Inf Progtech Übung: Rekursiver Taschenrechner

Klaus Kusche

Schreib ein Programm, das als *Taschenrechner* benutzt werden kann: Es bekommt eine Rechnung als Eingabe, rechnet sie *unter Beachtung der Vorrangregeln, Klammern usw.* aus, und zeigt das Ergebnis an.

Im Detail soll dein Programm Folgendes leisten:

- Dein Programm soll ganze Zahlen, Klammern, ein unäres Minus (vor Zahlen, Klammern und anderen Elementar-Konstrukten), ***** und **/** sowie **+** und **−** kennen.

  Achtung bei **/**: Auf Division durch **0** prüfen!
- Es soll intern mit Gleitkomma-Zahlen rechnen.
- Das Programm wird ohne Angaben auf der Befehlszeile aufgerufen.

  Der Input, d.h. die zu berechnende Rechnung, wird *zeichenweise* vom Terminal gelesen.

  Tabulatoren und Zwischenräume im Input werden dabei ignoriert bzw. überlesen. Auch Leerzeilen sollen erlaubt sein.
- Wenn der Input syntaktisch ungültig ist, soll eine Fehlermeldung mit der Fehler-Position ausgegeben werden (zumindest mit dem Zeichen, bei dem der Input nicht mehr verarbeitet werden konnte), und das Programm soll enden.

Hinweise:

- Analysiere den Input-Text direkt zeichenweise, *ohne den Umweg über die Zerlegung in Tokens* (d.h. statt getrenntem Lexer und Parser gibt es nur eine Ebene).
- Ich stelle als Ausgangsbasis ein *Programmfragment* zur Verfügung.
  Verwende die bereits definierten Funktionen und globalen Variablen.

  In den Kommentaren des Programmfragmentes stehen auch die *Syntax-Regeln* für die Maximalvariante des Programmes.

Wenn die Grundfunktion passt, kannst du noch folgende *optionale Features* einbauen:

- Das Programm soll auch *Gleitkomma-Zahlen* in der in C üblichen Schreibweise (in den wichtigsten Varianten: mit oder ohne "**e**", mit oder ohne Komma-Teil) als Input verarbeiten können.

  Der Exponent kann entweder kein Vorzeichen oder **+** oder **−** enthalten.
- Weiters soll es "**E**" und "**P**" für die *Konstanten* e und π (**M_E** und **M_PI**) kennen, vielleicht auch noch **| |** für den *Absolutwert* (Funktion **fabs**) und **[ ]** für den *ganzzahligen Anteil* (Funktion **trunc**).
- Bau dein Programm so, dass es *mehrere Rechnungen* als Input liest und berechnet, und zwar *eine Rechnung pro Zeile* (d.h. beim **\n** muss die Rechnung zu Ende sein, und in der nächsten Zeile beginnt eine neue Rechnung).

  "**!**" in einer Rechnung soll dabei für das *Ergebnis der vorigen Rechnung* stehen.
- Implementiere **^** für *Potenzieren* (Vorrang vor ***** und **/**).

Achtung:
^ bindet (im Unterschied zu allen anderen Operatoren) *von rechts nach links!*

- Implementiere *Variablen*:

  Endet eine Rechnung bzw. Zeile auf "**=** *c*" (wobei *c* irgendein *Kleinbuchstabe* für eine von 26 Variablen ist), wird das Ergebnis in dieser Variable gespeichert (entgegen aller Programmiersprachen weisen wir *von links nach rechts* zu, d.h. links vom = steht die Rechnung und rechts die Variable, denn wenn die Variable links stünde, kann man nicht mehr mit einem Zeichen Vorausschau zwischen Rechnung und Zuweisung unterscheiden).

  Variablen können überall statt Zahlen in Rechnungen vorkommen.

- Implementiere als Beispiel eine *Funktion*, z.B. **`Sqrt(`** … **`)`** für die Wurzel.

  Zur Unterscheidung von Variablen werden Funktionen *groß* geschrieben.