Hausübung 1 – Taschenrechner

PROG1 UE

WS16/17

Entwickeln Sie einen Taschenrechner mit Speicherfunktion. Beim Programmstart können auf der Kommandozeile Zahlen angegeben werden, welche in den internen Speicher des Rechners geschrieben und somit für Rechenoperationen benutzt werden können. Hier ist zuerst zu überprüfen, ob alle angegeben Parameter wirklich nur Zahlen sind, ansonsten ist das Programm mit einer Fehlermeldung zu beenden. Beachten Sie auch den Fall, dass gar keine Zahlen angegeben werden, welcher natürlich nicht zu einem Programmabbruch führen darf!

Das Programm soll nun in einer Schleife zeilenweise (mit fgets()) lesen. Jede Zeile kann aus folgenden Elemente bestehen (getrennt durch Leerzeichen):

- 1. Element: Ganzzahl, Fließkommazahl oder Kleinbuchstabe
- 2. Element: Rechenoperator
- 3. Element: Ganzzahl, Fließkommazahl oder Kleinbuchstabe

Zum Beenden des Programms soll es außerdem möglich sein, statt einer Berechnung das Wort “quit” einzugeben. Wird dieses Wort als 1. Element gelesen, soll sich das Programm ordnungsgemäß beenden.

Um den eingelesenen String in diese drei Elemente zu teilen, schauen Sie sich die Funktion strtok() an. Jeder dieser drei Teile muss nun auf richtige Eingabe geprüft werden.

1 Ganzzahl, Fließkommazahl oder Kleinbuchstabe

Hier muss überprüft werden, ob in diesem Teil nur Ziffern oder ein Punkt (als Komma bei Fließkommazahlen) vorhanden sind. In diesem Fall kann der String in eine Zahl umgewandelt werden (siehe atoi(), atol() und atof()). Besteht der erste Teil neben Ziffern und einem Punkt auch aus nicht-gültigen Zeichen, so ist eine Fehlermeldung auszugeben.

Im Falle eines Kleinbuchstaben ist zu prüfen, ob wirklich nur ein Buchstabe zwischen a und z vorhanden ist - alles andere führt zu einer Fehlermeldung. Sollte der String nur aus einem Kleinbuchstaben bestehen, ist im internen Speicher nachzusehen, ob diese Stelle bereits besetzt ist. Ist dies der Fall, wird die entsprechende Zahl, welche sich hinter diesem Kleinbuchstaben verbirgt, für die aktuelle Rechenoperation herangezogen. Andernfalls ist eine Fehlermeldung auszugeben.

2 Rechenoperator

Gültige Rechenoperationen sind +, -, * und /. In allen anderen Fällen (z.B. auch wenn der zweite Teil aus mehr als einem Zeichen besteht) ist eine Fehlermeldung auszugeben.

Falls beim Prüfen der Teile keine Fehler aufgetreten sind, kann das Ergebnis berechnet werden. Andernfalls ist von Neuem mit der Verarbeitung zu beginnen (Fehlermeldung nicht vergessen).

3 Berechnung

Sind alle drei Teile der Eingabe ausgewertet, kann das Ergebnis berechet werden. Achten Sie auch auf die Möglichkeit, dass Fließkommazahlen als Ergebnis herauskommen können. Das Ergebnis der Rechnung ist in einer neuen Zeile auszugeben und zusätzlich in einem neuen Speicherplatz in den

internen Speicher zu schreiben (der alphabetische Index ist ebenfalls auszugeben).

4 Interner Speicher

Der interne Speicher wird in einem dynamisch angeforderten (und wachsenden) Array von passendem Typ verwaltet. Überlegen Sie sich für das Array einen Datentyp, welcher alle Ergebnisse korrekt speichern kann (also Ganzzahlen als auch Fließkommazahlen). Zu Beginn soll der interne Speicher entweder mit den Zahlen der Kommandozeile initialisiert oder leer sein. Jedes Element im Array hat einen alphabetischen Index, beginnend mit a. Wird nun eine Berechnung durchgeführt, soll der interne Speicher um eine Stelle vergrößert, und das Ergebnis darin gespeichert werden.

Wenn die Stelle z (also die 26. Position im Array) vergeben wurde, ist eine Meldung auszugeben und das Programm zu beenden.

Wird somit an erster oder dritter Stelle ein Kleinbuchstabe in der Berechnung angegeben, muss der interne Speicher "befragt" werden, um die entsprechende Zahl zu erhalten. Ist die geforderte Speicherstelle noch nicht vergeben, ist eine Fehlermeldung auszugeben.

Am Ende des Programms ist auf das korrekte Freigeben von dynamischem Speicherplatz zu achten.

5 Beispiel

Ein Problelauf könnte folgendermaßen aussehen:

```
$ /a.out 1 2 3 4 5 6 7 8 9
a = 1.000000
b = 2.000000
c = 3.000000
d = 4.000000
e = 5.000000
f = 6.000000
g = 7.000000
h = 8.000000
i = 9.000000
Eingabe> a * b
j = 2.000000
Eingabe> c + 3.2
k = 6.200000
Eingabe> 1.2 / d
l = 0.300000
Eingabe> 5 * a
m = 5.000000
Eingabe> 3 - a
n = 2.000000
Eingabe> quit
```

6 Hinweis

- Überlegen Sie, welche Funktionalität für das Programm erforderlich ist, und teilen Sie diese Aufgaben in geeignete Funktionen. Eine Funktion soll im besten Fall eine Aufgabe erledigen. Achten Sie auf passende Rückgabewerte, um Erfolgs- bzw. Fehlerfall an den Aufrufer zu signalisieren.
- Achten Sie bei der Übergabe von Strings (char *) darauf, ob diese in der jeweiligen Funktion verändert werden, und berücksichtigen Sie das bei der Wahl des Parametertyps entsprechend (Stichwort: const).
- Achten Sie bei der Implementierung des internen Speichers darauf, das dieser korrekt freigegeben wird – im Erfolgs- und auch im Fehlerfall (wenn erforderlich).