Informatik I

Reiner Hüchting

DHBW Mannheim

27. Februar 2023

Record-Datentypen

Record-Datentypen

Vor- und Nachteile von Arrays

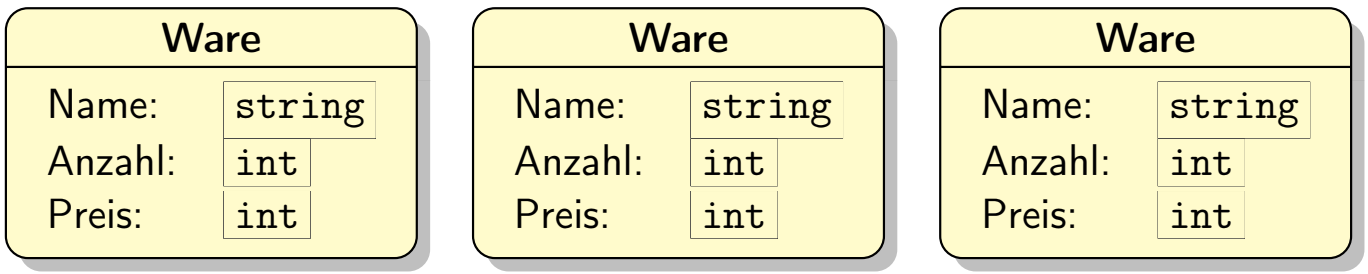
- Arrays sind geeignet, um Listen variabler Länge zu repräsentieren.
- Einschränkung: Alle Elemente haben den gleichen Datentyp.

Record-Datentypen

Vor- und Nachteile von Arrays

- Arrays sind geeignet, um Listen variabler Länge zu repräsentieren.
- Einschränkung: Alle Elemente haben den gleichen Datentyp.
- In der Praxis oft umgekehrte Anforderung: Feste Länge, aber Elemente haben verschiedene Datentypen (Records).

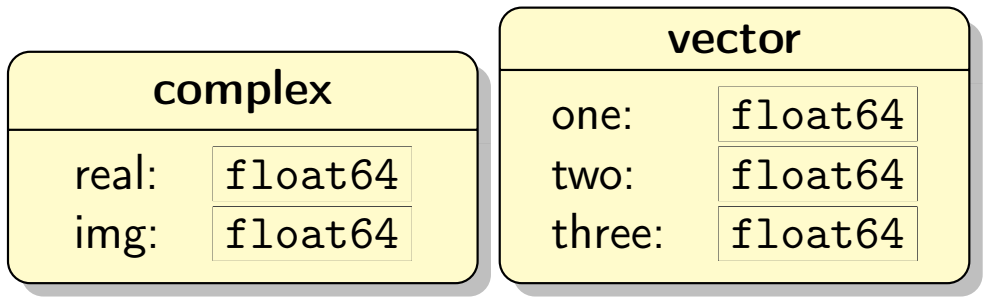
Beispiel: Datenstrukturen für eine Lagerverwaltung:



Record-Datentypen

Anwendungen von Record-Datentypen

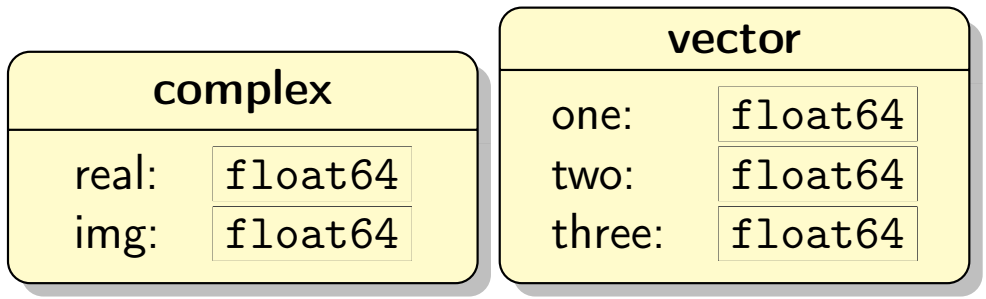
- zusammengesetzte Zahlen:



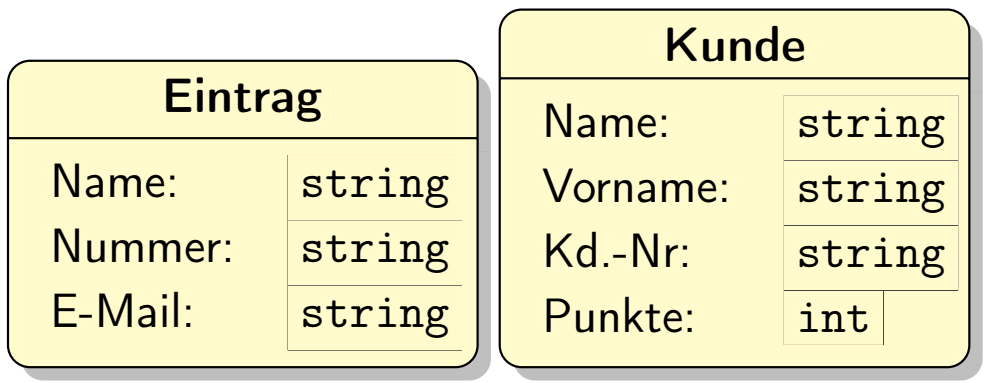
Record-Datentypen

Anwendungen von Record-Datentypen

- zusammengesetzte Zahlen:



- Einträge in Listen (z.B. Telefonbuch, Kundenkartei):



Record-Datentypen

Ein Record-Datentyp wird auch Struktur genannt:

Record-Datentypen

Ein Record-Datentyp wird auch Struktur genannt:

- Definition mit den Schlüsselwörtern `type` und `struct`.

```
1  type Kunde struct {
2      name string
3      vorname string
4      kdnr string
5      punkte int
6  }
```

Record-Datentypen

Ein Record-Datentyp wird auch Struktur genannt:

- Definition mit den Schlüsselwörtern `type` und `struct`.
- Zugriff auf Felder mit dem Punkt-Operator.

```
1  func main() {
2     var k1 Kunde
3     k1.name = "Mustermann"
4     k1.vorname = "Max"
5     k1.kdnr = "123a"
6     k1.punkte = 42
7
8     fmt.Println(k1)
9  }
```

Record-Datentypen

Weitere Möglichkeiten der Initialisierung:

```
1  k2 := Kunde{
2     "Beispiel",
3     "Barbara",
4     "456b",
5     77,
6  }
```

Record-Datentypen

Weitere Möglichkeiten der Initialisierung:

```
1  k3 := Kunde{
2      vorname: "Monika",
3      name: "Musterfrau",
4      punkte: 38,
5      kdnr: "135c",
6  }
```