Fortran95 简介－全文版

By 陈鲸太

FORTRAN 的演进

FORTRAN 的起源，要追溯到 1954 年 IBM 公司的一项计划。由 JOHN BACKUS 领导的一个小组，尝试着在 IBM 704 计算机上面发展一套程序，它可以把使用接近数学语言的文字，翻译成机械语言。这个计划在刚开始并不被大家看好，但他们在 1957 年交出了成果，也就是第一套 FORTRAN 编译器，FORTRAN 语言也就因此诞生了。FORTRAN 语言的执行效率普遍的令各界满意，它证明了这项计划的可行性，也成为第一个被广泛使用的高级语言。FORTRAN 的名字来自于英文的 FORMULA TRANSLATOR 这两个字，而这两个字恰是数学公式翻译器的意思。

旧版的 FORTRAN77 是在 1978 年由美国国家标准局(ANSI)所正式公布的，之后改版有 1992 年提出的 FORTRAN90 以及 1997 年的 FORTRAN95，本文是为了 FORTRAN 95 所撰写。

编译器简介

1、VISUAL FORTRAN

VISUAL FORTRAN 一开始是起源于 MICROSOFT 的 FORTRAN POWERSTATION 4.0，这套工具后来卖给 DIGITAL 公司来继续发展，下一个版本称为 DIGITAL VISUAL FORTRAN 5.0，DIGITAL 后来被 COMPAQ 合并，所以接下来的 6.0 及 6.5 版就称为 COMPAQ VISUAL FORTRAN。而 COMPAQ 目前又跟 HP 合并，也许下一个版本会称为 HP VISUAL FORTRAN。

VISUAL FORTRAN 被整合在一个叫作 MICROSOFT VISUAL STUDIO 的图形接口开发环境中，VISUAL STUDIO 提供一个统一的使用接口，这个接口包括文书编辑功能，PROJECT 的管理、除错工具等等，所以在使用上其实跟上学期的 VISUAL C++满类似的，同学们上课用过 VISUAL C++，对 VISUAL FORTRAN 应该不会陌生。

VISUAL FORTRAN 6.5 除了完全支持 FORTRAN 95 的语法外，扩充功能方面提供完整的 WINDOWS 程序开发工具，专业版还内含 IMSL 数值链接库。另外它还可以和 VISUAL C++直接互相连结使用，也就是把 FORTRAN 和 C 语言的程序代码混合编译成同一执行档案。

2、在工作站使用 FORTRAN

学校计中工作站也提供 FORTRAN COMPILER，使用方式很简单，只需要在存放 FORTRAN 档案的目录下面输入下面叙述即可：

ccsun33 [u8623033/fortran]% f77 filename.for
这个指令使用 f77 的 compiler，其中 filename.for 就是我们所编写的 FORTRAN 程序档案
ccsun33 [u8623033/fortran]% f90 filename.for
这个指令使用 f90 的 compiler
而指令按下 enter 键之后，会把结果 COMPILE 到 a.out 这个档案里面，我们想要执行这个程序，只要在命令提示字符后打 a.out 就可以看到执行结果：
ccsun33 [u8623033/fortran]% a.out
如果在 compile 的过程中想要把执行档改成别的档名，不要每次都变成 a.out，那我们可以输入下列指令
ccsun33 [u8623033/fortran]%f77 filename1.for –o filename2
或者
ccsun33 [u8623033/fortran]%f90 filename1.for –o filename2
则 filename2 会变成我们的执行档名字
在 COMPILE FORTRAN 的时候，我们必需登入学校计中 ccsun26~ccsun35 的机器才可以使用(学校规定)，并且因为软件总数只有两套，同时间只有两个人能可 compile。

FORTRAN 基本事项

字符集
字符集是指使用 FORTRAN 的时候，所能使用的所有字符有符号。FORTRAN 所能使用的字符集有
1、英文 26 个字母：大小写不分
2、数字：0 到 9
3、22 个特殊符号：有冒号、等号、加号、减号、惊叹号…等等
书面格式
FORTRAN 程序代码的写作格式有两种，FREE FORMAT(自由格式)以及 FIXED FORMAT(固定格式)。简单来说，FIXED FORMAT 是属于旧式的写法，它在写作版面上有很多限制。FREE FORMAT 是 FORTRAN90 之后的新写法，取消了许多旧的限制。FORTRAN 程序代码附加档名为*.F 或*.FOR 的档案，就是指以 FIXED FORMAT 来写作的程序，若以*.F90 为附加档名的档案，就是以 FREE FORMAT 来写作的程序。建议现在都应该改用 FREE FORMAT 来写作程序。
FIXED FORMAT
固定格式之中，规定了程序代码每一行中每个字符字段的意义。如下表所示：

第 1 个字符	如果是字母 c,C 或*(星号)，表示此行是批注

第 1-5 个字符	如果这边是数字，表示这一行的代号，否则应为空白
第 6 个字符	如果是 0 以外的字符，表示这一行程序会接续上一行
第 7-72 个字符	FORTRAN 程序代码的写作区域
第 73 个字符之后	不使用，超过部份会被忽略，有的编译器会有错误讯息

FIXED FORMAT 是为了配合早期需要使用打洞卡来输入程序才发明出来的格式。现在都应该要使用 FREE FORMAT 来写作程序。早期的计算机，还没有使用显示器作为输出装置，不能像现在一样直接利用键盘来修改程序。早期的程序是利用打洞卡片一张一张的记录下来，再拿给计算机执行。有着打洞卡的淘汰，FIXED FORMAT 也没有必要再继续使用下去。不过同学们还是可以大概了解一下，因为仍有些旧程序是用这种格式来写作。

FREE FORMAT

FREE FORMAT 基本上允许非常自由的写作格式，它没有再去规定每一行的第几个字符有什么作用。需要注意的事项只有下面几点：

1、惊叹号「!」后面的文字都是批注。

2、每行可以写作 132 个字符。(注意! 并不是无限长)

3、行号放在每行程序的最前面。

4、一行程序代码的最后如果是符号&，代表下一行程序会和这一行连接。

例子：sample1.f90

```
! Free Format
program main
write(*,*) “hello” ！打印出 hello 这个字
write(*,*) &
 “hello”
wri&
te(*,*) “hello”
end
```

输出、输入及宣告

输出基本范例

输出使用 write 指令，如下例

sample2.f90

```
program main
```

write(*,*) “hello”
stop
end

FORTRAN 程序通常以 PROGRAM 叙述来开头，PROGRAM 后面还要接一个自订的程序名称。这个名称可以完全自订，不需要和档名有任何的关系，这个名字表示「主程序」的名字。我在这里取名为 MAIN，这个名字可以看个人喜好修改。
FORTRAN 程序最后还要有 END 这个叙述，表示程序代码写到这一行结束。
WRITE 指令就是作为输出用，WRITE(*,*)之中两个星号各有各自的意义，前面的星号表示输出的位置使用内定值，也就是屏幕，后面的星号表示不特别设定输出格式。另外，下面三种输出格式其实会得到一样的结果：
WRITE(*,*) “HELLO”
WRITE(6,*) “HELLO”
WRITE(UNIT=6, FMT=*) “HELLO”
关于 WRITE 还有几点要注意
1、每一次执行 write 指令之后，会自动换到下一行来准备做下一次的输出。
2、因为双引号是用来包装字符串用的，所以想要印出双引号的时候，要连用两个双引号。
例：想要印出 MY NAME IS “CASTER”.
就要下达 WRITE(*,*) “ MY NAME IS “”CASTER””. ” 这个叙述
3、FORTRAN90 可以使用双引号或单引号来包装字符串，FORTRAN77 标准中只能使用单引号，不过大部份的 FORTRAN77 还是可以接受双引号。

范例中还有另一个指令 STOP，STOP 是终止程序的意思，它可以出现在程序的任何地方，程序执行到这个指令就会中止。除非必要，不要把 STOP 指令使用在主程序结束之外的其它地方。因为一个程序如果有太多的终止点会容易出错。
STOP 指令在这个地方可以省略，因为主程序的程序代码执行完毕后，程序会自动终止。加上这个指令只是为了更明确表示程序到此结束而已。
END 是用来包装程序代码使用的，说明程序代码已经写作完毕。FORTRAN 90 标准中，可以使用下面三种方法来表示程序代码写作结束，FORTRAN77 只能使用第一种方法。
END
END PROGRAM
END PROGRAM MAIN ！MAIN 是主程序的名字

PRINT 指令用法大致上和 WRITE 相同，只是专门针对屏幕作输出，因此少了指定输出的能力，它也具有限定输出格式的功能，其语法如下：
PRINT *,”输出字符串 ”

宣告

1、整数

整数的宣告法很简单，如下：

integer a !宣告 a 为一个整数，内定范围为 2^32~ －2^32

2、浮点数

real a !宣告 a 为一个浮点数，默认值为单精度浮点数

real*4 a !宣告 a 为一个单精度浮点数，大小为 4byte

real*8 a !宣告 a 为一个双精确度浮点数，大小为 8byte

单精度可记录的数值最大为 3.4*10^38，最小为－3.4*10^38

3、复数

FORTRAN 是少数有提供复数型态的程序语言，宣告方法如下

complex a

设定复数的方法如下：

a=(x,y) !x 为实部，y 为虚部

例如我们设 a=(3,8)就表示 a 是 3+8i

范例：sample3.f90

```
program main
complex a,b
a=(1.0,1.0)
b=(3.0,4.3)
write(*,*) “a+b=”, a+b
write(*,*) “a-b=”, a-b
write(*,*) “a*b=”,a*b
write(*,*) “a/b=”, a/b
stop
end
```

则执行结果如下

a+b= (4.0,5.3)

a-b= (-2.0,-3.3000001)

a*b= (-1.3000002,7.3)

a/b= (0.2655511,-0.047289926)

4、字符及字符串

宣告一个字符的方法如下

CHARACTER a

宣告字符串的方法如下

CHARACTER(10) A !宣告 A 这个字符串的最大长度为 10

另外使用下面几种宣告语法也是相同的结果

CHARACTER*10

CHARACTER(LEN=10)

CHARACTER*(10)

宣告好之后，我们要给予这个变量一个初始值的方法如下：

A=”字符串内容” !双引号在 FORTRAN90 适用

A=’字符串内容’ !单引号在部份 FORTRAN77 及所有 FORTRAN 90 都适用

输入指令

输入指令的基本语法如下：

INTEGER A

READ (*,*) A !读入一个整数，并存到 A 内

READ (5,*) A !同上

READ(UNIT=5, FMT=*) A !同上

第一个星号表示输入的来源使用内定的装置，第二颗星号来源表示不指定输出格式。键盘的输入位置是 5，也就是预设位置，所以可以用星号代替键盘代码。

格式化输出

格式化输出入的控制字符非常丰富，但是常用的并不多，在这里我们只示范几个比较常用的部份。

A. 关于 I

WRITE(*,”(I5)”) 100 !用 5 个字符的字段来输出一个整数

OUTPUT : _ _ 1 0 0

WRITE(*,”(I3)”) 100000

OUTPUT: * * *

输出 10000 需要 5 个字段，但是输出格式只给三个字的字段，因此印出三颗星号作为警告

WRITE(*,”(I5.4)”) 3 !输出五个字符字段，至少输出 4 位，不足补 0

OUTPUT: _ 0 0 0 3

B. 关于 F

WRITE(*,”(F9.3)”) 123.45 !输出 9 个字符字段，包括小数部份 3 个位数

OUTPUT: _ _ 1 2 3 . 4 5 0

C. 关于 E

WRITE(*,”(E15.7)”) 123.45

! 用科学计号表示法，输出 15 个字符字段，小数部份占 7 位

OUTPUT : _ _ 0 . 1 2 3 4 5 0 0 E + 0 3

D. 关于 A
WRITE (*,"(A10)") "HELLO" !用 10 个字符宽度输出字符串
OUTPUT: _ _ _ _ _ H E L L O
WRITE (*,"(A3)") "HELLO"
OUTPUT : H E L

E. 关于 B
WRITE (*,"(B6.5)") 3 !把 3 变成二进制输出，字 6 个字符宽，至少输出 5 位
OUTPUT : _ 0 0 0 1 1

F. 关于 X
WRITE (*,"(3X)") 20 !输出前先填 3 个空格符
OUTPUT: _ _ _ 2 0

变量名称的取名原则

变量名称的长度限制随着各家编译器而有所不同。FORTRAN 77 规定至少要支持到 6 个字符，FORTRAN 90 则规定最少要支持到 31 个字符。变量的名字最好是取成一个有意义的英文单字，这样可以减少程序写作时出错的机会。

IMPLICIT 指令

FORTRAN 标准中有一项不太好的功能，它的变量并不一定要经过宣告之后才能使用，编译器会依变量名称的第一个字母来自动决定这个变量的型态。第 1 个字母若为 I, J, K, L, M, N 的变量会被视为整数型态，其它的变量则会被当成浮点数来使用。来看下面范例：SAMPLE4.f90

```
PROGRAM MAIN
   I=11+22
   WRITE(*,*) "11+22=", J
STOP
END
```

程序执行结果会得到

11+22=0

这个结果当然是错误的，错误是出在程序的第 3 行，原本应该是要输出变量 I，却不小心打成 J，而 J 仍未设定任何数值，所以会输出 0 来。打错字是写程序的过程当中最容易发生的错误，这一类的错误通常很难查觉出来，尤其是在写作大程序的时候。所以建议在 FORTRAN 程序中，开始作宣告之前，都加入下面这个叙述：

IMPLICIT NONE

加入这个叙述之后，会把内定型态的功能关闭，因此这个范例程序若加入这一

行，那么在 COMPILE 的过程中就会发生错误，我们必需事先宣告所有会使用的变量才可以。
IMPLICIT 指令要马上接在 PROGRAM 指令的下一行，不能把它放在其它位置。

常数的宣告
常数的宣告有下列两种方式

A. REAL PI
   PARAMETER(PI=3.14159)
   这个方式先宣告 PI 是一个浮点数，再宣告它是一个常数，也就是我们所说的圆周率
B. REAL, PARAMETER ： ： PI=3.14159
   这个方式同时宣告 PI 为浮点数及常数

程序结构

FORTRAN 的程序结构应该如下

```
PROGRAM xxxx
IMPLICIT NONE
  INTEGER  INT1, INT2              宣告区
  REAL  REAL1, REAL2

  READ(*,*) INT1, INT2             程序主体
  WRITE (*,*) INT1+INT2
  ……….

STOP
END
```

其中宣告区不可与程序主体交错。

FLOW CONTROL

IF…THEN…ELSE

基本语法如下

```
IF(逻辑判断式) THEN
    执行动作 1
ELSE
    执行动作 2
END IF
```

```
SAMPLE5.f90
PROGRAM MAIN
IMPLICIT NONE
    REAL HIEGHT
    REAL WEIGHT

    READ(*,*) HEIGHT
    READ(*,*) WEIGHT
    IF (WEIGHT > HEIGHT-100) THEN
        WRITE(*,*) “TOO FAT!”
    ELSE
        WRITE(*,*) “UNDER CONTROL”
    END IF

STOP
END
```

逻辑表达式

FORTRAN 90 的逻辑运算符号共有下列几种：

= = 相等
/ = 不相等
> 大于
> = 大于等于
< 小于
< = 小于等于
.AND. 如果两边式子都成立，整个条件就成立
.OR. 两边的式子只要有一个成立，整个条件就成立
.NOT. 如果后面的式子不成立，整个式子就算成立
.EQV. 两边式子的逻辑运算结果相同时，整个式子就成立
.NEQV. 两边式子的逻辑运算结果不同时，整个式子就成立

FORTRAN 77 要用缩写来作判断，不能使用逻辑符号
.EQ. 等于
.NE. 不等于
.GT. 大于
.GE. 大于等于
.LT. 小于
.LE. 小于等于

DO 循环
DO 循环基本语法如下：

```
DO 起始值, 终止值, 累加值
    执行程序代码
END DO
```

举例如下：

```
DO I=10, 5, -1
    WRITE(*,*) I
END DO
```

在这个程序中，我们设定初始值是 I 这个变量为 10，然后每次减 1，一直到 I=5 为止，因此程序会印出：
10
9
8
7
6
5

DO WHILE …

DO WHILE 的基本语法如下：

```
DO WHILE (逻辑运算)
      程序代码
END DO
```

因为跟同学们上学期学过的 C 语言几乎一样，因此在这里不再多描述。

数组的宣告与使用

宣告数组

宣告数组有下列几种方法：

integer a(10) !宣告 a 这个数组有 10 个元素

integer , dimension(10) :: a ！同上，另一种作法

而在 FORTRAN 77 当中，我们必需用下面这种方法

integer a

dimension a(10)

请记住在 FORTRAN 当中的数组是从 1 开始算，也就是 a(1), a(2)一直到 a(10)

使用数组

我们要使用已经宣告出来的数组，直接利用其 index 即可，例如：

a(1)=18

也可以利用 DATA 这个叙述

INTEGER A(3)

DATA A /36, 24, 36/

在 FORTRAN 90 中，还可以省略 DATA 这个叙述

INTEGER :: A(3) = (/36, 26, 36/)

使用这个方式必需注意，括号跟除号之间不能有空格，并且冒号不能省略。

函式

子程序(SUBROUTINE)的使用

写程序时，可以把某一段常常被使用、具备特定功能的程序代码独立出来，包装成子程序，以后只要经由呼叫的 CALL 指令就可以执行这一段程序代码。

一个包含子程序的 FORTRAN 程序在结构上大概如下：

```
PROGRAM MAIN
      主程序代码
```

```
END

SUBROUTINE  SUB1()
      程序代码
END SUBROUTINE
SUBROUTINE  SUB2()
      程序代码
END SUBROUTINE
```

主程序不一定要放在程序的最开头，它可以安排在程序中的任意位置，可以先写子程序再写主程序也无妨。子程序的最后一个指令通常是 RETURN，表示程序要返回原来呼叫它的地方来继续执行程序。在主程序内呼叫 SUBROUTINE 就使用 CALL 这个指令。

FORTRAN 在传递参数的时候是使用传址呼叫(CALL BY REFERENCE)，这个意思是说呼叫时所传递出去的参数，和子程序中接收的参数，它们会使用相同的内存地址来记录数据。

范例：SAMPLE6.f90

```
PROGRAM MAIN
IMPLICIT NONE
      INTEGER :: A=1
      WRITE (*,*) “A 的初始值是”, A
      CALL ADD(A)
      WRITE(*,*) “A 后来的值是”, A
      STOP
END

SUBROUTINE ADD(NUM)
IMPLICIT NONE
      INTEGER NUM
      NUM = NUM + 1
      RETURN
END SUBROUTINE
```

这个程序的输出会是

A 的初始值是 1

A 后来的值是 2

很明显的看到 A 的值被 SUBROUTINE 所改变了!!

自订函数 FUNCTION

自订函数的运作基本上跟 SUBROUTINE 非常类似，它也是要经由呼叫才能执行，也可以独立宣告变量，参数传递的方法也如同子程序一样，它和子程序只有两点不同：

1、呼叫自订函数之前必需先宣告。

2、自订函数执行后会传回一个数值。

函数的宣告方法如下：

REAL , EXTERNAL :: ADD

其中 EXTERNAL 这个字表示我们宣告的东西是个函数。

FORTRAN 77 使用分开的宣告方式：

REAL ADD

EXTERNAL ADD

范例：SAMPLE7.f90

```
PROGRAM MAIN
IMPLICIT NONE
REAL, EXTERNAL:: TRIPPLE
        real:: A=1.38
        WRITE (*,*) "A 的初始值是", A
        WRITE (*,*) "呼叫函数",TRIPPLE(A)
        WRITE(*,*) "A 后来的值是", A
        STOP
END

REAL FUNCTION TRIPPLE(NUM)
IMPLICIT NONE
        REAL NUM
        TRIPPLE = NUM * 3
        RETURN
END
```

程序的执行结果如下：

A 的初始值是 1.38

呼叫函数 4.14

A 后来的值是 1.38

我们可以发现，使用了 function，让我们 A 这个变量不产生改变，而得到我们想要的输出，而这个也是使用函数的不成文规定：传递给函数的参数，只要读取它的数值就好了，不要去更变它的资料。传入函数中的参数就是所谓的自变量，而函数传回的值是应变量。自变量是自由变化的，它的值应该不会在使用函数的过程中被改变，如果想要改变传入的参数时，最好使用子程序，而不是使用函

数来完成工作。这个是写作函数及子程序时的不成文规定。