

# Grundlegende Struktur von Algorithmen

## Sonderübung

Dana Liebscher

PR10

05.11.2024

## 1. Wiederholung

## 2. Kontrollstrukturen

- Verzweigungen
  - IF-THEN-ELSE Konstrukt
  - Übungsaufgabe
  - SELECT CASE - Anweisung
  - Übungsaufgabe
- Schleifen
  - DO loops
  - DO WHILE - Schleife
  - Zählschleifen
  - Übungsaufgabe

# Was haben wir in der letzten Übung besprochen?

- Grundlegende Struktur eines FORTRAN Programms
- Was ist eine Variable?
- Datentypen: INTEGER, REAL, CHARACTER, LOGICAL
- Attribute von Variablen

# Kontrollstrukturen

Kontrollstrukturen dienen der Steuerung von Programmabläufen.

Manchmal wollen wir Anweisungen ...

- **wiederholt** ausführen oder
- nur unter einer bestimmten **Bedingung** ausführen.

Kontrollstrukturen können mittels Flussdiagrammen oder Struktogrammen visualisiert werden.

# Verzweigungen

## BLOCK-IF:

## IF-THEN:

```
IF (Bedingung) THEN  
    Anweisungsblock  
END IF
```

## IF-THEN-ELSE:

```
IF (Bedingung) THEN  
    Anweisungsblock  
ELSE  
    Anweisungsblock  
END IF
```

# Verzweigungen

## BLOCK-IF:

### Mehrfache Verzweigung:

```
IF (Bedingung) THEN
    Anweisungsblock
ELSE IF (weitere Bedingung) THEN
    Anweisungsblock
[ ...
ELSE IF (weitere Bedingung) THEN
    Anweisungsblock
... ]
ELSE
    Anweisungsblock
END IF
```

# Verzweigungen

## Beispiel

```
IF (note <= 0 .OR. note > 6) THEN  
    WRITE(*,*) 'Da ging etwas schief!'  
ELSE IF (1 <= note .AND. note <= 2) THEN  
    WRITE(*,*) 'Krasser Typ!'  
ELSE IF (3 <= note .AND. note <= 4) THEN  
    WRITE(*,*) 'Reicht!'  
ELSE  
    WRITE(*,*) 'Das war wohl nichts.'  
END IF
```

# Verzweigungen

## Logisches IF

Verkürzte Verzweigung mit nur **einer** Anweisung:

```
IF (Bedingung) Anweisung
```

Hierbei muss die Anweisung noch in der selben Zeile stehen!

## Beispiel

```
IF (note <= 0 .OR. note > 6) EXIT
```



# Verzweigungen

## Aufgabe

Wir möchten unseren Taschenrechner erweitern. Zu Beginn des Programms ist neben den zwei Zahlen auch ein Character zur Auswahl der Rechenoperation einzulesen. In Abhängigkeit dessen Wertes werden die Zahlen entweder addiert, subtrahiert, multipliziert oder dividiert. Division durch 0 muss ausgeschlossen werden.

Führt euer Programm anschließend auf dem Terminal aus und testet dessen Funktion mit ausgewählten Eingabewerten.

# Verzweigungen

## SELECT CASE:

```
SELECT CASE (note)
  CASE (1:2)
    WRITE(*,*) 'Krasser Typ!'
  CASE (3:4)
    WRITE(*,*) 'Reicht!'
  CASE (5:6)
    WRITE(*,*) 'Das war wohl nichts.'
  CASE DEFAULT
    WRITE(*,*) 'Da ging etwas schief!'
END SELECT
```

SELECT CASE kann stets durch ein BLOCK-IF ersetzt werden, aber nicht andersherum.

CASE DEFAULT sollte angegeben werden, um Laufzeitfehler zu verhindern.

# Verzweigungen

## Aufgabe

Programmiere deinen Taschenrechner zur Übung mit `SELECT CASE` statt mit `IF`-Verzweigungen.

# Schleifen (DO loops)

## Klassische Schleife

```
DO
    Anweisungsblock
END DO
```

Bei jedem Schleifendurchlauf muss überprüft werden, ob die Abbruchbedingung erfüllt ist und die Schleife sodann verlassen wird.

## Beispiel

```
READ(*,*) a
DO
    IF (a < 11) EXIT
    a = a / 2
END DO
```

# Schleifen (DO loops)

## DO WHILE - Schleife

```
DO WHILE (Bedingung)  
    Anweisungsblock  
END DO
```

## Beispiel

```
READ(*,*) a  
DO WHILE (a > 11)  
    a = a / 2  
END DO
```

## Infinite Loop

```
READ(*,*) a  
DO WHILE (a / = 1) ! Infinite Loop außer bei 2er Potenzen  
    a = a / 2  
END DO
```

! Achtung: Bedingungen sollten Sinn machen und irgendwann zum Abbruch der Schleife führen!

# Schleifen (DO loops)

## Zählschleifen

```
DO k = a, e [, i]
```

a - Anfang

e - Ende

i - Inkrement (Schrittweite)

```
DO k = a, e [, i]  
    Anweisungsblock  
END DO
```

Zählvariable k darf im Innern der Zählschleife nicht verändert werden.  
Zustand der Zählvariable k nach Beendigung der Schleife **undefiniert**.

# Schleifen (DO loops)

Der Endwert  $e$  muss nicht exakt erreicht werden, es reicht, wenn er überschritten wird.

$a$  und  $e$  werden vor Beginn der Schleife einmalig berechnet und dann nicht mehr verändert.

## Beispiel

```
DO k = 1, 9, 2  
    WRITE(*,*) k    ! Ausgabe aller ungeraden Zahlen zwischen 1 und 9  
END DO
```

# Schleifen (DO loops)

## Aufgabe

- Wir möchten nun wiederholt mit unserem Taschenrechner rechnen. Nach jeder ausgeführten Rechnung soll eine erneute Eingabe erfolgen. Überlegt euch eine geeignete Abbruchbedingung.
- Schreibt ein Programm, das alle Tripel dreier natürlicher Zahlen findet, deren Summe 50 ist.

**ZUSATZ:** Überlegt euch eine Möglichkeit, wie jede Zahlenkombination nur genau einmal ausgegeben wird (d. h. die Reihenfolge spielt keine Rolle).

TIPP: Nutzt Zählschleifen!

Zählt die Anzahl der Tripel und gebt diese anschließend auf dem Terminal aus.

- **ZUSATZ:** Schreibt ein Programm, das alle vierstelligen Zahlen ausgibt, die man aus den Ziffern 1,2,3,4 bilden kann. (Nutzt Zählschleifen!)