Inf ProgTech Übung: Wörtersumme (Backtracking)

Klaus Kusche

Gesucht ist ein Programm, dass für Wörtersumme-Denksportaufgaben alle Lösungen mittels Backtracking ermittelt: Gegeben sind drei Worte, deren *Buchstaben so durch Ziffern zu ersetzen* sind, dass die *Addition* der beiden ersten Worte das dritte Wort ergibt.

Dabei darf jede Ziffer nur einem Buchstaben zugeordnet werden (d.h. zwei Buchstaben dürfen nicht dieselbe Ziffer darstellen, die drei Worte dürfen daher aus insgesamt maximal zehn verschiedenen Buchstaben bestehen), und der erste Buchstabe eines jeden Wortes darf nicht durch 0 ersetzt werden.

Beispiele mit eindeutiger Lösung sind *"zeroes + ones = binary"*, *"send + more = money"*, *"apple + lemon = banana"* oder *"sechs + sechs = zwoelf"*, solche mit mehreren / vielen Lösungen "tisch + fisch = essen", "vater + mutter = eltern" oder *"hundert + hundert = tausend"*.

Lade dir von meiner Webseite das **Rahmenprogramm** herunter
und ergänze die beiden fehlenden Funktionen:

- **`bool loesung(int pos)`**

  Das ist die eigentliche, *rekursive Lösungsfunktion*: Sie probiert pro Aufruf alle Möglichkeiten (Ziffern) für den Buchstaben mit dem Index **`pos`** in **`buchst`** und **`ziffer`**, von **`anfang[pos]`** (Ziffer 0 oder 1) bis 9. Die Elemente in **`buchst`** und in **`ziffer`** mit einem Index kleiner **`pos`** sind beim Aufruf schon gefüllt, diejenigen mit einem Index größer **`pos`** werden durch rekursive Aufrufe berechnet, nachdem die Stelle **`pos`** mit einer zulässigen Ziffer belegt wurde.

  Eine Ziffer muss *zwei Kriterien* erfüllen, um eine zulässige Belegung eines Buchstabens zu sein:

  - Es muss eine *noch nicht verwendete Ziffer* sein.
    Das lässt sich über das Array **`belegt`** direkt feststellen.

    Wenn das erfüllt ist, kann die gewählte Ziffer für den aktuellen Buchstaben einerseits in das Array **`ziffer`** eingetragen und andererseits im Array **`belegt`** als vergeben markiert werden (beides muss man nach dem Versuch wieder *rückgängig* machen, für unbelegte Buchstaben muss **`-1`** in **`ziffer`** gespeichert werden!).
  - Bevor man rekursiv den nächsten Buchstaben in Angriff nimmt, muss man prüfen, ob diese Ziffer überhaupt *noch zu einer Lösung führen* kann oder ob die Rechnung damit schon falsch ist. Dazu dient die Funktion **`sinnvoll()`**: Liefert sie **`false`**, kann man sich den rekursiven Aufruf sparen, die Ziffer sofort rückgängig machen und gleich die nächste probieren.

  Wenn *alle Buchstaben* erfolgreich belegt sind, soll **`ausgabe()`** aufgerufen werden.

  Der Returnwert gibt an, ob eine *Lösung gefunden* wurde: **`true`** = ja, **`false`** = nein.

- **`bool sinnvoll(void)`**

  Diese Funktion prüft, ob die bisher in **`ziffer`** gespeicherten Ziffern *noch zu einer Lösung führen* können (**`true`**) oder schon eine falsche Rechnung ergeben (**`false`**).

  - Dazu muss man *von hinten nach vorne* (weil man ja den *Übertrag* spaltenweise von hinten nach vorne ermitteln und mitsummieren muss!) alle Spalten einzeln nachrechnen, die *schon komplett mit Ziffern* belegt sind, bis man auf eine Spalte mit fehlenden Ziffern trifft: Stimmt eine schon fertig belegte Spalte nicht, gibt man **`false`** zurück, passt der bisher belegte hintere Teil der Rechnung (bzw. alles, falls schon alle Buchstaben mit Ziffern belegt sind), returniert man **`true`**.

    Achtung: In der *vordersten Spalte* darf *kein Übertrag* entstehen!

  - Noch etwas besser (aber viel komplizierter!) ist es, immer alle Spalten so weit wie möglich zu prüfen, auch wenn noch einzelne Ziffern fehlen und man deshalb Summe und Übertrag nicht exakt berechnen kann (dadurch fällt z.B. sofort auf, wenn das Summenwort eine Stelle länger als die beiden Summanden ist und vorne mit etwas anderem als 1 belegt wird).