Dr. Andreas Borchert Institut für Angewandte Informationsverarbeitung 21. November 2006
Christian Ehrhardt

Allgemeine Informatik III (WS 2006/2007)

Abgabetermin 28.Nov. 2006

Pell'sche Gleichung (2 Punkte)

Wir betrachten Lösungen der sog. Pell'schen Gleichung $x^2 - 2y^2 = 1$ in ganzen Zahlen. Eine kleine Lösung ist dabei $x = 3$ und $y = 2$. Außerdem gilt: Wenn (x_n, y_n) eine Lösung der Gleichung ist, dann ist auch (x_{n+1}, y_{n+1}) mit $x_{n+1} = 3x_n + 4y_n$ und $y_{n+1} = 2x_n + 3y_n$ eine Lösung der Gleichung. Es lohnt sich ggf., das kurz nachzurechnen.
Schreiben Sie ein Programm, das mit Hilfe dieser Formel immer größere Lösungen der Gleichung berechnet und die linke Seite auswertet. Die Auswertung soll dabei mit verschiedenen Datentypen (`int`, `long long int`, `float` und `double`) durchgeführt werden. Warum kommt es zu abweichenden Ergebnissen?

Ein einfacher Taschenrechner (8 Punkte)

In diesem Teil des Blattes wird ein Programm entwickelt, das einen auf der Standardeingabe angegebenen arithmetischen Ausdruck berechnet. Der Ausdruck besteht nur aus `double`-Zahlen, Klammern und den Operatoren `+`, `-`, `*`, `/` und `^` (wobei `^` für "hoch" steht, `2^4` ergibt also `16`). Selbstverständlich wird dabei die Vorrangregel für Operatoren (Potenz vor Punkt vor Strich) beachtet, der Ausdruck `2+3*4*5-5*2^2` ergibt also `2+60-20=42`.

Einzelne Tokens

Zunächst sollte der Ausdruck in einzelne Teile (in der Fachsprache auch Tokens genannt) zerlegt werden. Ein Teil ist dabei entweder eine `double`-Zahl, wie sie z.B. `scanf` mit `%lf` einliest, ein Operator, eine öffnende Klammer oder eine schließende Klammer. Weiterhin gelten die folgenden Regeln für die Aufeinanderfolge von Tokens:

- Das erste Token überhaupt, das erste Token nach einer öffnenden Klammer sowie das erste Token nach einem Operator muß entweder eine Zahl oder eine öffnende Klammer sein.
- Auf eine schließende Klammer oder eine Zahl muß dagegen entweder eine schließende Klammer oder ein Operator folgen.

Es ergibt sich also die folgenden Tabelle:

Letztes Token	Mögliche Tokens
Keines (Anfang des Ausdrucks)	Zahl oder öffnende Klammer
Operator	Zahl oder öffnende Klammer
öffnende Klammer	Zahl oder öffnende Klammer
Zahl	schließende Klammer oder Operator
schließende Klammer	schließende Klammer oder Operator

Wir beginnen also zunächst mit einem Programm, das einen Ausdruck nur in seine Tokens zerlegt und dabei auf syntaktische Korrektheit überprüft. Dazu wird Buch geführt, was für Tokens als nächstes erlaubt sind. Ist eine Zahl erlaubt, dann wird **immer zuerst** versucht eine Zahl einzulesen. Ist keine Zahl erlaubt oder geht das einlesen einer Zahl schief, dann besteht das nächste Token aus genau einem Zeichen, das dann statt dessen eingelesen wird. Hier muß noch überprüft werden, ob das gelesene Zeichen an dieser Stelle auch erlaubt ist.

Zwei kleine Hilfsfunktionen

Für die Bewertung des Ausdrucks werden sich zwei kleine Funktionen sicherlich als nützlich erweisen:

- Eine Funktion, die zu einem Operator eine Zahl zurückliefert, die die Priorität des Operators angibt. Die Operatoren `+` und `-` haben dabei die niedrigste, der Operator `^` die höchste Priorität.
- Eine Funktion, die zwei `double`-Zahlen und einen Operator (in Form eines `char`) übergeben bekommt und das Ergebnis zurückliefert, das man erhält, wenn die beiden Zahlen mit diesem Operator verknüpft werden.

Auswertung von Ausdrücken ohne Klammern

Zunächst werten wir nur Ausdrücke ohne Klammern aus, die Behandlung von Klammern folgt später.

Zwei Stacks

Um einen Ausdruck ohne Klammern auszuwerten, werden zwei Stacks (realisiert z.B. mit Hilfe eines fest dimensionierten Arrays) benötigt. Einer der Stacks nimmt nur Zahlen auf, der andere nur Operatoren. Jedes Token wird, sobald es gelesen wurde, oben auf dem passenden Stack abgelegt.

Regel für den Operatorenstack

Lediglich für den Operatorenstack gilt die Regel, daß ein Operator erst dann auf den Stack gelegt werden kann, wenn der vorher oben liegende Operator eine echt niedrigere Priorität hat, als der neue Operator. Liegt also z.B. ein `+` oben auf dem Stack, dann kann zunächst kein `+` oder `-` auf den Stack gelegt werden, wohl aber z.B. `*` oder `^`.

Abbauen des Stacks

Um diese Bedingung einzuhalten, muß nach dem Einlesen eines Operators, der nicht direkt auf den Stack gelegt werden kann, zunächst der Stack so weit abgebaut werden, bis es erlaubt ist, den neuen Operator auf den Stack zu legen. Dazu werden die beiden Werte, die sich ganz oben auf dem Zahlenstack befinden und der Operator, der sich oben auf dem Operatorstack befindet, von ihren Stacks entfernt, und die beiden Zahlen werden mit Hilfe des Operators verknüpft. Das Ergebnis wird wieder oben auf den Zahlenstack gelegt. Durch eine solche Operation wird also sowohl der Zahlenstack alsauch der Operatorstack um ein Element kleiner.
Diese Operation wird so lange wiederholt, bis der gerade eingelesene Operator auf den Operatorstack gelegt werden kann.
Am Ende des Ausdrucks werden dann die noch auf dem Stack verbliebenen Operatoren auf die gleiche Weise abgebaut. Es verbleibt eine Zahl auf dem Zahlenstack, die das Ergebnis des Ausdrucks ist.

Beispiel

Die Auswertung des oben bereits erwähnten Ausdrucks `2+3*4*5-5*2^2` würde wie folgt aussehen:

Gelesenes Token	Zahlenstack	Operatorstack
2	2	
+	2	+
3	2 3	+
*	2 3	+ *
4	2 3 4	+ *
*	2 12	+
	2 12	+ *
5	2 12 5	+ *
-	2 60	+
	62	
	62	-
5	62 5	-
*	62 5	- *
2	62 5 2	- *
^	62 5 2	- * ^
2	62 5 2 2	- * ^
Ende	62 5 4	- *
	62 20	-
	42	

Wir erhalten also tatsächlich das erwartete Ergebnis.

Klammern

Das hinzufügen von Klammern funktioniert jetzt sehr einfach:

- Eine öffnende Klammer wird grundsätzlich auf den Operatorstack gelegt, egal was dort vorher für Operatoren lagen.
- Liegt eine öffnende Klammer bereits ganz oben auf dem Operatorstack, dann ist als nächstes jeder Operator (oder auch eine weitere öffnende Klammer) erlaubt.
- Bei einer schließenden Klammer wird der Operatorstack bis zur nächsten öffnenden Klammer abgebaut.

Sonstiges

Der Operator `^` kann mit Hilfe der C-Funktion `pow` implementiert werden. Bitte `math.h` und beim Übersetzen die Option `-lm` nicht vergessen!
Ein klassischer Taschenrechner arbeitet genau nach dem oben beschriebenen Verfahren. Dabei wird immer die oberste Zahl auf dem Zahlenstack im Display angezeigt. Diese ändert sich, wenn der Stack teilweise abgebaut wird. Die Qualität des Taschenrechners hängt unter anderem davon ab, wieviel Platz auf dem Zahlenstack vorhanden ist, bei den meisten Modellen ist es mit Klammern nicht besonders schwer, diesen zum Überlauf zu bringen.

Viel Erfolg!