# V. Felder und Strukturen

## 1. Felder

`array_ini.cc, array_suche.cc`

— zusammenfassen von Daten gleichen Typs
— Zuweisungsoperator "=" nicht definiert
— Operator "==" ist definiert, führt aber nicht den komponentenweisen

Vergleich zweier Felder durch (sondern vergleicht zwei Zeiger)

## 2. `struct`

`struct_geburt.cc, struct_vektor.cc`

— zusammenfassen von Daten unterschiedlichen Typs

— ⟨`Name der Strukturvariablen`⟩`.`⟨`Komponente`⟩

**3. `union`:** wie `struct`, jedoch wird nur ein Speicherplatz definiert — `union_wert.cc`

**4. `enum`:** aussagekräftige Namen für Integerwerte — `enum_bsp.cc`

**5. `typedef`:** Definition eigener Typen — `typedef_1.cc, typedef_2.cc`

# VI. Funktionen

● Teilaufgaben, die sich wiederholen ➪ Funktion ● strukturiert programmieren
● modular programmieren ● "freie Funktionen" und "Methoden"

## 1. Freie Funktionen

1.1 Deklaration, Prototyp: Datentyp des Funktionswerts und der Parameter
1.2 Definition: `<Rückgabetyp><Ftk.-Name>(<Parameterliste mit Typ>){Anweisungen}`

(i) Parameterliste: Typ und Name müssen vorhanden sein

(ii) übergebene Parameter = lokale Datenobjekte in Funktion

(iii) Im Anweisungsblock: Def. von Variablen, geschachtelte Blöcke, ...

(iv) Dekl. und Def. können in einem Schritt erfolgen

(v) Falls sep. Dekl. $\longrightarrow$ Def. kann an bel. Stelle im Programm stehen

(vi) `return`-Anweisung führt zum Verlassen des Programms