微机 FORTRAN 上机编译指南

何光渝
胡旭光
阮　敏

兰州大学出版社

微机 FORTRAN 上机编译指南

何光渝　胡旭光　阮　敏

兰州大学出版社

(甘)新登字第 08 号

内容简介

本书系统地介绍了微机上使用的MS FORTRAN 各种版本的编译系统;针对 386 微机的NDP FORTRAN 386 编译系统及全屏幕编辑软件PE 和QE。第一章介绍了 MS FORTRAN3. 3 版本编译系统。第二章介绍了 MS FORTRAN4. 0 版本编译系统。第三章介绍了窗口式调试工具 Code View。第四章为 MS FORTRAN 语言与标准 FORTRAN 77 结构化语言的差异。第五章介绍了 NDP FORTRAN386 编译系统。第六章介绍了全屏幕编辑软件PE 和QE。本书可作为大专院校理工科各专业"计算机应用基础"课程的教材,也可供科研院所、工矿企业计算机工作人员和科技人员参考使用。

微机 FORTRAN 上机编译指南

何光渝　胡旭光　阮　敏

兰州大学出版社出版
(兰州大学校内)

兰州人民印刷厂印刷	甘肃省新华书店发行
开本:787×1092　毫米1/16	印张:18.5
1994年8月第1版	1994年8月第1次印刷
字数:462千字	印数:1—2000册
ISBN7-311-00590-6/T·10	定价:19.50元

前　言

FORTRAN 语言是国际上最流行的一种适用于数值计算的语言。从目前世界上各种计算机语言的发展状况来看，在作科学计算时，FORTRAN 语言仍然是效率最高的一种语言。但是目前很多人不喜欢用它，原因在于使用 FORTRAN 语言编写用于科学计算的源程序固然好处很多，然而当源程序有逻辑错误时调试起来不方便，且图形功能大部分人不掌握。实际上，为了解决这些问题 Microsoft 公司陆续推出了 MS FORTRAN3.3 版本编译系统(1985 年)、MS FORTRAN4.0 版本编译系统(1987 年)和 MS FORTRAN5.0 版本编译系统(1989 年)等一系列软件。本书旨在采用简明扼要的语言，使用户短时间内掌握 MS FORTRAN 编译和绘图，熟练使用断点设置、跟踪功能和人机对话询问变量值等调试功能，快速找出逻辑错误，调通程序。另外近年来，各单位都已拥有一批 386 微机，为了充分地利用 386 微机的优势，本书还介绍了 NDP FORTRAN 386 编译系统。

本书第一章介绍的 MS FORTRAN 3.3 版本编译系统是国内较流行的微机编译系统，它包含了 FORTRAN 77 语言子集，还有若干扩展，编译元命令的使用为调试程序和使用大数组创造了条件。第二章介绍了 MS FORTRAN 4.0 版本编译系统，它包含了 FORTRAN 77 语言全集及扩展，窗口式 Code View 调试工具能很容易地找到源程序中的逻辑错误，它还支持大程序，允许 FORTRAN 语言与汇编语言、C 语言及 Pascal 语言连接。MS FORTRAN 5.0 版本是对 MS FORTRAN 4.0 版本的发展，最主要的扩展是开发出了图形功能，因而没有单独介绍 MS FORTRAN 5.0 版本，只是在第二章最后加入 5.0 版本的扩展及图形功能的使用方法。第三章详细介绍了窗口式调试工具 Code View 的使用。第四章介绍了 MS FORTRAN 77 语言对标准 FORTRAN 77 语言的扩展。第五章介绍了 NDP FORTRAN 386 编译系统，它是专门为 386 微机开发的 FORTRAN 编译软件。32 位 386 微机的性能指标已达到小型机的水平，如果还采用 MS FORTRAN 编译系统是无法充分发挥 386 的卓越性能，只能把 386 微机降级为快速 286 使用。例如，一般 386 微机都有 2—4 兆内存，NDP FORTRAN 386 突破了 640K 的内存限制，程序可以使用大内存，分配 1 兆以上大数组。本章简明扼要地介绍了 NDP FORTRAN 386 的编译连接、优化和内部函数，同时还介绍了图形功能的使用，并有综合作图举例。第六章介绍了两个全屏幕编辑软件 PE 和 QE，供用户编辑源程序之用。读者若需要本书各章所述的各种软件，可直接与作者联系。(西安石油学院石油工程系何光渝 邮编 710061)

参加本书编写工作的有何光渝(第一、三章，第四章 § 4.1、§ 4.3，第六章 § 6.1)，胡旭光(第五章，第二章 § 2.7、§ 2.8)，阮敏(第二章，第四章 § 4.1，第六章 § 6.2)。全书由何光渝统稿。在整个编写过程中曾参阅了近期出版的参考书，并直接引用了其中一些内容，在此特向有关作者致谢。

由于作者水平有限，缺点和错误在所难免，恳望读者批评指正。

编　者

1993 年 7 月 20 日

目　录

第一章　MS FORTRAN 3.3 版本编译系统

§1.1　MS FORTRAN 编译系统简介

随着科学技术的发展，作为电子计算机应用的一个主要方面——科学计算也日益发展，从目前世界上计算机语言的发展状况来看，在作科学计算时，FORTRAN 语言仍然是效率最高的一种语言。因而，微机上的 FORTRAN 编译系统也日臻完善。

FORTRAN 语言是编译语言，源程序要经过编译、连接，产生可运行文件后才能由机器执行。意即用 FORTRAN 语言编写的源程序必须经过"翻译"，变成机器语言，计算机才能"识别"并执行。而不象 BASIC 程序那样，当文件输入后，只要按 RUN 命令就能运行。MS FORTRAN 编译系统就能起到对 FORTRAN 源程序的"翻译"功能，经"翻译"后所产生的运行文件，可以作为 DOS 的外部命令使用，因此具有很高的运行效率和通用性。

1982 年美国 MICROSOFT 公司为 IBM—PC 微机开发了 MS FORTRAN 2.0 版本编译系统，随后 1985 年又开发了 MS FORTRAN 3.3 版本编译系统。为了 FORTRAN77 结构化语言全集的使用，相继又于 1987 年、1989 年推出 MS FORTRAN 4.0 版本和 MS FORTRAN5.0 版本编译系统。这一系列的版本都各有优缺点，但总的来说，功能越来越多，越来越完善，使用越来越方便。

§1.2　MS FORTRAN3.3 版本编译系统的使用

MS FORTRAN3.3 版本编译系统共有两张 5.25 寸低密软盘组成，它主要由下列文件组成：

FOR1.EXE	第 1 次扫描程序
PAS2.EXE	第 2 次扫描程序
PAS3.EXE	第 3 次扫描程序
FORTRAN.LIB	
DOS2FOR.LIB	专为 MS-DOS2.0 系统接口的基本库
8087.LIB	8087 使用的库程序
MATH.LIB	仿真 8087 用库程序
ALTMATH.LIB	加速库
DECMATH.LIB	十进制数学库
FORTRAN.MAP	连接列表文件
ALTMATH.MAP	连接列表文件
DECMATH.MAP	连接列表文件
8087.MAP	连接列表文件
MATH.MAP	连接列表文件
LINK.EXE	连接程序(2.4 版)

为了和 MS FORTRAN 2.0 版本相似，也可将 PAS2.EXE 改名为 FOR2.EXE。采用 MS FOR-

TRAN3.3 版本进行科学计算机的步骤如下：

1. C>MD FOR33 ↘　　建立子目录 FOR33

2. C>CD FOR33 ↘　　进入子目录 FOR33

3. C>COPY A:*.* ↘　　将上述文件拷贝至子目录中

4. C>COPY A:PE2.* ↘　　将第六章中所介绍的编辑软件 PE 或 QE 拷入该子目录，以便用来编写源程序。（也可采用其他编辑软件）源程序的扩展名为.FOR。

5. 使用 MS FORTRAN3.3 版本对源程序进行两次编译，形成目标文件，扩展名为.OBJ。

6. 使用 LINK 将目标文件连接形成执行文件，其扩展名为.EXE。在 DOS 系统下，键入执行文件名，即可运行该程序。

以上为在硬盘上进行操作的步骤。以上操作也可在软盘上进行，因目前只有双软盘无硬盘的微机极少，故不作介绍。下面介绍具体操作过程。

1. 编译过程

编译过程分二步进行，第一步 FOR1 是编译系统对源程序进行第一次扫描，功能是进行语法检查，若无语法错误，则可进入第二步，否则就修改源程序，再重新进行 FOR1 编译。第二步为 PAS2（或者 FOR2），这是编译系统对程序进行优化处理，产生以.OBJ 为扩展名的目标文件。同时还产生 PASIBF.TMP 和 PASIBF.OLD 文件供 PAS3 使用。第三次扫描 PAS3 可以运行也可以不运行，这取决于是否产生目标码列表文件（通常不需要该文件）。如不运行 PAS3，则可删除 PASIBF.TMP 和 PASIBF.OLD 文件，以节省空间。具体过程如下：

第一步：

```
C:\FOR33>FOR1 ↘
Microsoft FORTRAN77 V3.31 August 1985
(C)Copy right Microsoft Corp 1982、1983、1984、1985
Source filename[.FOR]:S
Object filename[S.OBJ]:
Source listing[NUL.LST]:
Object listing[NUL.COD]:
Pass One No Errors Detected
        63 Source Lines
```

首先出现的提示为 Source filename[.FOR]：这时键入你准备编译的文件名，不需扩展名.FOR，例如上面所显示，表示我们准备编译的文件为 S.FOR。按下回车键后，将继续出现第二行提示，Object filename[S.OBJ]：其中 Object filename 表示将要建立的目标文件名，此时若按 Enter 键，意指默认文件名为 S.OBJ；如果想给目标文件另起名，这时要加上文件类型扩展名.OBJ，再按 Enter 键。接着屏幕出现第三行提示：Source listing[NUL.LST]：表示编译后产生源列表文件。如果不要列表文件，则按 Enter 键；若要列表文件，则键入 S，此时编译系统将会产生源列表文件 S.LST。列表文件的内容是源程序，错误信息及其他信息。最后出现的一行为：Object listing[NUL.COD]：意为编译后产生的目标列表文件。如果不需要则按 Enter 键；若需要可键入 S，则编译后产生目标列表文件 S.COD。目标列表文件的内容为汇编清单。当编译结束时，若有错，将显示错误信息，用户必须重新启动编辑软件将源程序中的语法错误改正后，再进行 FOR1 编译。若没有错，FOR1 编译结束，这时将产生两个中间文件 PASIBF.SYM（符号表文件）和 PASIBF.BIN（中间代码文件）。

第二步：

```
C:\FOR33>FOR2 ↘
```

```
Code Area Size=#0767(1895)
Cons Area Size=#0081( 129)
Data Area Size=#0328( 808)
Pass Two No Errors Detected.
```

其中:第一条为源程序的目标代码长度,第二条为源程序中常数区的长度,第三条为静态分配数据区的长度。第二次编译如不出错,则 FORTRAN 编译全部完成,下面即可进行连接和运行。

注意:上述第一次编译可使用另一种简洁办法,即直接键入:

C:\FOR33>FOR1 S;

这时,屏幕上不出现上述四条提示行,编译结束后,直接显示错误信息或通过信息。

此外,在实际操作时,必须对某一个源程序文件连续进行 FOR1 和 FOR2。然后再对另一个源程序文件进行同样的操作。若连续对两个不同的源程序文件进行 FOR1 编译,则产生的临时中间文件 PASIBF.SYM 和 PASIBF.BIN 是针对后一个源程序文件,再进行 FOR2,则最后形成的是后一个源程序的目标文件.OBJ。

2. 连接过程

```
C:\FOR33>LINK ↓
Microsoft(R)8086 Object Linker Version 3.04
Copyright (C) Microsoft Corp 1983,1984,1985. All rights reserved
Object Modules[.OBJ]:S
Run File [S.EXE]
List File[NUL.MAP]
Libraries [.LIB]:
```

第一行 Object Modules[.OBJ]:意为输入目标文件名,不需要扩展名.OBJ。按下 Enter 键后出现第二行提示 Run File[S.EXE]:意为输入可执行文件的文件名,按 Enter 键,默认为 S.EXE;也可输入不同的文件名。下一个提示为 List File[NUL.MAP]:意为建立 MAP 文件,若不需要,则按 Enter 键;若需要 MAP 文件,可键入 S,表示建立 MAP 文件(列表文件),其名为 S.MAP。最后一行为 Libraries[.LIB]:按 Enter 键表示连接缺省文件 FORTRAN.LIB 及 MATH.LIB,如果要连接其他库文件,这时可键入库文件名。回车后,编译系统开始连接。连接结束,若有错,屏幕上将显示错误信息,读者可进行修改,然后,再进行 FOR1、FOR2 和 LINK,直到无错为止。若无错,屏幕上重新显示 DOS 提示符 C:\FOR33>。

和 FOR1 一样,LINK 也可使用简洁形式:

C:\FOR33>LINK S;

这时屏幕上不显示上述四行提示,连接结束时直接给出错误信息,若无错将重新显示 DOS 提示符。

连接过程中需要注意的是:

(1)MS FORTRAN3.3 版本提供了六个库文件。FORTRAN.LIB 是系统标准库;DOS2FOR.LIB 是专为 MS DOS2.0 改变系统接口的辅助库,其余四个全部是数学库。

MATH.LIB 称为仿真库,用它连接的运行程序能适应两种情况:当系统加有 8087 协处理器时,能自动启动 8087 作 80 位浮点运算;当系统无 8087 时,能用软件对 8087 仿真。因此,用 MATH.LIB 连接的程序,具有较高的运算精度,但无 8087 时,运行效率比较低,且运行文件体积比较大。

8087.LIB 是专门为加有 8087 的机器配的数学库,用它连接的程序运行速度快,精度高,可运行文件短,但运行时必须有 8087。

ALTMATH.LIB 称为交替库，它专为无 8087、而要求有较高运行速度的用户而设计，用它连接的程序浮点运算精度比较低。

DECMATH.LIB 称为十进制库，专为银行、财会系统设计。因为在这些部门中，数值的表示要精确，而一般的浮点表示在规格化时往往有截断误差。

(2)在进行科学计算时，经常有一些子程序是很多方法中通用的，我们可以对它们单独进行 FOR1 和 FOR2，形成目标文件，最后再将主程序和它们连接起来。例如：主程序为 S.FOR，子程序为 A1.FOR 和 A2.FOR，对它们分别经过 FOR1 和 FOR2 编译后，产生 S.OBJ、A1.OBJ 和 A2.OBJ，这时可以

```
C:\FOR33>LINK ↓
Microsoft(R)8086 Object Linker Version 3.04
Copyright(C) Microsoft Corp 1983, 1984, 1985. All right reserved
Object Modules[.OBJ]:S+A1+A2
Run File[S.EXE]:
List File [NUL.MAP]:
Libraries[.LIB]:
```

连接成功后，第一个文件名为执行文件名，这里执行文件名为 S.EXE。也可键入简洁形式：

```
C:\FOR33>LINK S+A1+A2;
```

连接成功后，回到 DOS 提示符下，不给出任何信息。

3. 程序的运行

连接成功后，产生扩展名为.EXE 的相应执行文件，若要运行，只需在 DOS 提示符下键入执行文件名，按下 Enter 键即可运行，如

```
C:\FOR33>S ↓
```

若需要打印输出结果，则打开打印机，按下 Ctrl+Print 键后，再按下 Enter 键，然后键入执行文件名，回车后，程序开始执行，屏幕上显示结果的同时，打印机同时打印出计算结果。

有时需要将计算结果置入一文件里，可在观察计算结果后进行编辑删除等，然后再打印或反复打印，这样可在运行执行文件时键入：

```
C:\FOR33>执行文件名>新文件名
```

当运行执行文件时，屏幕上将不显示输出结果，输出结果都存入新文件中。

§1.3 编译元命令

表 1·1 列出了 MS-FORTRAN3.3 版本的全部编译元命令。这些元命令可在 FORTRAN 源文件中出现，且第一列以 $ 字符开头，编译元命令不允许有续行，也不允许在命令行中任意加空格或其他字符。

编译元命令的作用是引导编译系统以某种特定方式处理 FORTRAN 源文件，它广泛用于程序的调试、跟踪或增强运行文件的某些功能。因此，正确使用编译元命令，可以有效地缩短复杂程序的调试周期，提高运行程序的质量。

表1·1 MS-FORTRAN3.3版本编译元命令一览表

命令格式	说　　明
$DEBUG	使运行程序检查以下各项： (1)整型运算溢出及被0除的情况。
$〔NO〕DEBUG	(2)检查赋值转移语句的去向。 (3)当运行出错时在显示器显示源文件名及行号。直到$NODEBUG时终止检查<缺省值为$NODEBUG>。
$ DECMATH	指示编译器按十进制浮点格式产生实型常数。
$ DO66	使DO语句具有FORTRAN66的功能。
$ FLOATCALLS	指定编译器在浮点运算时用CALL调用仿真库的子程序；
$〔NO〕FLOATCALLS	$NOFLOATCALLS使编译器产生在线中断<缺省值为：$FOLATCALLS>。
$ INCLUDE:' file'	使编译器在指定位置插入file文件再编译。
$ LARGE〔name〔' name〕...〕	指定以name为名的数组为大数组，其存储空间可超出64K。
$〔NOT〕LARGE	$NOTLARGE的作用相反<缺省值为$NOTLARGE>。
$ LINESIZE:n	使列表文件的后续页有n列宽度。
$ LIST	使后续源文件在编译中产生列表文件。$NOLIST则停止后续程序产生列表文件<缺省值为$LIST>。
$〔NO〕LIST	
$ MESSAGE:' chr'	向标准设备显示' chr' 字符。
$ PAGE	列表文件开始新的一页。
$ PAGESIZE:n	使列表文件每页n行，n最小值15<缺省值为66>。
$ STORAGE:n	给源程序中的每个整型、逻辑型变量分配n个存储单元，n为2或4<缺省值为4>。
$ STRICT	使MS-FORTRAN不能实现FORTRAN77子集或全集所设有的功能。
$〔NO〕STRICT	$NOSTRICT终止$STRLCT，使MS FORTRAN具有FORTRAN77子集及其全集所设有的功能。
$ SUBTITLE:' chr'	使列表文件的后续页以chr为子标题。
$ TLTLE:' chr'	使列表文件以chr为标题。

编译元命令可划分为两大类，一类目的在于为程序的编译、调试提供方便，如$DEBUG，$INCLUDE，$LINESIZE，$LIST，$MESSAGE，$PAGE，$PAGESIZE，$SUBTITLE，$TITLE等；另一类用于增强运行文件的某些功能，如$DECMATH，$DO66，$FLOATCALLS，$LARGE，$STORAGE，$STRICT等。

这两类元命令中我们感兴趣的是后一类，前一类中有的元命令的意义显而易见，因此我们只介绍个别元命令的使用方法，而后一类中关系比较密切、且常用的，我们将给出实例。

$LINESIZE，$PAGESIZE两个元命令用来设置列表文件的每行的字符数(40—132)和每页的行数(大于15)。

$SUBTITLE，$TITLE的作用是在列表文件的指定页上加标题和子标题，标题的内容由这两个元命令所带的字串给定。标题和子标题都出现在页首，直到出现下一个同类命令为止。

$PAGE用于引导列表文件重开一页。

$MESSAGE:' string' 为前端编译提供跟踪信息，当FOR1运行期间碰到该元命令时，即将string的内容显示在光屏上。有计划地在源程序中分段安排这个元命令，就可把前端出错的范围确定下来。

$INCLUDE:' filename' 主要为编译提供方便，前端编译期间，把filename指定的文件插入源程序$INCLUDE所在的位置，然后接着编译，该元命令常用于公用块的编译，例如，有以下一个公用块，在主程序和若干子程序中都要使用。

```
COMMON/BLK1/A1,A2,B(1000)
COMMON/BLK2/A(16000)
COMMON/BLK3/IB(1200),I,J,K,I1,I2,I3,L1,L2,L3
```

若以其中一个子程序为例：

【例 1·1】

```
1:C       EXAMPLE1.1
2:        SUBROUTINE SUB1(X,Y,Z)
3:        COMMON/BLK1/A1,A2,B(1000)
4:        COMMON/BLK2/A(16000)
5:        COMMON/BLK3/IB(1200),I,J,K,I1,I2,I3,L1,L2,L3
6:        DO 1 I=1,I1
7:1       A(I)=X*Y+Z
8:        RETURN
9:        END
```

此时可将公用块建立为一个单独的文件，以 COMMO. FOR 命为文件名，而将例 1·1 改为例 1·2，则编译、连接时与例 1·1 的结果一样。

【例 1·2】

```
1:C       EXAMPLE 1·2
2:        SUBROUTINE SUB1(X,Y,Z)
3:$INCLUDE:'COMMO.FOR'
4: $MESSAGE:'COMMON WAS INCLUDED SUCCESS!'
5:        DO 1 I=1,I1
6:1       A(I)=X*Y+Z
7:        RETURN
8:        END
```

上例中，第四行的元命令，会使 FOR1 运行期间在光屏上显示：

COMMON WAS INCLUDED SUCCESS！ 字样。

$DEBUG 用于对运行文件的跟踪，它有以下三项主要功能：

(1)检查被零除或整型数溢出。

(2)检查赋值转移语句的去向，如出现不合理的转向，则显示出错。

(3)当运行出错时，显示该语句所在的源文件名及行号。

$DEBUG 可出现在程序的任何位置，其作用直到出现 $NODEBUG 为止。如例 1·3 的程序，至少有两个错误会在运行时出现，一是 i*j 的值超出整型数的表示范围；二是实型数被零除。当我们在程序中加有 $DEBUG 元命令时，则运行结果附在该例的末尾，它能正确指出原文件名及行号，如没有使用该元命令，则只能显示出错误，而不能指出它的位置。

【例 1·3】

```
1:C       TEX18.FOR
2:$DEBUG
3:              I=3276894
4:              J=8754341
5:              K=I*J
6:              X=0.0
```

```
7:                  Y=1./X
8:                  ASSIGN 10 TO I
9:                  GOTO I,(10,20)
10:                 I=2
11:                 GOTO(10,20),I
12: 10              STOP' GOTO 10'
13: 20              STOP' GOTO 20'
14:                 END
? Error:long integer math overflow
Error Code 2201
Line      5 in MAIN of TEX18.FOR
PC=0AA2:00B1;SS=0B20,FP=F632,ST=F634
B>
```

如果我们排除了第一个错误，则又可找到第二个错误，如例 1·4 所示：

【例 1·4】

```
1:C         TEX19.FOR
2:$DEBUG
3:C                 I=3276894
4:C                 J=8754341
5:C                 K=I*J
6:                  X=0.0
7:                  Y=1./X
8:                  ASSIGN 10 TO I
9:                  GOTO I,(10,20)
10:                 I=3
11:                 GOTO (10,20)I
12:10               STOP' GOTO 10'
13:20               STOP' GOTO 20'
14:                 END
?ERROR:REAL DIVIDE BY ZERO
ERROR CODE 2100
LINE     7 IN MAIN OF TEX19,FOR
PC=0610:0045;SS=0B1A,FP=F644,SP=F640
```

$DEBUG 元命令的加入，会使运行文件的代码增加，因此，一个程序运行考核正确之后，一般应将它删除，并重新编译。

$DO66元命令引导编译系统按 FORTRAN66的语义处理 DO 语句。我们知道，对于 DO 语句 FORTRAN77在以下两点与 FORTRAN66不同：

(1)FORTRAN66中的 DO 语句的循环域至少被执行一次，而 FORTRAN77规定循环域的执行次数为：

MAX0(((终值－初值＋步长)/步长),0)

因此，当终值大于初值且步长为负，或终值小于初值而步长为正时，循环域均不被执行。

(2)FORTRAN66允许循环域延拓。在一定条件下，允许用控制语句转入或转出循环体，而 FORTRAN77却不允许转入循环体。

如果程序中需要执行FORTRAN66中DO语句的功能，则可在源程序的说明语句或可执行语句之前加入$DO66元命令。它只能出现在其他元命令之前，而且源程序中只能出现一次。

$FLOATCALLS 引导编译系统以调用子程序库的方式处理浮点运算。由于CALL指令比中断快，而且与过程的接口比较简单，当用仿真库MATH.LIB连接目标块时，$FLOATCALLS几乎可以全部省去直接仿真的附加操作，如果机内没有8087，那么此元命令可使浮点运算速度快约25%，但运行文件要增大。如果机器装有8087，此时浮点运算用8087实现，而无需调库；如果机器装有8087协处理器，用$NOFLOATCALLS元命令会使运行速度更快，且运行文件较短。

$LARGE 用于开辟大体积的数组，这是MS FORTRAN3·3版中一个很有特色的编译元命令。一般情况下，数组体积的分配受到内存分段管理的限制，一段的容量为64KB，因此在IBM—FORTRAN2·0及其早期版本中，一个数组的最大体积只能占用64KB，如果在MS FORTRAN3·3版中不使用$LARGE，其规定也和上述情况一样。

$LARGE 引导编译器在内存中分配大数组并产生最有效的寻址代码。它有两种使用方式：第一种形式是$LARGE，称为一般方式，它可出现在程序的任何位置，且影响源文件中所有子程序中的数组(除用$NOTLARGE指定的数组外)；另一种形式是$LARGE<arry>,<arry>…称为指定方式，即把跟在$LARGE后以arry为数组名的数组指定为大数组，这种方式只适用于该元命令存在的子程序段。

MS FORTRAN3·3版还规定，二维及多维数组的最大上界必须小于65536，而一维数组的上界仅受内存体积的限制。

对于常数组，当其体积大于64KB时，编译系统会自动按大数组处理，但这种内存分配方式不适用于可调数组。

例1·5为在具有512KB内存容量的机器上开辟大数组的检查程序，当第二行元命令删除时，运行结果出错(K=300)，若加上该元命令则运行正确。

【例1·5】

```
1: C
2: $LARGE
3:              PROGRAM U
4:              COMMON/BLOCK1/A(100000)
5:              WRITE(*,*)' INPUTING THE K'
6:              READ(*,'(BN,I7)')K
7:              CALL Y(A,K)
8:              STOP
9:              END
10:
11:             SUBROUTINE Y(A,N)
12:             DIMENSION A(N,N)
13:             X=1·0
14:             DO 1 I=1,N
15:             DO 1 J=1,N
16:             A(I,J)=X
17:1            X=X+1.
18:             DO 2 I=1,N
19:2            WRITE(*,3)(A(I,J),J=1,N)
```

```
20:3            FORMAT(1X,10F6,0)
21:             RETURN
22:             END
```

$STORAGE 用于分配整型和逻辑型变量的字节数，当使用 $STORAGE:2时，指定程序中每个整型和逻辑变量为二个字节。若使用 $STORAGE:4时，则每个整型和逻辑型变量占用四个字节。由于二字节变量的操作代码比四字节变量短，因而，当程序中大量使用整形或逻辑型变量运算时，使用 $STORAGE:2会使程序的运行效率提高，如果程序中大量使用整型数组，此时也会节省相当内存。

$STORAGE 元命令必须出现在说明语句之前，且只能在源程序中出现一次，它的缺省值是 $STORAGE:4。

例1·6是上述几个元命令的使用实例，用仿真库连接目标块(以 K＝100为标准，下同)，运行时间为370秒；用交替库连接，仅需76秒。

【例1·6】

```
1: C
2: $STORAGE:2
3: $LARGE
4: $FLOATCALLS
5:              COMMON/BLOCK1/A(100000)
6:              WRITE(*,*)'  INPUTING THE K'
7:              READ(*,*)K
8:              CALL Y(A,K)
9:              STOP
10:             END
11:
12:             SUBROUTINE Y(A,N)
13:             DIMENSION A(N,N)
14:             X=1.0
15:             DO 1 I=1,N
16:             DO 1 J=1,N
17:             X1=3.1415926*X/180.
18:             A(I,J)=SIN(X1)
19:1            X=X+1
20:             DO 2 I=1,N
21:2            WRITE(*.3) (A(I,J),J=1,N)
22:3            FORMAT(1X,10F7.3)
23:             RETURN
24:             END
```

$STRICT 与 $NOTSTRICT

MS FORTRAN3.3具有标准 FORTRAN77所没有的某些功能，下述五项功能是 MS FORTRAN3·3所特有的：

(1)字符表达式可赋给非字符变量。

(2)字符表达式和非字符表达式可以互相比较。

(3)允许字符变量与非字符变量存在于同一个COMMON 块。

(4)字符变量和非字符变量可以等价。

(5)非字符变量可用字符数据初始化。

$STRICT 元命令引导系统不使用上述五项功能，而用标准 FORTRAN77的规定处理。$NOTSTRICT 则是允许系统具有上述五项功能、编译的缺省值是$NOTSTRICT。

MS FORTRAN3·3编译系统提供了适应于多种环境的运行库和二十条编译元命令。这给用户调试、跟踪、考核 FORTRAN 源文件带来了很大方便。在 MS FORTRAN3·20版以前的版本中，由于数组的体积受64KB 的限制，使得中大规模数据量(10万左右)的处理，很难在微型机上实现，现在就可借助 $LARGE 元命令方便地办到。因此，MS FORTRAN 为在微型计算机中开发大型系统创造了条件，随着32位大容量(4—16MB)微机的出现，它将显得更富于生命力。

§1·4 错误信息

编译出错信息

编译格式检查：如果代码含有格式错误，编译器将显示一个特定的警告信息。该信息的形式是：

··· Warning-invalid format error(n)

这里的 n 是运行时下列错误代码之一：

1200，

1204到1213，

1215到1224，

1226到1228，或者1232

代码	错误信息
1	读源程序致命错
2	标号域中有非数字字符
3	续行太多
4	遇到致命的文件结束
5	续行中有标号
6	编译元命令遗漏字段
7	不能打开文件
8	不识别的元命令
9	输入文件用无效的格式
10	文件嵌套的层次太多
11	整常数错
12	实常数错
13	常数的位数太多
14	标识符太长
15	字符常数没有括起来
16	字符常数的长度为零
17	输入无效的字符
18	要求整常数

19	要求标号
20	标号错
21	要求给出类型
22	要求整常数
23	在语句末端有多余的字符
24	要求“)”
25	在 IMPLICIT 语句中已经用过的字母
26	要求“)”
27	要求字母
28	要求标识符
29	要求维数
30	数组已定维
31	维数太多
32	不相容的变量
33	标识符已有类型
34	标识符已说明
35	这里不允许内部函数
36	标识符必须是一个变量
37	标识符必须是变量或当前的 FUNCTION
38	要求“/”
39	有名公用块已被保存
40	变量已在 COMMON 语句中出现过
41	变量出现在两个不同的 COMMON 块中
42	下标数目与说明相矛盾
43	下标越界
44	强迫两个单元占用同一地址
45	强迫按相反方向定位
46	强行分配有矛盾的存储单元
47	要求语句号
48	字符项和数字项在同一个 COMMON 块中
49	字符项与非字符项矛盾
50	表达式中出现无效符号
51	表达式中出现子程序名
52	要求整型或实型
53	要求整型、实型或字符型
54	类型不相容
55	要求逻辑表达式
56	下标太多
57	下标太少
58	要求变元

59	要求“=”
60	字符项的长短必须一致
61	赋值类型不匹配
62	要求子程序名
63	不允许出现哑元
65	仅对哑数组作假定大小说明
66	仅对哑数组作可调大小的说明
67	假定的大小必须是最后维
68	可调界必须是参数或在COMMON 语句中
69	可调界必须是简单整变量
70	主程序多于一个
71	有名公用块的大小必须一致
72	不允许哑元
73	不允许 COMMON 变量
74	不允许 SUBROUTINE、FUNCTION 或 INTRINSIC 名
75	下标越界
76	重复数必须大小等于1
77	要求常数
78	类型矛盾
79	变量个数不匹配
80	不允许标号
81	无这样的内部函数
82	内部函数类型矛盾
83	要求字母
84	函数类型与前面调用的矛盾
85	SUBROUTINE/FUNCTION 已经定义过
87	变量类型矛盾
88	SUBROUTINE/FUNCTION 与前面使用的相矛盾
89	不可识别的语句
90	不允许 CHARACTER FUNCTION
91	缺少 END 语句
93	调用中实元个数少于哑元个数
94	调用中实元个数多于哑元个数
95	变元类型矛盾
96	SUBROUTINE/FUNCTION 没定义
98	CHARACTER 长度错
100	语句顺序错
101	不可识别的语句
102	不允许跳入模块
103	标号已用于 FORMAT 语句