Programmieren C: String-Funktionen: Umwandlung Text <==> Dezimalzahl

Klaus Kusche

Teil 1: String ==> Zahl

Gesucht ist eine Funktion, die das macht, was **`atoi`** bisher für uns gemacht hat:
Sie bekommt als *Parameter* einen *String*, der eine ganze Dezimalzahl enthält,
verwandelt ihn in einen **`int`**, und liefert diesen als *Returnwert*.

- Wir gehen dazu den String von vorne nach hinten *Zeichen für Zeichen* durch.
- Bei jedem Zeichen prüfen wir, ob es eine *Ziffer* ist, verwandeln es
  in den entsprechenden *Ziffernwert* (wie berechnet man aus dem ASCII-Code
  einer Ziffer die Zahl?), und rechnen diesen *zum bisherigen Ergebnis dazu* (wie?).

  Um festzustellen, ob ein Zeichen eine *Ziffer* ist,
  solltest du die entsprechende *vordefinierte Funktion* verwenden.

  Wenn der String ein Zeichen enthält, das *keine Ziffer* ist,
  sollte eine *Fehlermeldung* (mit dem fehlerhaften Zeichen!) ausgegeben werden.
  Die *Umwandlung* wird *beendet* und die *bisher umgewandelte Zahl zurückgegeben*.

Das *Hauptprogramm* soll diese Funktion der Reihe nach *für alle Worte der Befehlszeile*
aufrufen und ihr Ergebnis ausgeben.

Zusatzaufgabe:

Als Zusatzaufgabe könntest du noch ein **`'-'`** an der *ersten* Stelle eines Wortes
(und nur dort!) richtig behandeln. Am einfachsten ist das, wenn man sich am Anfang
das *Vorzeichen* merkt (und zwar am besten als *Multiplikationsfaktor* **`1`** oder **`-1`**)
und *nach* der Umwandlung zum Ergebnis dazurechnet.

Teil 2: Zahl ==> String

Der zweite Schritt ist die *umgekehrte Umwandlung*, von einem **`int`** in einen *Text*,
so wie es **`printf`** macht. Dieser Schritt ist etwas schwieriger, wenn der Ergebnis-String
nur so lang sein soll, wie es zur Darstellung der jeweiligen Zahl nötig ist.

Es gibt mehrere Varianten:

- Die Ziffern von vorne nach hinten berechnen. Das ist kompliziert und langsam.
- Zuerst die Länge feststellen, dann die Ziffern von hinten nach vorne gleich an
  die richtige Stelle in das Ergebnis speichern. Das ist auch kompliziert und langsam.
- Die Ziffern mit einer rekursiven Funktion von vorne nach hinten berechnen.
  Das ist der einfachste und eleganteste Weg, aber speichermäßig sehr ineffizient.
- Die Ziffern *von hinten nach vorne* berechnen, *zwischenspeichern*, und wenn man
  fertig ist und weiß, wie viele es sind, *an die richtige Stelle des Ergebnisses kopieren*,
  oder die Ziffern *von hinten nach vorne* berechnen, in *verkehrter* Reihenfolge
  von links nach rechts speichern, und nachher *umdrehen*. So wollen wir es machen.

Wir schreiben eine *Funktion*, die mit einem *String für das Ergebnis*, der *maximalen Länge* dieses Strings (für *Fehlerprüfungen*, ob die Zahl überhaupt in den String *hineinpasst*, hier wird die *Arraygröße* des Ergebnis-Strings incl. Platz für die Endemarkierung übergeben), und der zu verwandelnden *Zahl* als *Parameter* aufgerufen wird.

Wenn die Zahl in den String passt, liefert die Funktion den übergebenen, befüllten *Ergebnis-String als Returnwert*. Passt die Zahl *nicht*, soll die Funktion **NULL** returnieren.

- *Extrahiere* in einer Schleife immer wieder *die hinterste Ziffer*
  aus der umzuwandelnden Zahl, streiche sie aus der Zahl weg, verwandle sie in das entsprechende *ASCII-Zeichen* (wie?), und speichere dieses *im Ergebnis-String* (wahlweise von ganz hinten nach vorne oder von vorne nach hinten).

  Wiederhole das, bis von der ursprünglichen Zahl nichts mehr übrig ist (0).

  Achtung: Das Ergebnis muss *immer mindestens eine Stelle* haben,
  auch wenn die ursprüngliche Zahl von Anfang an 0 ist!
  (welche Art von Schleife ist dafür sinnvoll?)

  Prüfe vor jedem Speichern, ob das Zeichen noch im String *Platz* hat,
  und beende die Funktion erfolglos, wenn nicht.

- *Korrigiere* dann den String mit einer weiteren Schleife:

  Wenn du *von hinten nach vorne* gespeichert hast, musst du die Zeichen *nach links schieben*, damit die erste Ziffer am Anfang des Strings steht.

  Wenn du die Ziffern *von vorne nach hinten* gespeichert hast, musst du sie vor Ort *umdrehen* (eine Schleife zum Umdrehen eines Strings stand einmal auf einer Folie, sie hatte zwei gegenläufige Schleifenzähler).

  Beides solltest du *ohne* Verwendung vordefinierter Funktionen programmieren.

- Achte darauf, dass dein Ergebnis immer ein *gültiger String* ist!
  Im Idealfall sollte der Ergebnis-String sogar im Fehlerfall nullterminiert werden (außer es wurde eine Länge kleinergleich 0 übergeben).

Wir erweitern unser *Hauptprogramm* aus dem ersten Schritt:

- Leg einen *Hilfsstring für das Ergebnis* an und speichere den **int**,
  den die erste Funktion liefert, mit der zweiten Funktion in diesem String.

  Mach diesen Hilfsstring so kurz (kleiner als 10 Zeichen),
  dass du *auch den Überlauf-Fall testen* kannst!

- Wenn die Umwandlung erfolgreich ist, gib das ursprüngliche Befehlszeilen-Wort, den umgewandelten **int**, und den zurückgewandelten String mit **printf** aus.

  Sonst gib den **int** und eine *Fehlermeldung* aus.

Zusatzaufgabe:

- Es ist wieder klug, das *Vorzeichen* ganz am Anfang zu prüfen und *ganz vorne* im Ergebnis zu speichern und dann die Zahl *ohne Vorzeichen zu verwandeln*, *dahinter* zu speichern und zu verschieben oder umzudrehen.