Systemnahe Programmierung in C (SPiC)

10 Variablen

Jürgen Kleinöder, Daniel Lohmann, Volkmar Sieh

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Sommersemester 2020

http://www4.cs.fau.de/Lehre/SS20/V_SPIC

Variablendefinition [≈Java]

- **Variable** := Behälter für Werte (↦ Speicherplatz)
- Syntax (Variablendefinition):

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

SK_{opt}	Speicherklasse der Variable, `auto`, `static`, oder leer	[≈Java]
Typ	Typ der Variable, `int` falls kein Typ angegeben wird (↦ schlechter Stil!)	[=Java] [≠Java]
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]

Variablendefinition (Forts.)

- Variablen können an verschiedenen Positionen definiert werden
  - Global — außerhalb von Funktionen, üblicherweise am Kopf der Datei
  - Lokal — zu Beginn eines `{ Blocks }`, direkt nach der öffnenden Klammer — C89
  - 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
}
```

Variablendefinition (Forts.)

- Variablen können an verschiedenen Positionen definiert werden
  - Global — außerhalb von Funktionen, üblicherweise am Kopf der Datei
  - Lokal — zu Beginn eines `{ Blocks }`, direkt nach der öffnenden Klammer — C89
  - 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** ↪ ??