Frage **1**

Teilweise richtig

Erreichbare Punkte: 7,00

Lee die Listenrechnerin

Programmieren Sie einen einfachen Rechner, der zuerst einen Reihe von Operationen und Werte einliest, diese in einer verketteten Liste speichert und sie dann auswertet!

Aufgabe

Ihr Rechenprogramm liest zeilenweise Zahlen und Operatoren ein, speichert diese in einer verketteten Liste und wertet diese dann als einen einzigen Ausdruck aus.

Eingabe

Als Eingabe kommen immer abwechselnd Integer-Zahlen und Operatoren ('+', '-', '*', '/'), angefangen mit einer Zahl. Dann folgen beliebig viele Zahlen und Operatoren.
Diese Eingaben sollen in einer verketteten Liste gespeichert werden. Jeder Knoten der Liste kann sowohl einen char für den Operator als auch einen Integer für eine Zahl speichern. Zusätzlich enthält der Knoten einen enum-Typen der angibt, ob es sich um einen Operator oder eine Zahl handelt.
Jeder neue Knoten wird am Ende der Liste eingefügt und nach jedem Einfügen soll die Liste ausgegeben werden.
Die Eingabe wird durch ein '=' beendet (und folgt immer auf eine Zahl). Dieser Operator wird nicht mehr in die Liste eingefügt.

Auswertung

Wurde die Eingabe beendet soll die Liste in zwei Schritten ausgewertet werden. Dabei wird sie vom Anfang zum Ende hin ausgewertet. Es gelten folgende Regeln:

- Es werden nur Integerrechnungen durchgeführt.
- Divisionen durch 0 werden zu Divisionen durch 1.
- Im ersten Schritt werden nur Punktoperationen durchgeführt (Multiplikation und Division).
- Im zweiten Schritt werden nur Strichoperationen durchgeführt (Addition und Subtraktion).
- Immer wenn ein Operator ausgewertet wird und eine Berechnung stattfindet soll die Liste danach neu ausgegeben werden. Zum Beispiel:
  - 2 + 8 / 4 * 5 - 3
  - 2 + 2 * 5 - 3
  - 2 + 10 - 3
  - 12 - 3
  - 9

Bei jedem Zwischenergebnis reduziert sich somit die Anzahl der Knoten um zwei, und am Ende der Auswertung sollte die Liste nur noch einen Knoten enthalten - der mit dem Endergebnis (siehe vorhergehendes Beispiel).

Hier ein Beispiel des Programmablaufs:

```
Enter number: 4

Current list: 4

Enter operator: +

Current list: 4+

Enter number: 10

Current list: 4+10

Enter operator: /

Current list: 4+10/

Enter number: 2

Current list: 4+10/2

Enter operator: -

Current list: 4+10/2-

Enter number: 3

Current list: 4+10/2-3

Enter operator: *

Current list: 4+10/2-3*

Enter number: 20

Current list: 4+10/2-3*20

Enter operator: =

Point operator result: 4+5-3*20

Point operator result: 4+5-60

Dash operator result: 9-60

Das operator result: -51

Final result: -51
```

Die Struktur der Knoten ist bereits definiert:

```
struct node {
   char operator;
   int number;
   enum node_type type;
   struct node* next;
};
```

Das enum type bestimmt dabei ob es sich um einen Knoten für einen Operator oder eine Zahl handelt. Das entsprechende Enum ist ebenfalls bereits definiert. Wurde eine Zahl eingegeben wird also ein Knoten erstellt, bei dem dann number gesetzt wird und der type auf number_type. Wurde ein Operator eingegeben wird nur operator gesetzt und der type auf operator_type. Es werden nie beide Felder belegt (also number und operator).

Signaturen für einige Funktionen wurden bereits definiert, Sie können sie verwenden müssen aber nicht. Zu den Signaturen:

- struct node* createOperatorNode(char operator);
  Allokiert Speicher für einen Knoten, belegt den Operator des Knotens und setzt den Typ auf operator_type. Ein Pointer auf den Knoten wird retourniert.
- struct node* createNumberNode(int number);
  Allokiert Speicher für einen Knoten, belegt die Zahl des Knotens und setzt den Typ auf number_type. Ein Pointer auf den Knoten wird retourniert.
- struct node* findFirstPointOperator(struct node* head);
  Durchsucht die Liste, deren Anfang head ist, nach dem ersten Punktoperator.
- struct node* findFirstDashOperator(struct node* head);
  Durchsucht die Liste, deren Anfang head ist, nach dem ersten Strichoperator.
- struct node* findPreviousNode(struct node* head, struct node* node);

Durchsucht die Liste, deren Anfang head ist, nach dem Knoten der vor node liegt.

- void removeAfterNode(struct node* node);
  Entfernt den Knoten nach node aus der Liste.
- struct node* addLast(struct node* head, struct node* newNode);
  Fügt den Knoten newNode am Ende der Liste ein, deren Anfang head ist.
- void printList(struct node* head);
  Gibt die gesamte Liste aus, deren Anfang head ist.

Wichtige Hinweise

- Verwenden Sie dieselbe Ein- und Ausgabestruktur wie im obenstehenden Beispiel, damit Sie die automatisierten Tests erfolgreich bestehen können. Der rote Text dient nur zur Hervorhebung des User Inputs im Beispiel, Sie müssen diese Färbung nicht im Programm abbilden!
- Geben Sie **Zeilenumbrüche immer vor einer neuen Zeile aus und nicht am Ende**. Wenn Sie sich daran halten, sollte die Formatierung leichter zur Übereinstimmung mit der erwarteten Ausgabe gebracht werden können. Das Programm startet daher auch mit einem Zeilenumbruch.
- In CodeRunner wird jede einzelne Eingabe von einem Zeilenumbruch bestätigt, der in der Ausgabe aber nicht sichtbar ist. Wenn Sie außerhalb von CodeRunner das Programm starten, selber Werte eingeben (siehe Beispielausgabe) und dabei Zeilenumbrüche produzieren, wirkt sich das auf die Ausgabe aus (mehr Zeilenumbrüche als beim automatischen Testen in CodeRunner).
- Achten Sie darauf den Speicher korrekt zu verwalten.

Testfälle

Antwort: (Abzugssystem: 0 %)

Antwort zurücksetzen

```c
#include <stdio.h>
#include <stdlib.h>

enum node_type {
    operator_type,
    number_type
};

struct node {
    char operator;          //Jeder Knoten enthält immer nur entweder einen
    int number;             //Operator oder eine Zahl, nicht beides
    enum node_type type;
    struct node* next;
};

struct node* createOperatorNode(char operator);
struct node* createNumberNode(int number);

struct node* findFirstPointOperator(struct node* head); // NULL, wenn nicht gefunden
struct node* findFirstDashOperator(struct node* head); // NULL, wenn nicht gefunden

struct node* findPreviousNode(struct node* head, struct node* node);
void removeAfterNode(struct node* node);
```

Prüfen

	Test	Eingabe	Erwartet	Erhalten	
✖	Single addition	3 + 2 =	Enter number: Current list: 3 Enter operator: Current list: 3+ Enter number: Current list: 3+2 Enter operator: Dash operator result: 5 Final result: 5		✖

	Test	Eingabe	Erwartet	Erhalten	
✖	Single subtraction	4 - 10 =	Enter number: Current list: 4 Enter operator: Current list: 4- Enter number: Current list: 4-10 Enter operator: Dash operator result: -6 Final result: -6		✖
✖	Single multiplication	7 * 12 =	Enter number: Current list: 7 Enter operator: Current list: 7* Enter number: Current list: 7*12 Enter operator: Point operator result: 84 Final result: 84		✖
✖	Single division	10 / 2 =	Enter number: Current list: 10 Enter operator: Current list: 10/ Enter number: Current list: 10/2 Enter operator: Point operator result: 5 Final result: 5		✖

	Test	Eingabe	Erwartet	Erhalten	
✖	Add and subtract	19 + 5 - 7 - 2 + 100 =	Enter number: Current list: 19 Enter operator: Current list: 19+ Enter number: Current list: 19+5 Enter operator: Current list: 19+5- Enter number: Current list: 19+5-7 Enter operator: Current list: 19+5-7- Enter number: Current list: 19+5-7-2 Enter operator: Current list: 19+5-7-2+ Enter number: Current list: 19+5-7-2+100 Enter operator: Dash operator result: 24-7-2+100 Dash operator result: 17-2+100 Dash operator result: 15+100 Dash operator result: 115 Final result: 115		✖

	Test	Eingabe	Erwartet	Erhalten	
✖	Multiply and divide	20 * 3 / 2 / 2 * -10 =	Enter number: Current list: 20 Enter operator: Current list: 20* Enter number: Current list: 20*3 Enter operator: Current list: 20*3/ Enter number: Current list: 20*3/2 Enter operator: Current list: 20*3/2/ Enter number: Current list: 20*3/2/2 Enter operator: Current list: 20*3/2/2* Enter number: Current list: 20*3/2/2*-10 Enter operator: Point operator result: 60/2/2*-10 Point operator result: 30/2*-10 Point operator result: 15*-10 Point operator result: -150 Final result: -150		✖

	Test	Eingabe	Erwartet	Erhalten	
✖	Everything mixed	7 * 3 + 2 - 10 / 5 * 3 + 2 =	Enter number: Current list: 7 Enter operator: Current list: 7* Enter number: Current list: 7*3 Enter operator: Current list: 7*3+ Enter number: Current list: 7*3+2 Enter operator: Current list: 7*3+2- Enter number: Current list: 7*3+2-10 Enter operator: Current list: 7*3+2-10/ Enter number: Current list: 7*3+2-10/5 Enter operator: Current list: 7*3+2-10/5* Enter number: Current list: 7*3+2-10/5*3 Enter operator: Current list: 7*3+2-10/5*3+ Enter number: Current list: 7*3+2-10/5*3+2 Enter operator: Point operator result: 21+2-10/5*3+2 Point operator result: 21+2-2*3+2 Point operator result: 21+2-6+2 Dash operator result: 23-6+2 Dash operator result: 17+2 Dash operator result: 19 Final result: 19		✖
✖	Divide Zero	5 / 0 =	Enter number: Current list: 5 Enter operator: Current list: 5/ Enter number: Current list: 5/0 Enter operator: Point operator result: 5 Final result: 5		✖

	Test	Eingabe	Erwartet	Erhalten	
✖	Divide Zero complex	3 + 2 / 0 / 0 + 3 * 10 / 0 =	Enter number: Current list: 3 Enter operator: Current list: 3+ Enter number: Current list: 3+2 Enter operator: Current list: 3+2/ Enter number: Current list: 3+2/0 Enter operator: Current list: 3+2/0/ Enter number: Current list: 3+2/0/0 Enter operator: Current list: 3+2/0/0+ Enter number: Current list: 3+2/0/0+3 Enter operator: Current list: 3+2/0/0+3* Enter number: Current list: 3+2/0/0+3*10 Enter operator: Current list: 3+2/0/0+3*10/ Enter number: Current list: 3+2/0/0+3*10/0 Enter operator: Point operator result: 3+2/0+3*10/0 Point operator result: 3+2+3*10/0 Point operator result: 3+2+30/0 Point operator result: 3+2+30 Dash operator result: 5+30 Dash operator result: 35 Final result: 35		✖
✖	Divide fractured	15 / 2 =	Enter number: Current list: 15 Enter operator: Current list: 15/ Enter number: Current list: 15/2 Enter operator: Point operator result: 7 Final result: 7		✖

	Test	Eingabe	Erwartet	Erhalten	

✔	`Memory check`	`3`
		`+`
		`2`
		`*`
		`10`
		`+`
		`10`
		`/`
		`2`
		`-`
		`1500`
		`=`
			`** MemCheck Success **`	`** MemCheck Success **`	✔

Einige verborgene Testfälle sind ebenfalls fehlgeschlagen.

Unterschiede anzeigen

◄ Sammlung: Fragen zum Eigenstudium T

Direkt zu:

Peer Review Zwischentest 3 ►