Uebung 06

Aufwand: 19 Stunden

Selina A

2-5-2023

Übung 06

BEISPIEL OPERATOREN ÜBERLADEN

LÖSUNGSIDEE:

Für alle drei Beispiele wird der zur Verfügung gestellte Scanner verwendet.

INFIX:

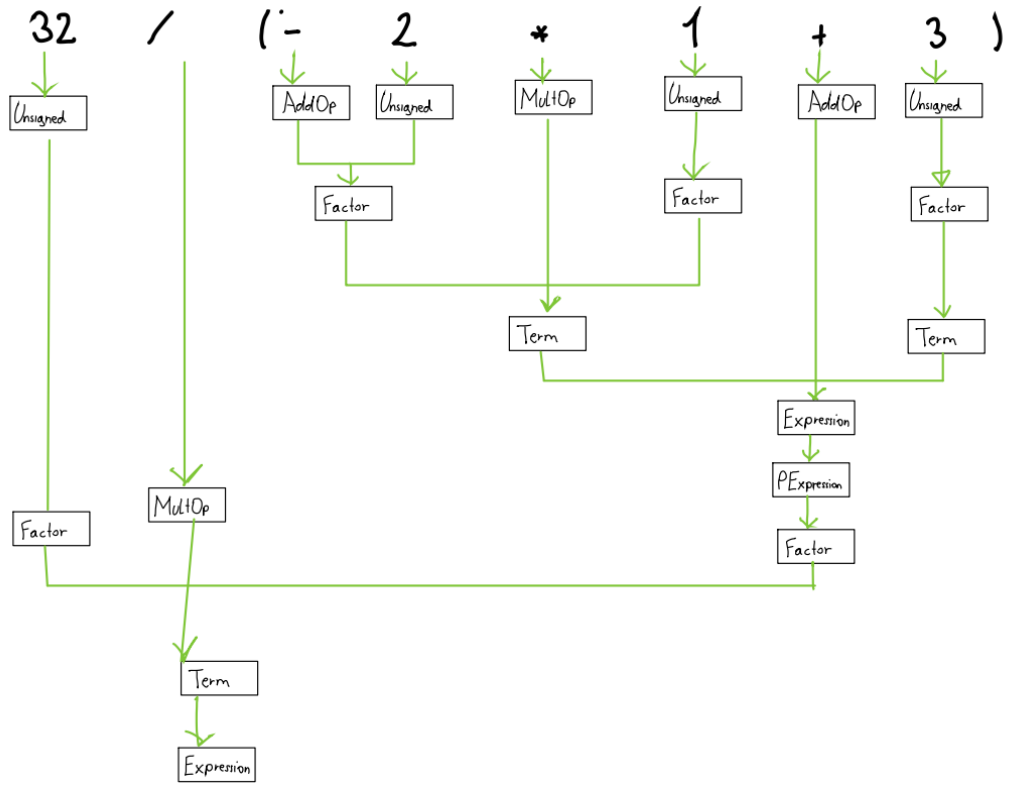
Grammatik:

Expression = Term { AddOp Term } .
Term = Factor { MultOp Factor } .
Factor = [AddOp] (Unsigned | PExpression) .
Unsigned = Digit { Digit } .
PExpression = ” (“ Expression ”)“ .
AddOp = ” +“ | ” −“ .
MultOp = ” ·“ | ” /“ .
Digit = ” 0“ | ” 1“ | ” 2“ | ” 3“ | ” 4“ | ” 5“ | ” 6“ | ” 7“ | ” 8“ | ” 9“ .

Es wird eine Rechnung in Infix eingegeben und danach wird nach der Reihe jedes Element der Grammatik zugeteilt. Zuerst wird geschaut ob es sich um eine Expression handelt. Wenn nicht wird eine Exception geschmissen. Und danach wird laut der oben genannten Grammatik fortgesetzt indem jeweils die richtige Funktion aufgerufen wird und das passende Element geparsed wird. Es wird jeweils kontrolliert, ob die Grammatik eingehalten wird. Wenn eine Division durch 0 entsteht, wird eine Exception geworfen.

Das Ergebnis wird schlussendlich auf der Konsole ausgegeben.

BSP: 32/(-2*1+3)



PRÄFIX:

Grammatik:

Expression = { OP Term } Term .
Term = Signed | Op Term Term
OP= MultOp| AddOp
Unsigned = Digit { Digit } .
Signed={-} Digit {Digit}
AddOp = ” +“ | ” −“ .
MultOp = ” ·“ | ” /“ .
Digit = ” 0“ | ” 1“ | ” 2“ | ” 3“ | ” 4“ | ” 5“ | ” 6“ | ” 7“ | ” 8“ | ” 9“ .

Das System funktioniert ähnlich wie beim Infix nur dass der Operator immer zuerst steht und danach die Zahlen. Es wird genauso nach der Grammatik voran gearbeitet und jedes Mal kontrolliert, ob es sich um den richtigen Grammatikteil handelt, wenn nicht wird wieder eine Exception geworfen.

MIT VARIABLEN:

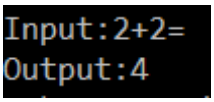
Fortsetzung von Infix Notation. Es wird eine Map in der Klasse angelegt, die jeweils aus Variablenname und den dazugehörigen Wert besteht. Beim Parsen wird nun auch zusätzlich kontrolliert ob es sich um eine Variable handelt und wenn ja ob diese auch in der Map steht, wenn nicht wird eine Exception geworfen. Wenn die Variable einen Wert besitzt, wird dieser anstelle des Variablennamen eingesetzt und normal wie auch bei BSP1 weiter gerechnet und geparsed und am Ende ein Ergebnis zurückgegeben.

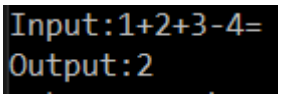
MAP:

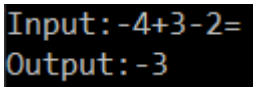
a	5
b	3
c	4

TESTFÄLLE: INFIX

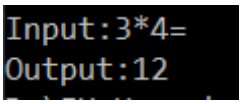
1. TEST ADDITION: MIT NEGATIVEN, MIT POSITIVEN UND MIT MEHREREN

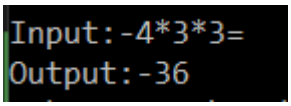




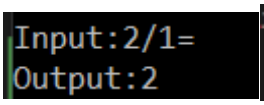


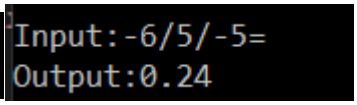
2. TEST MULTIPLIKATION: MIT NEGATIVEN, MIT POSITIVEN UND MIT MEHREREN



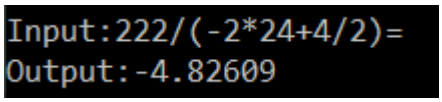


3. TEST DIVISION: MIT NEGATIVEN, MIT POSITIVEN UND MIT MEHREREN

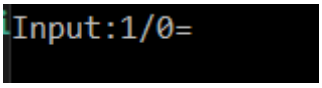




4. TEST BEI KOMPLIZIERTEREN RECHNUNGEN



5. TEST BEI NULL DIVISION

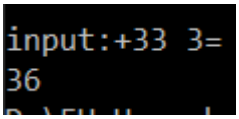


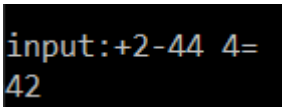
6. TEST BEI FEHLERN

- Test8: 3+++3=
- Test9: *3=+
- Test10: 4(3*3=

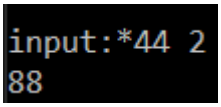
TESTFÄLLE: PRÄFIX

1. TEST ADDITION: MIT NEGATIVEN, MIT POSITIVEN UND MIT MEHREREN

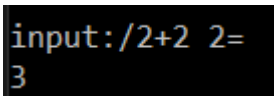


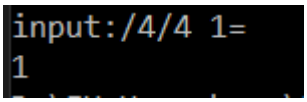


2. TEST MULTIPLIKATION: MIT NEGATIVEN, MIT POSITIVEN UND MIT MEHREREN

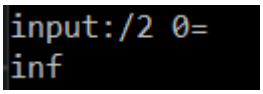


3. TEST DIVISION: MIT NEGATIVEN, MIT POSITIVEN UND MIT MEHREREN





4. TEST BEI NULL DIVISION



5. KOMPLIZIERTERE RECHNUNGEN

6. FEHLER

Grammatikfehler Operanten an der falschen Stelle, Term nicht vorhanden, Expression nicht an der richtigen Stelle,….

TESTFÄLLE: VARIABLE

1. TEST ADDITION: MIT NEGATIVEN, MIT POSITIVEN UND MIT MEHREREN

2. TEST MULTIPLIKATION: MIT NEGATIVEN, MIT POSITIVEN UND MIT MEHREREN

3. TEST DIVISION: MIT NEGATIVEN, MIT POSITIVEN UND MIT MEHREREN

4. TEST BEI NULL DIVISION

5. TEST WENN VARIABLE NICHT VORHANDEN IST

6. TEST WENN ANDERE FEHLER AUFTRETEN

→Grammatikfehler