

IO und Formatangaben

Sonderübung

Luise Zieger & Lea Happel

PROG Sonderübung

January 22, 2019

1. Wiederholung
2. Attribute zum Einlesen/Schreiben
 - Attribute beim Einlesen
 - Attribute beim Schreiben
3. Übungsaufgaben
4. Formatangaben

Wiederholung

Öffnen von Dateien

```
INTEGER:: ioopen
```

```
OPEN (UNIT=31, IOSTAT=ioopen, FILE='Beispiel.dat', &  
      & STATUS='new', ACTION='write')
```

```
IF (ioopen /= 0) THEN
```

```
    WRITE(*,*) 'Fehler beim Öffnen der Datei'
```

```
ELSE
```

```
    ...
```

```
END IF
```

- STATUS: OLD/ NEW/ REPLACE/ SCRATCH/ **UNKNOWN**
- ACTION: READ/ WRITE/ **READWRITE**
- IOSTAT-Variable zur Fehlerbehandlung (siehe Beispiel)

Wiederholung

Schließen von Dateien

```
CLOSE([UNIT=]u [,IOSTAT=iostat] [,ERR=err][,FILE=file][,STATUS=status])
```

Meistens verwenden wir bei CLOSE nur die Unitnummer und keine Attribute:

```
CLOSE(UNIT=31)
```

```
CLOSE(31)
```

Wiederholung

Lesen und Schreiben in Dateien

Bisher: `READ(UNIT=*,FMT=*)` bzw. `WRITE(UNIT=*,FMT=*)`

* steht für Standard, d.h. beim Lesen die Tastatur, beim Schreiben der Bildschirm

Um in eine Datei zu schreiben/ aus einer Datei zu lesen, wird das * bei Unit durch die entsprechende Unitnummer ersetzt.

Wenn nicht anders spezifiziert, geschieht lesen/schreiben zeilenweise!

```
WRITE(31,*) 'Hallo'  
WRITE(31,*) 'Welt!'
```

Attribute beim Einlesen

```
READ([UNIT=]u,[FMT=]f [,IOSTAT=iostat] [,ERR=err][,ADVANCE=ad])
```

- IOSTAT/ ERR werden meist dazu genutzt, das Dateiende zu identifizieren (daher: man liest in einer Schleife ein, bis es einen Einlesefehler gibt)
- bei IOSTAT bedeutet: iostat=0 erfolgreich eingelesen, iostat < 0 Datensatz zu Ende, iostat > 0 anderer Fehler
- mit ADVANCE ('NO'/'YES') lässt sich regeln, ob nach dem Einlesen die Zeile gewechselt werden soll ('YES') oder nicht ('NO')
- **Bei der Verwendung von ADVANCE='NO' müssen Formatangaben verwendet werden!**
- Default-Einstellung ist ADVANCE='YES'

Attribute beim Einlesen

Beispiel

```
INTEGER:: iopen, ioread, i
INTEGER, DIMENSION(20) :: m

OPEN(UNIT=31, IOSTAT=iopen, FILE='Beispiel.dat', &
     & STATUS='old', ACTION='read')
IF (iopen /= 0) THEN
    WRITE(*,*) 'Fehler beim Öffnen der Datei'
ELSE
    DO i = 1, 20
        READ(31,*, IOSTAT=ioread) m
        IF (ioread /= 0) EXIT
    END DO
END IF
```

Attribute beim Schreiben

```
WRITE([UNIT=]u,[FMT=]f [,IOSTAT=iostat] [,ERR=err][,ADVANCE=ad])
```

- mit ADVANCE ('NO'/'YES') lässt sich regeln, ob nach dem Schreiben die Zeile gewechselt werden soll ('YES') oder nicht ('NO')
- **Bei der Verwendung von ADVANCE='NO' müssen Formatangaben verwendet werden!**
- Default-Einstellung ist ADVANCE='YES'

Beispiel

```
INTEGER:: iopen,ioread,i

OPEN(UNIT=31,IOSTAT=iopen, FILE='Beispiel.dat', STATUS='new', ACTION='write')
IF( iopen == 0) THEN
    DO i=1,100
        WRITE (31,*) i
    END DO
END IF
```


Vergleich von Funktionsergebnissen

f_n bezeichne die n -te Fibonaccizahl. Der Goldene Schnitt ergibt sich als Grenzwert von $\lim_{n \rightarrow \infty} \frac{f_{n+1}}{f_n}$.

Berechne eine Näherung des Goldenen Schnitts, indem du die ersten hundert Quotienten in einer Schleife berechnest.

Speichere dabei jedes Zwischenergebniss, d.h. jeden der hundert Quotienten, in einer Datei ab.

Vergleiche deine Ergebnisse mit denen deines Nachbarn, um ihre Richtigkeit zu überprüfen. Schreibe dazu eine Funktion, die zuerst beide Dateien öffnet und dann in einer Schleife jeweils eine Zahl aus jeder Datei einliest und diese vergleicht. Stimmen die Zahlen immer überein, so soll die Funktion wahr zurückgeben. Stimmen die Zahlen nicht überein oder ist eine Datei länger als die andere, so soll die Funktion falsch zurückgeben.

Formatangaben

Wir wollen unsere Ausgaben hübsch formatieren.

```
WRITE(UNIT=*,FMT=*)
```

Dazu ersetzen wir das zweite * durch einen Format**string** (" nicht vergessen):

```
TYPE(student) :: studi  
REAL :: diameter
```

```
WRITE (UNIT = *, FMT = '(A,X,A,X,I2,X,A)') TRIM(studi%vorname), &  
& 'ist', student%alter, 'Jahre alt.'
```

```
WRITE(*,'(A,X,F6.2,X,A)') 'Der Durchmesser beträgt', diameter, 'cm.'
```

Formatangaben

```
INTEGER :: length2
```

```
REAL :: length1
```

```
CHARACTER(15) :: formatstring
```

```
formatstring = '(A,X,F5.3)'
```

```
WRITE(*,FMT=TRIM(formatstring)) 'Länge in km:', length1
```

```
formatstring(6:9) = 'l5 _ _'
```

```
WRITE(*,FMT=TRIM(formatstring)) 'Länge in m:', length2
```

Formatangaben

Wollt ihr zum Beispiel mehrere Ausgaben in einer Zeile vornehmen, benutzt ihr `ADVANCE = 'NO'`. Dazu sind Formatangaben nötig.

Beispiel

```
TYPE(student), DIMENSION(10) :: studis

DO i = 1, 10
    WRITE(*,'(A, A, I2, A)', ADVANCE = 'NO') studis(i)%vorname, &
        & '(', studis(i)%alter, '), '
END DO

WRITE(*,*)
```

Tanzwettbewerb

Bei einem Tanzwettbewerb soll eine fortlaufende Punktetabelle erstellt werden.

Erstelle einen ADT "Teilnehmer", bestehend aus einem Integer für die Startnummer und einem Character (Len=30) für den Namen. Lies dann zuerst die Anzahl der teilnehmenden Paare aus "Tanzpaare.txt" ein, allokiere entsprechende Vektoren und lies dann die Jungen und die Mädchen in jeweils einen Vektor ein. Die Tanzpaare sind in der Form "Startnummer_Paar Name_Frau Name_Mann" notiert.

Öffne dann eine neue Datei und schreibe in die erste Zeile die Namen der Mädchen und darunter die Namen ihrer Tanzpartner. Achte darauf, dass diese jeweils rechtsbündig untereinander stehen. Lies dann in einer Schleife die Ergebnisse der jeweiligen Runde von der Tastatur ein und schreibe diese ungefähr mittig (bastelt ein bisschen rum, so dass es hübsch aussieht) unter die Namen des jeweiligen Tanzpaares. Die Eingabe einer negativen Zahl soll das Programm beenden.

ZUSATZ: Ermittle, welches Paar den 1./2./3. Platz belegt hat und schreibe dies in die letzte Zeile der Datei.

Beispiel

Frau A	Frau B	Frau C
Herr A	Herr B	Herr C
1	2	3