

计算机软件技术基础

—— FORTRAN、数据结构、软件工程

汪大菊 葛卫民 陆明 赵国瑞 匙彦斌 编著

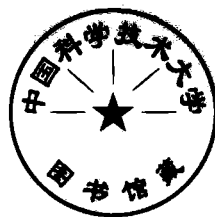


天津大学出版社

计算机软件技术基础

——FORTRAN、数据结构、软件工程

汪大菊 葛卫民 陆 明 赵国瑞 匙彦斌 编著



天津大学出版社

内 容 简 介

本书由三部分内容组成:FORTRAN 程序设计、数据结构和软件工程。全书以 FORTRAN 90 为蓝本较全面地介绍了 FORTRAN 语言的数据类型、语句、过程及程序结构,系统地讲述了程序设计方法、数值和非数值的常用算法。内容由浅入深,理论联系实际。每章均有一定数量的例题和习题。

本书可作为高等学校非计算机专业计算机程序设计教科书,也可作为软件开发人员学习的参考书。

图书在版编目(CIP)数据

计算机软件技术基础:FORTRAN、数据结构、软件工程 / 王大菊等编. —天津:天津大学出版社,2000.2
ISBN 7-5618-1266-3

I. 计... II. 王... III. FORTRAN 语言—程序设计
IV. TP312

中国版本图书馆 CIP 数据核字(2000)第 10166 号

出 版 天津大学出版社
出版人 杨风和
地 址 天津市卫津路 92 号天津大学内(邮编:300072)
电 话 发行部:022-27403647 邮购部:022-27402742
印 刷 河北省昌黎县印刷总厂
发 行 新华书店天津发行所
开 本 787mm×1092mm 1/16
印 张 15
字 数 390 千
版 次 2000 年 2 月第 1 版
印 次 2000 年 2 月第 1 次
印 数 1—5 000
定 价 18.00 元

前 言

根据教育部将高等院校非计算机专业大学计算机基础教育课程划分为三个层次(即计算机文化、计算机技术基础和计算机应用基础)的精神,结合理工科专业用计算机进行科学计算的需要,我们组织编写了本教材。作为计算机技术基础中的计算机软件技术基础课程的教科书,本书包括三部分内容,即 FORTRAN 语言的程序设计方法和程序设计中的常用算法、数据结构及软件工程。

对于大学生来说,程序设计是一门重要的基础课。该课程对学生全面了解计算机系统、提高软件设计技能、培养学生利用计算机所需要的逻辑思维能力和解决本专业实际问题能力都是必不可少的。对计算机程序设计能力的要求程度,也是目前我国中等教育与高等教育中计算机基础教育的区别所在。

本书以 FORTRAN 语言程序设计为主要内容,同时将数据结构、程序设计的常用算法和软件工程等方面的内容与程序设计语言结合起来,使学生在掌握一种程序设计语言的同时,学习更多的软件设计知识,为将来开发应用软件打下良好的基础。

本书作为大学低年级计算机文化基础课的后继课程,参考学时为 56 学时,其中讲课 32 学时,上机实习 24 学时,此外还应安排不少于 40 学时的课外上机实习。讲授时可按章节顺序安排,也可根据需要将第 8 章内容穿插在各章中介绍。

本书由汪大菊、葛卫民、陆明、匙彦斌、赵国瑞、王保旗编写,并由汪大菊、赵国瑞审阅了全书。

最后,诚恳欢迎各位读者对本书的缺点、错误批评指正。

编者

1999. 10

275-106

目 录

第 1 章 FORTRAN 语言基础知识	(1)
1.1 FORTRAN 语言简介	(1)
1.2 FORTRAN 源程序	(1)
1.3 FORTRAN 字符集	(5)
1.4 FORTRAN 数据类型	(6)
1.5 FORTRAN 数据对象	(6)
练习一	(16)
第 2 章 FORTRAN 基本语句	(18)
2.1 赋值语句	(18)
2.2 输入输出语句	(20)
2.3 END 语句、STOP 语句和 PAUSE 语句	(32)
2.4 顺序结构程序举例	(33)
练习二	(35)
第 3 章 选择结构语句	(38)
3.1 分块语句	(38)
3.2 SELECT 语句	(40)
3.3 其它几种控制转移语句	(42)
3.4 程序举例	(43)
练习三	(45)
第 4 章 循环结构	(48)
4.1 循环的引入	(48)
4.2 DO 语句	(49)
4.3 多重循环	(56)
4.4 程序举例	(57)
练习四	(60)
第 5 章 数组与结构	(65)
5.1 数组与数组说明符	(65)
5.2 数组的定义和引用	(66)
5.3 数组的存放与输入输出	(68)
5.4 DATA 语句	(71)
5.5 结构	(72)
5.6 指针	(76)
5.7 数组应用举例	(78)
练习五	(83)
第 6 章 FORTRAN 过程	(87)
6.1 内部函数	(87)

6.2	语句函数	(91)
6.3	函数子程序	(94)
6.4	子例行程序	(98)
6.5	虚参与实参结合的进一步讨论	(100)
6.6	公用语句和数据块子程序	(111)
6.7	过程中使用的其它语句	(114)
6.8	递归过程	(118)
6.9	模块	(120)
6.10	过程应用举例	(122)
	练习六	(126)
第 7 章	FORTRAN 文件	(131)
7.1	文件的概念	(131)
7.2	文件的基本操作	(132)
7.3	文件应用举例	(140)
	练习七	(144)
第 8 章	FORTRAN 程序设计基础和常用算法分析	(147)
8.1	程序设计的一般过程	(147)
8.2	结构化程序设计方法	(152)
8.3	FORTRAN 常用算法分析与设计	(153)
	练习八	(169)
第 9 章	数据结构	(171)
9.1	数据结构的概念	(171)
9.2	线性结构	(172)
9.3	数组	(182)
9.4	树形结构	(185)
* 9.5	图	(191)
9.6	排序	(200)
	练习九	(203)
第 10 章	软件工程	(206)
10.1	软件工程概述	(206)
10.2	结构化软件开发方法	(209)
10.3	软件测试	(223)
10.4	软件维护	(227)
10.5	面向对象的软件开发方法	(228)
	练习十	(233)

第 1 章 FORTRAN 语言基础知识

1.1 FORTRAN 语言简介

FORTRAN 是 FORMula TRANslation 的缩写,因此又称为公式翻译语言,是最早出现并得到广泛使用的计算机语言。它最初是为科学计算设计的,至今大量的科学计算应用程序仍用 FORTRAN 语言编写。

第一个 FORTRAN 文本发表于 1954 年,1956 年真正开始使用。1958 年出现的 FORTRAN II 和 1962 年出现的 FORTRAN IV 曾得到广泛使用。1966 年美国国家标准化协会(简称 ANSI)根据 FORTRAN IV 制定了 ANSI FORTRAN,简称 FORTRAN 66,实际上就是标准的 FORTRAN IV。随着计算机应用的发展,美国国家标准化协会于 1976 年修订了 FORTRAN IV,并于 1978 年正式公布新的 FORTRAN 文本,即 FORTRAN 77。FORTRAN 77 对 FORTRAN 66 做了较大改进,引进了字符数据处理的功能,增加了一些语句,尤其是块 IF 语句使 FORTRAN 成为较好的结构化程序设计工具。近些年,为了适应应用技术的需要,美国标准化组织又对 FORTRAN 77 做了进一步的改进和扩充,推出了 FORTRAN 90 标准,丰富了 FORTRAN 语言。

FORTRAN 语言最大的特点是简单、易学,不需要初学者具备太多的计算机知识就可以进行程序设计。FORTRAN 程序结构严谨、规范,不但适用于数值计算等科学计算问题,也可以用于非数值计算问题。

1.2 FORTRAN 源程序

1.2.1 源程序的组成

用 FORTRAN 语言编写的程序称为 FORTRAN 源程序。FORTRAN 源程序是一种段式结构(或称为块结构)的程序。每个 FORTRAN 源程序由一个主程序段和若干个子程序段组成。主程序段有且只能有一个,子程序段可以有零个或多个。每个程序段称为一个程序单位。每个程序单位可以独立编写,实现不同的功能,程序运行总是从主程序段开始。可以由主程序段调用子程序段,子程序段之间也可以互相调用。通过这种调用关系将各程序单位连接起来成为一个整体。

每一个程序单位都由若干条 FORTRAN 语句组成,按照对 FORTRAN 程序在编译、运行过程中所起作用不同,语句又分为可执行语句和非执行语句。可执行语句在程序执行时使计算机产生某种特定的操作,如赋值、输入输出、控制转移等。而非执行语句只是用来将某些信

息(如变量类型、数据输入输出格式、数组的维数和上下界等)通知编译系统,使编译系统在编译该程序时按照它所给出的信息进行处理,程序运行时它不会使计算机产生任何操作。FORTRAN 90 语言提供的语句见表 1.1。其中,带 * 的语句不符合结构化程序的设计原则,现在已不提倡使用,只是为了保持与以前版本的 FORTRAN 语言兼容才提供。

表 1.1 FORTRAN 90 语句

可执行语句	非执行语句
赋值语句	类型说明语句
* 语句标号赋值语句	数组说明语句
GOTO 语句	公用语句
* 计算 GOTO 语句	参数说明语句
* 赋值 GOTO 语句	INTRINSIC 语句
* 算术 IF 语句	EXTERNAL 语句
逻辑 IF 语句	SAVE 语句
SELECT CASE 语句	隐含类型说明语句
CASE 语句	* 等价语句
END SELECT 语句	PROGRAM 语句
块 IF 语句	FUNCTION 语句
ELSEIF 语句	SUBROUTINE 语句
ELSE 语句	BLOCKDATA 语句
ENDIF 语句	* ENTRY 语句
DO 语句 / EXIT 语句	DATA 语句
\ END DO 语句	FORMAT 语句
WHERE 语句	语句函数语句
CONTINUE 语句	MODULE 语句
DO WHILE 语句	USE 语句
STOP 语句	CONTAINS 语句
* PAUSE 语句	NAMelist 语句
END 语句	PRIVATE 语句
指针赋值语句	类型定义语句
CALL 语句	TYPE 语句
CYCLE 语句	POINTER 语句
RETURN 语句	ALLOCATABLE 语句
ALLOCATE 语句	
READ 语句	
DEALLOCATE 语句	
WRITE 语句	
PRINT 语句	
OPEN 语句	
CLOSE 语句	
INQUIRE 语句	
BACKSPACE 语句	
REWIND 语句	
ENDFILE 语句	

FORTRAN 语言对一个程序单位中各个语句所处位置有一定要求,不能随意放置。表 1.2 给出常用 FORTRAN 语句的位置。

表 1.2 常用 FORTRAN 语句的位置

注 释 行	PROGRAM, FUNCTION, SUBROUTINE, BLOCK DATA		
	FORMAT	PARAMETER	IMPLICIT
			其它说明语句
		DATA	语句函数语句 可执行语句
	END		

表中竖线隔开的语句可以交替出现,横线则划定了各类语句的位置关系。具体说明如下:

①PROGRAM、FUNCTION、SUBROUTINE、BLOCK DATA 语句只能出现在程序单位的开始位置;

②END 语句只能出现在程序单位的结束位置;

③FORMAT 语句可以出现在程序单位的开始语句和 END 语句之间的任何位置;

④说明语句(包括 PARAMETER 语句)优先于 DATA 语句和语句函数语句;

⑤IMPLICIT 语句应在其它说明语句之前出现;

⑥IMPLICIT 语句和 PARAMETER 语句可以交替出现,不分先后;

⑦PARAMETER 语句和其它说明语句可以交替出现;

⑧语句函数语句应在可执行语句之前、说明语句之后;

⑨DATA 语句可以出现在说明语句之后、END 语句之前的任何位置;

⑩注释行不是 FORTRAN 语句,可以出现在 FORTRAN 源程序中的任何位置,甚至是一个程序单位的首语句之前或 END 语句之后。

下面通过一个简单的 FORTRAN 程序增加对 FORTRAN 源程序组成的感性认识。

例 1. 求一元二次方程 $x^2 + 3x - 5.6 = 0$ 的根,并打印结果。

利用一元二次方程的求根公式可知:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

这里 $a=1, b=3, c=-5.6$, 令方程的两个根为 x_1 和 x_2 。解此方程的 FORTRAN 程序如下:

```

C THE ROOTS OF THE QUADRATIC EQUATION
PROGRAM EX1
A=1.0
B=3.0
C=-5.6
X1=(-B+SQRT(B*B-4.0*A*C))/(2.0*A)
X2=(-B-SQRT(B*B-4.0*A*C))/(2.0*A)
WRITE(*,50)X1,X2
50 FORMAT(1X,'X1=',F6.2,'X2=',F6.2)
STOP
END

```

该程序的第 1 行是注释行。第 2 行表示这是一个主程序段,该行也可省略不写。第 3 行到第 7 行是赋值语句,分别为变量赋值和进行计算,并将计算结果放在 X1 和 X2 中。第 8、9 行是输出及格式描述。第 10 行是停止语句,表示程序到此停止执行。第 11 行是结束行。

FORTRAN 规定每个程序单位的最后一行必须是 END 语句。

以上程序只有一个主程序单位。主程序单位可以由 PROGRAM 开头,后跟一个程序名,如上例的 EX1,也可以不用 PROGRAM 语句。若有子程序单位,则子程序单位必须用专门的子程序定义语句开头。

1.2.2 源程序的格式

FORTRAN 源程序必须按规定的格式书写,上机运行时也要按格式输入。

FORTRAN 90 的源程序有两种格式,即固定格式和自由格式。固定格式保留了原 FORTRAN 77 的要求。

固定格式的 FORTRAN 源程序按行书写,每行有 80 列,每一列只能写一个字符。按由左到右的顺序,80 列又分为四个区(表 1.3)。

表 1.3 FORTRAN 程序分区

1...5	6	7 8 9 10...72	73...80
C		THE ROOTS OF THE QUADRATIC EQUATION PROGRAM EX1 A=1.0 B=3.0 C=-5.6 X1=(-B+SQRT(B*B-4.0*A*C))/(2*A) X2=(-B-SQRT(B*B-4.0*A*C))/(2*A) WRITE(6,50)X1,X2 FORMAT(1X,2F6.2) STOP END	
50			

(1)语句标号区

第 1 列到第 5 列称为语句标号区。标号区通常是空白的。当本行的语句需要被其它语句引用时,需要在该语句前的标号区处设一个标号。标号取 1~99999 之间的任何整数。标号的大小不影响程序的执行顺序,但同一个程序单位内不得有重复的语句标号。不满 5 位的标号可以在 1 到 5 列的任何地方出现,作用都相同。例如表 1.3 中的 50 为 FORMAT 语句的标号。

(2)续行区

第 6 列为续行区。FORTRAN 规定一行内只能写一个语句,不能在一行内写两个以上的语句。如果一个语句太长,一行写不下时,可以在下一行接着写,接着写的行称为续行。此时,要在第 6 列,即续行区上写一个标志,表示本行是上一行的继续。续行标志为一个除零和空格以外的任意字符。FORTRAN 标准还规定,一个语句最多可以有 19 个续行,加上该语句的开始行,一个语句最多可占 20 行。续行不是一个独立的语句,它的标号区应该是空的。

(3)语句区

第 7 列到 72 列为程序的语句区。FORTRAN 的语句只能写在该区内,但不一定必须从第 7 列开始,可以由第 7 列以后的任何地方开始写,也可以在 72 列之前的任何列中断一个语句而从下一行接着写。

(4)空白区

从第 73 列到第 80 列是供程序员作标记或注释用的,不能写程序内容。上机时,该区内容

不送入计算机,编译程序对这部分内容也不做处理。

除以上四个区外,当任意行的第 1 列写有字母 C 或“*”时,表示该行为注释行。注释行的用途是供编制程序者对程序或程序的某一部分做必要的说明或注释,以帮助别人理解程序。注释行内容一般是文字,若一行写不下时,可以在下一行继续注释。继续注释的行仍在第 1 列用字母 C 或“*”作为标志。注释可随程序送入机器,机器不编译注释行,只保存并在列程序清单时原样打印出来。表 1.3 程序中的第 1 行内容为注释行。

用自由格式书写的 FORTRAN 源程序更加随意,与固定格式有以下不同。

①每个源程序行可以包括 0~132 个字符。

②一个语句可以在一行的任何位置出现,一行可以写几条语句,以“;”号作为语句间的分隔。“;”出现在一行的最后时,可以不写。例如某行有三条语句:

语句 1;语句 2;语句 3;

语句 3 后的“;”可以不写。若一行只有一条语句,最后的“;”可以不写。

③除可以使用固定格式的注释外,若以字符“!”开始的内容为注释内容,此种注释内容到源程序行的末端,注释行的续行仍为以“!”开始的行。

④若一条语句一行写不下,可以从下一行继续,此时在该行的最后写一个“&”符号作为标记,表明下一行是本行的继续。“&”符号对注释内容不起作用。FORTRAN 90 规定最多可以有 39 个继续行。

⑤若某语句被其它语句引用,可以使用语句标号。语句标号出现在被引用语句之前,这不会与语句内容混淆,因为在 FORTRAN 中不会出现由数字作为一个语句的开始。

1.3 FORTRAN 字符集

书写 FORTRAN 程序的字符的全体称为 FORTRAN 字符集。字符集包括:

①26 个英文字母 A~Z 或 a~z(除字符型数据外,FORTRAN 编译程序一般不区分大小写字母);

②10 个阿拉伯数字 0、1、2、3、4、5、6、7、8、9;

③13 个专用符号

□	空格	=	等号	-	下画线
+	加号	(	左括号	&	“与”号?
-	减号	)	右括号	;	分号
*	乘号或星号	,	逗号	<	小于
/	除号或斜杠	.	小数点	>	大于
:	冒号	'	单引号	?	问号
\$	货币符号	%	百分号	"	双引号
!	感叹号				

每个字符在源程序的语句行中占一列,不能将字符写在上角或下角,如 X_1 应写成 X1,占二个字符位置。

除字符串常数以外,任何 FORTRAN 语法成分必须用字符集中的字符表示,凡是非字符集中的符号必须用字符集中的符号替换。例如: π 、 α 、 β 等是数学中常用的符号,但在 FOR-

TRAN 程序中均为非法字符,必须转换成字符集中的字符才能使用。

1.4 FORTRAN 数据类型

数据是程序的处理对象,不同类型的数据在计算机中的处理是不同的。比如:不同类型的数据在计算机中所占存储单元的个数、表示形式、表示范围、运算方法以及输入输出格式等都不同。在程序设计语言中用数据类型这一抽象概念对所处理的数据进行分类,每种数据类型定义了一种特定类型数据的集合及该类数据处理规则的集合。

FORTRAN 语言提供了以下六种基本数据类型:

数据类型	类型名
整型	INTEGER
实型	REAL
双精度型	DOUBLE PRECISION
复型	COMPLEX
逻辑型	LOGICAL
字符型	CHARACTER

其中整型、实型、双精度型和复型称为数值型数据;逻辑型和字符型称为非数值型数据。

1.5 FORTRAN 数据对象

程序中一个具体的数据称为数据对象。FORTRAN 语言程序处理的数据对象有常量、变量、表达式和函数。

1.5.1 常量

常量是指程序运行过程中其值固定不变的量。按数据使用方法不同,常量又可分为直接常量和符号常量。直接常量也称为常数。每种 FORTRAN 数据类型都有自己的常数。

一、整型常数

整型常数用来表示一个正、负或零的整数值。除符号外,整型常数中不能包含其它任何非数字符号。例如:0、+3、85、-427、1250、-32767 都是正确的整型常数,而 0.0、-4.0、100.0、48,750、1,000,000 都是不正确的整型常数。

FORTRAN 语言的标准文本没有说明整数的表示范围,具体计算机系统的整数的表示范围有所不同。例如,对 16 位字长的计算机系统,一个整型数在机内以 16 位二进制代码表示时,所能表示的有符号的整数范围是 $-2^{15} \sim 2^{15} - 1$,即 $-32768 \sim +32767$ 。对 32 位字长的机器以 32 位二进制代码代表一个整数,允许范围为 $-2^{31} \sim 2^{31} - 1$,即 $-2147483648 \sim 2147483647$ 。因此,当使用 16 位字长的计算机时,若程序中出现一个大于 32767 或小于 -32768 的整型常数时,机器会出现“溢出”错误。

整型常数在计算机内是个精确值。

二、实型常数

实型常数用来表示一个带小数部分的实数。在 FORTRAN 中实型常数有两种表示形式,

即基本实常数和指数型实常数。

(1)基本实常数

基本实常数又称小数型实数,与日常习惯写法相同,由一个正(或负)号、整数部分、小数点和小数部分组成。正号可省略,并且允许没有整数部分或小数部分,但小数点不能没有。例如:

0.0 .02 314. -27.456 +0.00372

都是正确的基本实常数。

(2)指数型实常数

指数型实常数一般用于表示一个较大或较小的数。例如,以下数值对应的指数型实常数为:

数 值	指数型实常数
1.2×10^8	1.2E+08
-2.025×10^{16}	-2.025E+16
0.4415×10^{-5}	0.4415E-05
-10^4	-1E+04
15×10^{-6}	15E-06

可见指数型实常数由三部分组成:一个整数或小数型实常数后跟一个字母 E, E 后跟一个正或负的整数。

使用指数型实常数时要注意: E 后的指数部分必须是整型数,表示以 10 为底的乘方数。若出现 $1.4E+2.5$ 是错误的。另外 10^{-5} 不能写成 E-05, 而应写成 1E-05 或 1.E-5。若要将 0.25 写成指数型可以是 0.25E0 或 $2.5E-1$, 而不能写成 0.25E。也就是说, E 的两边必须都有内容。指数型实常数的指数部分一般不能超过 2 位整数。

实型数能表示的数值范围大大超过整型数。由于各种计算机系统分配给实型数的位数不同,因此表示的数值范围也不同。16 位字长的机器实型数 X 的允许范围是:

$$-10^{38} < X < 10^{38}$$

实型数表示的是计算机允许精度内的一个近似值,精确度与所用机器有关。

三、双精度型常数

一般计算机中,实型数的有效位数为 7~8 位。在实际应用中,这远远不能满足要求,而双精度型数在计算机内所占的存储位数通常为实型数的一倍,因而精度大大提高。

双精度型常数与实型数中指数表示法基本相同,只是把指数形式中的 E 换成 D。以下是双精度型常数:

3.141592653589798D0	表示	3.141592653589798
-1.23456789D-6	表示	$-1.23456789 \times 10^{-6}$
.01D-8	表示	0.01×10^{-8}
1.5D10	表示	1.5×10^{10}

使用时需要注意:双精度型常数必须写有 D,否则无论写多少位数,都不是双精度型常数。例如将数值 3.1415926535897 作为实型数存放时,只存放 3.141592,只有写成 3.1415926535897D0,机器才会将 14 位有效数字全部存起来。

同样双精度型常数 D 后的指数部分也必须是整型数,它的值可达到 3 位整数。

四、复型常数

一个复型常数由两个数组成。这两个数可以是实型数,也可以是整型数,分别表示复数的实部和虚部。复型常数的形式为:

(a,b)

其中 a 是实部, b 是虚部,中间用逗号隔开,而且必须用括号括起来。以下为复型常数与数学中复数对照关系:

(1.0,2.0)	表示复数 $1 + 2i$
(2.5, -3.07)	表示复数 $2.5 - 3.07i$
(0.0, -0.425)	表示纯虚部值 $-0.425i$
(9.8,0.0)	表示复数纯实部 9.8
(5,16)	表示复数 $5 + 16i$

复数实部和虚部都为零时,零不能省略。

五、逻辑型常数

逻辑型常数只有两个值,即真值和假值。在 FORTRAN 语言中用 .TRUE. 表示真值,用 .FALSE. 表示假值,其中左右两个圆点是必不可少的。

六、字符型常数

FORTRAN 语言中的字符型常数就是用单引号(')或双引号(")括起来的字符串。字符串中的字符不必是 FORTRAN 字符集中的字符,可以是 ASCII 码字符集中的任何可打印字符,也可以是汉字。例如:

'FORTRAN'

'X + Y = '

'9010020'

"天津大学"

"请输入变量 A 的值"

都是正确的字符型常数。

字符型常数的值就是该字符串,两边的引号不是字符串的一部分,只作为字符串的分界符。如果字符串中有单(双)引号,为了与分界符区别,用两个连续的单(双)引号表示,但在计算机处理时只按一个单(双)引号处理。

字符型常数的长度即字符串的长度是指引号内的字符个数。例如:"FORTRAN"的长度为 7;"X"VALUE IS"的长度为 10。注意该字符串中的两个连写的单引号按一个单引号处理,空格也算一个字符。而字符串"天津大学"的长度为 8,计算机中一个汉字按两个字符处理。

1.5.2 变量

一、变量及命名

变量是程序运行中可以改变其值的量。每个变量都有一个名字,称为变量名。变量名由字母和数字组成。FORTRAN 命名变量的规则是:

- ①变量名的第一个字符必须是字母,其余可以是字母、数字和下划线;
- ②变量名的长度最大为 31 个字符;
- ③程序中尽量避免使用 FORTRAN 保留字作变量名。

FORTRAN 中的数组、函数、程序和子程序的命名规则与变量名相同。

以下变量名是正确的：

A BBG5 FX1 SUM01 NN K0235

而下面的变量名是非法的：

5xy (以非字母开头)

A * 8B (含有非字母数字字符)

F(X) (含有括号)

α (字符集中没有)

π (字符集中没有)

二、变量的类型说明

FORTRAN 有六种数据类型,相应的变量也有六种。它们是整型变量、实型变量、双精度型变量、复型变量、字符型变量和逻辑型变量。在程序设计中,各种变量的用途不同,因此要确定每个变量的类型,以便在变量中存放与该变量类型相同的数据。确定变量类型有以下三种方法。

(1)隐含说明(仅适用于整型和实型)

FORTRAN 规定,如果不特别说明,程序中凡是以 I、J、K、L、M、N 六个字母开头的变量名均为整型变量,以其它字母开头的变量名为实型变量。例如,在程序中不加任何说明,直接引用变量

I MAT1 J001 NT LENG KEEP

时,FORTRAN 编译系统自动按整型变量处理;而直接引用变量

A1 XB001 SUM TOTAL FNX W

时,按实型变量处理。

所谓隐含说明即通过变量名的第 1 个字母就能知道这个变量的类型,通常称这个规定为“I—N 规则”。I—N 规则只适用于整型变量和实型变量。

(2)用类型说明语句说明(又称显式说明)

在 FORTRAN 程序中可以不加任何说明直接使用整型变量和实型变量,其类型可由 I—N 规则确定,但另外四种类型的变量必须先说明类型才能使用。FORTRAN 语言用类型说明语句显式地说明变量的类型。

类型说明语句的一般形式是：

类型名 变量名列表

其中,类型名可以是 INTEGER、REAL、DOUBLE PRECISION、COMPLEX 或 LOGICAL 中任一个。

变量列表是被说明的变量名字列表。当含有两个以上的变量时,变量名之间以逗号隔开。例如,以下类型说明语句

```
INTEGER      X1,Y2,N,IOM
REAL         M1,NT,X,Y
DOUBLE PRECISION  X11,X12
COMPLEX      F1,F2
LOGICAL      TT,FF
```

都是正确的。它们分别说明变量 X1、Y2、N、IOM 为整型变量;变量 M1、NT、X、Y 为实型变量;变量 X11、X12 为双精度型变量;变量 F1、F2 为复型变量;变量 TT、FF 为逻辑型变量。

字符型变量的类型说明形式与另外五种类型的变量说明略有不同,一般形式是:

CHARACTER * n, a1[* n1], a2[* n2]... , ar[* nr]

其中:n 为一正整数或取正整数的表达式,表示所说明变量可取字符串的最大长度,如果省略,则默认为 1。a1、a2、...、ar 为被说明的字符变量名。n1、n2、...、nr 为正整数或取正整数的表达式,指明每个被说明变量可取字符串的最大长度,如果省略,则默认为 n。

例如:

语句“CHARACTER * 5, A, B * 3, C”说明 A、B、C 均为字符型变量,A、C 可取的最大字符串长度为 5,而变量 B 可取的最大字符串长度为 3。

语句“CHARACTER X, Y, Z * 10”说明了 X、Y、Z 均为字符型变量,其中 X、Y 可取的最大字符串长度为 1,而变量 Z 可取的最大字符串长度为 10。

如果长度用表达式表示,则须用括号括起来,例如:

CHARACTER * (2 + 3) A, B, C * (2 * 3)

长度也可以写成一个(*),例如:

CHARACTER * (*) A, B

表示 A、B 为字符型变量,其长度可由另外的参数说明语句说明,或在函数及子程序调用时确定。

例如:

CHARACTER SS * (*)

PARAMETR(SS = 'YES')

要注意的是这种说明字符变量的方式只能用于两种情形。一种是在主程序中说明字符型符号常数;另一种是在过程子程序中说明字符型变量参数。

在使用字符型变量的时候,可以直接使用变量名,也可以提取字符变量中某一部分连续的字符串即字符子串。字符子串表示形式为:

a(e1:e2)

其中 a 为字符型变量名;e1、e2 为取正整数值的表达式。e1 代表子串在变量 a 中的起始位置,e2 代表子串在 a 中的结束位置。例如,若 AA 为字符型变量,值为“ABCDEFGH”,则子串 AA(3:6)的值为“CDEF”。

当字符子串中的 e1 = 1 时,可以省略不写,此时表示从字符变量的开始取子串到 e2 为止。如上例子串 AA(:3)的值为“ABC”。

同样,当字符子串中的 e2 为字符变量最大长度值时,可省略不写,此时表示从 e1 开始取子串到最末尾。如上例子串 AA(3:)的值为“CDEFGH”。

当 e1 = e2 时,表示只取一个字符,如子串 AA(3:3)的值为“C”。

当 e1、e2 都省略时,表示取字符变量的全部字符。例如,子串 AA(:)的值为“ABCDEFGH”。

无论以上哪种情形,表示子串时的冒号均不能省略。

在程序中,某个变量的类型一经用类型说明语句显示说明后,原来的隐含说明对该变量就不起作用了。例如,用类型说明语句“INTEGER A1, B2”将变量 A1、B2 说明为整型,在程序中若遇到 A1、B2 就按整型变量处理,而不再按 I—N 规则处理了。同一程序单位中整、实型变量

可以用类型说明语句显示说明。

FORTRAN 90 的变量说明除使用类型关键字外,后跟两个冒号和用逗号隔开的变量名表,例如:

```
INTEGER::x,y
```

(3)用 IMPLICIT 语句隐式说明变量类型

可以用 IMPLICIT 语句将 n 个以某字母开头或某几个字母开头的变量名指定为一种类型。例如:

```
IMPLICIT INTEGER(A,B)
```

该语句指定在程序段内以字母 A 和字母 B 开头的变量名均为整型变量名。这样在一个程序单位内,如果不作特别说明,凡遇到以 A、B 字母开头的变量名都按整型变量处理。又比如:

```
IMPLICIT REAL(H-K),INTEGER(W-Z)
```

表示从字母 H 到 K,即以 H、I、J、K 开头的变量名均为实型变量;而以 W、X、Y、Z 开头的变量名均为整型变量名。隐式说明也可用于双精度型或复型变量,例如:

```
IMPLICIT COMPLEX(C),DOUBLE PRECISION(N)
```

变量说明类型使用的注意事项如下。

①以上三种说明类型的方法在一个程序单位内可以混合使用。三种同时出现时的优先顺序为类型说明语句最优,其次为 IMPLICIT 语句。以上两种都没有被说明的变量,按 I—N 规则处理。例如:

```
IMPLICIT INTEGER(A-D)
```

```
REAL A2,B2
```

说明以字母 A 到 D 开头的变量为整型变量,而 A2 和 B2 为实型变量。

②一个变量名只能在类型语句中出现一次,也就是说一个变量只能属于一种类型。

③类型说明语句和 IMPLICIT 语句只在本程序段内有效。使用时这两种语句都应出现在程序前部说明语句部分,而 IMPLICIT 在所有类型说明语句之前。

④IMPLICIT 语句除可以说明 FORTRAN 六种数据类型变量外,还可以说明数组、语句函数名和外部函数名的类型。

1.5.3 内部函数简介

在程序设计中经常会遇到求三角函数、指数、对数、平方根等运算。FORTRAN 语言提供了一系列函数来完成这些运算。这些函数称为内部函数,又称库函数。

内部函数是机器本身的编译系统预先定义好的函数。使用这些函数时,只要正确地写出函数名字,并按要求提供自变量的值就可以了。例如计算 $\sin 1.0$ 的值,只要写成 $\text{SIN}(1.0)$ 就能给出值 0.8414709;写出 $\text{SQRT}(9.0)$ 就能给出 $\sqrt{9.0}$ 的值 3.0。

几个最常用的内部函数见表 1.4。

使用内部函数需注意以下几点。

①每个内部函数都要求一个或 n 个自变量,使用时自变量必须用括号括起来。例如求 $\sin x + \cos x$ 应写成 $\text{SIN}(X) + \text{COS}(X)$;又如求三个变量 x_1 、 x_2 和 x_3 的最大值,应写成 $\text{MAX}(X1,X2,X3)$ 。

②自变量可以是常数、变量、数组元素、表达式或内部函数。