Fortran Dokumentation

Helge Janicke, Niels-Peter de Witt, Karsten Wolke

2. Januar 2002

Inhaltsverzeichnis

Abbildungsverzeichnis

1 FORMAT Anweisung

Standardmäßig geschieht die Ein-/ Ausgabe von Daten *listengesteuert*, d.h die einzelnen Daten werden durch ein Trennzeichen (Komma oder Leerzeichen) getrennt. Für die Darstellung, Umwandlung werden Standardverfahren verwendet.

```
      READ(*,*) A, B
      C = SQRT(A*A + B*B)
      write(*,*) A, B, C
```

Formate bieten die Möglichkeit eigene Formatierungen zu spezifizieren. Sie werden bei READ/ WRITE Operation anstatt des zweiten * angegeben. Es gibt drei Möglichkeiten ein Format anzugeben:

1. Durch eine `FORMAT` Anweisung
2. Durch eine konstante Zeichenkette
3. Durch eine `character` Variable

```
      CHARACTER*5 form

c              12345
      form = '(I10)'
      read(*, form)    A
      read(*, '(I10)') A
      read(*, 1000)    A

 1000 FORMAT(I10)
```

Einige Möglichkeiten Formate zu spezifizieren werden im Folgenden erläutert.

1.1 Formate bei der Eingabe

I Format

```
I w INTEGER
```

Ganze Zahl in w Spalten.

F Format, E Format, G Format, D Format

```
F w.d FLOAT
E w.d FLOAT
G w.d FLOAT
D w.d FLOAT
```

F, E, G Angaben wirken bei der Eingabe gleich.
Reelle Zahl in w Spalten.
Fehlt die Angabe des Dezimalpunktes, so gibt d die Stellung des Dezimalpunktes der Mantisse an.

Der Exponent wird durch ein E oder ein Vorzeichen eingeleitet.

Das D Format verhält sich wie die oben genannten Formate nur wird der Exponent mit einem D eingeleitet.

A FORMAT

A[w] CHARACTER

Zeichenkette in w Spalten. Fehlt die Angabe von w so bestimmt die Länge der Variablen die Spaltenanzahl.

L FORMAT

L w LOGICAL

w Spalten werden werden von links beginnend nach T für .TRUE. und F für .FALSE. durchsucht.

1.2 Steuerzeichen für die Eingabe

w X	Es werden w Spalten der Eingabezeile überlesen.
/	Ende der Eingabezeile, Beginn einer neuen Zeile.
BN	Leerzeichen werden bei der Eingabe von Zahlen übergangen.
BZ	Leerzeichen werden bei der Eingabe von Zahlen als Nullen gewertet.

1.3 Formate bei der Ausgabe

I FORMAT

I w.m INTEGER

Ganze Zahl rechtsbündig in w Spalten. Es werden mindestens m Ziffern ausgegeben. Sind weniger Ziffern vorhanden wird links mit Nullen aufgefüllt.

INTEGER :: a = 123, b = -123, c = 123456

WRITE(*,"(I5)")	a			1	2	3
WRITE(*,"(I5.2)")	a			1	2	3
WRITE(*,"(I5.4)")	a		0	1	2	3
WRITE(*,"(I5.5)")	a	0	0	1	2	3
WRITE(*,"(I5)")	b		-	1	2	3
WRITE(*,"(I5.2)")	b		-	1	2	3
WRITE(*,"(I5.4)")	b	-	0	1	2	3
WRITE(*,"(I5.5)")	b	*	*	*	*	*
WRITE(*,"(I5.2)")	c	*	*	*	*	*

Abbildung 1: Beispiele für das I Format[outExampleI.jpg]

F FORMAT

```
F w.d FLOAT
```

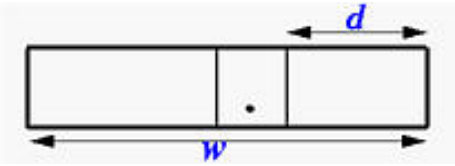

Abbildung 2: F Format bei der Ausgabe[outF.jpg]

Reelle Zahl rechtsbündig in w Spalten.
Mit d Stellen hinter dem Dezimalpunkt.
$w > d + 2$ Da Platz für Vorzeichen, Dezimalpunkt und mindestens eine Ziffer vor dem Dezimalpunkt sein muß.

REAL :: a = 123.345, b = -123.345

1	WRITE(*,"(F10.0)") a							1	2	3	.
2	WRITE(*,"(F10.1)") a						1	2	3	.	3
3	WRITE(*,"(F10.2)") a					1	2	3	.	3	5
4	WRITE(*,"(F10.3)") a				1	2	3	.	3	4	5
5	WRITE(*,"(F10.4)") a			1	2	3	.	3	4	5	0
6	WRITE(*,"(F10.5)") a		1	2	3	.	3	4	5	0	0
7	WRITE(*,"(F10.6)") a	1	2	3	.	3	4	5	0	0	0
8	WRITE(*,"(F10.7)") a	*	*	*	*	*	*	*	*	*	*
9	WRITE(*,"(F10.4)") b		-	1	2	3	.	3	4	5	0
10	WRITE(*,"(F10.5)") b	-	1	2	3	.	3	4	5	0	0
11	WRITE(*,"(F10.6)") b	*	*	*	*	*	*	*	*	*	*
		1	2	3	4	5	6	7	8	9	10

Abbildung 3: Beispiele für das F Format[outExampleF.jpg]

E FORMAT

```
E w.d[Ee] FLOAT
```

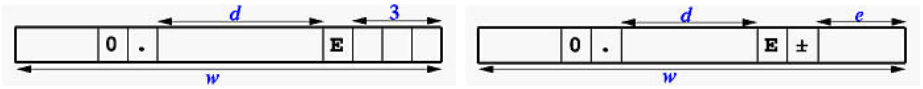

Abbildung 4: E Format bei der Ausgabe[outE.jpg, outESign.jpg]

Reelle Zahl rechtsbündig in w Spalten.
Mantisse mit d Stellen hinter dem Punkt.
Exponent zweistellig mit Vorzeichen bzw.
mit Angabe von E werden e Stellen für den Exponenten reserviert.
$w > d + 6$ Da Platz für Vorzeichen, eine Vorkommastelle (0),Dezimalpunkt, E, Vorzeichen des Exponenten und einen zweistelligen Exponenten reserviert werden muß.
bzw.
$w > d + e + 4$ Da der Exponent mit Vorzeichen und e Stellen dargestellt wird.

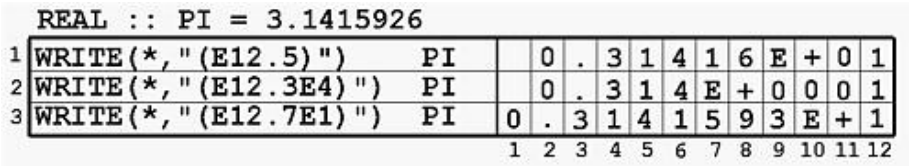

Abbildung 5: Beispiele für das E Format[outExampleE.jpg]

G FORMAT

```
F w.d FLOAT
```

Ausgabe in w Spalten mit d Stellen hinter dem Dezimalpunkt.
Ausgabe wie F oder E Format je nach Größe der Zahl. Zahlen im Intervall $[0.1, 10^d)$ werden im F Format dargestellt. Alle anderen im E Format.

A FORMAT

```
A[w]   CHARACTER
```

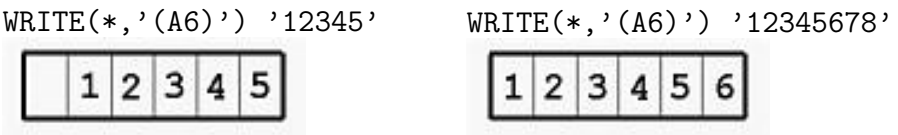

Abbildung 6: A Format bei der Ausgabe[outA6Less.jpg, outA6More.jpg]

Zeichenkette in w Spalten. Fehlt die Angabe von w so bestimmt die Länge der Variablen die Spaltenanzahl.

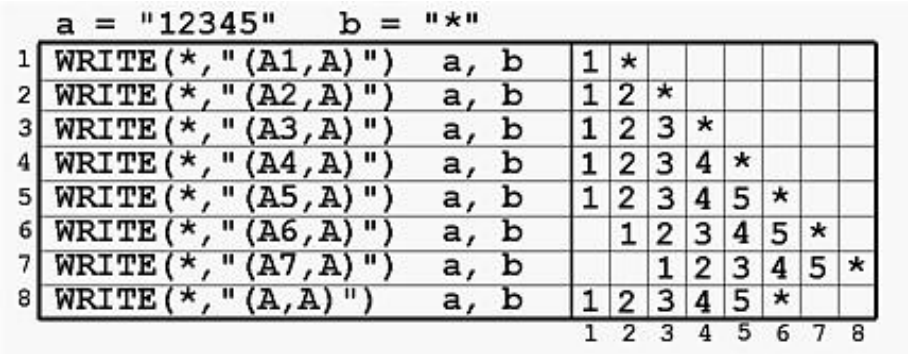

Abbildung 7: Beispiele für das A Format[outExampleA.jpg]

L FORMAT

```
L w    LOGICAL
```

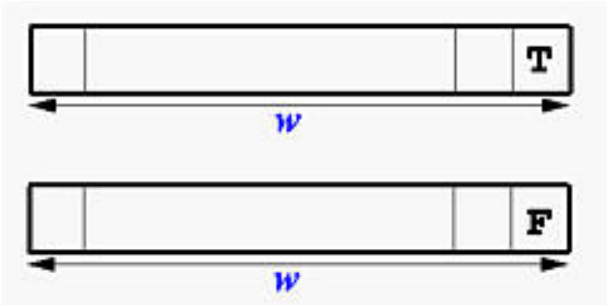

Abbildung 8: L Format bei der Ausgabe[outL.jpg]

Ausgabe in w Spalten mit T für .TRUE. und F für .FALSE.

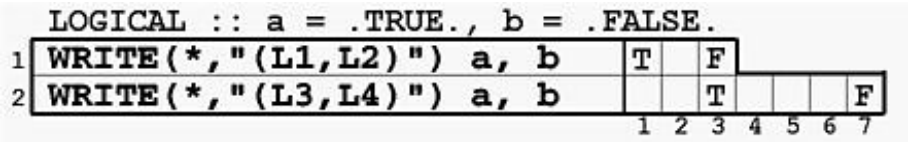

Abbildung 9: Beispiele für das L Format[outExampleL.jpg]

1.4 Steuerzeichen bei der Ausgabe

`w X`	Es werden w Leerzeichen ausgegeben.
`'text'`	text wird ausgeschrieben.
`/`	Ende der Ausgabezeile, Beginn einer neuen Zeile.
`\`	Die folgende Eingabe beginnt in der selben Zeile.
`SP`	positive und negative Zahlen erhalten Vorzeichen.
`SS`	nur negative Zahlen erhalten Vorzeichen.
`:`	Ausgabe beendet, wenn Liste abgearbeitet.

Ohne : Steuerzeichen

```
      INTEGER A(4), i
      DATA    A/1,2,3,4/

 1000 FORMAT(4(I4,','))
      WRITE(*,1000) (A(i),i=1,4)
```

ergibt als Ausgabe:

```
   1,   2,   3,   4,
```

Mit : Steuerzeichen

```
      INTEGER A(4), i
      DATA    A/1,2,3,4/

 1000 FORMAT(4(I4,:,','))
      WRITE(*,1000) (A(i),i=1,4)
```

ergibt als Ausgabe:

```
   1,   2,   3,   4
```

Abbildung 10: Ausgabe mit/ ohne : Steuerzeichen

1.5 Wiederholung von Formaten

Innerhalb einer Formatanweisung können Klammern gesetzt werden. Diese dienen dazu die Formatwiederholung auf eine Formatgruppe zu begrenzen. Es kann nur eine Gruppe festgelegt werden. Falls mehrere Klammern gesetzt werden gilt die äußerste.
Werden weniger Formate angegeben als Daten zu lesen/ schreiben sind, wird die Formatanweisung (Klammerung) wiederholt. Sind mehr Formate vorhanden, als Daten zu lesen/ schreiben sind, so werden nachfolgende Formate ignoriert.

Es ist möglich vor jede Formatgruppe einen Wiederholungsfaktor zu schreiben, der angibt, wie oft diese Gruppe wiederholt werden soll.

```
write(*,'(3I4.2)') 3, -5, 128
```

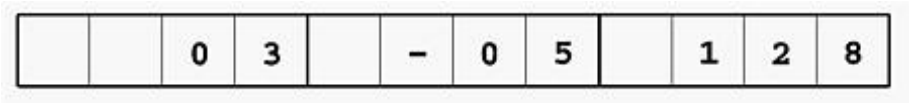

Abbildung 11: Wiederholungsformat 1[outRepetition3i4.jpg]

```
write(*,'(3F6.2)') 12.34, -0.95, 100.00
```

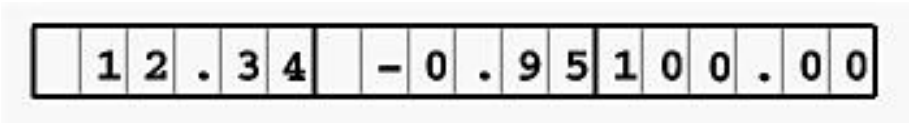

Abbildung 12: Wiederholungsformat 2[outRepetition3f62.jpg]

2 Implizite Schleife

Syntaktisch wird eine Implizite Schleife wie eine DO Schleife verwendet. Begrenzt wird die Schleife durch die umschließenden runden Klammern.
Eine Implizite Schleife und eine umschließende DO Schleife sind NICHT IDENTISCH! Dies wird an den beiden folgenden Beispielen deutlich.

DO Schleife

```
INTEGER A(3), j
DATA    A/1, 2, 3/

DO j=1, 3
         WRITE(*,*) A(j)
END DO
```

ergibt als Ausgabe:

```
1
2
3
```

Implizite Schleife

```
INTEGER A(3), j
DATA    A/1, 2, 3/

WRITE(*,*) (A(j), j=1, 3)
```

ergibt als Ausgabe:

```
1 2 3
```

Abbildung 13: Implizite Schleife vs. DO

3 Öffnen und Schließen von Kanälen

Kanäle werden in Fortran zur Ein- und Ausgabe verwendet, dabei ist es egal welches Gerät hinter der Ein-/ Ausgabe steht. Die Ansteuerung ist somit

geräteunabhängig.
Die Befehle READ und WRITE beziehen sich auf Kanäle, wobei der (*) die Standard Ein-/ Ausgabe des Systems darstellt.

3.1 Öffnen eines Kanals

```
OPEN(UNIT,FILE,STATUS,ACCESS,FORM,RECL,IOSTAT,ERR,BLANK)
```

UNIT = 10
Datei/ Gerät wird unter der Kanalnummer 10 geöffnet.

FILE = 'datei'
Dateiname ist Betriebsystemabhängig.

STATUS =

'OLD'	Öffnen einer bereits bestehenden Datei. Existiert die Datei nicht, kommt es zu einem Fehler.
'NEW'	Erzeugen einer neuen Datei. Existiert bereits eine Datei gleichen Namens, kommt es zu einem Fehler.
'SCRATCH'	Arbeitsdatei, die beim Schließen (CLOSE) wieder gelöscht wird.
'UNKNOWN' *default*	Wird der Status vom Betriebsystem bestimmt.

ACCESS =

'SEQUENTIAL' *default*	Fortlaufender Zugriff.
'DIRECT'	Direkter Zugriff (random access)

FORM =

'FORMATTED' *default when SEQUENTIAL*	Daten werden mit FORMAT umgewandelt.
'UNFORMATTED' *default when DIRECT*	Daten werden binär (wie im Speicher) übertragen.

RECL = 4
Nur bei DIRECT Zugriff. Die Datensatzlänge beträgt 4 Byte.

IOSTAT = i
Errorcode wird bei Fehler in i gespeichert.

ERR = 9000
Bei Fehler springe zu Label 9000.

BLANK =

'ZERO' *default*	Leerzeichen als 0 lesen.
'NULL'	Leerzeichen ignorieren.

Das Folgende Beispiel eröffnet unter der Nummer 10 eine alte Datei, mit dem Namen 'Daten.dat'. Der Zugriff ist sequentiell und die Übertragung ist formatiert.

```
c234567
      OPEN( 10, FILE='Daten.dat', STATUS='OLD',
     .ACCESS='SEQUENTIAL', FORM='FORMATTED')
```

3.2 Schließen eines Kanals

```
CLOSE(UNIT,STATUS,IOSTAT,ERR)
```

UNIT = 10

Schließt den Kanal mit der Kanalnummer 10. Die Nummer kann anschließend einer anderen Datei zugeordnet werden.

STATUS =

'KEEP' *default*	Datei bleibt erhalten außer wenn es sich um eine SCRATCH Datei handelt, dann darf KEEP nicht angegeben werden.
'DELETE' *default when SCRATCH*	Datei wird gelöscht.

IOSTAT = i

Errorcode wird bei Fehler in i gespeichert.

ERR = 9000

Bei Fehler springe zu Label 9000.

3.3 Informationen über Dateien/ Kanäle auslesen

Dafür gibt es die INQUIRE Anweisung, die allerdings so mächtig ist, das wir die Parameterlisten hier nicht näher erläutern möchten. Sie ist allerdings in /RRZN Fortran 77/ gut beschrieben.

3.4 Lesen und Schreiben

```
READ (UNIT, FMT, REC, END, IOSTAT, ERR)
WRITE(UNIT, FMT, REC,      IOSTAT, ERR)
```

UNIT

Die Kanalnummer zum Lesen/ Schreiben.

FMT
Die Formatanweisung die bei formatierter Ein-/ Ausgabe verwendet werden soll. Darf nur bei FORM = FORMATTED angegeben werden.

REC
Die Datensatznummer des Datensatzes, der gelesen/ geschrieben werden soll. Nur erlaubt bei ACCESS = DIRECT.

END
Label an das gesprungen werden soll, wenn Dateiende erreicht. Bei DIRECT ACCESS wird diese Angabe ignoriert.

IOSTAT = i
Errorcode wird bei Fehler in i gespeichert.

ERR = 9000
Bei Fehler springe zu Label 9000.

3.5 Besondere Befehle bei squentiellen Dateien

```
REWIND(UNIT,IOSTAT,ERR)
```

UNIT = 10
Setzt die Datei (10) auf den Anfang zurück.

IOSTAT = i
Errorcode wird bei Fehler in i gespeichert.

ERR = 9000
Bei Fehler springe zu Label 9000.

```
BACKSPACE(UNIT,IOSTAT,ERR)
```

UNIT = 10
Setzt die Datei (10) um einen Satz zurück, so daß der zuletzt verarbeitete Satz erneut zur Verfügung steht.
Es kommt zu keinem Fehler wenn die Datei am Anfang steht. Der Befehl wird dann ignoriert.

IOSTAT = i
Errorcode wird bei Fehler in i gespeichert.

ERR = 9000
Bei Fehler springe zu Label 9000.

```
ENDFILE(UNIT,IOSTAT,ERR)
```

UNIT = 10
Setzt eine Endmarke an die aktuelle Position der Datei (10).

IOSTAT = i
Errorcode wird bei Fehler in i gespeichert.

ERR = 9000
Bei Fehler springe zu Label 9000.