

# ARRAYS in Unterprogrammen

## Sonderübung

Luisse Zieger & Lea Happel

PROG Sonderübung

January 8, 2019

1. Wiederholung
2. Felder in Unterprogrammen
  - Felder als UP-Parameter
  - Felder als Funktionsergebnis
  - Automatische Felder
3. Übungsaufgabe

# Wiederholung: ARRAYS

- homogene Datenstruktur (alle Elemente haben denselben Typ)
- ein- oder mehrdimensional (Vektor, Matrix, Tensor, ...)
- entweder **statisch** oder **dynamisch**

```
<Datentyp>, DIMENSION( $a_1 : e_1, a_2 : e_2, \dots, a_N : e_N$ ) :: Array1
```

```
Feldeintrag=Array1( $i_1, i_2, \dots, i_n$ )
```

! Zugriff auf einzelne Felder

```
Feldeintrag=Array1( $i_1 : j_1, i_2 : j_2, \dots, i_n : j_n$ )
```

! Zugriff auf Teilfelder

- In Fortran **beginnen Arrays (per default) immer mit 1!**
- Ein 1-dimensionales Array (entspricht Vektor) kann man mit dem Feldkonstruktor vorbelegen

```
INTEGER, DIMENSION(1:3):: A
```

```
A=(/1,2,3/)
```

# Wiederholung: Dynamische ARRAY-Variablen

```
< Elementtyp >, DIMENSION ( : , : , ... , : ), ALLOCATABLE :: my_array
```

## Beispiel

```
INTEGER, DIMENSION (:), ALLOCATABLE :: dyn_vec
```

```
INTEGER, DIMENSION ( : , : ), ALLOCATABLE :: dyn_mat, A, B
```

! Feld allokalieren, d. h. Speicherplatz beschaffen

```
ALLOCATE (dyn_mat(4, 4), dyn_vec(0 : 6))
```

! Feld deallokieren, d. h. Speicherplatz wieder freigeben

```
DEALLOCATE (dyn_mat, dyn_vec)
```

# Wiederholung

## Standardfunktionen:

- SIZE, SUM, PRODUCT
- MINVAL, MAXVAL, MINLOC, MAXLOC
- für 2-dim. Felder: TRANSPOSE
- MATMUL, DOT\_PRODUCT
- für LOGICAL-Felder: ANY, ALL
- ...
- alle intrinsischen Operatoren (+, -, \*, /, ...) und die meisten intrinsischen Funktionen (z. B. ABS) auf Felder anwendbar (sofern Elementtyp stimmt)
- dabei wird eintragsweise gerechnet
- ACHTUNG: Operanden (bei binären Operatoren) müssen gestaltkonform sein
- Skalare sind mit bel. Feldern gestaltkonform

z. B.  $3 + v$ ,  $v + w$ ,  $v * w$ ,  $v == w$ ,  $ANY(v == w)$ , ...

# Felder in Unterprogrammen

## Felder als UP-Parameter

- statisches Feld oder
- Feld übernommener Gestalt: Gestalt zum Aufrufzeitpunkt des UPs durch a. A. festgelegt

```
FUNCTION mult (A, v)
  INTEGER, DIMENSION ( : , : ) :: A
  INTEGER; DIMENSION (0 : ) :: v
  ...
```

# Felder in Unterprogrammen

## Felder als Funktionsergebnis

Indexgrenzen müssen zum Aufrufzeitpunkt des UPs berechenbar sein. Möglich: Ausdrücke, die von d. UP-Parametern abhängen, z. B.

```
FUNCTION mult (A, v)
  INTEGER, DIMENSION ( : , : ) :: A
  INTEGER, DIMENSION ( 0 : ) :: v
  INTEGER, DIMENSION ( 1 : SIZE (A, 1)) :: mult
```

# Felder in Unterprogrammen

## Automatische Felder

- lokale Variablen des UPs
- Indexgrenzen müssen zum Aufrufzeitpunkt des UPs berechenbar sein, können von d. UP-Parametern abhängen

```
FUNCTION mult (A, v)
  INTEGER, DIMENSION ( : , : ) :: A
  INTEGER, DIMENSION ( 0 : ) :: v
  INTEGER, DIMENSION ( 1 : SIZE (A, 1)) :: mult
  INTEGER, DIMENSION ( SIZE (A, 2), SIZE (A, 1) ) :: T
  ...
  T = TRANSPOSE (A)
  ...
```



# Übungsaufgabe

Schreibe ein Hauptprogramm, das

- die Dimension der Matrix einliest
- eine quadratische Matrix dieser Dimension allokiert

Schreibe ein Modul mit

- einer Subroutine zum zeilenweisen Einlesen einer Matrix übernommener Gestalt
- einer Funktion zur Berechnung der Spur (Summe der Diagonaleinträge) einer Matrix
- testet, ob die Matrix orthogonal ist ( $AA^T = I$ )
- den Positivteil der Matrix zurückgibt