Bemerkungen zu den Übergabeparametern – 2

- FORTRAN gestattet auch, Funktionen als Übergabeparameter zu behandeln, sofern sie mit

  ```
  EXTERNAL <Funktionsname>
  ```

  im rufenden Programm deklariert wurden.

- auch intrinsische Funktionen (z.B. `SIN(X)`) können übergeben werden, wenn sie mit

  ```
  INTRINSIC <Funktionsname>
  ```

  im rufenden Programm deklariert wurden. Allerdings dürfen die Typkonversionsfunktionen (z.B. `INT`), die Maximums– und Minimumsfunktion sowie die lexikalischen Funktionen (z.B. `LGE`) nicht als Parameter übergeben werden.

- (User–definierte Funktionsnamen haben stets Vorrang vor den intrinsischen.)

- Beispiel:

```
      PROGRAM ZEROS
      EXTERNAL F1
        ⋮
      CALL BISEC(A,B,F1)
        ⋮
      END
C*************************************
      SUBROUTINE BISEC(X,Y,F)
      DOUBLE PRECISION X,Y,F,Y1,Y2
        ⋮
      Y1 = F(X)
      Y2 = F(Y)
        ⋮
      END
C*************************************
      DOUBLE PRECISION FUNCTION F1(Z)
        ⋮
      F1 = ...
        ⋮
      END
```

Bemerkungen zu den Übergabeparametern – 3

Problem: Übergabe eines Arrays als Parameter an eine Subroutine ⇒ Wie erhält man eine konsistente Dimensionierung des Feldes?

- **Dimension mitübergeben:**

```
      PROGRAM PROG
      DIMENSION A(100)
        ⋮
      N = 100
      CALL SUB1(A,N)
        ⋮
C*************************************
      SUBROUTINE SUB1(X,M)
      DIMENSION X(M)
        ⋮
      END
```

- **Dimension "kennen":**

```
      SUBROUTINE SUB1(X)
      DIMENSION X(100)
        ⋮
      END
```

- **"Feld mit übernommener Länge":**

```
SUBROUTINE SUB1(X)
DIMENSION X(*)
  ⋮
END
```

- **COMMON–Block bzw. INCLUDE–File:**

```
SUBROUTINE SUB1(X)
COMMON /PARA/ N
DIMENSION X(N)
  ⋮
END
```

Arbeiten mit Files

- **Öffnen von Dateien:**

```
OPEN(<Parameterliste>)
```

  - Kanalnummer: `UNIT=u`
  - Filename: `FILE='<Filename>'`
  - Fehlerbehandlung: `ERR=<Labelnummer>`
  - Rekordlänge: `RECL=<Rekordlänge>`
  - Status: `STATUS=t`
    - 'OLD' existiert schon
    - 'NEW' neu anlegen
    - 'UNKNOWN' (Default)
    - 'SCRATCH' wird nach dem Schließen des Kanals gelöscht
  - `FORM=a`
    - 'FORMATTED' (Default) ASCII–File
    - 'UNFORMATTED' Binärformat
  - `ACCESS=a`
    - 'DIRECT'
    - 'SEQUENTIAL' (Default)

```
OPEN(20,FILE='out.dat',STATUS='OLD',ERR=40)
```

- **Schließen von Dateien:**

```
CLOSE(<Kanalnummer>)
```

- **Datenzeiger zum Fileanfang:**

```
REWIND(<Kanalnummer>)
```

- **Datenzeiger um einen Eintrag zurücksetzen:**

```
BACKSPACE(<Kanalnummer>)
```

- **Eigenschaften einer Datei bestimmen:**

```
INQUIRE(FILE=<Filename>,<Parameterliste>)
```
oder
```
INQUIRE(UNIT=<Kanalnummer>,<Parameterliste>)
```

  - Beispiel:
    ```
    INQUIRE(FILE='out.dat',FORM=format)
    ```
    `format` ist dann entweder 'FORMATTED' oder 'UNFORMATTED'.

Weiteres zu READ und WRITE

- **Unformatierte Ein- und Ausgabe:**

```
READ(u) <I/O-Liste>
WRITE(u) <I/O-Liste>
```

  - Daten werden in maschineninterner Form (*binary*) geschrieben bzw. gelesen.
  - **Vorteil:**
    - kompakt, d.h. weniger Speicherplatz
    - schnelles Einlesen
  - **Nachteil:**
    - nicht direkt lesbar (Bildschirm)
    - nicht kompatibel zu anderen Architekturen
  - $\Rightarrow$ Anwendung: große Datenmengen
  - Bei jedem READ oder WRITE wird einen Record weitergegangen.

- **Listengesteuerte Übertragung:**

```
READ(u,<Steuerliste>) <I/O-Liste>
WRITE(u,<Steuerliste>) <I/O-Liste>
```

– Kanalnummer: `UNIT=u`
(0 $\leq u \leq$ 99, u kann INTEGER–Variable sein)

– Label der Formatanweisung: `FMT=<Label>`
(oder String mit FORMAT–Anweisung, z.B. `FMT='(a)'`)

– Zugriff auf bestimmten Datensatz r: `REC=r`
(File muß hierzu mit `ACCESS='DIRECT'` geöffnet sein.)

– Error–Parameter: `ERR=<Label>`

– Input/Output–Status: `IOSTAT=i`
(i gibt maschinenabhängig Auskunft über die aufgetretene Fehlerart; $i = 0$: kein Fehler).

```
      READ(12,1000,IOSTAT=IP,ERR=200)
        ⋮
200    WRITE(3,'(I5)') IP
```

– Dateiende: `END=<Label>`
(Programm wird bei angegebenem Label fortgesetzt, wenn *End–of–File* gelesen wurde. ⇒ Nützlich bei Lesen eines Files unbekannter Länge!

weitere FORTRAN–Statements...

- **Direkte Ausgabe auf Standard–Output:**

  ```
  PRINT <Formatlabel> <Ausgabeliste>
  ```

  Beispiel:

  ```
  PRINT 10,A,B,C
  PRINT*,A,B,C
  ```

- Unterbrechung des Programms mit Fortsetzungsmöglichkeit: `PAUSE <Text>`
- Zuordnung desselben Speicherplatzes zu verschiedenen Variablennamen: `EQUIVALENCE(x,y)`
  (nützlich bei großen Hilfsfeldern unterschiedlicher Dimensionierung ⇒ Speicherplatzersparnis!)
- Alternativer "Einstiegspunkt"
  in eine SUBROUTINE:
  `ENTRY <Name> (Variablenliste)`

Interne Dateien

- keine Datei im üblichen Sinne, sondern ein sequentieller File mit festgelegter maximaler Länge = CHARACTER–Variable!
- CHARACTER–Variable wird wie ein File beschrieben bzw. gelesen.
- erlaubt bequeme Übertragung von numerischen Ergebnissen in Strings (inklusive Rundung!) (⇒ Übergabe an Graphikprogramme)
- erlaubt bequeme Erstellung von fortlaufenden Dateinamen (z.B. out.001, out.002 . . .)
- Beispiel:

```
      ⋮
      CHARACTER*30 OUTTXT
      ⋮
      A = 17.85
      WRITE(OUTTXT,10) A
10    FORMAT(1X,'a=',F5.1)
      PRINT*,OUTTXT
      ⋮
```

ergibt:

```
a=17.9
```