Programmieren 1 Übung: Einfache Funktionen

Klaus Kusche

1.) Das kleinste gemeinsame Vielfache

Das kleinste gemeinsame Vielfache (kgV, englisch lcm = least common multiple) zweier ganzer Zahlen *a* und *b* ist die kleinste Zahl, die sowohl ein ganzzahliges Vielfaches von *a* als auch ein ganzzahliges Vielfaches von *b* ist.

Man braucht das kgV, wenn man Brüche auf einen gemeinsamen Nenner bringt.

Beispiele:
kgV(1, 11) = 11 kgV(9, 3) = 9 kgV(6, 9) = 18 kgV(7, 5) = 35 kgV(4, 4) = 4 kgV(12, 15) = 60

kgV(*a*, *b*) wird normalerweise als (*a* * *b*) / ggT(*a*, *b*) berechnet
(ggT…größter gemeinsamer Teiler).

Es gibt allerdings eine (sehr langsame!) "Idiotenmethode",
die ohne ggT, Multiplikation oder Division auskommt:

- Du brauchst zwei Variablen "Vielfaches von a" und "Vielfaches von b".
- Ist *a* oder *b* *gleich 0*, so ist das Ergebnis 0.
- Ist *a* oder *b* *negativ*, so entferne das Vorzeichen.
- "Vielfaches von a" wird am Anfang auf *a* gesetzt, "Vielfaches von b" auf *b*.
- Dann *wiederholt* man Folgendes,
  solange "Vielfaches von a" und "Vielfaches von b" voneinander *verschieden* sind:
  - Ist "Vielfaches von a" kleiner als "Vielfaches von b",
    so wird *a* einmal zu "Vielfaches von a" dazugezählt,
    sonst wird *b* einmal zu "Vielfaches von b" dazugezählt.
- Wenn beide gleich sind, hat man das kgV gefunden.

Schreib eine Funktion, die das kgV zweier Zahlen nach dieser Methode berechnet
und zurückliefert.

Schreib dazu ein **`main`**, das zwei Zahlen von der Befehlszeile einliest,
die kgV-Funktion damit aufruft, und deren Ergebnis ausgibt.

2.) Rechnen mit Uhrzeiten

Gesucht ist ein Programm, das *Uhrzeiten addieren oder subtrahieren* kann.

Das Programm wird mit *zwei Zeiten* auf der Befehlszeile gestartet.
Die Zeiten (gegeben als Stunde, Minute, Sekunde) werden als *höchstens sechsstellige ganze Zahlen* im Format "*hhmmss*" angegeben, eventuell mit einem *Minus* davor.

Du sollst sie nicht zeichenweise verarbeiten, sondern gleich mit **`atoi`** in eine Zahl verwandeln. *Ausgeben* soll das Programm (schön formatiert als "*hh*`:`*mm*`:`*ss*"!)
die beiden *eingegebenen Zeiten* sowie deren *Summe*, also z.B.

```
2:33:10 + 0:41:55 = 3:15:05
```

Das Addieren / Subtrahieren / Multiplizieren / Dividieren von Zeiten im Stunden/Minuten/Sekunden-Format ist ziemlich umständlich. Löse die Aufgabe daher so, wie es in fast allen Programmen und Betriebssystemen gemacht wird:

- Alle Zeitangaben werden gleich bei der Eingabe *in Sekunden* (in anderen Programmen auch Mikrosekunden o.ä.) *umgerechnet* und als ganz normaler **`int`** gespeichert.
- Intern gerechnet (in unserem Fall: Addiert) wird *nur mit Sekunden*: Mit diesen kann man ohne besondere Vorkehrungen wie mit jedem normalen **`int`** rechnen.
- Erst bei der Ausgabe werden die Sekunden-Zahlen wieder in die *hh*`:`*mm*`:`*ss*-Darstellung zerlegt.

Hinweise:

- Du sollst dazu *zwei Funktionen schreiben* und in deinem **`main`** aufrufen:
  - Eine Funktion wird mit einer ganzen, max. *sechsstelligen Zahl* aufgerufen (die *Sekunden* als Einer und Zehner, *Minuten* als Hunderter und Tausender, sowie *Stunden* als Zehn- und Hunderttausender enthält), und liefert als Ergebnis eine Zahl (**`int`**), die dieselbe Zeit *nur als Sekunden* angibt.

    Extrahiere dazu mittels **`/`** und **`%`** die Stunden, die Minuten und die Sekunden aus der ursprünglichen Zahl und berechne dann aus den drei getrennten Werten die Sekunden.

    Wenn du es richtig machst, sollte es auch für *negative Eingabezahlen* funktionieren (und eine *negative Zahl von Sekunden* liefern):
    Die Division rundet in C bei negativen Zahlen Richtung 0,
    und der Rest hat dasselbe Vorzeichen wie die zu dividierende Zahl.
  - Eine zweite Funktion wird mit einer Zahl aufgerufen, die die Anzahl der *Sekunden* angibt. Zurück kommt nichts, aber die Funktion soll die Zeit *schön formatiert ausgeben*:
    - Wenn die Zeit *negativ* ist, wird zuerst einmal ein *Minus* ausgegeben (rechne dann mit der positiven Zeit weiter!).
    - Wenn die Zeit *größergleich 24 Stunden* ist (und nur dann!), werden die *Tage* ausgegeben (“*d*`:`”) und von der Zeit abgezogen.
    - Dann wird die *Zeit* im Format “*hh*`:`*mm*`:`*ss*” ausgegeben (berechne die einzelnen Teile mit **`%`** und **`/`** aus den Sekunden!).

    Tipp: “**`%02d`**” beim **`printf`** bewirkt, dass ein **`int`** mit 2 Stellen und führender 0 ausgegeben wird, mit **`cout <<`** ist das eher mühsam.

Zusatzaufgabe:

- Die Funktion zur Verwandlung der Eingabe in Sekunden soll auch *prüfen*, ob die Eingabe *nicht mehr als sechsstellig* ist und ob die eingegebenen Stunden max. 23 und die eingegebenen Minuten und Sekunden kleiner 60 sind.
  Wenn nicht, soll das Programm mit einer Fehlermeldung enden.

  Achtung auf *negative Zeiten*, du musst die *Absolutwerte* prüfen!