

6. Übungsblatt

Ausgabe: 02.06.22**Abgabe:** 09.06.22 08:00

Die Lösungen bitte in der Vorlage `uebung-06.md` eintragen und diese in Eurem KSGM-Repository rechtzeitig committen und hochladen.

6.1 Binärprodukte

Im letzten Übungsblatt haben Sie ein Programm zur Multiplikation zweier Zahlen durch wiederholte Addition verifiziert. Ihr Kollege sagt, folgendes Program sei wesentlich effizienter:¹

```
/* Binary multiplication of n and m */
// {M = m ∧ N = n ∧ n > 0}
x = 0;
while (n != 0) {
    x = x + m * (n % 2);
    m = 2 * m;
    n = n / 2;
}
// (POST)
```

Als ad-hoc-Erweiterung unserer Sprache benutzt das Programm den Modulo-Operator (%) in C, der den Rest bei der Division berechnet. Die mathematische Notation dafür ist $x \bmod y$.

- (i) Geben Sie die Nachbedingung an.

POST = ?

- (ii) Finden Sie eine Invariante, und beweisen Sie die Korrektheit des Programmes.

Hinweise:

- Denkt daran, dass `/` die *ganzzahlige Division* (geschrieben $\div$) berechnet. Es ist im Allgemeinen *nicht* das Inverse zur Multiplikation (d.h. es gilt *nicht*, dass $a = b \cdot (a \div b)$). Aber es gilt:

$$a = b \cdot (a \div b) + a \bmod b \quad (1)$$

und in diesem Fall spezialisiert mit $b = 2$:

$$a = 2 \cdot (a \div 2) + a \bmod 2 \quad (2)$$

- Die Invariante ist nicht ganz einfach. Um eine Idee zu bekommen, schreibt euch für verschiedene konkrete Belegungen von n und m deren Werte während der Berechnung der Schleife jeweils am Anfang und am Ende auf. Wer trotzdem keine findet, kann die Veranstalter um Hilfe fragen.

¹Richtig schnell wird dieses Programm, wenn wir den Modulo-Operator und die Multiplikation und Division mit 2 durch entsprechende Operationen auf Bitebene (in C `&`, `>>`, `<<`) ersetzen, aber semantisch passiert da nichts, deshalb ist dieses Programm hier effizient genug.