

AIK Programmieren 1 Übung: Strukturen

Klaus Kusche

Bruchrechnen

Wir wollen die Aufgabe „einfacher Taschenrechner“ aus Übung 5 nochmals lösen, aber diesmal soll das Programm mit Brüchen statt mit **double**-Zahlen rechnen.

Wir brauchen also einen Struktur-Typ, der zwei int's enthält (Zähler und Nenner).
Mach auch ein **typedef** für diesen Struktur-Typ, damit du weniger schreiben musst!

Damit das Programm schön übersichtlich bleibt, wollen wir es in 7 Funktionen aufteilen:
Je eine Funktion zum Addieren, Subtrahieren, Multiplizieren und Dividieren zweier Brüche (Potenzieren lassen wir weg), eine Funktion, die einen String in einen Bruch verwandelt (zum Einlesen der Brüche von der Befehlszeile), eine Funktion zum Ausgeben eines Bruches, und unser **main**:

- Die 4 Rechen-Funktionen haben jeweils 2 Brüche (linker und rechter Operand) als Parameter und liefern einen neuen Bruch als Returnwert zurück.
Da Bruch-Strukturen ohnehin schön klein sind, können wir sie einfach „by Value“ übergeben und zurückgeben (keine Pointer).

Intern sehen die 4 Funktionen alle gleich aus: Eine Bruch-Struktur-Variable für das Ergebnis deklarieren, ihren Zähler und Nenner mit den richtigen Werten füllen, und die gefüllte Variable als Returnwert zurückgeben.

Zur Auffrischung der Mathematik:

$$z1/n1 + z2/n2 = (z1*n2 + z2*n1)/(n1*n2)$$

$$z1/n1 - z2/n2 = (z1*n2 - z2*n1)/(n1*n2)$$

$$z1/n1 * z2/n2 = (z1*z2)/(n1*n2)$$

$$z1/n1 / z2/n2 = (z1*n2)/(n1*z2)$$

- Die Ausgabe-Funktion hat einen Bruch als Parameter (wieder „by Value“) und keinen Returnwert. Wenn der Nenner des Bruches ungleich 1 ist, wird der ganze Bruch ausgegeben (zähler/nenner), sonst eine ganze Zahl (nur der Zähler).
- Die Eingabe-Funktion hat einen String (der nicht geändert wird) als Parameter und liefert einen neuen Bruch (wieder „by Value“) als Returnwert: Wie bei den Rechen-Funktionen wird eine Bruch-Variable angelegt, befüllt und zurückgegeben.

Der Eingabe-String kann entweder einen Bruch oder nur eine ganze Zahl enthalten.
Die beiden Fälle werden wie folgt unterschieden und behandelt:

- Suche im String das Zeichen '/' (wie heißt die Stringfunktion, die ein Zeichen in einem String sucht, und welchen Header braucht man dafür?) und merke dir die Fundstelle.
- Wenn kein '/' gefunden wird, wird die Eingabe als Zahl behandelt:
Der Zähler des Ergebnisses ist der in einen **int** umgewandelte String (**atoi**), der Nenner ist 1.

- Wird ein '/' gefunden, so ist der Zähler wie bisher das **atoi** des Strings (das **atoi** hört mit dem Umwandeln automatisch vor dem '/' auf).
Der Nenner ist das **atoi** des Strings ab einem Zeichen hinter der Fundstelle des '/' (wenn du den gemerkten Pointer auf die Fundstelle hast, wie rufst du **atoi** mit dem String ab dem Zeichen dahinter auf?).
- Das **main** hat im Wesentlichen dieselbe Struktur wie in der Lösung von Übung 5, du kannst die damalige Musterlösung als Ausgangsbasis nehmen.
 - Aus den beiden **double**-Variablen werden Bruch-Struktur-Variablen.
 - Der Fall für das Potenzieren (und damit auch **math.h**) fällt weg, und die Prüfung auf Division durch 0 vorläufig auch.
 - Statt dem **atoi** wird unsere Eingabe-Funktion aufgerufen und statt dem **printf** unsere Ausgabe-Funktion.
 - In den 4 Rechnungen werden statt + - * / unsere 4 Rechen-Funktionen aufgerufen.

Tip: Da ein auf der Befehlszeile eingegebener '*' nicht funktioniert, verwende in der Eingabe stattdessen wieder ein 'x' für die Multiplikation.

Zusatzaufgabe:

Unser Programm erkennt bisher keine Brüche mit 0 im Nenner (bzw. keine Divisionen durch 0) und liefert ungekürzte Brüche mit unnötig großem Zähler und Nenner (und eventuell mit negativem Nenner, was auch nicht schön ist). Wir wollen daher eine Funktion Kuerzen schreiben, die einen frisch berechneten Bruch prüft und kürzt.

Die Funktion soll aber nicht wie unsere bisherigen Funktionen einen Bruch „by Value“ übergeben bekommen und einen neuen, gekürzten Bruch als Returnwert zurückgeben, sondern sie soll direkt den Bruch im Aufrufer ändern. Sie bekommt also einen Pointer auf die Struktur des zu kürzenden Bruch übergeben und hat keinen Returnwert.

Aufgerufen wird die Funktion an 5 Stellen, nämlich in den 4 Rechen-Funktionen und in der Eingabe-Funktion, und zwar jeweils unmittelbar vor dem **return**.
Wie übergibst du der Funktion einen Pointer auf die zu kürzende Ergebnis-Variable?

Wir brauchen zuerst einmal eine Hilfsfunktion, die den ggT berechnet: Zwei **int**'s gehen hinein, ein **int** soll zurückkommen. Wie man den ggT zweier Zahlen ausrechnet, findest du z.B. in Übung 3 (Verfahren von Euklid: **while**-Schleife mit %-Rechnung).
Vorsichtshalber solltest du von beiden Parametern den Absolutbetrag nehmen, bevor du mit der **while**-Schleife beginnst (mit negativen Zahlen funktioniert sie nämlich nicht).

Die Funktion **Kuerzen** besteht aus 3 Schritten:

- Ist der Nenner 0, so soll eine Fehlermeldung ausgegeben und das Programm beendet werden.
- Ist der Nenner negativ, soll beim Zähler und beim Nenner das Vorzeichen umgedreht werden.
- Danach wird mit der ggt-Funktion der ggT von Zähler und Nenner berechnet und aus beiden herausdividiert.