Systemnahe Programmierung in C (SPiC)

90 „Guter" Code

Jürgen Kleinöder, Daniel Lohmann, Volkmar Sieh

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Sommersemester 2021

http://www4.cs.fau.de/Lehre/SS21/V_SPIC

Aufgabe

Schreiben Sie eine C-Funktion `sum`, die die Summe der ersten n ganzen Zahlen berechnet!

$$x = 0 + 1 + 2 + 3 + \ldots + (n-1) + n$$

Anfänger-Lösungversuch

```
int sum(
     int n) {int s,i;
  i=0; while(i <=(  n))
s=s + i++;
    return ((s)); }
```

Gruselig... ;-)

Unbedingt richtig einrücken!
So kann den Code auch kein guter Programmierer lesen!

1. Lösungversuch

Eine „typische" Java-Programmierer-Lösung könnte sein:

```
int sum(int n) {
    int s, i;
    for (i = 0; i <= n; i++)
        s = s + i;
    return s;
}
```

Gleicher Code in C/Java bis auf „*static*".

1. Lösungversuch

Testet man die Lösung für C, könnte für $n = 0, 1, 2, \ldots, 100$ der richtige Funktionswert herauskommen!

Wer so abgibt, bekommt nur einen Teil der Aufgabenpunkte!

Wichtig: Durch Testen kann man nur die *Anwesenheit*, nicht die *Abwesenheit* von Fehlern im Programm beweisen!

=> Nachdenken immer erforderlich!

1. Lösungversuch

Probleme:

- Compiler würde warnen: „*s* not initialized”.
- Funktion liefert falsches Ergebnis für sehr große *n*.
- Funktion liefert auf Ergebnis (0) für negative *n*.

=> 1. „Lösung” ist keine Lösung!

=> Warnungen vom Compiler *immer* aktivieren und beachten!
=> weniger testen, mehr denken!

2. Lösungversuch

```
int32_t sum(int16_t n) {
    if (n < 0) {
        ... /* error handling */
    }
    int32_t s = 0;
    int16_t i;
    for (i = 1; i <= n; i++)
        s = s + i;
    return s;
}
```

Besser:

- *s* ist initialisiert (=> Ergebnis ist immer das gleiche)
- durch `int16_t` und `int32_t` sind die Wertebereiche klarer
- durch `if (n < 0) ...` werden falsche Parameter abgefangen
- Summation ab '1'

Wer so abgibt, bekommt die vollen Aufgabenpunkte!

Weitere Verbesserungen

```
/** \brief Function returning the sum of the first 'n' integers. */
uint32_t sum(uint16_t n) {
    /* Clear sum 's'. */
    uint32_t s = 0;

    /* Add first 'n' integers to sum in 's'. */
    for (uint16_t i = 0; i <= n; i++) {
        s = s + i;
    }

    /* Return sum. */
    return s;
}
```

Verbesserungen:

- Kommentare, Einrückungen
- `uint16_t` und `uint32_t` vergrößern Wertebereich
- durch `uint16_t`-Parameter Fehlerabfrage überflüssig
- `uint16_t i` in `for`-Schleifenkopf
- Klammern `{`, `}` vermeiden ggf. Probleme bei Erweiterungen

Weitere Verbesserungen

```
/**
 * \brief
 * Function returning the sum of the first 'n' integers.
 *
 * \details
 * Sum of 1 + 2 + 3 + ... + n can be calculated by the idea:
 *   sum =   1   +   2   +   3   + ... +   n
 *   sum =   n   + (n-1) + (n-2) + ... +   1
 * 2*sum = (n+1) + (n+1) + (n+1) + ... + (n+1)
 * So sum = n * (n+1) / 2.
 */
uint32_t sum(uint16_t n) {
    uint32_t x = n; /* To avoid overflows below. */
    return x * (x + 1) / 2;
}
```

- für große n viel schneller!
- viel kürzerer Code!

Das sollte die Informatiker-Lösung sein... ;-)