IBM

个人微型計算机
参考資料

PC Fortran语言

上海长江电子计算机厂

前　言

这是一本为 IBM PC 编写的 Fortran 语言参考手册。运用这本手册应首先具有对 Fortran 习惯用语的一些知识。虽然这本手册并不是一种指导性的教学读物，然而每个章节对 Fortran 语言都分别作了详细的解说。本手册的内容按排如下：

- 第一章"引言"介绍符号的常规用法、Fortran 程序结构、数据类型、表达式和 Fortran 名称。
- 第二章"Fortran 程序的编译"阐述编译、连接和执行 Fortran 程序的技巧。
- 第三章"编译程序的元命令"阐述处理 Fortran 源文本的编译程序元命令。
- 第四章"语句"缮述控制、程序函数和子程序、I/O(输入/输出)以及说明语句，此外对 DATA(数据)语句亦作了缮述。
- 第五章"I/O(输入/输出)系统"阐述 Fortran I/O 系统，包括 Fortran I/O 的基本概念及 FORMAT(格式)语句。
- 第六章"内部函数"阐述可用于 Fortran 程序的内部函数。
- 附录 A"输出信息"计算机能向你发出的信息一览表。
- 附录 B"IBM Fortran 与 ANSI Fortran 的区别"缮述 IBM Fortran 与标准子集语言的区别。
- 附录 C"连接程序(LINK)"提供连接程序的信息。
- 附录 D"目标程序块的连接"缮述如何用不同的编译程序来连接目标程序块。
- 附录 E"实例会话"一个如何输入、执行、改正、重执行和如何运用 IBM-PC Fortran 程序的例子。

目　录

第一章 引 言

目 录

IBM-PC Fortran，本手册中称为IBM Fortran，在子集合一级上，与标准ANSI X 3.9-1978(Fortran77)相一致，同时也包括ANSI X 3.9-1978的一些其他特性。

IBM Fortran使Fortran66程序的转换更为方便，例如将Fortran66的语义转换成Do loops(循环语句)本身就是编译程序的一种功能。

IBM-PC连接程序，使你能将Fortran目标程序块与其他语言的程序块，以及IBM-PC的MACRO汇编程序和IBM-PC的Pascal编译程序联合起来，便于你用多种语言编制程序。

IBM Fortran包括Fortran编译程序和一个目标模块库，它构成了Fortran的运行时间库。编制一个可执行的Fortran程序，第一步是编译它的源程序块，然后再将得出的目标块与Fortran运行时间库连接起来，产生一个运行文件，这个运行文件可通过IBM-PC DOS进行调用。

目标模块和连接起来的目标模块库，可包括IBM-PC MACRO汇编程序的输出，IBM-PC Pascal编译程序的输出和IBM Fortran编译程序的输出。

符号常规用法

本手册的符号常规用法如下：

- 输入程序的大写字母和特殊字符形式如CONTINUE
- 小写斜体字母和单词是你提供给文本中实际语句的条款，一旦小写条款定义后，它的意义在程序讨论的全过程中就不再改变。例如PROGRAM pname

  举例：Iw表示描述整数编辑格式中的大写和小写，其中w是非零，无符号的整常数。

这样，在实际语句中，程序可能包含有I_3或I_{44}。描写实数编辑的格式为Fw.d，其中d是一个没有符号的整常数，在实际语句中则F7.4或F22.0就有效。应注意环节，作为一个特殊字符，要用斜体。

- 方括号表示任选项，例如A[w]表示A或A_{12}均有效，(作为一种说明字符格式的方法)
- 省略号(…)表示省略号前的任选项可能出现多项。

举例说：下面描述的计算GOTO(转向)语句，表示由逗号分开的S，S表示的语法项可能重复多次：

GOTO(S[，S]…)[，]i

- 空格在Fortran语句描写中一般没有什么规要意义。本章内的"Fortran程序结构"一节中所描述的一般空格重则，是控制空格的编译规则。

Fortran程序结构

字符集

Fortran源程序由一连串字符所组成，包括；

- 字母：从A到Z的52个大写和小写字母。
- 数字：0，1，2，3，4，5，6，7，8，9。
- 特殊字符：从ASCII字符集中保留下来的可打印字符。

Fortran，除了在字符常数中和何勒内斯(Hollerith)字段中以外，规定将上下文中的小写都翻译成大写。因此类似下述用户定义的名称均无法被IBM Fortran系统所区别：

ABCDE abcde AbCdE aBcDe

IBM Fortran字符集的排列次序，按照ASCII序列

行

我们把Fortran源程序可看作是一连串行，编译程序只认真处理每一行的前72个字符以后的所有字符它都不予识别。

若一行少于72个字符，则编译程序就以它的长度作为一定意义来处理。(本章的"字符表达式"一节内，将对此作进一步说明)

列

行的每一个字符均与列一一致，例如第一个字符在列1，第二个字符在列2，依次类推。

标号字符置于每行的1-6列内。于是要进行编译的字符就从第7列开始。在列表文件中，这个标号可扩充几个空格，所有其他标号也都有如此进入列表文件。

字符所占的列在Fortram中的定义：

列	用途
1-5	语句标号和注解指示符
6	继续指示符
7-72	源语句

初始化

凡在第六列内为空白或0的，非注解行，非元命令行，均称为开始行。该行的前五列必须全部空白或置有一个标号。除了逻辑IF后面的语句外，Fortran语句均由开始行开始。

注：若初始行后面用了继续行，则初始行的继续列内的(0)，只是用来增进其易读性。

例如：␣

```
␣␣␣␣␣␣ OIF  ((A.LT.0).AND.(B.LT.0)
␣␣␣␣␣␣ 1   .AND.(C.LT.O))
␣␣␣␣␣␣ 2   THEN
```

空格字符

除下列情况外空格字符在Fortran源程序中均无重要意义，仅用来增进Fortran程序的易读性，

其例外有：

- 常数串中的空白
- 何勒内斯字段中的空白
- 第六列内作为区别开始行与继续行标准的空格

注解行

注解行在任何情况下均不影响Fortran 程序的执行。

若符合下列中任一条者，均作注行论

- 在第一列中有一个“C”(或“c”)者
- 在第一列中有一个“*”者
- 全行空格者

注解行必须紧接一个开始行或另一个注解行，他们后面不能跟继续行。

注：Fortran 程序末尾的额外空格行，常导致编译时的错误，因它后面未跟开始行。Fortran会把它们当作注解行编译。

标号

语句标号是由 1 到 5 个数字构成的一个数列。在每一个程序单元中，都是独立的标号，每一个标号中至少有一个数字不是零。标号可置于开始行的 1-5 列中的任何位置。空格和首位为零均无意义。

继续行

非注解行，或除了第六列内为空格或 0 外不含任何字符的非元命令行，均为继续行。继续行的前五列必须是空格。

继续行为书写语句提供更大的空间，若一个语句超过一个单开始行，则它可以扩展到最多达九个继续行。

语句

Fortran 语句，包括一个开始行和多至九个继续行，语句字符占这些行的第 7 到 12 列，最多可达 660 个字符，END 语句须全部写在开始行上而不能在 END 语句开始行中出现其他语句。

程序单元

辅程序和主程序称为程序单元。

辅程序以 SUBROUTINE(子程序)语句或 FUNCTION(函数)语句开始，以 END 语句结束。

主程序可选择一个 PROGRAM(程序)语句开始。

Fortran 语言加强了语句和行句一定次序，形成了 Fortran 编译，一般讲，一次编译至多只能包括一个主程序和没有或几个辅程序(详见附录 D)以下为语句规程。

主程序和辅程序

主程序以一个 PROGRAM 语句开始，或除了 SUBROUTINE 或 FUNCTION 语句外的任何语句开始，并以END 语句结束。

语句次序

在一个程序单元中，不论是主程序还是辅程序，语句必须按下列次序出现。

1. SUBROUTINE 或 FUNCTION 语句，或 PROGRAM 语句，必须是程序单元的第一个语句。

2. FORMAT 语句可在 SUBROUTINE 语句，FUNCTION 语句或 PROGRAM 语句后的任何部位出现。

3. 一切说明语句必须放在所有 DATA 语句，语句函数语句和可执行语句之前。

4. 在说明语句中，IMPLICIT(隐)语句必须位于所有其他说明语句之前。

5．一切 DATA 语句必须在说明语句之后，和一切语句函数语句与可执行语句之前。

6．一切语句函数语句必须置于一切可执行语句之前。

程序单元中的语句次序

下表的意义即：

·上下各别表示的语句，必须按指明的次序出现。

·注解行或 FORMAT 语句可分散与其他同时出现的语句放在一起。

<table>
<tr><td rowspan="6">注解行</td><td colspan="2">程序，函数或子程序语句</td></tr>
<tr><td rowspan="5">FORMAT
(格式)语句</td><td>IMPLICIT(隐)语句</td></tr>
<tr><td>其他说明语句</td></tr>
<tr><td>DATA(数据)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="2">END(结束)语句</td></tr>
</table>

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

本节内将叙述各类型的特性、范围以及各类型常数的形式。

用于非格式化文件的 IBM Fortran 数据类型所需的存贮容量和随机存取存贮器的存贮容量见下表：

类　型		存贮容量(字节)
LOGICAL	(逻辑型)	2 或 4
LOGICAL*2	(逻辑型*2	2
LOGICAL*4	(逻辑型*4)	4
INTEGER	(整型数)	2 或 4
INTEGER*2	(整型数*2)	2
INTEGER*4	(整型数*4)	4
CHARACTER	(字符)	1
CHARACER*n	(字符*n)	n
REAL	(实数)	4
REAL*4	(实数*4)	4

整型

整型是正负数的子集，整数值是相应整数的真正代表，一个有效整数占 2-4 字节存贮量。

在 IBM Fortran中，整数可规定为 INTEGER*2，INTEGER*4 或 INTEGER 三种，前二种分别为2字节整数和 4 字节整数，后一种按 $STORAGE(存贮)元命令的建立可以

是2字节或4字节整数(既定值为4字节整数)。

一个2字节整数可为-32767到32767范围内的任何一个值，一个4字节整数则可为-2，147，483，647到2，47，483，647范围内的任何一个值。

整常数是前加任意算术符,正(+)号或负(-)号的一个或几个十进制数字,(一个或几个十进制数字，前面加上正(+)号或负(-)号，)在整常数中不允许有十进制小数点，下面为整常数的例子：

```
    123          +123          -123          0
00000123         32767         -32768
```

实型

实型数由一个实数的子集构成：是一个单精度实数，单精度实数值是期望实数的近似值，一个单精度实数值占4字节存贮量，其精确度为六个十进制数字。

实型数可以是一个基本实型数，或一个跟有指数部分的基本实型数，或一个跟有指数部分的整型数，例如：

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

全部代表同一个实型数：$-\frac{1}{100}$

单精度实数值的范围为：

```
3.0E-39   到   1.7E+38     (正值界)
1.7E+38   到   -3.0E-39    (负值界)
```

一个实型数由一个正(负)号跟一个整数部分，一个小数点，一个小数和一个正(负)指数部份组成，其整数和小数部分由一个或几个十进制数字组成，小数点为一个句号(。)，可以缺少整数部分或小数部分，但不能两者同时缺少，下面为一些基本实型数示范：

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

指数部分由字母"E"或"e"跟一个带正(负)号的一位或二位整型数组成，指数n表示它前面的值应乘10的n次方，下面是一些指数部分的范围：

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

逻辑型

逻辑型数据由二个逻辑值真和假组成，一个逻辑变量占2到4个字节存贮量。

IBM Fortran中可将逻辑型数据规定为LOGICAL*2，LOGICAL*4或LOGICAL三种，前二种分别为2字节和4字节逻辑型数据，第三种按$STORAGE(存贮)元命令的建立或是2字节逻辑型数据或是4字节逻辑型数据(既定值为4字节逻辑型数据)。

逻辑常数只有二个，即.TRUE.(真)和.FALSE.(假)，它们代表二种相应的逻辑值，.FALSE.的内部表示是一个全部为O(O'S)的字，而.TRUE.的内部表示为一个最低位上为一个1余全为O(O'S)字的。若逻辑变量包含任何其他位值，则其逻辑意义为未定。

逻辑变量的意义不受$STORAGE元命令的影响，$STORAGE元命令主要是允许同ANSI的需要相一致，使逻辑变量，实数变量和整数变量的大小相同。

字符型

符字型数据是一个 ASCII 字符串，字符值的长度与该字符串的字符数相等，一个特定常数或变量的长度是固定的，且必须在 1 到 127 字符之间。

一个字符串中，每一字符占一字节存贮量，若该字符串长度成奇数，则再加一个字节。字符变量始终与字边界相匹配。在一个字符组中允许有空格字符，并且是有意义的。

一个字符常数是用一对撇号''夹一个或几个字符构成，在字符常数中允许存在空格字符，每一个字符记数为 1，而字符常数本身内部的撇号则用二个不夹空格的连续撇号来代表，字符常数的长度同二个撇号间的字符数目相等，一个双重撇号，按单撇号字符计数，下面为一些字符常数的范围：

```
'A' ''          'Help'
'A Very long CHARACTER Constant' '' ''
```

最后一个例子""代表一个单撇号。

Fortran 允许源行占 72 列，较短的行则用空白填满到第 72 列。因此，当一个字符常数超过行界时，它的赋值按用空格填满到第72列的行同样赋值，并置于连续行之前，因此，字符常数：

```
200␣␣␣␣CH = 'ABC
     XDEF'
```

的长度为 63。

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）逻辑表示式

算术表达式

算术表达式或是一个整数，或是一个实数，算术表达式的最简形式有：

- 无符号的整常数或实常数
- 整数和实数的变量引用
- 整数和实数的数组元素引用
- 整数和实数的函数引用

变量引用或数组元素引用的值必须以算术表达形式出现，而整数变量引用的值必须用算术值定义，不是用原在 ASSIGN(赋值)语句中的语句标号值。

其他的算术表达式是以上述四种简单形式为基础，再加上括号和算术符号构成的：

算术符号

算符	意义	优先级别
**	取幂	最高级
/	除	中等级
*	乘	中等级

-	减或相反值	最低级
+	加或相同值	最低级

所有算符均为二目算符，置于其算术表达式的运算对象之间，+或-可以是一目符，置于其对象之前。

同等级之间的运算顺序，既可自左至右，也可自右至左，因此：

A/B*C　　与　　(A/B)*C　　相同

A**B**C　　与　　A**(B**C)　　相同

算术表达式在大部分程序设计语言中通常可按数学概念的形式表达，但Fortran不允许二个算符紧接着一起出现，因此：

A**(-B)是允许的，

但不能出现A**-B这种情况。

要注意，一目符减号，仍然是最低级运算符，因此：

-A*B　应翻译成　-(A*B)。

在表达式中可用括号来控制运算的结合，和求值的次序。

整数除法

二整数相除所得的结果是二值的数学商，取其整数部分，因此，7/3得2，(-7)/3得-2，9/10得0，9/(-10)得0。

同类型换算，其结果的数据类型不变

一个算术表达式中所有运算对象均属同一型数据者，该表达式所产生的结果值，仍属该类型。若其运算对象属于不同数据型，则其结果数据的类型按下述等级决定：

算术数据类型	等　级
REAL	最高级
INTEGER*4	中等级
INTEGER*2	最低级

若运算二个不同型的算术对象，则取最高级对象的类型为其结果值的数据类型，例如实数与整数运算所得的结果为实型数据。

表达式的数据型是求表达式值时最后一道运算所得结果的数据型，而运算的数据型可分为INTEGER*2，INTEGER*4或REAL。

整数运算只允许在整数运算对象间进行，除法所得的小数均应截去，不能进行舍入。因此，下面算式所得的结果值应为0而不是1。

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

注：INTEGER(不是*2或*4)的存贮接$STORAGE(存贮)元命令的用法而定，将于第三章作详细说明。

实数运算，可在实数运算对象间进行，也可在实数，整数混合运算中进行。若作混合运算，则先应将整数对象转换成实型数数据，在其小数部分加0，然后按实数运算求此表达式的值。

例如：

A=N/B

将N转换为实型数数据，然后在N和B之间用实数运算进行除法。

若表达式内含有混合数据类型，则求值时，按运算优先顺序进行运算，例如：

Y=X*(I+J)

先对I和J进行整数加法，然后将其和转换成实数，再进行'和'与X之间的实数乘法。

注：IBM Fortran对整型数溢出不作校验，因此若整型数值超过限界时，就会出现无法预计的结果。

字符表达式

字符表达式字符表达式所产生的值为字符型值，字符表达式的形式有下列几种：

- 字符常数
- 字符变量引用
- 字符数组元素引用
- 括入括号内的任何字符表达形式

字符表达式不存在运算结果。

关系表达式

关系表达式用于对二个算术表达式或二个字符表达式进行比较。

一个算术值不能同一个字符值进行比较，而关系表达式的结果则是逻辑型。

关系表达式可用下列任一个运算符进行值的比较。

关系运算符

运算符	意义
.LT.	小于
.LE.	小于或等于
.EQ.	等于
.NE.	不等于
.GT.	大于
.GE.	大于或等于

所有运算符均为二目标符，位于其比较对象之间，例如：

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

上述例子为有效表达式，在关系比较对象之间，不存在优先运算和结合运算，因此，

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

为无效表达式。此例违反类型运算规则，此外，关系表达式只可出现在逻辑表达式中。

以算术为对象的关系表达式中，可能一个对象为整型数，而另一个对象为实型数，在这种情况下，用关系表达式进行比较前，应先将整型数转换成实型数。

以字符为对象的表达式，是比较这些对象在ASCII排序中的位置，在序列中先出现的对象小于后出现者，若比较对象不等长时，则较短的对象后加空白字符到等长，然后进行比较。

逻辑表达式

逻辑表达式所产生的结果为逻辑型值，逻辑表达的最简形式有：

• 逻辑常数
• 逻辑变量引用
• 逻辑数组元素引用
• 逻辑函数引用
• 关系表达式

其他逻辑表达式是在上述简单形式的基础上加括号和逻辑运算符构成的：

逻辑运算符

运算符	意义	等级
.NOT.	非	最高级
.AND.	与	中等级
.OR.	或	最低级

其中 AND和OR 为二目算符，位于逻辑表达式对象之间，NOT 为一目算符，置于其对象前面，同级之间的运算则应自左向右进行，例如：

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

和

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

相同，再按上述优先等级次序例如下：

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

写成(.NOT.A).OR.(B.AND.C)也可以，虽然A.AND..NOT.B是允许的，但不允许二个NOT紧接着一起出现。

逻辑算符的意义是它们的标准数学语义，.OR.的意义为非唯一的，就是说，

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

求得的值是：

```
.TRUE.
```

数组元素名

数组元素名用来引用某数组中的一个元素，

```
arr(sub[, sub]…)
```

arr 是某数组的名称，Sub 为下标表达式：

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

下标表达式就是用来选择一个数组特殊元素的整型数表达式，下标表达式的数字必须同数组说明符中的维数相一致，下标表达式的值，必须介于1与它所代表的维数的上界之间。

函数引用

函数引用可出现在算术表达式或逻辑表达式中，函数引用的执行可使函数获得运算，而它的结果值则作为它内含表达式中的一个运算对象来使用。

```
fname([arg[, arg]]…)
```

fname 是用户或系统定义的外部函数，内部函数或语句函数的名称。

arg 为实元，实元可以是一个算术表达式，或一个用户或系统定义的函数或子程序。