式为

```
REWIND (alist)
```

其中:alist是下列说明符的表:

```
[UNIT=]u
```

```
IOSTAT=ios
```

```
ERR=S
```

alist必须恰好包含一个外部部件说明符,其它说明符至多包含一个。说明符的含义与OPEN语句相同。

四、BACKSPACE 语句

BACKSPACE语句称为回退语句,它使指定部件号的文件指针退回一个记录位置。语句形式为

```
BACKSPACE(alist)
```

其中,说明符alist的内容与REWIND语句相同。

注意:REWIND语句和BACKSPACE语句只能用于顺序存取文件。对于一个已连接但

不存在的文件，不允许使用 BACKSPACE 语句。虽然允许执行 REWIND 语句，但该语句执行后毫无结果。

9.3 文件输入/输出

在第3章中已经讨论了有格式输入/输出语句，该节在有关文件概念的基础上，将给出输入/输出语句的完整格式。

一、输入语句

输入操作由 READ 语句实现，它的一般形式为

READ (cilst) [iolist]

或 READ f [, iolist]

其中：iolist 是输入项表。当 iolist 被省略时，每执行一次 READ 语句就跳过一个记录。cilst 被称作控制信息表。具体形式列于表 9.3 中。

表 9.3 控制信息说明表

说明符名	形 式	说 明
部 件	[UNIT=]u(部件标识符)	必须要有
格 式	[FMT=]f(格式标识符)	无格式输入时不允许出现格式说明符
记 录	REC=rn(取值为正整数的表达式)	顺序存取时不允许出现此项
状 态	IOSTAT=ios(整型变量或数组元素)	任选
错 误	ERR=S(语句标号)	任选
文件结束	END=S(语句标号)	适用于顺序存取输入/输出

现对各项说明如下：

部件说明 [UNIT=]u：除了可以用星号外，其它与 OPEN 语句中的部件说明相同。

格式说明 [FMT=]f：f 为格式标识符，只有对格式文件进行输入/输出时才需要格式说明。对于无格式文件进行输入/输出操作时，不允许出现格式说明。

记录号说明 REC=rn：rn 是一个值为正整数的算术表达式。对于指定为直接存取方式的文件，输入/输出操作时此项必不可少。rn 的值即是要被读取的记录号。对于指定为顺序存取方式的文件，不允许出现此项说明。

状态说明 IOSTAT=ios：ios 是一个整型变量或整型数组元素。当执行 READ 语句时，系统将根据语句执行的情况给该变量赋一个值；当语句执行成功时赋给变量一个零。遇到文件结束符时赋给变量一个大于零的数；当 READ 语句在执行过程中出错时，赋给该变量一个小于零的数。其具体的正负数值由相应系统而定。

错误说明符 ERR=S：S 是与 READ 语句位于同一程序单位的可执行语句标号。其作用是当 READ 语句执行过程中发生错误时，系统自动转到该语句去执行。否则，系统将给出错误信息，并终止程序的运行。

文件结束说明符 END=S，S 是与 READ 语句位于同一程序单位的可执行语句标号。其

作用是当 READ 语句读到文件结束标志时,系统将自动转去执行此语句。若无此项选择时系统将给出出错信息,并终止程序运行。

例 9.3 假设将一个班学生的学号、姓名及数学、物理、化学三门课的考试成绩存放在 STUDENT.DAT 文件中(该文件可用编辑软件生成),其记录格式见图 9.1。

学号	姓名	数学	物理	化学
X001	ZHANGHUA	81.2	91.3	100.0
X002	WANGCHENG	92.9	86.4	76.3
⋮	⋮	⋮	⋮	⋮

图 9.1

学号为字符型,长度为 4;姓名为字符型,长度为 10;三门课成绩均为实型,宽度为 5,1 位小数。文件格式见图 9.2。请编一程序读取每一个人的成绩并显示。

X001	ZHANGHUA	81.2	91.3	100.0	X002	WANGCHENG	92.9
------	----------	------	------	-------	------	-----------	------

图 9.2

解 认为此文件存在于磁盘上,可将存取方式设置为有格式顺序存取方式,并用格式语句读其数据,程序如下:

```

C      EXAMPLE 9.3
      CHARACTER XH * 4, XM * 10
      REAL A, B, C
      WRITE(*, 30)
30     FORMAT(1X, 'XH', 12X, 'XM', 12X, 'MATHEMATICS', ' PHYSICS',
$      'CHEMICAL')
      OPEN(2, FILE='STUDENT.DAT', STATUS='OLD')
50     XH=' '
      XM=' '
      READ(2, 100, END=500) XH, XM, A, B, C
100    FORMAT(A4, A10, 3F5.1)
      WRITE(*, 200) XH, XM, A, B, C
200    FORMAT(1X, A4, 8X, A10, 10X, 3(F5.1, 6X))
      GOTO 50
500    END

```

运行结果如下:

XH	XM	MATHEMATICS	PHYSICS	CHEMICAL
X001	ZHANGHUA	81.2	91.3	100.0
X002	WANGCHENG	92.9	86.4	76.3
X003	LIMINGL	81.5	92.6	77.8

当用 OPEN 语句打开文件时,文件指针指向文件的开始位置。顺序存取时每输入/输出一条记录,文件指针就下移一个记录位置。当读到文件末尾时 END=500, 选择项发挥作

用。跳转到 500 语句，从而结束程序的运行。

二、输出语句

输出操作由 WRITE 语句和 PRINT 语句完成。

WRITE 语句的形式为

WRITE (cilst) [iolist]

PRINT 语句的形式为

PRINT f [, iolist]

其中：iolist 为输出项表；

cilst 为控制信息表列。它与表 9.3 中的项目、作用、用法基本上相同。

例 9.4 设文件格式与例 9.3 相同，把一个班 30 个学生成绩以顺序存取的方式存入该文件中，待数据录入完后，能直接查询并修改该班任一学生的成绩。

程序如下：

```

C      EXAMPLE 9.4
      CHARACTER XH * 4, XM * 10
      REAL A, B, C
      OPEN(9, FILE='STUDENT1. DAT', STATUS='NEW')
      PRINT *, 'XH XM MATH PHYS CHEM'
10     READ(*, 100, END=300) XH, XM, A, B, C
100    FORMAT(A4, A8, 3F5.1)
      WRITE(9, 200) XH, XM, A, B, C
200    FORMAT(A4, A10, 3F5.1)
      GOTO 10
300    CLOSE(9)
      OPEN(15, FILE='STUDENT1. DAT', STATUS='OLD', ACCESS='DIRECT',
$      FORM='FORMATTED', RECL=29)
20     PRINT *, 'Enter A Number(0 : 29)'
      READ(*, *) M
      IF(M. GT. 0. AND. M. LE. 29) THEN
          READ(15, 200, REC=M) XH, XM, A, B, C
          PRINT *, 'XH: ', XH
          PRINT *, 'XM: ', XM
          PRINT *, 'MATH: ', A
          PRINT *, 'PHYS: ', B
          PRINT *, 'CHEM: ', C
C1     READ(*, *) XH, XM, A, B, C      本次读入正确数据，重写回文件时即可修正
                                      原来的错误
C2     WRITE(15, 200, REC=M) XH, XM, A, B, C
          GOTO 20
      ELSE
          PRINT *, 'GOOD BYE'
          STOP

```



```
END IF
```

```
END
```

运行结果如下:

XH	XM	MATH	PHYS	CHEM
X001	ZHANGHUA	87.5	91.2	76.4
X002	WANGXIAO	96.3	65.4	100.0
X003	WANGJIAN	65.4	100.0	87.5

^ Z CTR-Z 表示数据输入结束, 程序转去执行 300 语句。

Enter A Number (0 : 29)

1

XH: X001

XM: ZHANGHUA

MATH: 87.500000

PHYS: 91.200000

CHEM: 76.400000

Enter A Number (0 : 29)

30

GOOD BYE

Stop — Program terminated.

9.4 文件的使用

一、文件的建立

文件的建立实际上就是通常所说的数据录入过程。其数据流动示意图如图 9.3 所示。

顺序存取和直接存取文件的一般操作步骤分别见图

9.4 和图 9.5。

有格式顺序文件的建立过程已在例 9.4 中举过例子。下边举一个无格式直接存取文件的例子。

例 9.5 设某工厂产品生产按日产值排列如下:

产品编码	产量	单价	产值
3	52	200.0	10 400.0
7	35	70.0	2 450.0
5	10	123.0	1 230.0
1	70	10.0	700.0

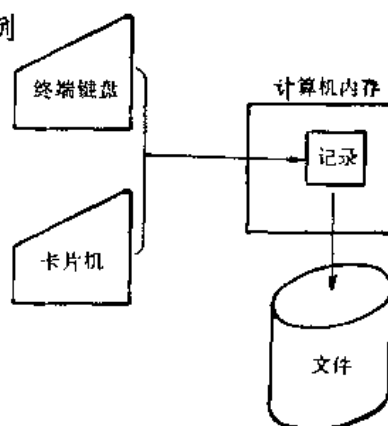


图 9.3

为了保密, 想以产品编码做为记录号, 将该数据以无格式直接存取方式存于 CP.DAT 文件中。

解 我们以整型变量 BM 和 CL 分别表示编码和产量, 以实型变量 DJ 和 CZ 代表单价

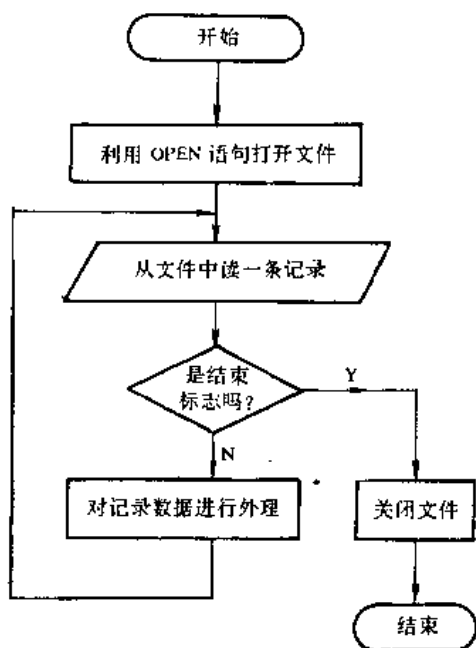


图 9.4

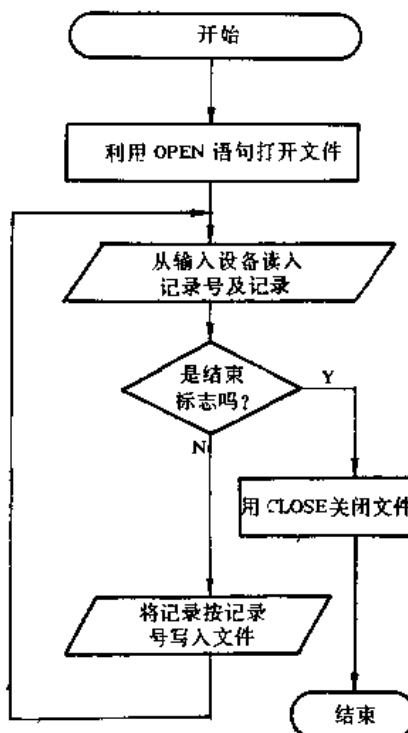


图 9.5

和产值。BM 的输入值为 -1 时表示录入结束。其程序如下：

```

C      EXAMPLE 9.5
      INTEGER BM,CL
      REAL DJ,CZ
      OPEN(7,FILE='CP.DAT',STATUS='NEW',ACCESS='DIRECT',RECL=16)
      PRINT *,'BM CL DJ      CZ      '
10     READ(*,*,END=120)BM,CL,DJ,CZ
      IF (BM.GT.0.AND.BM.LE.100) THEN
          WRITE(7,REC=BM)BM,CL,DJ,CZ
          GOTO 10
      ELSE
          PRINT *,'BM ERROR'
      END IF
120    PRINT *,'ENTER RECN '
      READ *,M
      READ(7,REC=M)BM,CL,DJ,CZ
      PRINT *,BM,CL,DJ,CZ
      CLOSE(7)
      END
  
```

在上述 OPEN 语句中不指明记录格式，默认为无格式文件，记录长度与输出项类型有关。一般整型、实型占 4 个字节，逻辑型和字符串中的一个字符占 1 个字节，双精度和复型数占 8 个字节。

上述程序运行结果如下：

```

BM CL DJ      CZ
3, 52, 200.0, 10400.0
7, 35, 70., 2450.
5, 10, 123.0, 1230.
1, 70, 10., 700.0
^ Z
ENTER RECN
5
          5          10          1230.000000          1230.000000

```

二、读取文件记录

文件一旦写入记录，就可以读取其中的记录。用直接方式建立的文件，一般可以采取顺序方式打开进行读取。用顺序方式建立的文件只要记录长度相等，也可以用直接方式打开进行读取。表控输入/输出语句由于记录长度不能人为控制，因此一般不用于直接文件的存取。

顺序存取和直接存取文件的读取步骤分别见图 9.6 和图 9.7。

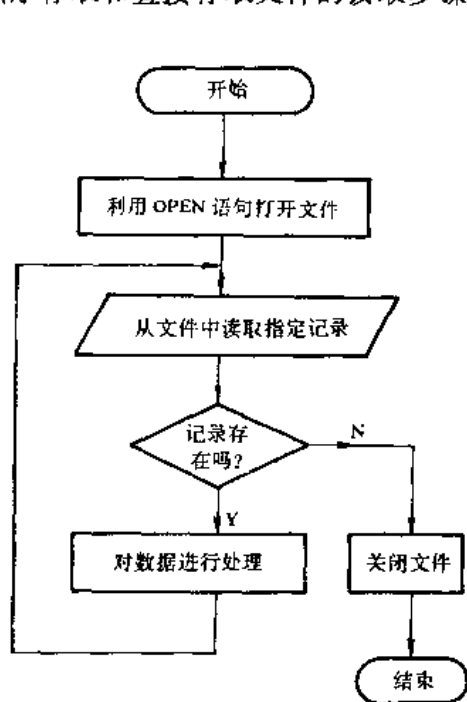


图 9.6

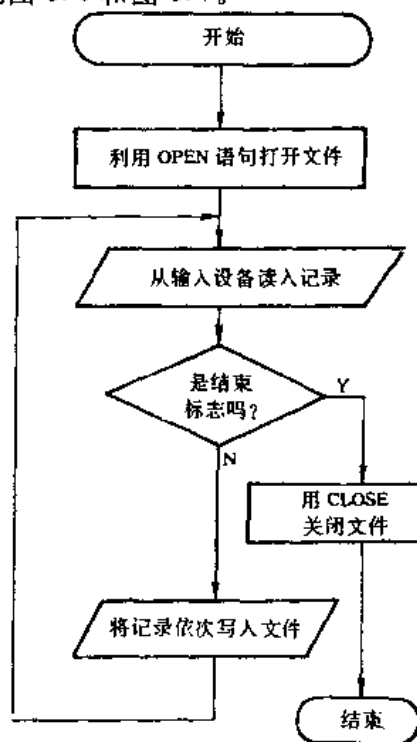


图 9.7

例 9.6 设在文件 F1.DAT 中随机存放着数 1~10 及其平方根。其格式如图 9.8 所示。

8	2.8284	3	1.7321	4	2.0000	...
← 记录 1 →		← 记录 2 →		← 记录 3 →		

图 9.8

现将该文件转贮于文件 F2.DAT 中，并按数的大小依次存放。其程序如下：

C EXAMPLE 9.6

```
OPEN(2, FILE='F11.DAT', STATUS='OLD')
```

```
OPEN(7,FILE='F12.DAT',STATUS='NEW',ACCESS='DIRECT',RECL=7
$ ,FORM='FORMATTED')
10 READ(2,100,END=300)M,X
100 FORMAT(11,F6.4)
WRITE(7,100,REC=M)M,X
GOTO 10
300 CONTINUE
CLOSE(7)
CLOSE(2)
END
```

习 题 九

1. 指出下列 OPEN 语句的错误：

- (1) OPEN(FILE='STUDENT', STATUS='OLD')
- (2) OPEN (FILE='AAA', STATUS='NEW', 15)
- (3) OPEN (32, FILE='AAA', ACCESS='DIRECT')
- (4) OPEN (24, FILE='AAA', RECL=25)
- (5) OPEN (15, FILE='TEMP', STATUS='SCRATCH')
- (6) OPEN (17, FILE='CLASS', FORM='UNFORMATTED', BLANK='ZERO')

2. 指出下面 OPEN 语句的含义：

- (1) OPEN (15, FILE='A.DAT', STATUS='OLD')
- (2) OPEN (16, FILE='B.DAT', STATUS='NEW', ACCESS='SEQUENTIAL',
FORM='FORMATTED')
- (3) OPEN (17, FILE='C.DAT', ACCESS='DIRECT', RECL=30, STATUS='OLD')
- (4) OPEN (18, FILE='D.DAT', ACCESS='DIRECT', FORM='FORMATTED',
ACCESS='DIRECT', RECL=30, STATUS='OLD')
- (5) OPEN (19, STATUS='SCRATCH')

3. 写出建立一个有格式顺序存取文件的步骤。

4. 一个文件不通过关闭语句，能否改变与其相联的部件号？一个文件在读的过程中，能否直接写入记录？

5. 一个直接存取文件可以顺序地处理吗？为什么？一个顺序存取文件可能按直接存取处理吗？为什么？

6. 将一个班学生的学号及一门课程的成绩存入顺序文件中。然后从该文件中读出数据，并对之排队。要求将排队结果在打印输出的同时，保存到另外一个文件中。

7. 利用随机函数产生 n 个随机数 (n 由键盘输入)，按由小到大的顺序放入直接文件中，每个记录放一个数。然后采用二分法对此直接文件进行查找，待查找的数由键盘输入。由人工控制查找是否结束。

8. A、B、C 三个文件中的数都按顺序存放，一个记录中包含一个数。请把三个文件中的内容合并成一个文件，合并后的文件内容仍然要有序。

最后三题要写出程序。

10

FORTRAN 高级编程指南

各种计算机语言其所以能得到生存和发展是因为它们各有所长。例如，C 语言适合于开发系统软件，FOXPRO 适合于数据处理，而 FORTRAN 语言的数值计算能力则更强。对于一个大型软件来说，由于种种原因，往往需要采用不同的几种语言来编程。

所谓混合语言程序设计，就是指采用两种或者两种以上的编程语言组合编程，彼此相互调用，进行参数传递，共享数据结构及数据信息，从而形成一种程序实体的过程。针对混合语言编程的技术实施，有人直接把它称为程序接口(program interface)，也有人称这种技术为组合程序设计(combined programming)。此外，图形往往能够比文字更直观、更准确地向人们表达信息。一个大型软件往往也需要具备图形功能。Microsoft FORTRAN 5.10 中提供了上述两个功能，现介绍如下。

10.1 FORTRAN 与汇编语言混合编程

汇编语言具有能直接面向机器硬件、面向系统低层编程，程序短，运行速度快等优点。因而在编制某些对时、空指标具有明显要求的程序段时，不得不采用汇编语言。FORTRAN 语言支持与汇编语言的混合编程。但所用 FORTRAN 编译器应与汇编系统相兼容。本节给出了 MS - FORTRAN 与 MS 汇编语言混合编程的指导与例子。

一、FORTRAN 调用汇编子程序时应遵循的原则

编写 MS - FORTRAN 的汇编语言子程序的最简单方法是使用 Microsoft 宏汇编 5.0 以上版本提供的简化的段指令。

使用 INTERFACE 语句、FORTRAN 程序可以调用一个外部汇编过程。但若不改变 FORTRAN 的缺省属性，则 INTERFACE 语句并不是严格必需的。在用 FORTRAN 程序调用外部汇编时应注意如下事项：

- (1) 将被 FORTRAN 调用的过程说明为 FAR(远过程)。

(2) 遵循 FORTRAN 调用约定:

①退出时, 过程必须将 SP 恢复到参量入栈之前的值, 这可以用 RET SIZE 指令来完成。其中, SIZE 是所有参数大小的总和。

②参量入栈的顺序和它们在 FORTRAN 程序调用语句中出现的顺序相同。即第一个参数出现在栈底, 最后一个参数出现在栈顶, 且栈底位于内存的高端位置上。

③缺省时, 若 FORTRAN 模块以大模式或巨模式编译, 则 FORTRAN 参数通过远地址方式传递; 若 FORTRAN 模块以中模式编译, 则参数将通过近地址来传递。

(3) 遵循 FORTRAN 命名约定。在缺省方式时, FORTRAN 能识别名字的前 31 个字符。

二、举例

例 10.1 编写一个 FORTRAN 程序, 在此程序中调用一个汇编子程序, 该子程序的功能是完成 $A * 2^B$ 计算。

解 在汇编语言中完成 $A * 2^B$ 运算的方法是将 A 左移 B 位。具体程序如下:

```
C    FORTRAN 程序段
      INTERFACE TO INTEGER * 2 POWER2(A, B)
      INTEGER * 2 A, B
      END
      INTEGER * 2 A, B
      A=3
      B=5
      WRITE (*, *) '3 times 2 to the power of 5 is', POWER2(A, B)
      END
```

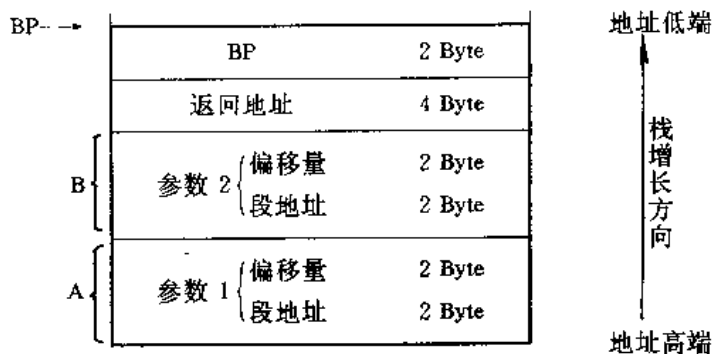
汇编过程如下:

```
• MODEL LARGE
• CODE
  PUBLIC POWER2
  POWER2 PROC
    PUSH BP          ;保存 BP 原值
    MOV BP, SP       ;设置栈址参考指针
    LES BX, [BP+10]   ;取地址到 ES:BX 中
    MOV AX, ES:BX     ;将参数 1 取入 AX 中
    LES BX, [BP+6]    ;取参数 2 地址到 ES:BX 中
    MOV CX, ES:BX     ;将参数 2 取入 CX 中
    SHL AX, CL        ;完成  $A * 2^B$  计算并将结果存入 AX 中
    POP BP           ;恢复 BP 原值
    RET 8             ;返回, 并将 SP 恢复到参数入栈之前的值
```

运行结果如下:

```
3 times 2 to the power of 5 is 96
```

在上述程序中 FORTRAN 与汇编语言传通参数的过程图示如下:



三、操作步骤：

假定你有 MS - FORTRAN 5.1 和 MS - MASM 系统，则可参考下述步骤将 FORTRAN 程序和汇编语言程序编译，连接并生成可执行文件。

(1) 分别输入 FORTRAN 及汇编语言源程序(假定存放 FORTRAN 程序文件名为 FT.FOR，汇编语言源程序文件名 MA.ASM)

(2) 用下述命令编译 FORTRAN 源程序

```
FL/C FT.FOR;
```

其中，参数/C 表示只对 FT.FOR 进行编译，而不进行连接。“;”表示默认的目标文件名为 FT.OBJ。

(3) 用 MASM 系统对 MA.ASM 程序进行汇编，命令格式为：

```
MASM MA.ASM;
```

执行上述命令后，生成的目标文件名为 MA.OBJ

(4) 用 LINK 命令对上述两个目标文件进行连接，命令格式为：

```
LINK FT.OBJ MA.OBJ;
```

执行上述命令后将生成 FT.EXE 文件。

(5) 执行 FT.EXE 文件，命令格式为：

```
FT
```

10.2 FORTRAN 与 C 语言混合编程

如果你的主程序是用 MS - FORTRAN 5.1 系统语言编写的，则可调用用 MS - C 6.0 系统编写的 C 子程序。若子程序中需要返回值时，则应在 INTERFACE 语句中将其说明为函数子程序；若无值返回时，应说明为子例行子程序(见例 10.2、例 10.3)。其操作过程也是首先将 FORTRAN 和 C 程序分别编译，然后用 LINK 程序进行连接。在连接时应给出需要连接的 FORTRAN 库和 C 库。例如：

```
LINK/NOD mod1 mod2,,,LLIBFORE+LLIBC
```

该命令将连接两个目标模块 mod1, mod2 以及 FORTRAN 系统的大模式仿真数学库和 C 的大模式程序库。

例 10.2 在 FORTRAN 主程序中输入两个不相等的整数，调用 C 子程序将两个参数中的较小者改为较大者的值。


```

C   FORTRAN 中对将被调用的 C 子程序的接口说明
    INTERFACE TO SUBROUTINE MAXPARAM[C](I, J)
    INTEGER * 2  I[NEAR,REFERENCE]
    INTEGER * 2  J[NEAR,REFERENCE]
    END

C   FORTRAN 主程序
    INTEGER * 2  I, J
    I=5
    J=7
    WRITE( *, * )'I=', I, 'J=', J
    CALL MAXPARAM(I, J)
    WRITE( *, * )'I=', I, 'J=', J
    END

C   C 子程序
    void maxparam(P1, P2)
    int near * p1; / 指向整数的近地址指针 /
    int near * p2;
    { if ( * p1 > * p2)
        * p2 = * p1;
      else
        * p1 = * p2
    }

```

例 10.3 FORTRAN 程序调用计算阶乘的 C 函数

```

C   FORTRAN 中对 C 函数的接口说明
    INTERFACE TO INTEGER * 2 FUNCTION FACT[C](N)
    INTEGER * 2  N
    END

C   FORTRAN 主程序
    INTEGER * 2  FACT
    INTEGER * 2  I, J
    I=3
    J=4
    WRITE( *, * )'The factorial of I is ', FACT(I)
    WRITE( *, * )'The factorial of J is ', FACT(J)
    END

C   计算阶乘的 C 子程序
    int fact(n)
    int n;
    { int result = 1
      while(n)
        result *= n--;
      return(result);
    }

```

10.3 图形的功能

图形可以使屏幕和数据更形象生动,更便于理解。MS - FORTRAN 5.10 中的 GRAPHICS. LIB 提供了一套完整的图形函数集。它支持基于像素的图形、实际坐标图形和字体。

一、FORTRAN 绘图的基本概念

1. 显示设备简介

(1) 视频模式

微型计算机的显示设备由显示器和插在主机板上的图形适配器组成。图形适配器的功能就是支持在显示器上的文本或图形显示。常见的图形适配器有以下几种类型:

CGA(Color Graphics Adapter)	彩色图形适配器
EGA(Enhanced Graphics Adapter)	增强图形适配器
MDA(Monochrome Display Adapter)	单色显示卡
VGA(Video Graphics Array)	视频阵列

一个图形适配器可进入一种或多种不同的“视频模式”。视频模式控制着显示器的分辨率和色彩数等。

FORTRAN 5.10 支持 17 种不同的视频模式,按其功能可以分为两大类:

文本模式——显示字符;

图形模式——显示或关闭单个的像素。

通常开机后,默认的是黑白显示 80 列文本方式。

(2) 视频的分辨率

分辨率是描述显示器的一个重要指标,它是指显示一帧图像的线数及每线上的像点数。如 VGA 的最大分辨率为 $640 * 480$,其含义是:水平方向共有 480 条线,每条线有 640 个像点。

2. FORTRAN 作图的步骤

- 1) 测试图形适配器的类型;
- 2) 根据图形适配器类型设置所需的图形模式;
- 3) 使用绘图函数作图;
- 4) 返回先前缺省的视频模式。

绘图程序的编译,必须显式连接图形库 GRAPHICS. LIB。编译命令的格式如下:

FL/FPc <文件名>.FOR GRAPHICS. LIB

3. 绘图函数的引用

FORTRAN 5.10 绘图功能的实现,是混合语言编程应用的一个最典型的例子。FORTRAN 5.10 使用的是 MSC 语言的图形函数库。从混合语言编程中我们知道,在 FORTRAN 中必须作一个接口说明才能引用 MSC 的图形库中的函数。为了简化程序设计,FORTRAN 5.10 提供了两个包含文件:FORTRAN. FI 和 FGRAPH. FD。FGRAPH. FI 是接口说明文件,图形库的所有的接口说明都在其中。FGRAPH. FD 中包含了图形库中所有的符号常量的定义和值。因此,在绘图程序中,只要将这两个文件包含进来,就无需再作接口的说明了。

4. 检测当前的视频模式

为了避免不兼容的问题,在进入某种视频方式之前,应调用 `getvideoconfig` 例程。`videoconfig` 的结构元素见表 10.1。结构 `videoconfig` 中返回了有关系统的当前信息,该结构定义在 `FGRAPH.FD` 中。`videoconfig` 的结构定义如下:

```

STRUCTURE/videoconfig/
    INTEGER * 2 numxpixels      ! number of pixels on X axis
    INTEGER * 2 numypixels      ! number of pixels on Y axis
    INTEGER * 2 numtextcols     ! number of text columns available
    INTEGER * 2 numtextrows     ! number of text rows available
    INTEGER * 2 numcolors       ! number of actual colors
    INTEGER * 2 bitsperpixel    ! number of bits per pixel
    INTEGER * 2 numvideopages   ! number of available video pages
    INTEGER * 2 mode            ! current video mode
    INTEGER * 2 adapter         ! active display adapter
    INTEGER * 2 monitor         ! active display monitor
    INTEGER * 2 memory          ! active video memory in K bytes
END STRUCTURE

```

表 10.1 `videoconfig` 的结构元素

元 素	说 明
<code>numxpixels</code>	X 轴像素个数
<code>numypixels</code>	Y 轴像素个数
<code>numtextcols</code>	文本有效列数
<code>numtextrows</code>	文本有效行数
<code>numcolors</code>	实际颜色数
<code>bitsperpixel</code>	每个像素所需位数
<code>numvideopages</code>	有效视频页数
<code>mode</code>	当前视频模式
<code>adapter</code>	活动的显示适配器
<code>monitor</code>	活动的显示监视器
<code>memory</code>	适配器的存储器(KB)

5. 设置视频方式

一旦已知道了当前的视频设置,就必须告诉图形适配器从文本方式切换到图形方式,即调用 `setvideomode`。此子程序将一个常量传递到适配器,告诉适配器以哪种方式显示(见表 10.2)。

包含文件 `FGRAPH.FD` 中用符号常量定义了相应的值。每种方式的屏幕尺寸都用图形方式的像素和文本方式的列数给出。

表 10.2 文本与图形视频方式

常 量	视 频 方 式	方式类型/硬件
\$DEFAULTMODE	恢复原方式	均适用
\$TEXTBW40	40 列文本, 16 灰度	文本/CGA
\$TEXTC40	40 列文本, 16/8 色	文本/CGA
\$TEXTBW80	80 列文本, 16 灰度	文本/CGA
\$MRES4COLOR	300×200, 4 色	图形/CGA
\$MRESNOCOLOR	320×200, 4 灰度	图形/CGA
\$HRESBW	640×200, 黑白	图形/CGA
\$TEXTMONO	80 列文本, 黑白	文本/MDA
\$HERCMONO	720×348, HGC 黑白	图形/HGC
\$MRES16COLOR	320×200, 16 色	图形/EGA
\$HRES16COLOR	640×200, 16 色	图形/EGA
\$ERESNCOLOR	640×350, 黑白	图形/EGA
\$ERESNCOLOR	640×350, 4 或 16 色	图形/EGA
\$VRES16COLOR	640×480, 黑白	图形/VGA
\$VRES16COLOR	640×480, 16 色	图形/VGA
\$MRES256COLOR	320×200, 256 色	图形/VGA/MCGA
\$ORESCOLOR	640×400, 16 色之一	图形/Olivetti

若 Setvideomode 返回零, 则硬件不支持所选择的方式。可以继续选择另一种视频方式, 直到了程序返回非零值。为了选择最高分辨率, 应该从高到低选择最高分辨率, 对该子程序进行调用。此时, 使用 SELECT CASE 结构最为方便。若硬件配置不支持任何视频方式, 则程序不能显示图形。表 10.2 给出各个视频方式名。

6. 图形坐标系

屏幕上描述像素位置的坐标系有三种: 物理坐标、视口坐标和窗口坐标。坐标系统标识了像素在屏幕上的位置, 它们具有下述约定: 位置从 0 开始(而不是 1), 若有 640 个像素, 则它们的编号为 0~639; x 坐标总是出现在 y 坐标之前。

(1) 物理坐标

假定程序调用 Setvideomode 将屏幕置为 VGA 图形方式 \$VRES16COLOR, 屏幕包含 640×480 个像素, 每个像素用坐标 x 和坐标 y 表示, 如图 10.1, 屏幕的左上角称为“原点”, 原点的坐标是(0, 0)。物理坐标是直接指示每个像素的整数(即屏幕不能显示分数形式的像素)。若用变量来表示坐标位置, 则变量必须说明为整型。

当首次进入图形方式时, “视口”或画图区域即为物理屏幕。子程序 setvieworg 改变视口原点位置。它带两个整数, 分别表示新原点的 x 和 y 坐标。例如, 下列语句将原点移到物理屏幕位置(50, 100)处。

```
CALL setvieworg(50, 100)
```

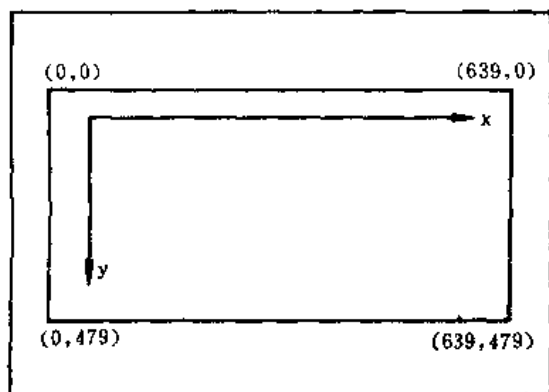


图 10.1 物理屏幕坐标

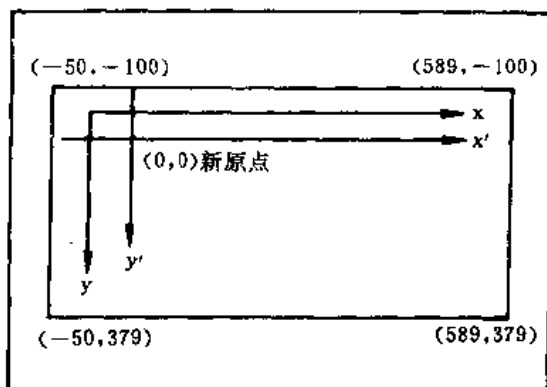


图 10.2 setviewport 改变后的坐标

此时，屏幕效果如图 10.2，新的坐标用 (x', y') 表示，相当于坐标平移。像素的数量并未改变，但用于指定点的坐标已经改变。 x 轴的范围是 $-50 \sim +589$ （而不是 $0 \sim 639$ ）， y 轴的范围是 $-100 \sim +379$ 。若适配器不是 VGA，则这些数可能不同，但效果是相同的。

新的原点将影响使用视口坐标的所有图形子程序，包括 `moveto`、`lineto`、`rectangle`、`ellipse`、`arc` 等图形子程序。例如，若在视口原点重新确定之后调用 `rectangle` 子程序，并传递值 $(0, 0)$ 和 $(40, 40)$ ，矩形左上角将在距屏幕左端 50 个像素、距屏幕顶端 100 个像素的位置。可见原点不在屏幕的左上角。

(2) 视口坐标

视口是屏幕上的实际作图区域。视口坐标表示当前视口的像素。`setviewport` 子程序在物理屏幕边界内创建一个新的视口。标准视口有两个显著特征：① 视口的原点位于左上角；② 缺省剪裁区域即为视口的边界。例如：

```
setviewport (50, 50, 200, 100)
```

建立一个如图 10.3 所示的视口。`setviewport` 子程序的参数是对角线两点的物理坐标。一旦视口建立后，视口原点位于左上角。

(3) 窗口坐标

引用物理屏幕坐标和视口坐标的函数只能接收整型参数。但是，许多应用程序可能需要频繁地使用浮点数，如速度、质量等。`setwindow` 子程序允许对屏幕进行窗口“覆盖”当前视口；窗口图形可以出现在定义为当前视口的屏幕部分。

例如，若要用图形绘出全年 12 月的平均温度，温度范围是 $-50 \sim +45^\circ\text{C}$ ，则可以在程序中增加下面一行：

```
dummy=setwindow(.TRUE., 1.0, -50.0, 12.0, 45.0)
```

第一个参量是反转标志，它的最小值放在左下角；其后是 x 和 y 的最小值和最大值；小数点表示这些数是浮点数。此时，屏幕的结构如图 10.4 所示。

注意，一月和十二月分别在屏幕的左、右两端。在类似的应用程序中， x 轴的范围最好设置为 0.0 和 13.0 ，以便提供一些附加空间。

若用 `setpixel_w` 画一个点，或用 `lineto_w` 画一条线，则数值将自动按比例变换为已建立的窗口。

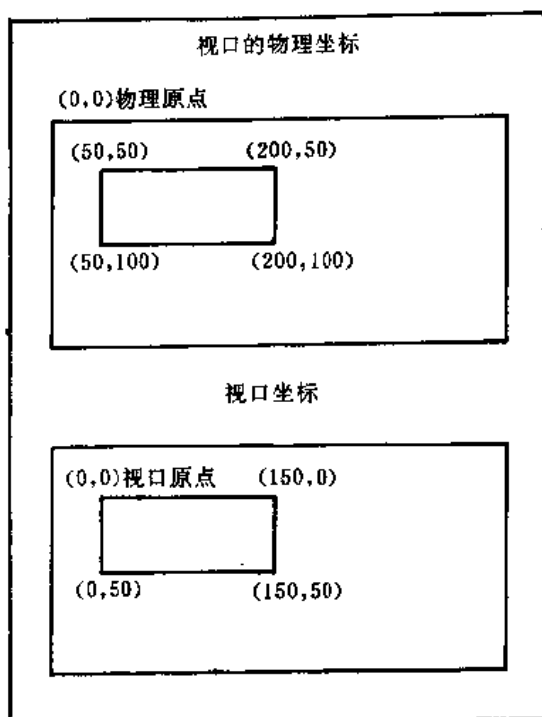


图 10.3 视口坐标

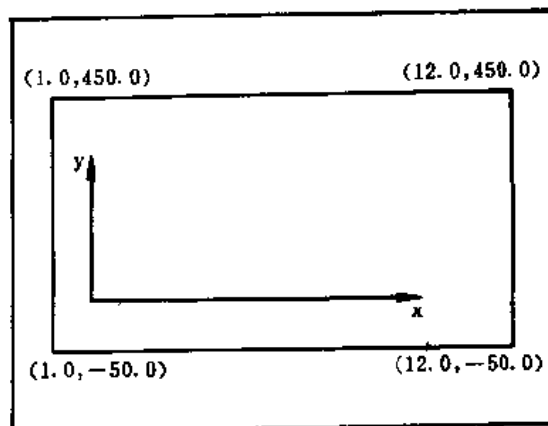


图 10.4 窗口坐标

使用浮点窗口坐标时,应遵循以下步骤:

- (1) 用 `setvideomode` 进入一种图形方式。
- (2) 用 `setviewport` 创建一个视口区域。若使用整个屏幕,则此步可省略。
- (3) 用 `setwindow` 创建一个实际窗口,并传递一个 LOGICAL 型反转标志和四个 DOUBLE PRECISION 型 x 和 y 的最小、最大值。

窗口坐标图形具有较大的灵活性。例如,根据数据类型,坐标轴可以缩小(如 151.25 到 151.45)或放大(-50000.0 至 +80000)。此外,通过改变窗口坐标,可以获得放大、缩小、拉近或平移图形的效果。

有了上述这些基本概念,我们就可以绘制一幅黑白的图形了。限于篇幅,有关 FORTRAN 绘图的图形函数和子程序及其它的一些概念就不再一一的介绍。

二、一个图形程序示例

例 10.4 编写一个图形程序 SINE。

和许多程序一样,图形程序也可采用模块形式,单个功能独立的子程序有利于设计和调试。下面的程序及其子程序说明了程序的过程,包括初始化、画图和关闭图形程序。

程序 SINE 生成了一条正弦曲线,其过程使用了许多公共图形子程序。主程序采用了模块化结构,它调用了 5 个函数,分别执行实际的图形命令。

程序 SINE 的主程序如下:

```
CC SINE.FOR—基本图形命令演示程序
INCLUDE 'FGRAPH.FI'
```

```

CALL graphicsmode( )
CALL drawlines( )
CALL sinewave( )
CALL drawshapes( )
CALL endprogram( )
END

```

1. 激活一种图形方式

在显示图形之前，程序必须指定一种图形方式，然后调用 `setvideomode` 将适配器设置为该图形方式。

程序 SINE 调用子程序 `graphicsmode` 打开图形方式。子程序选择当前适配器可用的最高的分辨率。

```

SUBROUTINE graphicsmode( )
INCLUDE 'FGRAPH. FD'
INTEGER * 2    dummy, maxx, maxy
RECORD /videoconfig/myscreen
COMMON        maxx, maxy
C
C 测试适配器类型，并设置为最高分辨率
C
CALL getvideoconfig(myscreen)
SELECT CASE(myscreen.adapter)
CASE ( $ CGA )
    dummy=setvideomode( $ HRESBW )
CASE ( $ OCGA )
    dummy=setvideomode( $ ORESCOLOR )
CASE ( $ EGA, $ OEGA )
    IF(myscreen.monitor .EQ. $ MONO) THEN
        dummy=setvideomode( $ ERESNOCOLOR )
    ELSE
        dummy=setvideomode( $ ERESCOLOR )
    END IF
CASE ( $ VGA, $ OVGA, $ MCGA )
    dummy=setvideomode( $ VRES2COLOR )
CASE ( $ HGC )
    dummy=setvideomode( $ HERCMONO )
CASE DEFAULT
    dummy=0
END SELECT
IF(dummy .EQ. 0)STOP 'Error: cannot set graphics mode'
C
C Determine te minimum and maximum dimensions.
CALL getvideoconfig(myscreen)

```

```

maxx=mymcreen.numxpixels-1
maxy=mymcreen.numypixels-1
END

```

子程序先调用 `getvideoconfig`，并传递 `videoconfig` 结构 `mymcreen` 的地址。在 `videoconfig` 结构中，称为 `adapter` 的元素告诉所使用的适配器的类型。程序根据适配器信息，用 `SELECT CASE` 结构来选择一种适当的图形方式。

为获得程序所选择的视频方式的分辨率，必须将 `videoconfig` 结构传递到 `getvideoconfig` 子程序。

```

CALL getvideoconfig(mymcreen)
maxx=mymcreen.numxpixels-1
maxy=mymcreen.numypixels-1

```

若计算机使用了 EGA 卡，则子程序 `graphicsmode` 将视频方式设置为 `$ERESNOCOLOR`。屏幕的水平尺寸是 640 像素，而垂直尺寸是 350 像素。上述两个赋值语句，分别将这些值减 1 后赋给 `maxx` 和 `maxy`。水平分辨率可以是 640。但像素是按 0 到 639 编排的，所以，变量 `maxx` 最大的像素号为像素的总数减 1。

将当前使用的图形适配器设置为最高分辨率，使用下述程序段则更简单：

```

SUBROUTINE InitGraphics(bkColor, Color)
  INCLUDE 'FGRAPH.FD'
  INTEGER * 2 dummy
  INTEGER * 4 bkColor, Color
C
C   Find graphics mode, inti graphics
C
  IF (setvideomode($MAXRESMODE).EQ.0)
    STOP 'Error; cannot set graphics mode'
  dummy=setcolor(Color)
  dummy=setbkcolor(bkColor)
  CALL CLEARSCREEN($GCLEARSCREEN)
END

```

该子程序的功能是设置最高的分辨率，绘图的背景色和前景色(绘图用色)。

两个短小的函数将 1000×1000 像素的假想屏幕转换为有效的视频方式。从现在开始程序假定在每个方向上有 1 000 个像素。为画出屏幕上的点，`newx` 和 `newy` 将每个点改变为实际的坐标(像素)。

```

CC   NEWX—This function finds new x coordinates.
      INTEGER * 2 FUNCTION newx(xcoord)
      INTEGER * 2 xcoord, maxx, maxy
      REAL * 4 tempx
      COMMON maxx, maxy
      tempx=maxx/1000.0
      tempx=xcoord * tempx+0.5
      newx=tempx

```



```

      END
CC    NEWY—This function finds new y coordinates.
      INTEGER * 2 FUNCTION newy(ycoord)
      INTEGER * 2 ycoord, maxx, maxy
      REAL * 4 tempy
      COMMON maxx, maxy
      tempy = maxy/1000.0
      tempy = ycoord * tempy + 0.5
      newy = tempy
      END

```

2. 在屏幕上画线

SINE 程序接着调用子程序 drawlines。顾名思义，这个子程序用于在屏幕上画线。在本例中，子程序沿屏幕边界画出一个矩形，以及三条水平线将矩形分成四个部分。

```

CC    DRAWLINES—This subroutine draws a box and several lines.
      SUBROUTINE drawlines( )
      INCLUDE 'FGRAPH.FD'
      EXTERNAL newx, newy
      INTEGER * 2 dummy, newx, newy, maxx, maxy
      RECORD /xycoord/xy
      COMMON maxx, maxy

C
C    Draw the box.
C
      dummy = rectangle( $GBORDER, 0, 0, maxx, maxy)
      CALL setvieworg(0, newy(INT2(500)), xy)

C
C    Draw the lines.
C
      CALL moveto(0, 0, xy)
      dummy = lineto(newx(INT2(1000)), 0)
      CALL setlinestyle( #AA3C)
      CALL moveto(0, newy(INT2(-250)), xy)
      dummy = lineto(newx(INT2(1000)), newy(INT2(-250)))
      CALL setlinestyle( #8888)
      CALL moveto(0, newy(INT2(250)), xy)
      dummy = lineto(newx(INT2(1000)), newy(INT2(250)))
      END

```

矩形子程序的第一个参量是“填充标志”。它等于 \$GBORDER 或 \$GFILLINTERIOR。若矩形只需四条边，则可选择 \$CBORDER；若要画一实心矩形（用当前色和填充模式填充），则可选择 \$GFILLINTERIOR。

第二个和第三个参量是矩形左上角的 x 和 y 方向的坐标。第四和第五个是矩形右下角的坐标。由于两个参数分别是 (0, 0) 和 (maxx, maxy)，所以调用 rectangle 即沿屏幕画出其

边界。

```
dummy=rectangle(SGBORDER, 0, 0, maxx, maxy)
```

调用子程序 setfinety, 将线型实线改为短杠线(dashed line)。一个 16 位序列告诉子程序画线的模式: 1 表示一个实线段, 而 0 表示一个空线段。因此, 对计算机来说, 实线相当于 1111111111111111; 短杠线则类似于 1111111100000000(长杠线)或 1111000011110000(短杠), 您可以选择 1 和 0 的任意组合。这个 16 位二进制数也可以表示为一个十六进制数或十进制数。下例中, 十六进制常量 #AA3C 等于二进制值 1010101000111100。但是也可以使用 43 580 表示该参量。

为了画线, 首先必须设置线型, 然后再移动到线的起点, 再从起点到终点画一条线。例如, 程序 SINE 使用了下述画线的语句:

```
CALL setlinestyle( #AA3C)
CALL moveto(0, newy(INT2(-250)), xy)
dummy=lineto(newx(INT2(1000)),
+ newy(INT2(-250)))
```

子程序 moveto 将一虚拟像素光标移到屏幕上的某个点上(实际上屏幕看不见); 子程序 line 画一条直线。负值 -250 看起来像一个不可能的屏幕坐标, 但是, 程序将调用 setvieworg, 改变视口坐标系, 使屏幕的顶部为 -500, 而底部为 +500。这好像有点不合常规, 但此例说明了该子程序可以将视口坐标改变为任意的大小或方向。

3. 画正弦曲线

有了坐标轴和方框之后, SINE 程序开始画正弦线。于程序 sinewave 计算两个周期中的 x 和 y 的位置, 并将这些点画在屏幕上。

```
CC    SINEWAVE—This subroutine calculates and plots a sine wave.
      SUBROUTINE sinewave( )
      INCLUDE 'FGRAPH.FD'
      INTEGER * 2 dummy, newx, newy, locx, locy, i
      DOUBLE PRECISION rad, PI
      EXTERNAL newx, newy
      PARAMETER(PI=-3.14159)

C
C    Calculate each position and display it on the screen.
C
      DO i=0, 999, 3
        rad=-SIN(PI*i/250.0)
        locx=newx(i)
        locy=newy(INT2(rad*250.0))
        dummy=setpixel(locx, locy)
      END DO
      END
```

在上述子程序中, $\text{SIN}(\text{PI} * i / 250.0)$ 中的 $\text{PI} * i / 250.0$ 的含意是在 $0 \sim 4\pi$ 内绘制 SIN 图形。 $0 \sim 4\pi$ 被分配到长度为 1 000 个单位的轴上, 故有 $4\pi = 1\,000$ 个单位, 1 个单位就相当于 $\text{PI} / 250.0$ 个弧度。本例中的步长取的是 3 个单位。

4. 增加形状

在画出正弦曲线之后, 程序 SINE 调用 drawshapes, 在屏幕画两个矩形和两个椭圆。填充标志分别为 \$GBORDER 和 \$GFILLINTERIOR;

```
CC    DRAWSHAPES—This subroutine draws two boxes and two ellipses.
      SUBROUTINE drawshapes( )
      INCLUDE 'FGRAPH.FD'
      EXTERNAL newx, newy
      INTEGER * 2 dummy, newx, newy

C
C    Create a masking (fill) pattern.
C
      INTEGER * 1 diagmask(8), linemask(8)
      DATA diagmask / #93, #C9, #64, #B2, #59, #2C, #96, #4B/
      DATA linemask / #FF, #00, #7F, #FE, #00, #00, #00, #CC/

C
C    Draw the rectangles.
C
      CALL setlinestyle( #FFFF)
      CALL setfillmask(diagmask)
      dummy=rectangle( $GBORDER, -
+          newx(INT2(50)), newy(INT2(-325)),
+          newx(INT2(200)), newy(INT2(-425)))
      dummy=rectangle( $GFILLINTERIOR,
+          newx(INT2(550)), newy(INT2(-325)),
+          newx(INT2(700)), newy(INT2(-425)))

C
C
C    Draw the ellipses.
C
      CALL setfillmask(linemask)
      dummy=ellipse( $GBORDER,
+          newx(INT2(50)), newy(INT2(325)),
+          newx(INT2(200)), newy(INT2(425)))
      dummy=ellipse( $GFILLINTERIOR,
+          newx(INT2(550)), newy(INT2(325)),
+          newx(INT2(700)), newy(INT2(425)))
      END
```

注意 setlinestyle 将线型置为实线。若不用此程序, 则第一个矩形将用短横线画出。

子程序 ellipse 使用 rectangle 子程序类似的参数来画出一个椭圆。它们均需要一个填充标志和边界矩形的两个角。

5. 退出图形方式

SINE 程序调用最后一个 endprogram, 等待用户按一回车键, 然后将屏幕设置回正常方式。

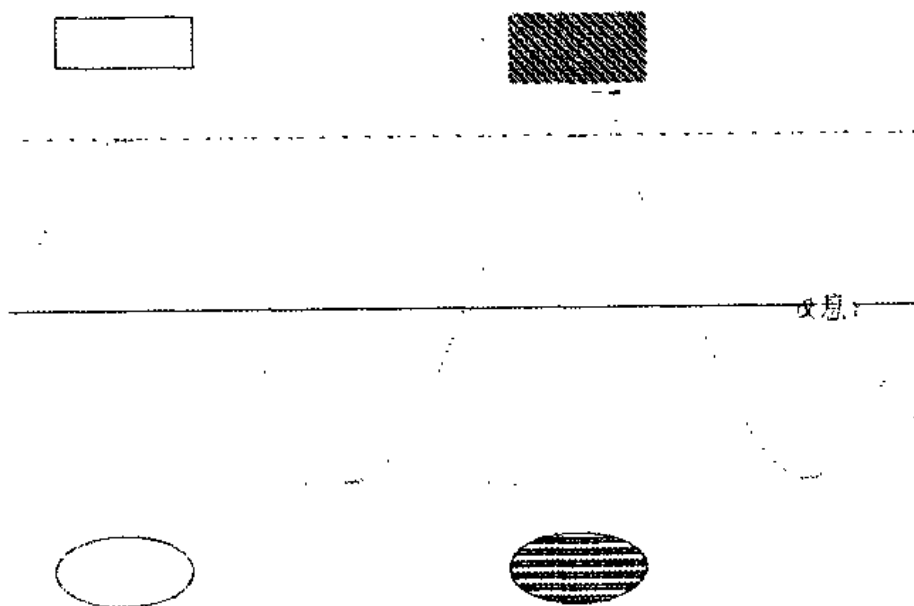
```

CC  ENDPROGRAM—This subroutine waits for the ENTER key to be
      SUBROUTINE endprogram( )
      INCLUDE 'FGRAPH.FD'
      INTEGER * 2 dummy
      READ( * , * )          ! Wait for ENTER key
      dummy=setvideomode( $ DEFAULTMODE)
      END

```

注意 恢复屏幕是图形程序中的重要一步，否则，其它程序若设定监视器及其分辨率，则将导致不可预料的后果。因为这些设置会和已经对当前图形方式进行的改变发生冲突，而恢复到原来的方式则避免了冲突。

由例 10.4 的 SINE 图形程序和增加图形程序生成的正弦曲线和两个矩形、两个椭圆图形如下图所示：



10.4 FORTRAN 中的汉字处理技术

一、文本方式下的汉字处理

80 年代中期，IBM PC 机引入我国。电子工业部六所开发出了基于 PC DOS 2.0 的 CCDOS 2.1 中文操作系统。它基本上解决了汉字的输入、显示和打印问题。基于当时的硬件水平，显示器适配器为 CGA，CCDOS 2.1 的汉字处理是在图形方式下将汉字按其点阵输出的。每屏可显示 11 行，每行 20 个汉字。CCDOS 的推出，为我国计算机的普及和汉字处理技术的发展奠定了一个坚实的基础。

随着计算机硬件技术的发展,在 80 年代末,90 年代初期,涌现出了一大批优秀的汉字操作系统,如:SPDOS, 2.13H, UC DOS 2.0 等。这一时期的汉字处理仍是基于图形方式的。然而大多数优秀的软件都是基于文本方式的。所以要使用最新的软件,就必须要将西文软件进行“汉化”。所谓汉化,就是将原版西文软件中有关输入输出的部分,改造成图形方式下的输入输出。通常,由于没有原码,汉化工作困难重重,人力物力耗费巨大,且周期很长,这就使得国内对最新软件的使用,至少比国外滞后半年甚至更长。为彻底改变这一现状,使中西文软件完全兼容,90 年代初期,推出了一大批“字符型”汉字操作系统。如:UCDOS 3.0、中国龙 2.0、天汇、联想等。经过几年的发展,基于 DOS 的汉字操作系统在技术上已经成熟。西文软件无需再进行汉化,就可以进行汉字的输入输出,使得国内对最新软件的应用基本上与国外同步。因此,只要选择一种字符型的汉字操作系统,在 FORTRAN 的文本方式下就可以进行汉字的输入输出。本书的所有例题都是在 UC DOS 3.1 下完成的。

最后需要说明的是:在 FORTRAN 中,汉字只能作为字符串来使用,而不能作为变量名来使用。

二、图形方式下的汉字处理

1. 汉字的编码

(1) 国标码

1981 年以来,我国先后颁布了 GB 2312《信息处理交换用汉字编码字符集基本集》,GB 1988—89《信息处理交换用的七位汉字编码字符集》和 GB 2311《信息处理交换用的七位汉字编码字符集的扩充方法》等国家标准。标准规定了汉字信息处理交换用的基本图形符号及其二进制编码。这些有标准的交换码,称为国标码。

国标码中共收集了 7 445 个汉字和图形符号。其中汉字有 6 763 个,分为两个级别:一级字库为常用汉字 3 755 个;二级字库为常用汉字 3 008 个。国标码中的任何一个汉字或图形符号,都是用两个 7 位的字节来表示的。

国标码中有一张代码表。表中,纵坐标代表区,编号为 1~94;横坐标代表位,编号为 1~94。每个区位的焦点表示一个汉字的代码,称为区位码。例如,“啊”的区位码是 1601 (16 区 01 位)。其国标码是 3021H。

(2) 汉字机内码

为保障汉字系统中西兼容,在处理汉字机内码时,既要保障西文机内码(ASCII 码)的使用,也要允许国标码的使用。显然,在一个系统中同时存在国标码和 ASCII 码,这将会产生二义性。例如,机内码 30H 和 21H,它既可以表示汉字“啊”的国标码,又可以表示西文“O”和“!”的 ASCII 码。因此,通常将汉字国标码的两个字节的高位分别置 1,用它来作为机内码标识汉字。这样既解决了汉字机内码与西文机内码冲突的问题,又保证了汉字机内码与国标码之间非常简单的映射关系。

(3) 编码的转换

根据上述定义,下面给出了代码的转换公式。

• 区位码转换成国标码

$$\text{国标码第一字节} = \text{区号} + 20\text{H} \quad (10-1)$$

$$\text{国标码第二字节} = \text{位号} + 20\text{H} \quad (10-2)$$

• 区位码转换成机内码

$$\begin{aligned}\text{机内码第一字节} &= (\text{区号} + 20\text{H}) + 80\text{H} \\ &= \text{区号} + 160 \quad (10-3)\end{aligned}$$

$$\begin{aligned}\text{机内码第二字节} &= (\text{位号} + 20\text{H}) + 80\text{H} \\ &= \text{位号} + 160 \quad (10-4)\end{aligned}$$

• 机内码转换成区位码

$$\text{区码} = \text{机内码第一字节} - 160 \quad (10-5)$$

$$\text{位码} = \text{机内码第二字节} - 160 \quad (10-6)$$

2. 汉字库及存储

(1) 汉字点阵表示方法

在 EGA 和 VGA 显示器下, 西文字符采用 8×14 点来显示的, 而对于笔画结构极为复杂的汉字来说, 是不足以描述其轮廓的。通常, 用于显示器的汉字点阵采用 16×16 点阵, 如图 10.5 所示。而用于打印时, 则采用 24×24 , 48×48 等点阵。点阵数越大, 描述汉字就越精细, 输出质量也就越好。

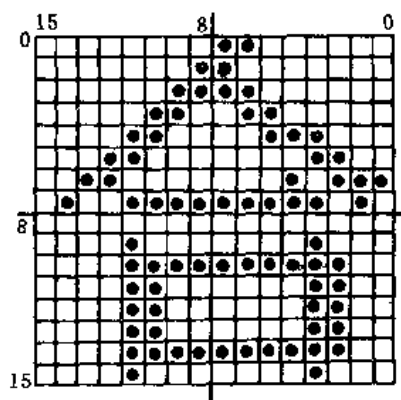


图 10.5 汉字点阵字形

(2) 汉字点阵的机内表示

对点阵的描述, 通常采用二进制的 0 和 1 表示, 1 表示有点, 0 表示空白(无点)。因此, 16×16 点阵共需 32 个字节来描述, 24×24 点阵则需要 72 个字节来描述。

16×16 点阵的字节排列格式见图 10.6, 24×24 点阵的字节排列格式见图 10.7。

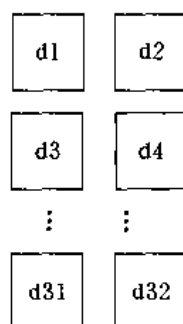


图 10.6 16×16 点阵字节排列格式

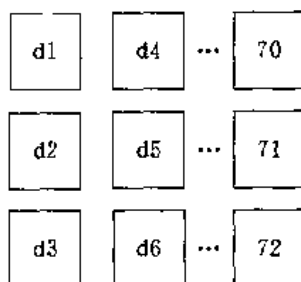


图 10.7 24×24 点阵字节排列格式

我们看到, 16×16 点阵的字节排列是水平方向的, 即一个字节代表水平方向上的 8 个位置上的 0 和 1。而 24×24 点阵则是纵向排列的, 即一个字节代表垂直方向上的 8 个位置的 0 和 1 的。 24×24 点阵的这种表示方法, 主要是为了打印机在输出打印时的方便。但若采用 24×24 点阵进行显示时, 则必须转换成水平方向排列的。

(3) 汉字的存储——汉字库

汉字库是按照汉字区位码的顺序, 将汉字点阵的信息按顺序存放在一个文件中。这个文件被称为汉字库。汉字库中汉字是从第 16 区开始存放的。这在读取汉字库时是应该注意的。

由于汉字操作系统没有一个国家标准, 因此, 各个系统提供的字库都不完全相同, 上机时, 应使用式(10-5)和式(10-6)进行测试, 以保证计算出来的汉字在字库中的偏移量是正确的。本书中采用的 2.13H 中的汉字库。

三、图示方式下汉字显示的示例

例 10.5 在屏幕的指定位置显示 16 点阵和 24 点阵的汉字。

解 算法分析

以 16 点阵的汉字串的显示为例，在屏幕的指定位置显示汉字串 s。其步骤如下：

- (1) 初始化图形系统
- (2) 打开指定的汉字库
- (3) I=1 TO S 的串长 STEP 2 DO
 - (3.1) 计算第 I 个汉字的点阵信息在字库中的起始位置(偏移量)
 - (3.2) 读取汉字的点阵信息存入一数组中
 - (3.3) 根据汉字的点阵信息在屏幕的指定位置显示汉字
- (4) 关闭图形系统

细化后的步骤如下：

1. 初始化图形系统，调用子程序 InitGraphics

2. 打开指定的汉字库采用如下方式：

```
OPEN (10, FILE='HZK16', ACCESS='DIRECT',
      FORM='BINARY', STATUS='OLD', RECL=1, IOSTAT=IOCHECK)
```

FORM=BINARY 是 FORTRAN 5.0 中新提供的一种打开文件的方式。它使得 FORTRAN 可以按字节对文件进行处理。

3. I=1 TO S 的串长 STEP 2 DO

- 3.1 计算第 I 个汉字的点阵信息在字库中的起始位置(偏移量)

根据式(7-5)和式(7-6)将机内码转换成区位码：

```
ICHAR(S(I;1))-161          汉字的区号
ICHAR(S(I+1;I+1))-161      汉字的位号
TEMP1=(ICHAR(S(I;I))-161)*94  每区中有 94 个汉字
TEMP2=ICHAR(S(I+1;I+1))-161
```

计算汉字字模在字库中的偏移量，FORTRAN 中的记录号是从 1 开始的

```
OFFSET=(TEMP 1+TEMP 2)*32+1  每个汉字的点阵占 32 个字节
```

- 3.2 读取汉字的点阵信息存入一数组中

```
DO J=1,32
  READ(10, REC=OFFSET)DOT(J)
  OFFSET=OFFSET+1
END DO
```

也可以采用隐 DO 循环直接给数组赋值。

- 3.3 根据汉字的点阵信息在屏幕的指定位置显示汉字串

按图 10.6 的 16×16 点阵字节排列格式，将 DOT 中的汉字的点阵信息显示在指定位置。为处理上的方便，用 EQUIVALENCE(DOT, D)语句将一维数组 DOT(32)变换成二维数组 D(2, 16)。具体的程序段如下：

```
DO I1=1, 16
```

```

C      X1, Y1 第 I 行的起始坐标。
          X1=X
          Y1=Y+11
          DO J=1, 2
              DO K=7, 0, -1
C      BTEST(N, K)位测试函数, N 为整型数。若 N 的第 K 位为 1, 则返回 TRUE,
C      否则返回 FALSE。
                  IF (BTEST(D(J, 11), K)) THEN
                      DUMMY=SETPIXEL(X1, Y1)
                  END IF
C      水平坐标右移一位。
                      X1=X1+1
                      END DO ! END K
                  END DO ! END J
          END DO ! END I1
C      水平坐标右移 16 位, 下一汉字的起始位置
          X=X+16
          END DO ! END DO 1

```

4. 关闭图形系统, 调用子程序 CloseGraphics

24×24 点阵汉字串的显示与 16×16 点阵汉字串的显示的算法一样, 只是点阵数和排列方式不同, 故不再赘述。

完整的程序如下:

```

C      Program Chinese_Character_display
C
          INCLUDE 'FGRAPH.FI'
          INCLUDE 'FGRAPH.FD'
          CHARACTER * 80 S, S1
          DATA S/'程序设计指南'/
          DATA S1/'微型计算机'/
C
C      初始化图形系统, 使用最高的分辨率, 背景色为蓝色, 前景色为黄色。
C
          CALL InitGraphics($BLUE, 14)
C
C      从(230,180)点处开始输出汉字串, 背景色为蓝色, 汉字串的颜色为黄色。
C
          CALL CCStr16(S, 230, 180, 14)
C
C      从(215, 220)点处开始输出汉字串, 背景色为蓝色, 汉字串的颜色为白色。
C
          CALL CCStr24(S1, 215, 220, 15)
C
C      从(240, 260)点处开始输出汉字串, 背景色为蓝色, 汉字串的颜色为黄色。

```



```

C
      CALL CCStr16(S1, 240, 260, 14)
C
C      从(205, 300)点处开始输出汉字串, 背景色为蓝色, 汉字串的颜色为白色。
C
      CALL CCStr24(S, 205, 300, 15)
      READ * ! Press Enter key
C
C      关闭图形系统
C
      CALL CloseGraphics( )
END
C
C      初始化图形系统, 使用最高的分辨率, 背景色为 bkColor, 前景色为 Color。
C
SUBROUTINE InitGraphics(bkColor, Color)
  INCLUDE 'FGRAPH. FD'
  INTEGER * 2 dummy
  INTEGER * 4 bkColor, Color
C
C      Find graphics mode init graphics
C
  IF(setvideomode( $ MAXRESMODE) .EQ. 0)
+   STOP 'Error: cannot set graphics mode'
  dummy=setcolor(Color)
  dummy=setbkcolor(bkColor)
  CALL CLEARSCREEN( $ GCLEARSCREEN)
END
C
C      关闭图形系统
C
SUBROUTINE CloseGraphics( )
C      Close graphics
  INCLUDE 'FGRAPH. FD'
  INTEGER * 2 dummy
  dummy=setvideomode( $ DEFAULTMODE)
END
C
C      16 点阵汉字串显示模块
C      S: 汉字串; X, Y: 坐标; COLOR: 汉字串颜色
C
SUBROUTINE CCStr16(S, X, Y, COLOR)
  INCLUDE 'FGRAPH. FD'

```

```

      CHARACTER * 80 S
      INTEGER * 1 DOT(32), D(2, 16)
      INTEGER * 2 IOCHECK
      INTEGER * 4 X, Y, I, J, K, II, X1, Y1, COLOR, STRL, DUMMY
      INTEGER * 4 OFFSET, TEMP1, TEMP2
      EQUIVALENCE(DOT, D)

C
C  FORM=BINARY 是 FORTRAN 5.10 中新提供的一种打开文件的方式。使得 FORTRAN
C  可以按字节对文件进行处理。
C
      OPEN(10, FILE='HZK16', ACCESS='DIRECT',
+      FORM='BINARY', STATUS='OLD', RECL=1, IOSTAT=IOCHECK)
      IF (IOCHECK.NE. 0)THEN
          STOP 'File not found !!!'
      END IF

C
C  忽略尾部空格后的串长
C
      STRL=LEN_TRIM(S)
      DO I=1, STRL, 2

C
C      ICHAR(S(I,I))-161 汉字的区号
C      ICHAR(S(I+1,I+1))-161 汉字的位号
C
      TEMP1=(ICHAR(S(I,I))-161)*94
      TEMP2=ICHAR(S(I+1,I+1))-161

C
C      计算汉字字模在字库中的偏移量
C      FORTRAN 中的记录号是从 1 开始的
C
      OFFSET=(TEMP1+TEMP2)*32+1

C
C      从字库中读取 16 点阵汉字字模，共 32 个字节。
C
      DO J=1, 32
          READ(10, REC=OFFSET)DOT(J)
          OFFSET=OFFSET+1
      END DO
      DUMMY=setcolor(COLOR)

C  Draw pixels.
      DO II=1, 16

C
C      X1, Y1 第 I 行的起始坐标。

```

```

C
      X1=X
      Y1=Y+I1
      DO J=1, 2
        DO K=7, 0, -1

C
C      BTEST(N, K)位测试函数, N 为整型数。若 N 的第 K 位为 1, 则返回 TRUE; 否则,
C      返回 FALSE。
C
          IF(BTEST(D(J, I1), K)) THEN
            DUMMY=SETPIXEL(X1, Y1)
          END IF

C
C      水平坐标右移一位。
C
          X1=X1+1
        END DO ! END K
      END DO ! END J
    END DO ! END I1

C
C      水平坐标右移 16 位, 下一汉字的起始位置
C
      X=X+16
    END DO ! END DO I
  CLOSE(10)
END

C
C 24 点阵汉字串显示模块
C S: 汉字串; X, Y: 坐标; COLOR: 汉字串颜色
C
      SUBROUTINE CCStr24(S, X, Y, COLOR)
      INCLUDE 'FGRAPH.FD'
      CHARACTER * 80 S
      INTEGER * 1 DOT(72), D(3, 24)
      INTEGER * 4 X, Y, I, J, K, I1, X1, Y1, COLOR, STRL, DUMMY
      INTEGER * 4 OFFSET, TEMP1, TEMP2
      EQUIVALENCE(DOT, D)

      OPEN(10, FILE='HZK24S', ACCESS='DIRECT',
+      FORM='BINARY', STATUS='OLD', RECL=1, IOSTAT=IOCHECK)
      IF (IOCHECK .NE. 0) THEN
        STOP 'File not found !!!'
      END IF

```

```

C      忽略尾部空格后的串长
C
      STRL=LEN_TRIM(S)
      DO I=1, STRL, 2
C
C      ICHAR(S(I,I))汉字的区号
C      ICHAR(S(I+1,I+1))汉字的位号
C
      TEMP1=(ICHAR(S(I,I))-176)*94
      TEMP2=ICHAR(S(I+1,I+1))-161
C
C      计算汉字字模在字库中的偏移量
C      FORTRAN 中的记录号是从 1 开始
C
      OFFSET=(TEMP1+TEMP2)*72+1
C
C      从字库中读取 24 点阵汉字字模, 共 72 个字节
C
      DO J=1, 72
        READ(10, REC=OFFSET)DOT(J)
        OFFSET=OFFSET+1
      END DO
      DUMMY=setcolor(COLOR)
C      Draw pixels.
      DO I1=1, 24
        X1=X+I1
        Y1=Y
        DO J=1, 3
          DO K=7, 0, -1
            IF (BTEST(DOT(J, I1), K)) THEN
              DUMMY=SETPIXEL(X1, Y1)
            END IF
            Y1=Y1+1
          END DO ! END K
        END DO ! END J
      END DO ! END I1
      X=X+30
    END DO ! END DO I
    CLOSE(10)
  END

```

说明: 运行本例时, 当前目录下必须要有汉字库文件 HZK16 和 HZK24S。读者可以将 CCStr16(S, X, Y, COLOR)和 CCStr24(S, X, Y, COLOR)作为一个标准的子程序来使用。此外, 读者还可修改本例, 如调整字间的距离, 建立一个自己专用的小字库等。在计算机控

制系统的设计中构造专用的小字库是非常有用的。采用下面的程序来调用 CCStr16(S, X, Y, COLOR)和 CCStr24(S, X, Y, COLOR), 将会看到一种非常奇妙的汉字输出效果。

```
C   Program Chinese_Character_display
C
      INCLUDE 'FGRAPH.FI'
      INCLUDE 'FGRAPH.FD'
      CHARACTER * 80 S, S1
      DATA S/'程序设计指南'/
      DATA S1/'微型计算机'/
      CALL InitGraphics( $BLUE, 14)
      CALL CCStr24(S1, 215, 90, 2)
      CALL CCStr24(S1, 216, 91, 14)
      CALL CCStr24(S1, 215, 140, 12)
      CALL CCStr24(S1, 216, 141, 14)
      CALL CCStr16(S, 230, 180, 14)
      CALL CCStr16(S, 231, 180, 14)
      CALL CCStr24(S1, 215, 220, 15)
      CALL CCStr16(S1, 240, 260, 14)
      CALL CCStr24(S, 205, 300, 12)
      CALL CCStr24(S, 206, 301, 14)
      CALL CCStr24(S, 205, 350, 2)
      CALL CCStr24(S, 206, 351, 14)
      READ * ! Press Enter key
      CALL CloseGraphics( )
END
```

11 上机操作指南

Microsoft FORTRAN 5.0 版本提供了很强的文件编辑及编译连接功能。用户可利用系统提供的装配程序方便地在自己的软盘或硬盘上生成自己的运行环境(包括编辑、编译、连接、数学库函数及图形函数等系统)。利用系统编辑器,用户可方便地对自己的源程序文件、原始数据文件等进行各种编辑操作。调用编译连接系统对自己的文件进行编译连接,并生成可执行文件。然后可直接运行此可执行文件。下边我们假设将系统安装在 C 盘上(操作系统为 DOS),并以对文件 EX1.FOR 操作为例说明系统的安装、编辑、连接及执行过程。

文件 EX1.FOR 由一个主程序和一个函数辅程序及子程序辅程序组成。函数辅程序用于求 $\prod_{i=1}^n i(n!)$, 子程序辅程序用于求 $\sum_{i=1}^n i(1+2+\cdots+n)$ 。

11.1 系统安装

若你手上有经销商提供的 13 张 Microsoft FORTRAN 5.0 系统盘时,请按下述步骤做:

① 在 A 驱中插入 DOS 盘并启动。

② 当 DOS 启动成功、屏幕上出现提示符 A>时,取出 DOS 盘,插入标有 SETUP 的盘并键入 SETUP(↵表示回车,下同)。

③ 在这一步操作中,系统将通过入机对话提示你是否要安装某些文件,安装到何处。且在大部分情况下,系统有一个默认的设置,若你不改变这种设置时,打入 C 键,继续进行下一步;若要改变时,打入相应键即可。具体步骤为:

当打入 SETUP(↵)后,系统显示 SETUP 程序的功能,并提示按 C 键继续。接着显示主菜单。若是首次建立系统,在主菜单下按 F 键。系统提示是在 MS-DOS 下运行,还是在 OS/2 系统下运行。若是在 MS-DOS(PC-DOS 兼容)下运行,按 D 键。系统提示是要在硬盘上还是在 5 英寸或 3 英寸软盘上安装系统。假定是在硬盘上则按 H 键。

系统让你确认硬盘,默认为 C。若要安装于 D 盘上,可用 BACKSPACE 或 ← 键消掉 C,打入 D 即可。

系统提示将要安装的文件表列,打入 C 键继续。

系统提示将要安装的二进制文件目录路径,系统默认为 C:\BIN,若不修改时直接按 C 键。

系统提示是否要安装 Microsoft 编辑器,若要安装时按 Y 键。若你更喜欢使用别的编辑器,例如 EDLIN, ED, Q 等时可按 N 键。

当按 Y 键后系统提示可供选择的编辑器文件名。假如选“QUICK”系列时按 Q 键,则系统为你建立一个名为 M 的编辑器,以后可调用此编辑器建立或修改你的源程序文件等。

系统提示你作图涉及到的系统文件的安放目录路径。系统默认为 C:\INCLUDE,若不改变时直接按 C 键。

系统提示将要建立的临时文件目录路径,默认为 C:\TMP,若不改变时按 C 键。

系统提示是否要安装鼠标器,若不安装时按 N 键。

系统提示库文件目录路径为 C:\LIB,若不改变时按 C 键。

系统提示你对上述二进制文件(BIN)、作图涉及系统文件(INCLUDE)、临时文件(TMP)及库文件(LIB)所选目录路径。若无误直接按 C 键,否则选相应键进行修改。

系统提示插入 compiler 1 到当前驱动器中,准备好后按 C 键。

系统提示插入 compiler 2 到当前驱动器中,准备好后按 C 键。

系统提示插入标有 utilities 1 的盘到当前驱动器中,然后按 C 键。

系统提示插入标有 utilities 2 的盘到当前驱动器中,然后按 C 键。

系统提示插入标有 EDitor 的盘到当前驱动器中,然后按 C 键。

系统提示插入标有 Additional libraries 的盘到当前驱动器中,然后按 C 键。

系统提示可执行文件(excutable)和支持文件(support)安装完毕,直接按 C 键继续。

系统提示 SETUP 系统下一步将要建立库文件,准备好后按 C 键继续。

系统提示在建立 FORTRAN 库时你所做的选择,包括库模式、数学软件包、错误信息处理和与 C 语言混合编辑能力四项选择。按 C 键即开始对上述四项做选择。

系统提示大模式是当前所做的选择,若不改变直接按 C 键。

系统提示 8087/80287/80387 数学软件包是当前的选择,若你的机器上未安装协处理器,可按 E(选择仿真数学库)或 A(备用数学库),假设按 E 键。

系统提示对将要用的数学库所做的选择,确信无误后直接按 C 键。

系统提示对错误信息的处理当前的选择是保存错误信息文本,若是按 C 键。

对于和 C 语言的混合编辑,系统当前的选择是放弃对 C 编辑能力,若无误时直接按 C 键。

系统提示对你所做上述选择后将要建立的 FORTRAN 库名为 LLIBFORE. LIB。

其中:

库名的第一个字母(L, M, S)表示所选择的库模式:

L 表示所选库为大模式(Large)或巨模式(Huge);

M 表示所选库为中模式(Medium);

S 表示所选库为小模式(Small);

第 2 至第 7 位 LIBFOR 表示是 FORTRAN 库；
 最后一个字母 (7, E, A) 表示所选的数学库类型；
 7 表示是 8087/80287/80387 数学库；
 E 表示是仿真数学库；
 A 表示是备用数学库；
 扩展名总是 LIB。

系统提示插入标有 Large-Model Libraries 的盘在当前驱动器中，然后按 C 键。

系统提示建库工作已做，打 C 键继续。

系统提示在运行编译程序之前，应在 AUTOEXEC.BAT 或 STARTUP.CMD 文件中对部分文件目录路径进行设置。内容如下：

```
PATH C:\BIN
SET LIB=C:\LIB
SET TMP=C:\TMP
SET INCLUDE=C:\INCLUDE
SET INIT=C:\BIN
```

当记下该内容后按 C 键继续。系统提示应在你的当前盘根目录下的 CONFIG.SYS 文件中，设置可同时打开文件数及可用缓冲区数。

```
files=20
buffers=10
```

看清楚后打 C 键继续。系统提示你以后可用 DOS 的 CD 命令转入你要存放源程序的目录中，然后按 C 键继续。

至此编辑、编译、连接及库文件系统已生成，打 Q 键即可退出 SETUP 过程。

MS - FORTRAN 5.1 系统的安装过程基本与上述过程相同，但增加了缺省安装方式。若用户不选择与 C 语言混合编程及图形功能时，选用缺省方式安装更简易，快捷。

11.2 输入及修改源程序

若在安装过程中，你选择了为系统安装快速编辑器，则在安装完成并按系统要求设置好环境变量后，你就可调用编辑器 M 输入或修改你的源程序文件。

一、输入源程序

① 在系统提示符 C> 下用 DOS 的 CD 命令转入你自己的目的子目录下。若你要在根目录下建立并调试程序，则可省去此步操作。

② 当位于目的目录之下后键入 M EX1.FOR↵。

③ 系统提示该文件不存在是否要创建，键入 Y↵。

④ 按 Tab 键或空格键跳到指定列上输入源程序。程序及输入格式如下：

```
C      EXAMPLE 11.1
      PROGRAM MAIN
      INTEGER S, P, FACT, N
```



```





PRINT *, 'ENTER NUMERIC N '
READ( *, *) N
P=FACT(N)
CALL SUM (N, S)
WRITE ( *, 100) P, S
100    FORMAT(1X, 'P=', I5, 4X, 'S=', I5)
END
FUNCTION FACT(N)
    INTEGER FACT, N, I, Q
    Q=1
    DO 10  I=1, N, 1
        Q=Q * I
10    CONTINUE
    FACT=Q
END
SUBROUTINE SUM(N, S)
    INTEGER N, S, I
    S=0
    DO 10  I=1, N, 1
        S=S+I
10    CONTINUE
END

```

⑤ 源程序存盘：当源程序输入完后，直接按功能键 F8 存盘退出。若按 F9 F8 系统提示是否要保存当前文件，若按 N 则表示放弃修改不存盘退出。

二、修改源程序文件

M 编辑器具有全屏幕编辑功能，可方便地实现行插入或删除，字符的插入、修改与删除操作。要修改文件时可先在系统提示符下用命令 M 文件名将文件调出，再进行修改。

① 光标移动：可直接用键盘上的     四个键将光标方便地在屏幕上按箭头所指方向进行移动。利用 Page Down 和 Page Up 键实现前后翻页。

② 行插入：行插入可用下述两种方法之一：

方法 1：当光标位于一行的行首时，若在插入状态（屏幕右下角提示有 insert），则直接按回车键即可插入一个空行；若光标位于行中，则会将该行内容从光标所在处分为两行。

方法 2：无论光标在行的什么位置上，在或不在插入状态下都可直接按 CTR+N 键，在光标处插入一个空行。

③ 行删除：要删除一行时，先将光标移到要删除的行上，再按 CTR-Y 键即可删除该行。

④ 字符插入：首先用光标移动键将光标移到要插入的字符位置，再使编辑器处于插入状态。（可按 Ins 键，该键作用是奇偶的，按一下时进入插入状态，屏幕右下角提示 insert，若再按一下时退出插入状态，屏幕右下角提示相应消失，表示又回到了改写状态。）然后，即可输入要插入的内容。

⑤ 字符删除：首先用光标移动键将光标移动到要删除的字符位置，然后，按 DeL 键即可删除光标处的字符。

⑥ 修改字符：首先用光标移动键将光标移到要修改的字符位置，确信系统在改写状态（即不在插入状态）时，直接输入新内容以取代老内容。

当文件修改完后可按 F8 存盘退出，则系统用修改过的文件替换老文件。若按 F9 F8 则可放弃修改而退出，原文件的内容将保持不变。

M 文件编辑器具有极强的文件编辑能力，例如可进行文件的合并，开设多窗口，实现字或行的移动及复制等。由于篇幅所限，在此不再详述。有兴趣的读者当进入 M 编辑器后可按 F1 键，得到更多的操作功能说明信息。

11.3 编译与连接

源程序修改好后即可进行编译与连接工作。在 Microsoft FORTRAN 5.00 中该项操作极其简单，可用 FL 命令一次完成。命令格式如下：

FL 选择项 文件名[, 文件[, ...]]

选择项见表 11.1。

表 11.1

选 择 项	作 用
/FPa	产生浮点调用；选择库为 XLIBFORA. LIB
/FPc	产生浮点调用；选择库为 XLIBFORE. LIB
/FPc87	产生浮点调用；选择库为 XLIBFOR7. LIB
/FPi	产生直插式指令；选择库为 XLIBFORE. LIB
/FPi87	产生直插式指令；选择库为 XLIBFOR7. LIB

在表 11.1 中，X 表示通配符。例如，XLIBFORA. LIB 是表示 LLIBFORA. LIB 或 MLIBFORA. LIB，即 X 可为 L 或 M。FL 命令行中只能使用这些选择项中的一个，选择项可存在于该命令行中的任何位置上。各个 /FP 选项及其作用见表 11.2。

表 11.2 /FP 选项及其作用

选 项	方 式	优 点	协处理器使用	所 选 库
/FPi87	直插式	内存需求最小，速度最快	必 须	LLIBFOR7. LIB 或 MLIBFOR7. LIB
/FPc87	调 用	比 /FPi87 慢，但允许连接时改变库	必 须	LLIBFOR7. LIB 或 MLIBFOR7. LIB
/FPi	直插式	代码比 /FPi87 大，但可不需协处理器，无协处理器时也可达到的最大精度	有则使用	MLIBFORE. LIB
/FPc	调 用	慢于 /FPi，但允许连接时改变库	有则使用	LLIBFORE. LIB 或 MLIBFORE. LIB
/FPa	调 用	无协处理器时最快但牺牲了精度	忽略协处理器	LLIBFORA. LIB 或 MLIBFORA. LIB

在本例中我们选择的命令及参数如下：

```
FL /FPc EX1.FOR ↵
```

当编译正确完成后，即可执行该文件。若还有错，则可再次调用 M 对源文件进行修改，重新进行编译。

在编译时，各程序模块可存在于一个文件中，如上例 EX1.FOR 中包含了主模块及函数辅程序 FACT 和子程序辅程序 SUM 三个模块。也可将它们分开，将主模块存于 EX1.FOR，将函数辅程序 FACT 存于 EX2.FOR，将子程序辅程序 SUM 存于 EX3.FOR，然后发如下命令：

```
FL /FPc EX1.FOR, EX2.FOR, EX3.FOR ↵
```

则系统首先分别对三个模块进行编译，然后将它们统一连接，生成一个名为 EX1.EXE 的可执行文件。

若有多个文件同时进行编译连接时，系统总是取 FL 命令中第一个文件主名为可执行文件主名。

11.4 执行程序

当你历尽艰辛，终于正确完成了上述各步操作之后，这时就可在系统提示符下打入命令：

```
EX1 ↵
```

得到运行结果如下：

```
ENTER NUMERIC N
5
P= 120   S= 15
```

习 题 答 案

习 题 二 答 案

1. 合法的整常数有

2225 -25 -97612 10000 0 005326

合法的实常数有

.137 2E-7 1.562 -.137 +0162.2E-02 +258. 3.6E-22

下述为不合法的:

E-10(指数不能单用,应写成 1.E-10)

101.2E+2.(指数不是整数)

2,360(不允许用逗号)

23.34.(只允许一个小数点)

111-333(字符“-”不合法)

2. 合法的整型变量名有

NUMBER I123 ITER IP L1369P

合法的实型变量名有

PI AMPS FRED ZETA Q COUNTS POWER

下述为不合法的:

2X13(不是以字母开头)

M63-2 X+Y K(2) INC* L3.6 JIM'S

(出现了不是字母或数字的字符)

BSQUARED(超过了六个字符)

3.

$$(1) X1 = (-B + \text{SQRT}(B * B - 4. * A * C)) / (2. * A)$$

$$(2) A = (3. * X * Y * * 2 * (Z + 1.0) + (0.5 * Y * Z)) / (1.0 + X + X * * 3)$$

$$(3) B = 1.5 * X * (X - 1.0) * \text{SQRT}(7.0 * Y - \text{LOG}(\text{COS}(X)))$$

$$(4) C = 5. * X * * 2 * \text{SQRT}((\text{COS}(X * * 2 - Y * * 2)) * * 3 + \text{ATAN}(X * \text{COS}(X))) / (\text{EXP}(X + 1.0) + \text{EXP}(Y + 1.0) + 1.0)$$

$$(5) D = A * (X + D / (2. * A)) * * 2 + (4. * A * C - B * B) / (4. * A * * 2)$$

$$(6) E = \sin(X) * * 2 - \cos(X) * * 2$$

$$(7) F = \exp(X) - \exp(-X)$$

$$(8) G = \text{SQRT}(\text{ABS}(P - Q))$$

4.

$$(1) 9. \quad (2) 3. \quad (3) -2 \quad (4) -100.$$

$$(5) 2.4 \quad (6) -2.4 \quad (7) 100. \quad (8) 83.4$$

$$(9) 2. \quad (10) 1.$$

$$5. I = K/J * (100 * X - (K * * 2) / (3.0 * J))$$

若先把整数均改为实数再进行运算, 得 $I=103$, 否则 $I=89$ 。

$$6. (1) .\text{TRUE} . \quad (2) .\text{FALSE} . \quad (3) .\text{TRUE} .$$

$$7. 0 \quad 2 \quad .75$$

$$2.666667 \quad 0 \quad 1.$$

$$8. 1.5 \quad 1.5 \quad 5. \quad 6.$$

$$4. \quad 5.5 \quad 6.2$$

9. 下述为合法的 FORTRAN 77 常数:

'TODAY' S DATE IS 8:3:94' 是长度为 22 的字符常数

2.35675D2 是双精度常数

.TRUE. 是逻辑常数

(3.65, 2E5) 是复常数

下述为不合法的:

'ALL' S WELL THAT ENDS WELL' (应当是 ALL'S)

'THE VALUE OF X = (最后丢失一个撇号)

(5.3 2.1) (两个数之间应有逗号)

.F. (应当写成 .FALSE.)

$$10. (1) 7. \quad (2) 6. \quad (3) 0.1111111 \quad (4) 1.$$

$$(5) -1.5 \quad (6) -1.5 \quad (7) -.75 \quad (8) 0.5$$

$$(9) 0. \quad (10) \text{错误} \quad (11) 729. \quad (12) -4.5$$

$$(13) 6561. \quad (14) \text{错误} \quad (15) \text{不确定}$$

$$11. (1) 6. \quad (2) 3. \quad (3) 4.$$

$$(4) -5 \quad (5) 0.0277778 \quad (6) 0.$$

$$(7) 0. \quad (8) 1. \quad (9) 0.$$

$$12. S=0. \quad J=1 \quad JK=1$$

13. 两个不等式均可能成立。

14.

$$(1) A. \text{GE} . B$$

$$(2) (C+D) . \text{LT} . (E+F)$$

$$(3) (U+V) . \text{NE} . (X+Y)$$

$$(4) ((32+S) * (63+T) + \sin(37 * 3.14159/180)) . \text{GT} . (\exp(X) * 36 * \log(X))$$

- (5) $3 * X + 6 * Y .LE. 7 * X * * 2 + 8 * Y * * 2 + 9$
 (6) $7 * SIN((X + Y) * * 2) .EQ. ATAN(SQRT(A * * 2 + B * * 2) / ABS(C))$
- 15.
- (1) $(X .GE. 0) .AND. (X .LE. 15)$
 (2) $(X .LT. 0) .OR. (X .GT. 7)$
 (3) $((X .GE. 2) .AND. (X .LE. 9)) .AND. (Y .GE. 1) .AND. (Y .LE. 10)$
 (4) $SQRT((X2 - X3) * * 2 + (Y2 - Y3) * * 2) .GT. SQRT((X1 - X2) * * 2 + (Y1 - Y2) * * 2)$
 (5) $(X * Y) .GT. 0$
 (6) $(.NOT. ((X .EQ. 0) .AND. (Y .EQ. 0))) .AND. ((X * Y) .EQ. 0)$
16. (1) T (2) F (3) T
- 17.
- (1) 2. 3.
 (2) -1 1 -1 1 ...
 (3) 20 21 22 23
 (4) 3. 2. 1.
 (5) 10. 1. 8. 2. 6. 3.
 (6) 3 3 3 ...
 (7) 1 3 5
 (8) 101 102 103 104
- 21.
- (1) F (2) F (3) T
 (4) T (5) T (6) F
- 23.
- (1) $A .AND. (B .OR. C) .OR. D .AND. (E .OR. F)$
 (2) $((A .OR. B .OR. C) .AND. ((D .AND. E) .OR. F) .AND. G) .OR. H$
 (3) $(A .AND. B .AND. C) .OR. ((D .OR. E) .AND. F) .OR. G$

习题三答案

2.

- (1) 2 048.560 000 120 804.050 000
 (2) 3 096.840 000 220 44.524 000
 (3) 输入出错

3.

- (1) 30 5 320
 (2) -2.500 000 49.103 600 43 202.670 000
 (3) 30 49.103 600
 (4) -2.500 0 49.103 6 5 320

(5) 30 -2.500 0 5 320

(6) x= -2.500 000 i= 30

8.

(1) i= 42

j= 567

x= 35.910 000

k= 2

l= 526

w= 360.438 100

(2) m= 4

n= 256

k= 735

f= 9.120 000 E - 01

j= 526

i= 360

h= 4.380 000 E - 01

l= 1

习题四答案

13. 27 次

习题五答案

2. (1)

b(4)= 7

a(8)= 21.500 000

a(60)= 33.400 000

b(10)= 20

a(i+1)= -8.400 000

b(i-1)= 5

a(2*i-3)= -8.400 000

b(3+i)= 8

(2)

a(i)= 10.300 000

a(b(i))= 13.400 000

a(b(b(i))+1)=62.300 000

3. (1) 1 (2) 1 (3) 一次也不执行。

2

4

3	7
4	10
5	
6	
7	
8	
9	
10	
5. (1) 1	1
1	2
2	1
2	2
3	1
3	2
(2) 1	
	1
	1
	2
	3
	4
2	
	1
	2
	3
	4
3	
	1
	2
	3
	4
2	
	1
	1
	2
	3
	4
2	
	1
	2


```

          3
          4
      3
          1
          2
          3
          4
(3) I=  6      J=  5      K=  4      L=  5      N= 20

```

习题六答案

1.

- (1) 用维数说明的变量不能够在虚元表中出现。
- (2) 虚元不允许是常数。
- (3) 函数不能调用自身。
- (4) 正确, $T(X)$ 为已定义过的语句函数。
- (5) 正确。
- (6) 正确, 形式变量在程序里可以作为实际变量使用。
- (7) 正确, 原因同上。
- (8) 形参名不能相同。

3.

(1) F (2) T (3) F (4) T (5) T (6) T (7) T (8) T (9) F

11.

(1)

- ① $F(X) = X * * 3 - 2.0 * X * X + 7.0 * X + 4.0$
- ② $X = X2 - (X2 - X1) / (F2 - F1) * F2$
- ③ $F1 - F0$
- ④ $X2 = X$
- ⑤ GOTO 10

(2)

- ① $N = N + 1$
- ② SUM (S, 1, NC, NR) / NC
- ③ REAL A (NK, J)
- ④ SUM = S
- ⑤ CHARACTER * 6 NUM(K)
- ⑥ TA = NUM(I)
- ⑦ NUM(J) = TA
- ⑧ A(I) = A(J)

(3)

- ① EXTERNAL FUN
- ② CALL ROOT (0.5, X, FUN, 1E-6, M)
- ③ X=X1
- ④ ((N. CT. 100) .AND. (X1-X .GT. EPS))
- ⑤ FUN

习题七答案

1.
 - X1= (-1.000 000, 2.000 000)
 - X2= (-1.000 000, -2.000 000)
5.
 - SEE JANE RUN
 - 5
 - 6
 - JACKBSEESBJANE.
6.
 - (1) NOM=0
 - (2) INDEX(LINE(L:80), WORD)
 - (3) LINE(1:L-1)//'NEW'//LINE(L+3:80)
 - (4) L+3
 - (5) OOTO 10

习题八答案

2. (1) F (2) T (3) T
3. 2 4 6

习题九答案

1.
 - (1) 缺少部件说明符。
 - (2) 部件说明符省去字符“UNIT=”时必须放在表的首项。
 - (3) 缺少记录长度说明符。一个连接成直接存取的文件必须要指出记录长度。
 - (4) 存取方式说明符不出现，隐含是顺序存取的。顺序存取的文件不应出现记录长度说明符。
 - (5) 一个临时文件不应有名字。
 - (6) 对一个无格式文件，空白符说明符是无意义的。
- 2.

- (1) 执行该 OPEN 语句, 将把一个已存在的名为 A.DAT 的文件作为格式顺序存取方式连接到部件 15 上。
- (2) 执行这个 OPEN 语句将在部件 16 上建立一个名为 B.DAT 的顺序存取格式文件。
- (3) 执行这个 OPEN 语句将把一个已存在的名为 C.DAT 的文件, 作为无格式直接存取方式连接到部件 17 上。其记录长度为 30 个单位。
- (4) 执行这个 OPEN 语句将把一个已存在的名为 D.DAT 的文件, 作为有格式直接存取方式连接到部件 18 上。记录长度为 30 个字符。
- (5) 执行这个 OPEN 语句, 将把一个有格式顺序存取方式的临时文件连接到部件 19 上。

附录 A

内部函数表

内部函数	定 义	变元个数	属名	特定名	类 型	
					变 元	函 数
类 型 转 换	转换为整数 $\text{int}(a)$ (参见注①)	1	INT	— INT IFIX IDINT —	整 型 实 型 实 型 双精度型 复 型	整 型 整 型 整 型 整 型 整 型
	转换为实数 (参见注②)	1	REAL	REAL FLOAT — SINGL —	整 型 整 型 实 型 双精度型 复 型	实 型 实 型 实 型 实 型 实 型
	转换为双精度 (参见注③)	1	DBLE	— — — —	整 型 实 型 双精度型 复 型	双精度型 双精度型 双精度型 双精度型
	转换为复数 (参见注④)	1 或 2	CMPLX	— — — —	整 型 实 型 双精度型 复 型	复 型 复 型 复 型 复 型
	转换为整数 (参见注⑤)	1		ICHAR	字符型	整 型
	转换为字符 (参见注⑤)	1		CHAR	整 型	字符型
截 断	$\text{int}(a)$ (参见注①)	1	AINT	AINT DINT	实 型 双精度型	实 型 双精度型
最近 同型 取整	$\text{int}(a+.5)$ 当 $a \geq 0$ $\text{int}(a-.5)$ 当 $a < 0$	1	ANINT	ANINT DNINT	实 型 双精度型	实 型 双精度型
最近 取整	$\text{int}(a+.5)$ 当 $a \geq 0$ $\text{int}(a-.5)$ 当 $a < 0$	1	NINT	NINT IDNINT	实 型 双精度型	整 型 整 型

续表

内部函数	定 义	变元个数	属名	特定名	类 型	
					变 元	函 数
绝对值	$ a $ (参见注⑥) $(ar^2 + ai^2)^{1/2}$	1	ABS	IABS ABS DABS CABS	整 型 实 型 双精度型 复 型	整 型 实 型 双精度型 实 型
求 余	$a_1 - \text{int}(a_1/a_2) * a_2$ (参见注①)	2	MOD	MOD AMOD DMOD	整 型 实 型 双精度型	整 型 实 型 双精度型
双精度 乘 积	$a_1 * a_2$	2		DPROD	实 型	双精度型
符号传送	$ a_1 $ 当 $a_2 \geq 0$ $- a_1 $ 当 $a_2 < 0$	2	SIGN	ISIGN SIGN DSIGN	整 型 实 型 双精度型	整 型 实 型 双精度型
正 差	$a_1 - a_2$ 当 $a_1 > a_2$ 0 当 $a_1 \leq a_2$	2	DIM	IDIM DIM DDIM	整 型 实 型 双精度型	整 型 实 型 双精度型
选最大值	$\max(a_1, a_2, \dots)$	≥ 2	MAX	MAX0 AMAX1 DMAX1 AMAX0 MAX1	整 型 实 型 双精度型 整 型 实 型	整 型 实 型 双精度型 实 型 整 型
选最小值	$\min(a_1, a_2, \dots)$	≥ 2	MIN	MIN0 AMIN1 DMIN1 AMIN0 MIN1	整 型 实 型 双精度型 整 型 实 型	整 型 实 型 双精度型 实 型 整 型
子串下标	子串 a_2 在串 a_1 中的 位置(参见注⑩)	2		INDEX	字符型	整 型
长 度	字符实体的长度	1		LEN	字符型	整 型
复型变元 的虚部	ai (见注⑥)	1		AIMAG	复 型	实 型
复型变元 的共轭	(ar, ai) (见注⑥)	1		CONJG	复 型	复 型
平方根	$(a)^{1/2}$	1	SQRT	SQRT DSQRT CSQRT	实 型 双精度型 复 型	实 型 双精度型 复 型
指 数	$e^{**}a$	1	EXP	EXP DEXP CEXP	实 型 双精度型 复 型	实 型 双精度型 复 型

续表

内部函数	定 义	变元个数	属名	特定名	类 型	
					变 元	函 数
自然对数	$\log(a)$	1	LOG	ALOG	实 型	实 型
				DLOG	双精度型	双精度型
				CLOG	复 型	复 型
常用对数	$\log_{10}(a)$	1	LOG10	ALOG10	实 型	实 型
				DLOG10	双精度型	双精度型
正 弦	$\sin(a)$	1	SIN	SIN	实 型	实 型
				DSIN	双精度型	双精度型
				CSIN	复 型	复 型
余 弦	$\cos(a)$	1	COS	COS	实 型	实 型
				DCOS	双精度型	双精度型
				CCOS	复 型	复 型
正 切	$\tan(a)$	1	TAN	TAN	实 型	实 型
				DTAN	双精度型	双精度型
反正弦	$\arcsin(a)$	1	ASIN	ASIN	实 型	实 型
				DASIN	双精度型	双精度型
反余弦	$\arccos(a)$	1	ACOS	ACOS	实 型	实 型
				DACOS	双精度型	双精度型
反正切	$\arctan(a)$	1	ATAN	ATAN	实 型	实 型
	$\arctan(a_1/a_2)$	2	ATAN2	DATAN	双精度型	双精度型
				ATAN2	实 型	实 型
双曲正弦	$\sinh(a)$	1	SINH	DTAN2	双精度型	双精度型
				SINH	实 型	实 型
双曲余弦	$\cosh(a)$	1	COSH	DSINH	双精度型	双精度型
				COSH	实 型	实 型
双曲正切	$\tanh(a)$	1	TANH	DCOSH	双精度型	双精度型
				TANH	实 型	实 型
按字序 大于等于	$a_1 \geq a_2$ (参见注②)	2		DTANH	双精度型	双精度型
按字序 大于	$a_1 > a_2$ (参见注②)	2		LGE	字符型	逻辑型
按字序 小于等于	$a_1 \leq a_2$ (参见注②)	2		LGT	字符型	逻辑型
按字序 小于	$a_1 < a_2$ (参见注②)	2		LLE	字符型	逻辑型
				LLT	字符型	逻辑型

附录 A 的注:

① 对整型的 a , $\text{int}(a) = a$ 。对实型或双精度型的 a , 有两种情况: 若 $|a| < 1$, $\text{int}(a) = 0$; 若 $|a| \geq 1$, $\text{int}(a)$ 就是这样的整数, 它的绝对值不超过 a 的绝对值的最大整数, 它的符号和 a 的符号相同。

对复型的 a , $\text{int}(a)$ 是对 a 的实部应用上述规则得到的值。

对实型的 a , $\text{IFIX}(a)$ 和 $\text{INT}(a)$ 相同。

② 对实型的 a , $\text{REAL}(a)$ 即 a 。对整型或双精度型的 a , $\text{REAL}(a)$ 的精度和 a 作为实数据时可包含的有效部分的精度一样。对复型的 a , $\text{REAL}(a)$ 是 a 的实部。

对整型的 a , $\text{FLOAT}(a)$ 和 $\text{REAL}(a)$ 一样。

③ 对双精度型的 a , $\text{DBLE}(a)$ 就是 a 。对整型或实型的 a , $\text{DBLE}(a)$ 的精度和 a 的有效部分作为一个双精度数据可包含的精度一样。对复型的 a , $\text{DBLE}(a)$ 的精度和 a 实部的有效部分作为一个双精度数据可包含的精度一样。

④ CMPLX 可以有一个或两个变元。若有一个变元, 则它可以是整型、实型、双精度型或复型。若有两个变元, 则这两变元必须是相同类型的, 并且可以是整型、实型或双精度型的。

对复型的 a , $\text{CMPLX}(a)$ 就是 a 。对整型、实型或双精度型的 a , $\text{CMPLX}(a)$ 是实部为 $\text{REAL}(a)$, 虚部为零的复数值。

$\text{CMPLX}(a_1, a_2)$ 是实部为 $\text{REAL}(a_1)$, 虚部为 $\text{REAL}(a_2)$ 的复数值。

⑤ ICHAR 提供了从字符到整数的转换手段, 转换是根据处理系统的排序序列中的字符位置进行的。在排序序列中, 第一个字符对应于位置 0, 最后一个字符对应于位置 $n-1$ 。其中, n 是排序序列中的字符数目。

$\text{ICHAR}(a)$ 的值是范围 $0 \leq \text{ICHAR}(a) \leq n-1$ 内的整数。其中, a 是长度为 1 的字符型的变元。 a 的值必须是处理系统中能够表示的字符。该字符在排序序列中的位置即是 ICHAR 的值。

对处理系统中能够表示的任意字符 C_1 和 C_2 , (C_1, LE, C_2) 是真, 当且仅当 $(\text{ICHAR}(C_1), \text{LE}, \text{ICHAR}(C_2))$ 是真; (C_1, EQ, C_2) 为真, 当且仅当 $(\text{ICHAR}(C_1), \text{EQ}, \text{ICHAR}(C_2))$ 为真。

$\text{CHAR}(i)$ 回送的值为处理系统排序序列中第 i 个位置上的字符。值为字符型, 长度为 1。 i 必须是一整表达式, 其值必须属下述范围:

$$0 \leq i \leq n-1$$

$$\text{ICHAR}(\text{CHAR}(i)) = i, \text{ 对 } 0 \leq i \leq n-1$$

$$\text{CNAR}(\text{ICHAR}(C)) = C, \text{ 对任一处理系统中能表示的字符 } C。$$

⑥ 复数值用一对有序实数 (ar, ai) 表示。其中, ar 是实部, ai 是虚部。

⑦ 所有的角度都用弧度表示。

⑧ 复型函数的结果是其主值。

⑨ 在一个内部函数引用中的所有变元必须有相同的类型。

⑩ $\text{INDEX}(a_1, a_2)$ 回送一个整数值, 它指出在字符串 a_1 中和串 a_2 一样的子串的开始位置。如果 a_2 在 a_1 中不止出现一次, 则回送值为其第一次出现的开始位置。

若 a_2 不在 a_1 中出现, 则回送值 0; 若 $\text{LEN}(a_1) < \text{LEN}(a_2)$, 同样回送值 0。

⑪ LEN 函数的变元值, 在该函数引用执行时不要求是有定义的。

⑫ 若按 ASCII 代码 $a_1 \geq a_2$, $\text{LGE}(a_1, a_2)$ 回送真值, 否则回送假值。按上述规定, 若 $a_1 > a_2$, $\text{LGT}(a_1, a_2)$ 回送真值, 否则回送假值; 若 $a_1 \leq a_2$, $\text{LLE}(a_1, a_2)$ 回送真值, 否则回送假值; 若 $a_1 < a_2$, $\text{LLT}(a_1, a_2)$ 回送真值, 否则回送假值。

附 录 B

字符 ASCII 代码及 EBCDIC 代码对照表

字 符	ASCII 码			EBCDIC 码	
	八进制数	十进制数	十六进制数	八进制数	十六进制数
换行(LF)	012	010	0A	045	25
回车(CR)	015	013	0D	015	0D
ESC	033	025	1B	047	27
空格	040	032	20	100	40
!	041	033	21	132	5A
"	042	034	22	177	7F
#	043	035	23	173	7B
\$	044	036	24	133	5B
%	045	037	25	154	6C
&	046	038	26	120	50
'	047	039	27	175	7D
(050	040	28	115	4D
)	051	041	29	135	5D
*	052	042	2A	134	5C
+	053	043	2B	116	4E
,	054	044	2C	153	6B
-	055	045	2D	155	6D
.	056	046	2E	113	4B
/	057	047	2F	141	61
0	060	048	30	360	F0
1	061	049	31	361	F1
2	062	050	32	362	F2
3	063	051	33	363	F3
4	064	052	34	364	F4
5	065	053	35	365	F5
6	066	054	36	366	F6

续表

字 符	ASCII 码			EBCDIC 码	
	八进制数	十进制数	十六进制数	八进制数	十六进制数
7	067	055	37	367	F7
8	070	056	38	370	F8
9	071	057	39	371	F9
:	072	058	3A	172	7A
;	073	059	3B	136	5E
<	074	060	3C	114	4C
=	075	061	3D	176	7E
>	076	062	3E	156	6E
?	077	063	3F	157	6F
@	100	064	40	174	7C
A	101	065	41	301	C1
B	102	066	42	302	C2
C	103	067	43	303	C3
D	104	068	44	304	C4
E	105	069	45	305	C5
F	106	070	46	306	C6
G	107	071	47	307	C7
H	110	072	48	310	C8
I	111	073	49	311	C9
J	112	074	4A	321	D1
K	113	075	4B	322	D2
L	114	076	4C	323	D3
M	115	077	4D	324	D4
N	116	078	4E	325	D5
O	117	079	4F	326	D6
P	120	080	50	327	D7
Q	121	081	51	330	D8
R	122	082	52	331	D9
S	123	083	53	342	E2
T	124	084	54	343	E3
U	125	085	55	344	E4

续表

字 符	ASCII 码			EBCDIC 码	
	八进制数	十进制数	十六进制数	八进制数	十六进制数
V	126	086	56	345	E5
W	127	087	57	346	E6
X	130	088	58	347	E7
Y	131	089	59	350	E8
Z	132	090	5A	351	E9
[133	091	5B		
\	134	092	5C	340	E0
]	135	093	5D		
^ 或 †	136	094	5E		
_ (下划线)	137	095	5F		
	140	096			
a	141	097	61	201	81
b	142	098	62	202	82
c	143	099	63	203	83
d	144	100	64	204	84
e	145	101	65	205	85
f	146	102	66	206	86
g	147	103	67	207	87
h	150	104	68	210	88
i	151	105	69	211	89
j	152	106	6A	221	91
k	153	107	6B	222	92
l	154	108	6C	223	93
m	155	109	6D	224	94
n	156	110	6E	225	95
o	157	111	6F	226	96
p	160	112	70	227	97
q	161	113	71	230	98
r	162	114	72	231	99
s	163	115	73	242	A2
t	164	116	74	243	A3

续表

字 符	ASCII 码			EBCDIC 码	
	八进制数	十进制数	十六进制数	八进制数	十六进制数
u	165	117	75	244	A4
v	166	118	76	245	A5
w	167	119	77	246	A6
x	170	120	78	247	A7
y	171	121	79	250	A8
z	172	122	7A	251	A9
{	173	123	7B	300	C0
	174	124	7C	117	4F
}	175	125	7D	320	D0
~	176	126	7E		

注：有几个字符无相应的 EBCDIC 代码，本表中该栏空着。

附 录 C

FORTRAN 77 语句形式表

ASSIGN S TO i	标号赋值语句
BACKSPACE u	回退语句
BACKSPACE (alist)	
BLOCK DATA (sub)	数据块子程序语句
CALL sub(((a[, a]...)))	引用子程序语句
CHARACTER(* len[,])nam[, nam]...	字符型说明语句
CLOSE (clist)	文件关闭语句
COMMON [/[cb]/] nlist	公共语句
[[[,]/[cb] /nlist]...	
COMPLEX v [, v]...	复型说明语句
CONTINUE	继续语句
DATA nlist/clist/[[[,]nlist/clist]...	数据初值语句
DIMENSION a(d) [, a(a(d))]...	数组说明语句
DO s[,]i=e ₁ , e ₂ [, e ₃]	循环语句
DOUBLE PRECISION v[, v]...	双精度型说明语句
ELSE	否则语句
ELSE IF(e) THEN	ELSE IF 语句
END	结束语句
END IF	END IF 语句
ENDFILE u	文件结束语句
ENDFILE (alist)	
ENTRY en(((d[, d]...)))	入口语句
EQUIVALENCE(nlist)[, (nlist)]...	等价语句
EXTERNAL Proc[, Proc]...	外部语句
FORMAT fs	格式语句
fun ((d[, d]...))=c	语句函数语句

[typ] FUNCTION fun ((d[, d]...))	函数子程序语句
GO TO ; [(,)(s[, s]...)]	赋值 GOTO 语句
GO TO s	无条件 GOTO 语句
GO TO(s[, s]...)[,]e	计算 GO TO 语句
IF (e) st	逻辑 IF 语句
IF (e) S ₁ , S ₂ , S ₃	算术 IF 语句
IF (e) THEN	块 IF 语句
IMPLICIT typ (a[, a]...)[typ, (a[, a]...)]...	类型隐含说明语句
INQUIRE(iflist)	按文件询问语句
INQUIRE(iulist)	按设备询问语句
INTEGER v[, v]...	整型说明语句
INTRINSIC fun [, fun]...	内部语句
LOGICAL v[, v]...	逻辑型说明语句
OPEN (olist)	文件打开语句
PARAMETER(P=e[, P=e]...)	参数语句
PAUSE [n]	暂停语句
PRINT f[, iolist]	打印语句
PROGRAM pgm	程序语句
READ (cilst) [iolist]	读语句
READ f[, iolist]	
REALV [, v]...	实型说明语句
RETURN [e]	返回语句
REWIND u	反绕语句
REWIND (alist)	
SAVE [a[, a]...]	保存语句
STOP [n]	停止语句
SUBROUTINE sub (([d[, d]...]))	子程序语句
V=e	数值赋值语句
V=e	逻辑赋值语句
V=e	字符赋值语句
WRITE(cilst) [iolist]	输出语句



参 考 文 献

- [1] Calderbank V J. A Course on Programming in FORTRAN (Second Edition). Chapman and Hall Ltd, 1983
- [2] Boillot M. Understanding FORTRAN (Second Edition). West Publishing Compang, 1981
- [3] 邓自立. 程序设计语言 FORTRAN 77. 北京: 高等教育出版社, 1992
- [4] 高福成, 姜玉泉, 梁静毅. 最新 FORTRAN 程序设计实用教程. 天津: 天津科学技术出版社, 1993
- [5] 汪康. Microsoft FORTRAN 5.0 用户手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [6] 许晓. Microsoft FORTRAN 5.0 程序员参考手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [7] 王乃文. Microsoft FORTRAN 5.0 调试工具 Code View 使用指南. 北京: 中国科学院希望高级电脑技术公司, 1991
- [8] 谭浩强, 田淑清. FORTRAN 语言. 北京: 清华大学出版社, 1993
- [9] 钱乐秋, 张玲珊, 金锦良. FORTRAN 77 语言程序设计基础. 上海: 复旦大学出版社, 1992
- [10] 赵雄芳. FORTRAN 77 程序设计教程. 长沙: 国防科技大学出版社, 1988
- [11] 高元华, 王练辉. FORTRAN 77 编程操作基础. 北京: 北京邮电学院出版社, 1992
- [12] 徐士良. FORTRAN 常用算法程序集. 北京: 清华大学出版社, 1992
- [13] 秦克诚. FORTRAN 程序设计. 北京: 电子工业出版社, 1987
- [14] 崔秀梅, 周梓星, 郭纯, 彭先定. MS-FORTRAN 77 程序设计语言. 长沙: 中南工业大学出版社, 1990
- [15] 余冬梅, 韩卫. FORTRAN 77 语言程序设计. 北京: 中国科学技术出版社, 1996



参 考 文 献

- [1] Calderbank V J. A Course on Programming in FORTRAN (Second Edition). Chapman and Hall Ltd, 1983
- [2] Boillot M. Understanding FORTRAN (Second Edition). West Publishing Compang, 1981
- [3] 邓自立. 程序设计语言 FORTRAN 77. 北京: 高等教育出版社, 1992
- [4] 高福成, 姜玉泉, 梁静毅. 最新 FORTRAN 程序设计实用教程. 天津: 天津科学技术出版社, 1993
- [5] 汪康. Microsoft FORTRAN 5.0 用户手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [6] 许晓. Microsoft FORTRAN 5.0 程序员参考手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [7] 王乃文. Microsoft FORTRAN 5.0 调试工具 Code View 使用指南. 北京: 中国科学院希望高级电脑技术公司, 1991
- [8] 谭浩强, 田淑清. FORTRAN 语言. 北京: 清华大学出版社, 1993
- [9] 钱乐秋, 张玲珊, 金锦良. FORTRAN 77 语言程序设计基础. 上海: 复旦大学出版社, 1992
- [10] 赵雄芳. FORTRAN 77 程序设计教程. 长沙: 国防科技大学出版社, 1988
- [11] 高元华, 王练辉. FORTRAN 77 编程操作基础. 北京: 北京邮电学院出版社, 1992
- [12] 徐士良. FORTRAN 常用算法程序集. 北京: 清华大学出版社, 1992
- [13] 秦克诚. FORTRAN 程序设计. 北京: 电子工业出版社, 1987
- [14] 崔秀梅, 周梓星, 郭纯, 彭先定. MS-FORTRAN 77 程序设计语言. 长沙: 中南工业大学出版社, 1990
- [15] 余冬梅, 韩卫. FORTRAN 77 语言程序设计. 北京: 中国科学技术出版社, 1996



参 考 文 献

- [1] Calderbank V J. A Course on Programming in FORTRAN (Second Edition). Chapman and Hall Ltd, 1983
- [2] Boillot M. Understanding FORTRAN (Second Edition). West Publishing Compang, 1981
- [3] 邓自立. 程序设计语言 FORTRAN 77. 北京: 高等教育出版社, 1992
- [4] 高福成, 姜玉泉, 梁静毅. 最新 FORTRAN 程序设计实用教程. 天津: 天津科学技术出版社, 1993
- [5] 汪康. Microsoft FORTRAN 5.0 用户手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [6] 许晓. Microsoft FORTRAN 5.0 程序员参考手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [7] 王乃文. Microsoft FORTRAN 5.0 调试工具 Code View 使用指南. 北京: 中国科学院希望高级电脑技术公司, 1991
- [8] 谭浩强, 田淑清. FORTRAN 语言. 北京: 清华大学出版社, 1993
- [9] 钱乐秋, 张玲珊, 金锦良. FORTRAN 77 语言程序设计基础. 上海: 复旦大学出版社, 1992
- [10] 赵雄芳. FORTRAN 77 程序设计教程. 长沙: 国防科技大学出版社, 1988
- [11] 高元华, 王练辉. FORTRAN 77 编程操作基础. 北京: 北京邮电学院出版社, 1992
- [12] 徐士良. FORTRAN 常用算法程序集. 北京: 清华大学出版社, 1992
- [13] 秦克诚. FORTRAN 程序设计. 北京: 电子工业出版社, 1987
- [14] 崔秀梅, 周梓星, 郭纯, 彭先定. MS - FORTRAN 77 程序设计语言. 长沙: 中南工业大学出版社, 1990
- [15] 余冬梅, 韩卫. FORTRAN 77 语言程序设计. 北京: 中国科学技术出版社, 1996



参 考 文 献

- [1] Calderbank V J. A Course on Programming in FORTRAN (Second Edition). Chapman and Hall Ltd, 1983
- [2] Boillot M. Understanding FORTRAN (Second Edition). West Publishing Compang, 1981
- [3] 邓自立. 程序设计语言 FORTRAN 77. 北京: 高等教育出版社, 1992
- [4] 高福成, 姜玉泉, 梁静毅. 最新 FORTRAN 程序设计实用教程. 天津: 天津科学技术出版社, 1993
- [5] 汪康. Microsoft FORTRAN 5.0 用户手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [6] 许晓. Microsoft FORTRAN 5.0 程序员参考手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [7] 王乃文. Microsoft FORTRAN 5.0 调试工具 Code View 使用指南. 北京: 中国科学院希望高级电脑技术公司, 1991
- [8] 谭浩强, 田淑清. FORTRAN 语言. 北京: 清华大学出版社, 1993
- [9] 钱乐秋, 张玲珊, 金锦良. FORTRAN 77 语言程序设计基础. 上海: 复旦大学出版社, 1992
- [10] 赵雄芳. FORTRAN 77 程序设计教程. 长沙: 国防科技大学出版社, 1988
- [11] 高元华, 王练辉. FORTRAN 77 编程操作基础. 北京: 北京邮电学院出版社, 1992
- [12] 徐士良. FORTRAN 常用算法程序集. 北京: 清华大学出版社, 1992
- [13] 秦克诚. FORTRAN 程序设计. 北京: 电子工业出版社, 1987
- [14] 崔秀梅, 周梓星, 郭纯, 彭先定. MS-FORTRAN 77 程序设计语言. 长沙: 中南工业大学出版社, 1990
- [15] 余冬梅, 韩卫. FORTRAN 77 语言程序设计. 北京: 中国科学技术出版社, 1996



参 考 文 献

- [1] Calderbank V J. A Course on Programming in FORTRAN (Second Edition). Chapman and Hall Ltd, 1983
- [2] Boillot M. Understanding FORTRAN (Second Edition). West Publishing Compang, 1981
- [3] 邓自立. 程序设计语言 FORTRAN 77. 北京: 高等教育出版社, 1992
- [4] 高福成, 姜玉泉, 梁静毅. 最新 FORTRAN 程序设计实用教程. 天津: 天津科学技术出版社, 1993
- [5] 汪康. Microsoft FORTRAN 5.0 用户手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [6] 许晓. Microsoft FORTRAN 5.0 程序员参考手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [7] 王乃文. Microsoft FORTRAN 5.0 调试工具 Code View 使用指南. 北京: 中国科学院希望高级电脑技术公司, 1991
- [8] 谭浩强, 田淑清. FORTRAN 语言. 北京: 清华大学出版社, 1993
- [9] 钱乐秋, 张玲珊, 金锦良. FORTRAN 77 语言程序设计基础. 上海: 复旦大学出版社, 1992
- [10] 赵雄芳. FORTRAN 77 程序设计教程. 长沙: 国防科技大学出版社, 1988
- [11] 高元华, 王练辉. FORTRAN 77 编程操作基础. 北京: 北京邮电学院出版社, 1992
- [12] 徐士良. FORTRAN 常用算法程序集. 北京: 清华大学出版社, 1992
- [13] 秦克诚. FORTRAN 程序设计. 北京: 电子工业出版社, 1987
- [14] 崔秀梅, 周梓星, 郭纯, 彭先定. MS-FORTRAN 77 程序设计语言. 长沙: 中南工业大学出版社, 1990
- [15] 余冬梅, 韩卫. FORTRAN 77 语言程序设计. 北京: 中国科学技术出版社, 1996



参 考 文 献

- [1] Calderbank V J. A Course on Programming in FORTRAN (Second Edition). Chapman and Hall Ltd, 1983
- [2] Boillot M. Understanding FORTRAN (Second Edition). West Publishing Compang, 1981
- [3] 邓自立. 程序设计语言 FORTRAN 77. 北京: 高等教育出版社, 1992
- [4] 高福成, 姜玉泉, 梁静毅. 最新 FORTRAN 程序设计实用教程. 天津: 天津科学技术出版社, 1993
- [5] 汪康. Microsoft FORTRAN 5.0 用户手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [6] 许晓. Microsoft FORTRAN 5.0 程序员参考手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [7] 王乃文. Microsoft FORTRAN 5.0 调试工具 Code View 使用指南. 北京: 中国科学院希望高级电脑技术公司, 1991
- [8] 谭浩强, 田淑清. FORTRAN 语言. 北京: 清华大学出版社, 1993
- [9] 钱乐秋, 张玲珊, 金锦良. FORTRAN 77 语言程序设计基础. 上海: 复旦大学出版社, 1992
- [10] 赵雄芳. FORTRAN 77 程序设计教程. 长沙: 国防科技大学出版社, 1988
- [11] 高元华, 王练辉. FORTRAN 77 编程操作基础. 北京: 北京邮电学院出版社, 1992
- [12] 徐士良. FORTRAN 常用算法程序集. 北京: 清华大学出版社, 1992
- [13] 秦克诚. FORTRAN 程序设计. 北京: 电子工业出版社, 1987
- [14] 崔秀梅, 周梓星, 郭纯, 彭先定. MS-FORTRAN 77 程序设计语言. 长沙: 中南工业大学出版社, 1990
- [15] 余冬梅, 韩卫. FORTRAN 77 语言程序设计. 北京: 中国科学技术出版社, 1996



参 考 文 献

- [1] Calderbank V J. A Course on Programming in FORTRAN (Second Edition). Chapman and Hall Ltd, 1983
- [2] Boillot M. Understanding FORTRAN (Second Edition). West Publishing Compang, 1981
- [3] 邓自立. 程序设计语言 FORTRAN 77. 北京: 高等教育出版社, 1992
- [4] 高福成, 姜玉泉, 梁静毅. 最新 FORTRAN 程序设计实用教程. 天津: 天津科学技术出版社, 1993
- [5] 汪康. Microsoft FORTRAN 5.0 用户手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [6] 许晓. Microsoft FORTRAN 5.0 程序员参考手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [7] 王乃文. Microsoft FORTRAN 5.0 调试工具 Code View 使用指南. 北京: 中国科学院希望高级电脑技术公司, 1991
- [8] 谭浩强, 田淑清. FORTRAN 语言. 北京: 清华大学出版社, 1993
- [9] 钱乐秋, 张玲珊, 金锦良. FORTRAN 77 语言程序设计基础. 上海: 复旦大学出版社, 1992
- [10] 赵雄芳. FORTRAN 77 程序设计教程. 长沙: 国防科技大学出版社, 1988
- [11] 高元华, 王练辉. FORTRAN 77 编程操作基础. 北京: 北京邮电学院出版社, 1992
- [12] 徐士良. FORTRAN 常用算法程序集. 北京: 清华大学出版社, 1992
- [13] 秦克诚. FORTRAN 程序设计. 北京: 电子工业出版社, 1987
- [14] 崔秀梅, 周梓星, 郭纯, 彭先定. MS-FORTRAN 77 程序设计语言. 长沙: 中南工业大学出版社, 1990
- [15] 余冬梅, 韩卫. FORTRAN 77 语言程序设计. 北京: 中国科学技术出版社, 1996



参 考 文 献

- [1] Calderbank V J. A Course on Programming in FORTRAN (Second Edition). Chapman and Hall Ltd, 1983
- [2] Boillot M. Understanding FORTRAN (Second Edition). West Publishing Compang, 1981
- [3] 邓自立. 程序设计语言 FORTRAN 77. 北京: 高等教育出版社, 1992
- [4] 高福成, 姜玉泉, 梁静毅. 最新 FORTRAN 程序设计实用教程. 天津: 天津科学技术出版社, 1993
- [5] 汪康. Microsoft FORTRAN 5.0 用户手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [6] 许晓. Microsoft FORTRAN 5.0 程序员参考手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [7] 王乃文. Microsoft FORTRAN 5.0 调试工具 Code View 使用指南. 北京: 中国科学院希望高级电脑技术公司, 1991
- [8] 谭浩强, 田淑清. FORTRAN 语言. 北京: 清华大学出版社, 1993
- [9] 钱乐秋, 张玲珊, 金锦良. FORTRAN 77 语言程序设计基础. 上海: 复旦大学出版社, 1992
- [10] 赵雄芳. FORTRAN 77 程序设计教程. 长沙: 国防科技大学出版社, 1988
- [11] 高元华, 王练辉. FORTRAN 77 编程操作基础. 北京: 北京邮电学院出版社, 1992
- [12] 徐士良. FORTRAN 常用算法程序集. 北京: 清华大学出版社, 1992
- [13] 秦克诚. FORTRAN 程序设计. 北京: 电子工业出版社, 1987
- [14] 崔秀梅, 周梓星, 郭纯, 彭先定. MS-FORTRAN 77 程序设计语言. 长沙: 中南工业大学出版社, 1990
- [15] 余冬梅, 韩卫. FORTRAN 77 语言程序设计. 北京: 中国科学技术出版社, 1996



参 考 文 献

- [1] Calderbank V J. A Course on Programming in FORTRAN (Second Edition). Chapman and Hall Ltd, 1983
- [2] Boillot M. Understanding FORTRAN (Second Edition). West Publishing Compang, 1981
- [3] 邓自立. 程序设计语言 FORTRAN 77. 北京: 高等教育出版社, 1992
- [4] 高福成, 姜玉泉, 梁静毅. 最新 FORTRAN 程序设计实用教程. 天津: 天津科学技术出版社, 1993
- [5] 汪康. Microsoft FORTRAN 5.0 用户手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [6] 许晓. Microsoft FORTRAN 5.0 程序员参考手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [7] 王乃文. Microsoft FORTRAN 5.0 调试工具 Code View 使用指南. 北京: 中国科学院希望高级电脑技术公司, 1991
- [8] 谭浩强, 田淑清. FORTRAN 语言. 北京: 清华大学出版社, 1993
- [9] 钱乐秋, 张玲珊, 金锦良. FORTRAN 77 语言程序设计基础. 上海: 复旦大学出版社, 1992
- [10] 赵雄芳. FORTRAN 77 程序设计教程. 长沙: 国防科技大学出版社, 1988
- [11] 高元华, 王练辉. FORTRAN 77 编程操作基础. 北京: 北京邮电学院出版社, 1992
- [12] 徐士良. FORTRAN 常用算法程序集. 北京: 清华大学出版社, 1992
- [13] 秦克诚. FORTRAN 程序设计. 北京: 电子工业出版社, 1987
- [14] 崔秀梅, 周梓星, 郭纯, 彭先定. MS-FORTRAN 77 程序设计语言. 长沙: 中南工业大学出版社, 1990
- [15] 余冬梅, 韩卫. FORTRAN 77 语言程序设计. 北京: 中国科学技术出版社, 1996



参 考 文 献

- [1] Calderbank V J. A Course on Programming in FORTRAN (Second Edition). Chapman and Hall Ltd, 1983
- [2] Boillot M. Understanding FORTRAN (Second Edition). West Publishing Compang, 1981
- [3] 邓自立. 程序设计语言 FORTRAN 77. 北京: 高等教育出版社, 1992
- [4] 高福成, 姜玉泉, 梁静毅. 最新 FORTRAN 程序设计实用教程. 天津: 天津科学技术出版社, 1993
- [5] 汪康. Microsoft FORTRAN 5.0 用户手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [6] 许晓. Microsoft FORTRAN 5.0 程序员参考手册. 北京: 中国科学院希望高级电脑技术公司, 1991
- [7] 王乃文. Microsoft FORTRAN 5.0 调试工具 Code View 使用指南. 北京: 中国科学院希望高级电脑技术公司, 1991
- [8] 谭浩强, 田淑清. FORTRAN 语言. 北京: 清华大学出版社, 1993
- [9] 钱乐秋, 张玲珊, 金锦良. FORTRAN 77 语言程序设计基础. 上海: 复旦大学出版社, 1992
- [10] 赵雄芳. FORTRAN 77 程序设计教程. 长沙: 国防科技大学出版社, 1988
- [11] 高元华, 王练辉. FORTRAN 77 编程操作基础. 北京: 北京邮电学院出版社, 1992
- [12] 徐士良. FORTRAN 常用算法程序集. 北京: 清华大学出版社, 1992
- [13] 秦克诚. FORTRAN 程序设计. 北京: 电子工业出版社, 1987
- [14] 崔秀梅, 周梓星, 郭纯, 彭先定. MS-FORTRAN 77 程序设计语言. 长沙: 中南工业大学出版社, 1990
- [15] 余冬梅, 韩卫. FORTRAN 77 语言程序设计. 北京: 中国科学技术出版社, 1996