

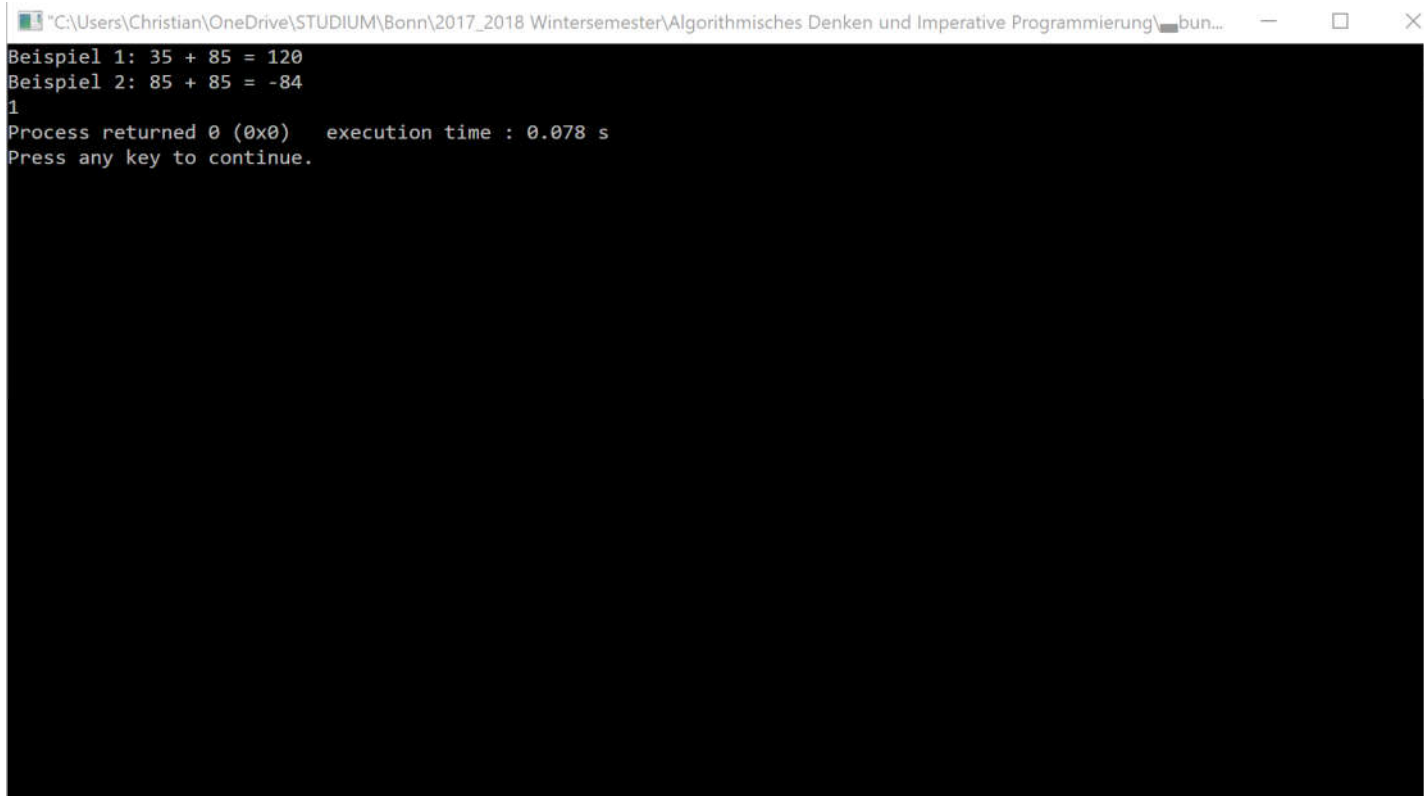
Blatt 2

Donnerstag, 2. November 2017 12:57

Aufgabe 1:

Um normale Ergebnisse zu erhalten deklarieren wir die Variablen als unsigned: "char x1,x2,result;"

Der unveränderte Code liefert jedoch:



```
"C:\Users\Christian\OneDrive\STUDIUM\Bonn\2017_2018 Wintersemester\Algorithmisches Denken und Imperative Programmierung\bun...
Beispiel 1: 35 + 85 = 120
Beispiel 2: 85 + 85 = -84
1
Process returned 0 (0x0) execution time : 0.078 s
Press any key to continue.
```

Erfasster Bildschirmausschnitt: 02.11.2017 13:01

Warum ist dies so?

Der normale ASCII-Code läuft von 0 - 127. Wir haben jedoch beim Typ char in C den "erweiterten ASCII-Code", der 256 Werte annehmen kann.

Ein Char hat in C die Länge ein Byte (= 8 Bit) und wird, wenn wir damit rechnen oder es ausgeben als vorzeichenbehaftete Ganzzahl behandelt (signed). Wir gehen dabei von einer Darstellung Interpretation im Zweierkomplement aus, wobei das ganz linke Bit bei einer 1 eine negative Zahl anzeigt. Beim Zweierkomplement wäre dies theoretisch so, dass das erste Bit gleich -128 (max. niedriger Wert beim Zweierkomplement mit 8 Bit) gezählt wird und alle Bits rechts davon dann aufaddiert werden. Sprich $10000000 = -128$ $11111111 = 11111111 - 1$.

Wenn wir jetzt zwei Chars z.B. Q = 81 und R = 82 miteinander addieren, erhalten wir einen Wert = 163. Dieser ist größer als 127. Also wird das erste Bit im fürs Char reserviertem Speicher benutzt und auf 1 gesetzt. (Das ist schlichtweg die Binärdarstellung von einer Zahl größer 127.)

Da für C char eine vorzeichenbehaftete Ganzzahl ist, gibt er, wenn man sich das Resultat aus Zahl ausgeben lässt, natürlich die Interpretation im Zweierkomplement aus, welche dann negativ ist.

Dieses Verhalten lässt sich vermeiden, indem man das Ergebnis in eine int Var. Schreibt oder "unsigned char" statt normaler char verwendet.

