

# Abstrakte Datentypen

## Sonderübung

Nadine Schärmann & Thomas Klütz

PR10

05. Januar 2021

## 1. ADT - Abstrakte Datentypen

- Allgemeine Syntax
- Beispiel
- Nutzen von ADT
- Komponentenzugriff
- Beispiel
- Übungsaufgabe
- Typkonstruktor

# Abstrakte Datentypen (auch: benutzerdefinierte Typen)

- definiert heterogene Datenstruktur
- heterogen bedeutet: Komponenten können unterschiedliche Typen haben

## Allgemeine Syntax

```
TYPE < Typname >  
    [PRIVATE]
```

```
    < Typvereinbarung für Komponenten >
```

```
END TYPE
```

# Abstrakte Datentypen (auch: benutzerdefinierte Typen)

## Beispiel

```
TYPE fraction  
    INTEGER :: nominator  
    INTEGER :: denominator  
END TYPE
```

Die Komponententypen können verschieden sein:

```
TYPE bike  
    CHARACTER(15) :: brand  
    INTEGER :: size  
    ...  
END TYPE
```

# Nutzen von ADT

- möglichst viel von der inneren Struktur verstecken
- Informationen, die zusammen gehören, aber unterschiedliche Datentypen haben, an einem Ort speichern (Zum Beispiel wird jeder Student über Name, s-Nummer und Matrikelnummer gekennzeichnet. Diese Infos sollten daher zusammen gespeichert werden.)
- erstellen von Datentypen, die in der Praxis gebraucht werden, aber nicht in F95 vorhanden sind (z.B. Rationale Zahlen, Intervallarithmetik, ...)

# Komponentenzugriff

**Problem:** Es gibt fast keine intrinsischen (d.h. vordefinierte) Funktionen dazu  
→ Alle gewünschten Funktionalitäten müsst ihr selbst programmieren!

Dazu benötigt ihr den **Komponentenzugriff**:

- die einzelnen Komponenten besitzen intrinsische Datentypen (d.h. Real, Integer, ...) und für diese gibt es intrinsische Funktionen  
→ ihr könnt damit arbeiten
- Allgemeine Syntax: **<NameDerVariablen> % <NameDerKomponenten>**

# Beispiel für Komponentenzugriff

## Beispiel

```
TYPE Student
  INTEGER :: Alter
  CHARACTER(15) :: Vorname, Name
END TYPE
```

...

```
FUNCTION aelter (a,b)
  TYPE(Student) :: a,b
  LOGICAL :: aelter

  aelter = a%Alter > b%Alter
END FUNCTION
```

# Übungsaufgabe

Wir möchten ein Teilnehmerverzeichnis für unsere Sonderübung anlegen. Dabei sollen zu jedem Teilnehmer folgende Angaben gespeichert werden:

- Vorname (max. 30 Zeichen),
- Name (max. 30 Zeichen),
- Alter,
- Semester

Programmiere für diesen ADT folgende Funktionalitäten:

- eine Subroutine PUT die folgenden Satz ausgibt: " *Max Muster*, 20 Jahre, ist Student im 1. Semester."
- eine Funktion, die die Studiendauer zweier Studenten vergleicht
- eine Funktion, die Nach- und Vornamen zweier Studenten lexikalisch vergleicht



# Typkonstruktor

Um im Programmablauf Variablen eures ADTs zu definieren, ohne sie einlesen zu müssen, eine selbstgeschriebene Funktion zu verwenden oder sie komponentenweise belegen zu müssen, könnt ihr den Typkonstruktor verwenden.

Allgemeine Syntax: `<NameDesADT>(<Komponente1>,<Komponente2>,...)`

## Beispiel

```
TYPE Student
  INTEGER :: Alter
  CHARACTER(8) :: Vorname, Name
END TYPE
```

```
TYPE(Student) :: MaxMuster
```

```
MaxMuster = Student(20,'Max','Muster')
```