

# Systemnahe Programmierung in C

## 14 Verbundtypen

**J. Kleinöder, D. Lohmann, V. Sieh, P. Wägemann**

Lehrstuhl für Informatik 4  
Systemsoftware

Friedrich-Alexander-Universität  
Erlangen-Nürnberg (FAU)

Sommersemester 2026

<http://sys.cs.fau.de/lehre/ss26>



# Strukturen: Motivation

- Gehören Variable „irgendwie“ zusammen,
  - wäre es auch besser diese **zusammenzufassen**
  - „problembezogene Abstraktionen“
  - „Trennung der Belange“
- Dies geht in C mit **Verbundtypen** (Strukturen)

↪ 4-1

↪ 12-4

```
// Structure declaration
struct Student {
    char   lastname[64];
    char   firstname[64];
    long   matnum;
    int    passed;
};

// Variable definition
struct Student stud;

// Pointer definition
struct Student *pstud;
```

Ein **Strukturtyp** fasst eine Menge von Daten zu einem gemeinsamen Typ zusammen. Die Datenelemente werden **hintereinander** im Speicher abgelegt.



# Strukturen: Motivation

- Gehören Variable „irgendwie“ zusammen,
  - wäre es auch besser diese **zusammenzufassen**
  - „problembezogene Abstraktionen“
  - „Trennung der Belange“
- Dies geht in C mit **Verbundtypen** (Strukturen)

↪ 4-1

↪ 12-4

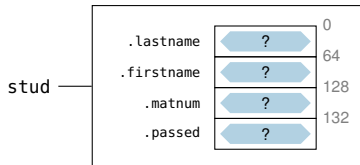
```
// Structure declaration
struct Student {
    char   lastname[64];
    char   firstname[64];
    long   matnum;
    int    passed;
};

// Variable definition
struct Student stud;

// Pointer definition
struct Student *pstud;
```

Ein **Strukturtyp** fasst eine Menge von Daten zu einem gemeinsamen Typ zusammen.

Die Datenelemente werden **hintereinander** im Speicher abgelegt.



# Strukturen: Variablendefinition und -initialisierung

- Analog zu einem Array kann eine Strukturvariable bei Definition elementweise initialisiert werden

↔ 13-8

```
struct Student {  
    char    lastname[64];  
    char    firstname[64];  
    long    matnum;  
    int     passed;  
};
```

```
struct Student stud = { "Meier", "Hans",  
                        4711, 0 };
```

Die Initialisierer werden nur über ihre Reihenfolge, nicht über ihren Bezeichner zugewiesen.  
↪ **Potentielle Fehlerquelle** bei Änderungen!



# Strukturen: Variablendefinition und -initialisierung

- Analog zu einem Array kann eine Strukturvariable bei Definition elementweise initialisiert werden

↔ 13-8

```
struct Student {  
    char   lastname[64];  
    char   firstname[64];  
    long   matnum;  
    int    passed;  
};
```

```
struct Student stud = { "Meier", "Hans",  
                        4711, 0 };
```

Die Initialisierer werden nur über ihre Reihenfolge, nicht über ihren Bezeichner zugewiesen.  
↪ **Potentielle Fehlerquelle** bei Änderungen!

- Analog zur Definition von **enum**-Typen kann man mit **typedef** die Verwendung vereinfachen

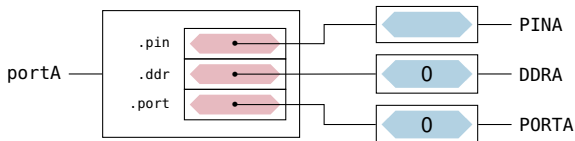
↔ 6-8

```
typedef struct {  
    volatile uint8_t *pin;  
    volatile uint8_t *ddr;  
    volatile uint8_t *port;  
} port_t;
```

```
port_t portA = { &PINA, &DDRA, &PORTA };  
port_t portD = { &PIND, &DDRD, &PORTD };
```



# Strukturen: Elementzugriff



- Auf Strukturelemente wird mit dem `.`-Operator zugegriffen [≈Java]

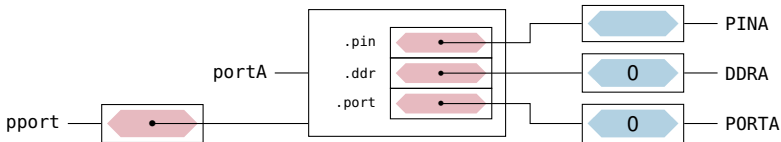
```
port_t portA = { &PINA, &DDRA, &PORTA };
```

```
*portA.port = 0; // clear all pins  
*portA.ddr = 0xff; // set all to output
```

**Beachte:** `.` hat eine höhere Priorität als `*`



# Strukturen: Elementzugriff

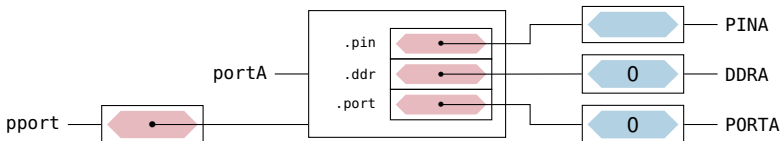


- Bei einem Zeiger auf eine Struktur würde Klammerung benötigt

```
port_t *pport = &portA; // p --> portA  
  
>(*pport).port = 0;      // clear all pins  
(*pport).ddr = 0xff;    // set all to output
```



# Strukturen: Elementzugriff



- Bei einem Zeiger auf eine Struktur würde Klammerung benötigt

```
port_t *pport = &portA; // p --> portA  
  
*(*pport).port = 0;      // clear all pins  
*(*pport).ddr = 0xff;   // set all to output
```

- Mit dem `->`-Operator lässt sich dies vereinfachen  $s \rightarrow m \equiv (*s).m$

```
port_t *pport = &portA; // p --> portA  
  
*pport->port = 0;        // clear all pins  
*pport->ddr = 0xff;     // set all to output
```

`->` hat **ebenfalls** eine höhere Priorität als `*`



# Strukturen als Funktionsparameter

- Im Gegensatz zu Arrays werden Strukturen *by-value* übergeben

```
void initPort(port_t p) {  
    *p.port = 0;           // clear all pins  
    *p.ddd = 0xff;        // set all to output  
  
    p.port = &PORTD;     // no effect, p is local variable  
}  
  
void main(void) { initPort(portA); ... }
```



# Strukturen als Funktionsparameter

- Im Gegensatz zu Arrays werden Strukturen *by-value* übergeben

```
void initPort(port_t p) {
    *p.port = 0;           // clear all pins
    *p.ddd = 0xff;        // set all to output

    p.port = &PORTD;     // no effect, p is local variable
}

void main(void) { initPort(portA); ... }
```

- Bei größeren Strukturen wird das **sehr ineffizient**
  - Z. B. Student ( $\leftrightarrow$  14-1): Jedes mal 134 Byte allozieren und kopieren
  - Besser man übergibt einen Zeiger auf eine konstante Struktur

```
void initPort(const port_t *p){
    *p->port = 0;         // clear all pins
    *p->ddd = 0xff;       // set all to output

    // p->port = &PORTD;  compile-time error, *p is const!
}

void main(void) { initPort(&portA); ... }
```



- Strukturelemente können auf Bit-Granularität festgelegt werden
  - Der Compiler fasst Bitfelder zu passenden Ganzzahltypen zusammen
  - Nützlich, um auf einzelne Bit-Bereiche eines Registers zuzugreifen
- Beispiel
  - EICRA

**External Interrupt Control Register A**  
Steuert Auslöser für externe Interrupt-Quellen  
INT0 und INT1. [1]



```
typedef struct {  
    uint8_t ISC0      : 2;    // bit 0-1: interrupt sense control INT0  
    uint8_t ISC1      : 2;    // bit 2-3: interrupt sense control INT1  
    uint8_t reserved : 4;    // bit 4-7: reserved for future use  
} EICRA_t;
```



- [1] *ATmega328PB 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash*. Atmel Corporation. Okt. 2015. URL: [https://sys.cs.fau.de/extern/lehre/ss25/spic/uebung/spicboard/Atmel-42397-8-bit-AVR-Microcontroller-ATmega328PB\\_Datasheet.pdf](https://sys.cs.fau.de/extern/lehre/ss25/spic/uebung/spicboard/Atmel-42397-8-bit-AVR-Microcontroller-ATmega328PB_Datasheet.pdf).

