第一章　前　言

IBM 个人计算机上的 Fortran 与标准 ANSI X3.9—1978 (FORTRAN—77) 的一个子集相当，在这本手册中称为 IBM Fortran，它也包括完全 ANSI X3.9—1978 级的一些特征。

可以看出，IBM Fortran 已被增强，以致易于对现有的 Fortran 66 程序进行转换。例如：Fortran 66 中的 DO循环的语义是由编译程序选择的。

IBM-PC 的连接程序允许将 Fortran 目标模块与其它语言的目标模块结合在一起,如与 IBM-PC 的宏汇编程序和 IBM-PC 的 Pascal 目标模块均可结合，使需要用不同语言来书写程序不同部分的用法得以实现。

IBM Fortran 由 Fortran 编译程序和目标模块库构成，该目标模块建立 Fortran 运行时的库。建立一个可执行 Fortran 程序的第一步是编译它的（一个或多个）源程序模块，编译的结果形成目标模块，它与 Fortran 运行库相连接后产生一个运行文件，这一运行文件，可以为 IBM-PC 的 DOS调用。

和包括 IBM Fortran 编译程序的输出一样，被连接的目标模块和库可以包括 IBM-PC 宏汇编程序和 IBM-PC Pascal 编译程序的输出。

一、表示法的几点约定

本手册上所用的表示法的约定如下：

• 出现在程序中的是大写字母和专用字符。(如：CONTINUE.)

• 小写的斜体字母和字是应在正文中描述的实际语句中提供的项。一旦一个小写项被定义，它将在该程序的整个论述中保持其含义。(如：PROGRAM pname.)

例如，描述整数的编辑的格式中的大写和小写字母表示为Iw,其中 w是一个非零无符整常数。

于是，在一个实际语句中，一个程序可以包含 I 3 或I 44。描述实数的编辑格式是 Fw•d，这里 d是一个无符整常数，在实际语句中 F 7.4 或 F 22.0 是正确的。注意，要照样采用作为专用字符的点号。

• 方括号中是可选择的项，例如 A[w]表明 A 或 A 12（作为指定的一个字符格式）都是允许的。

• 省略号（•••）指出省略号前面可选择的项可以出现一次以上。

例如，下述的计算 GOTO 语句表明，用 s表示的、用逗号隔开的语法项可以重复任意多次。

GOTO (s[, s]•••)[,] i

• 空格在 FORTRAN 语句的描述中通常没有任何含义，在本章‘ FORTRAN 程序结构 ”一节中描述的关于空格的一般规则给出空格的解释。

二、FORTRAN程序结构

1. 字符集

一个 Fortran 源程序包含一系列字符，这些字符是：

• 字 母 ：A—Z 的 52个大写和小写字母。

• 数字：0、1、2、3、4、5、6、7、8、9。

• 专用字符：ASCII字符集中其余的可打印字符。

除了字符串常数和 Hollerith 字段，对一切上下文中的小写字母，Fortran均作为大写字母来解释。因此，下面的用户定义名在 IBM FORTRAN 系统中不加区分。

ABCDE abcde AbCdE aBcDe

IBM Fortran 字符集的排序序列为 ASCII 序列。

2. 行

一个 Fortran 源程序也可以看作是行的序列，在一行中仅前面 72个字符为编译程序认为是有效的。编译程序将忽略第 72 列以后的所有字符。

如果一行少于 72 个字符，编译程序把其长度看作是有效的(对这些的解释见本章‘字符表达式’中字符常数的描述)。

3. 列

在一已知行中的字符被分为列。例如，第一个字符是在第 1 列，第二个字符在第 2 列，依此类推。

一标记字符可以放在一行的第 1—6列，它导致下一字符被解释为在第 7 列，这个标记在列表文件中将被扩展到适当的空格数，所有其它标记被传递，如同传递到列表文件一样。

列	用 途
1—5	语句标号和注释指示符
6	继续指示符
7—72	源语句

4. 初始行

初始行是在第 6 列为一空格或一个 0 的任意行，并且它不是一个注释行或一个元命令行。这行的前五列是全部空格或包含一个标号。除去跟在逻辑 IF 语句之后的语句，Fortran 语句以一个初始行开始。

注：当在初始行之后使用续行时，在初始行继续列上使用一个 0 来改善可续性。

例：

```
     0  IF ((A. LT. 0). AND. (B. LT. 0)
     1  .AND. (C. LT. 0))
     2  THEN
```

5. 空格

除了下面指出的空格以外，在 Fortran 源程序中空格是无效的，它可用来改善 Fortran 程序的可读性。

例外的情形是：

- 字符串常数中的空格。
- Hollerith 字段中的空格。
- 在第 6 列的空格，它区分初始行与续行。

6. 注释行

注释行对 Fortran 程序的执行无任何影响。

遇下面任一情况，一个行被当作一注释行。

- 第 1 列出现一个“C”（或“c”）。
- 第 1 列出现一个“*”。
- 一行全部为空格。

注释行后必须紧跟一个初始行或另一注释行，绝对不能跟一个续行。

注：在一个 Fortran 程序结尾的额外的空行导致编译时错误，因为 Fortran 把它们解释为注释行，尽管其后未跟一个初始行。

7. 标号

一个语句标号是一个 1—5 位数字的序列，在各程序单位中它是唯一的。至少一位数为非 0。一个标号可以放在一个初始行的第 1—5 列的任何位置。空格和先行 0 无效。

8．续行

一个续行可以是非注释行或元命令行的任意行，它在第6列是除空格或 0 以外的任何字符，续行的前 5 列必须是空格。

续行提供了书写一个语句的更大空间，如果一个语句在单个初始行写不下时，最多可以扩展到九个续行。

9．语句

一个 Fortran 语句包含一个初始行和最多九个续行。语句的字符是在这些行的第 7 列到第 72 列见到的字符，最多 660 个字符。END 语句必须整个地写在一个初始行中，没有任何其它语句可以有出现一个 END语句的初始行。

10．程序单位

子程序与主程序都叫做程序单位。

一个子程序用 SUBROUTINE语句或 FUNCTION 语句开始，用 END语句结束。

一个主程序可任选地用一个 PROGRAM 语句开始。

Fortran 语言规定了语句和行之间有一个确定的顺序以建立一个 Fortran编纂。通常，一个编纂差不多包含一个主程序，零个或几个子程序（关于程序单位和子例程的编纂详见附录 D)。语句的排序规则见下述。

11．主程序和子程序

一个主程序以一个 PROGRAM 语句开始，或者以不是 SUBROUTINE 或 FUNCTION 语句的任何语句开始，用一个 END语句作结束。

12．语句次序

在一个程序单位内，无论是一个主程序还是子程序，语句必须以与下列规则一致的顺序出现：

(1) SUBROUTINE 语句或 FUNCTION 语句或 PROGRAM语句（如果有的话）必须作为程序单位的第一个语句。

(2) FORMAT语句可以出现在 SUBROUTINE 语句或 FUNCTION 语句或 PROGRAM 语句（如果有的话）之后的任何位置。

(3) 全部说明语句必须先于所有 DATA 语句、语句函数语句和可执行语句。

(4) 在说明语句中，IMPLICIT 语句必须先于所有其它说明语句。

(5) 全部 DATA 语句必须出现在说明语句之后，先于所有语句函数语句和可执行语句。

(6) 所有语句函数语句必须先于全部可执行语句。

13. 程序单位中的语句次序

下列图表说明如下：

· 在其它语句之上或之下示出的语句必须按标出次序出现。

· 注释行或格式语句可以散布在对着它们出现的其它语句中间。

<table>
<tr><td rowspan="6">注释行</td><td colspan="2">PROGRAM, FUNCTION 或 SUBROUTINE 语句</td></tr>
<tr><td rowspan="5">FORMAT
语　句</td><td>IMPLICIT 语句</td></tr>
<tr><td>其它说明语句</td></tr>
<tr><td>DATA 语句</td></tr>
<tr><td>语句函数语句</td></tr>
<tr><td>可执行语句</td></tr>
<tr><td colspan="3">END 语句</td></tr>
</table>

三、 数据类型

在 IBM Fortran 中有四个基本数据类型。

1) 整型

2) 实型

3) 逻辑型

4 字符型

本节描述每一类型常数的性质、值的范围和格式。

对于无格式文件与随机存取存储，IBM Fortran 各数据类型的存储要求是：

类　　型	存储(字节)
LOGICAL	2 或 4
LOGICAL * 2	2
LOGICAL * 4	4
INTEGER	2 或 4
INTEGER * 2	2
INTEGER * 4	4
CHARACTER	1
CHARACTER * n	n
REAL	4
REAL * 4	4

1. 整型

数据类型 integer （整型）是正数和负数的子集。一个整型值是相应整数的精确表示。一个整型变量占有 2 个或 4 个字节的存储。

IBM Fortran 中，一个整数可以指定为 INTEGER*2，INTEGER*4 或 INTEGER。前两种分别指定 2 个和 4 个字节的整数，第三种指定 2 个或 4 个字节的整数，取决于 $STORAGE 元命令的设置（缺省元命令是 4 个字节）。

一个 2 字节的整数可以包括范围是 -32,767 到 32,767 的任意值，一个 4 字节整数可包括范围是 -2,147,483,647 到 2,147,483,647 的任意值。

整型常数是由一位或多位十进制数前面加上一个任选的算术符“正”(+)或“负”(-)构成的，在整型常数中不允许出现小数点，下面是整型常数的例。

```
        123      +123      -123      0
   00000123      32767    -32767
```

2. 实型

数据类型 real （实型）是由实数的一个子集构成：单精度实数。一个单精度实数值是一个要求的实数的近似，一个单精度实数值占有 4 个字节存储。精度是六位十进制数。

一个实型常数是一个基本实常数，一个基本实常数跟一个指数部分或一个整常数跟一个指数部分，如：

```
      +1.000E-2        1.E-2        1E-2
      +0.01          100.0E-4     .0001E+2
```

都表示同样的实数值：1/100

单精度实数值的范围是：

```
    3.0E-39—1.7E+38   （正范围）
  -1.7E+38  -3.0E-39（负范围）
```

一个实型常数包括一个任选符号跟一个整数部分，一个小数点，一个小数部分和一个任选指数部分。整数和小数部分由一位或多位十进制数构成，小数点是一个点号（.），无论整数部分还是小数部分都可以省略，但不能两者均省略。

下面是一些基本实常数的例。

```
-123.456     +123.456     123.456
-123.        +123         123.
-.456        +.456           .456
```

指数部分由字母‘E’或‘e’后面跟一个任选地冠以符号的一位或二位整常数，一个指数表示 其前面的值乘以 10 的整数次幂，一些指数部分的例如下：

```
E12    E-12    E+12    E0
```

3. 逻辑型

数据类型 logical （逻辑型）由两个逻辑值 .TRUE.（真）和 .FALSE.（假）构成，一个逻辑型变量占有 2 个或 4 个存储字节。

一个逻辑数在 IBM Fortran 中指定为 LOGICAL*2，LOGICAL*4 或 LOGICAL。前二者分别指定 2 个或 4 个字节的逻辑数，第三个指定可以是 2 个或 4 个字节的逻辑数，这要取决于 $STORAGE 元命令的设置（如果缺省该元命令，则为 4 个字节）。

只有两个逻辑常数，.TRUE. 和 .FALSE.；它们表示两个相应的逻辑值。.FALSE. 的内部表示是一个全“0”的字，.TRUE. 的内部表示是一个最低位为 1、其余全“0”的字，如果一个逻辑变量在任何其它位有值，则其逻辑值为未定义。

逻辑变量的含义不受元命令 $STORAGE 的影响，其出现主要是为了可与 ANSI 要求兼容，即 LOGICAL，REAL 与 INTEGER 变量是同样大小。

4。 字符型

数据类型 character（字符型）是 ASCII 字符的序列。字符值的长度等于序列中的字符数。一个具体的常数或变量的长度是固定的，它必须是 1—127 个字符。

字符变量序列中的每个字符占有一个存储字节，如果长度为奇数，则增加一个字节。字符变量总是定位在字的边界，在一个字符值中允许有空格字符，它是有意义的。

一个字符常数是由一对引号括起来的一个或多个字符，在字符型常数中允许空格字符。每个字符计数为 1。字符型常数中的一个引号要用两个相邻的引号来表示，两个引号之间没有空格。一个字符型常数的长度等于引号间的字符数，双引号计作一个引号字符。一些简单的字符常数是：

```
'A'    ' '    'Help'
'A very long CHARACTER constant'    ''''
```

最后一个例中，''''，表示一个引号。

Fortran 源程序每行的上限是 72 列，少于 72 列的行用空格填充直到第 72列，因此，当一个字符常数延伸到跨过行的边界时，其值的分配如同该行填充空格直到第 72列并接在续行之前一样。

例如：

```
200  CH='ABC
     XDEF'
```

字符常数的长度为 63。

四、表 达 式

Fortran有四种表达式：

1) 算术表达式

2）字符表达式

3）关系表达式

4 逻辑表达式

1. 算术表达式

一个算术表达式产生一个整型值或实型值。最简单的算术表达式是：

- 无符整型或实型常数。
- 整型或实型变量的引用。
- 整型或实型数组元素的引用。
- 整型或实型函数的引用。

在算术表达式中出现的一个变量引用或数组元素引用的值必须是有定义的。还有，一个整型变量的值，必须由一个算术值定义，而不是原先在一个 ASSIGN 语句中赋给的一个语句标号值。

其它算术表达式由上述简单的形式，采用括号以及下面的算术运算符产生。

算术运算符：

运算符	操作	优先级
**	乘幂	最高
/	除	中
*	乘	中
-	减或负	最低
+	加或正	最低

所有出现在算术表达式两个操作数间的运算符是二元运算符。“ + ”或“ - ”也可以是一元运算符，放在它们的操作数之前。

同级优先级的操作是向左结合的，这点对乘幂例外，乘幂是向右结合的。

因此，

A/B * C 与 (A/B) * C 等价。

A ** B ** C 与 A** (B ** C) 等价。

同多数高级程序设计语言中一样，算术表达式可按通常的数学意义形成，只是 FORTRAN 禁止两个运算符接连出现。

因此，

A ** - B是禁止的，仅管

A ** (- B) 是允许的。注意，一元的负号也是优先级最低级的。因此，

- A * B 解释为 - (A * B)

括号可以用在表达式中以控制结合和求值的顺序。

整数相除：

两个整数相除的结果是两个值的数学商所截取的整数。因此，7/3 求得的值是 2， (- 7)/3求得的值是 - 2， 9/10 求得的值是 0，9/ (- 10) 求得的值也是 0。

类型转换和结果类型：

当一个算术表达式的所有操作数是同一类型时，由表达式产生的值也是该类型，当操作数是不同数据类型时，表达式产生的值的数据类型由下面级别决定：

算术数据类型	级 别
REAL	高
INTEGER * 4	中
INTEGER * 2	低

当一个操作具有两个不同数据类型的操作数时，结果值的数据类型取最高级的操作数的数据类型。例如，对一个整型元素和一个实型元素的操作产生数据类型为实型的值。

一个表达式的数据类型是计算表达式中最终实现的那个操作的结果的数据类型。操作的数据类型分类为：

INTEGER * 2 INTEGER * 4 或REAL

整型操作仅对整型操作数实行操作。由相除产生的小数截断成整数而不是舍入。

因此下例求得的是 0而不是 1。

```
1/4+1/4+1/4 + 1/4
```

注意：对于 INTEGER（没有 *2或 *4扩展）类型的存储取决于$STORAGE 元命令的使用，详见第三章。

实型操作用于对实型操作数或实型和整型操作数的组合实行操作。当一个操作有一个实型操作数和一个整型操作数时，整型操作数首先转换为实型数据类型，方法是小数部分每一位都给定为 .0 然后采用实型运算来计算此表达式。

例如： A = N/B

N 转换为实型数据类型，对N和B实行实型除法。

当一表达式包含混合数据类型时，所实行的整型和实型操作依运算符的优先级顺序进行。

例如： Y = X * (I + J)

先对 I 和 J 实行整型加法，和变为实型数据类型，对结果和 X 实行实型乘法。

注意： IBM Fortran 不检验整数溢出，如果超过整数值的范围，出现不可预料的结果。

2. 字符表达式

一个字符表达式产生一个字符型的值，字符型表达式的形式是：

- 字符常数
- 字符变量的引用
- 字符数组元素的引用
- 括在括号内的任意字符表达式。

字符表达式无运算符。

3. 关系表达式

关系表达式比较两个算术或字符表达式的值。

一个算术值不能同一个字符值比较。关系表达式的结果是逻辑型的。

关系表达式可以用如下所示的任一关系算符来比较值。

关系算符：

算符	所表示的操作
.LT.	小　于
.LE.	小于等于
.EQ.	等　于
.NE.	不等于
.GT.	大　于
.GE.	大于等于

在操作数间出现的一切算符是二元算符，例如：

```
A .GT. B
X .EQ. 7
```

上述例子是正确的表达式，这里没有相关操作数间的结合或相对的优先。因此，

```
A .LT. B .NE. C
```

是不正确的。上述例子违反了操作数的类型规则，关系表达式只能出现在逻辑表达式中。

具有算术操作数的关系表达式可以有一个整型操作数和一个实型操作数，在这种情况下，关系表达式计算之前，整型操作数要转换成实型。

具有字符操作数的关系表达式以 ASCII 的排序序列比较它们的操作数的位置。如果在排序序列中出现的早，则称该操作数小于另一个，如果不等长的操作数进行比较，则较短的操作数以加空格的方法扩展到与较长的操作数等长。

4。 逻辑表达式

一个逻辑表达式产生一个逻辑型的值。逻辑表达式的最简单形式是：

- 逻辑常数
- 逻辑变量的引用
- 逻辑数组元素的引用
- 逻辑函数的引用
- 关系表达式。

其它逻辑表达式由上述简单形式用括号和下述逻辑算符生成。

逻辑算符：

算符	操作	优先级
.NOT.	取反	高
.AND.	与	中
.OR.	或	低

出现在逻辑表达式两个操作数间的 .AND. 或 .OR. 算符是二元算符，.NOT. 是一元算符，放在其操作数之前。优先级相同的操作是向左结合的。例如：

```
A .AND. B .AND. C
```

等价于

```
(A .AND. B) .AND. C
```

作为优先规则的一个例子，

```
.NOT. A .OR. B .AND. C
```

被解释成

```
( .NOT. A) .OR. (B .AND. C)
```

两个 .NOT. 算符不能相邻。

然而，

```
A .AND. .NOT. B
```

是一个具有两个相邻算符而被允许的例子。

逻辑算符的含义是它们的标准数学语义，例如 .OR. 是“或”，即

```
.TRUE. .OR. .TRUE.
```

求得的值为 .TRUE.。

5. 数组元素名

数组元素名用来引用数组的一个元素。

```
arr(sub [, sub]...)
```

这里：arr是数组名，sub是下标表达式。

```
C ASSIGN THE 4TH ELEMENT OF
C ARRAY A THE VALUE 3.8
    A (4,1,1) =3.8
```

一个下标表达式是一个整型表达式，用来选择数组的一个特定元素。下标表达式的数目必须与数组说明符中的维数的数目相匹配。下标表达式的值必须界于 1 和它表示的维数的上界之间。

6. 函数引用

一个函数引用可以出现在算术或逻辑表达式中。一个函数引用的执行使得函数被计算，而得到的值用作含有它的表达式中的一个操作数。

```
fname ([arg[,arg]]... )
```

这里：fname 是用户或系统定义的外部函数名，内部函数名或语句函数名。

arg是一个实在变元。实在变元可以是一个算术表达式，一个数组或一个用户或系统定义的函数或子例程。这里是一些在表达式中函数引用的例。

```
1+SIN(.5)
A+B+USERFN(9)
.TRUE. .AND.BITFN(3)
VAR1/XYZ(3,A)
```

关于形式参数更详细的资料见第四章，实在变元的数目必须与函数定义中的情况一样，对应的类型必须一致。

7. 表达式的优先级

当算术算符、关系算符和逻辑算符出现在同一表达式时，其相对优先级如下：

算　　符	优先级
算术算符	高
关系算符	中
逻辑算符	低

8. 表达式的计算规则和限制

在一表达式中被引用的任一变量、数组元素或函数在引用时必须已被定义。整型变量必须用一算术值定义，而不是由ASSIGN语句赋值的一个语句标号值。

无数学含义的算术操作，例如被 0除，是不允许的。其它不允许的操作有 0 的 0次幂或负次幂，负数的实数次幂。

五、FORTRAN 名

一个 Fortran 名用来表示一个变量、数组、函数或子例程。

一个 Fortran 名由一个字母字符打头后跟字母数字字符（最多 6个字符）组成，空格可以出现在一个 Fortran名中，但无意义。

任意正确的字符序列可以用作任意 Fortran名，但是没有像其它语言那样的保留名。由 Fortran编译程序用作关键字的字母字符序列同用户定义的名字不相混淆。

编译程序由其上下文识别关键字，且决不限制用户定义名的使用，因此，一个程序例如可以用 IF、READ或 GOTO作数