

# Systemnahe Programmierung in C

## 10 Variablen

**J. Kleinöder, D. Lohmann, V. Sieh**

Lehrstuhl für Informatik 4  
Systemsoftware

Friedrich-Alexander-Universität  
Erlangen-Nürnberg

Sommersemester 2025

<http://sys.cs.fau.de/lehre/ss25>



- **Variable** := Behälter für Werte (↪ Speicherplatz)

- Syntax (Variablendefinition):

$SK_{opt} Typ_{opt} Bez_1 [= Ausdr_1]_{opt} [, Bez_2 [= Ausdr_2]_{opt} , \dots]_{opt};$

- $SK_{opt}$  Speicherklasse der Variable, [≈Java]  
**auto**, **static**, oder leer
- $Typ$  Typ der Variable, [=Java]  
**int** falls kein Typ angegeben wird [≠Java]  
(↪ schlechter Stil!)
- $Bez_i$  Name der Variable [=Java]
- $Ausdr_i$  Ausdruck für die initiale Wertzuweisung;  
wird kein Wert zugewiesen so ist der Inhalt  
von nicht-**static**-Variablen **undefiniert** [≠Java]



- Variablen können an verschiedenen Positionen definiert werden
  - Global                      außerhalb von Funktionen,  
   üblicherweise am Kopf der Datei
  - Lokal                        zu Beginn eines { Blocks },                      C89  
   direkt nach der öffnenden Klammer
  - Lokal                        überall dort, wo eine Anweisung stehen darf                      C99

```
int a = 0;           // a: global
int b = 47;         // b: global

void main(void) {
    int a = b;       // a: local to function, covers global a
    printf("%d", a);
    int c = 11;      // c: local to function (C99 only!)
    for (int i=0; i<c; i++) { // i: local to for-block (C99 only!)
        int a = i;   // a: local to for-block,
                    //   covers function-local a
    }
}
```



- Variablen können an verschiedenen Positionen definiert werden
  - Global                      außerhalb von Funktionen,  
   üblicherweise am Kopf der Datei
  - Lokal                        zu Beginn eines { Blocks },                      C89  
   direkt nach der öffnenden Klammer
  - Lokal                        überall dort, wo eine Anweisung stehen darf      C99

```
int a = 0;           // a: global
int b = 47;         // b: global

void main(void) {
    int a = b;       // a: local to function, covers global a
    printf("%d", a);
    int c = 11;      // c: local to function (C99 only!)
    for (int i=0; i<c; i++) { // i: local to for-block (C99 only!)
        int a = i;   // a: local to for-block,
                    // covers function-local a
    }
}
```

Mit globalen Variablen beschäftigen wir uns noch näher im Zusammenhang mit **Modularisierung** ↔ 12-5

