SWE3; WS 2022

Wolfgang Eder

S2110458039

Zeitaufwand: 8 Stunden

Programmierübung UE06

Inhaltsverzeichnis

Beispiel 1: „Arithmetische Ausdrücke (infix)“

Code

Das Programm befindet sich in der Solution SWE3_EDER_UE06 unter der Solution: „Arithmetische Ausdrücke (infix)“.

Lösungsansatz:

Für die Übungsaufgabe soll ein Programm erstellt werden, welches arithmetische Ausdrücke, geschrieben in Infix-Notation, parsen, evaluieren und korrekt ausgeben kann. Die dem Prinzip innewohnende Idee verfolgt es ein Parsermodul zu implementiert unter Zuhilfenahme des zur Verfügung gestellten Moduls „pfc-facilities/scanner“ als Lexer bzw. Scanner.

Dem zu Grunde liegt die Semiotik, welche vereinfacht für unser Beispiel, das Reich einfacher Zeichenketten berührt und wie folgt eine Grammatik definiert. Die hier skizzierte formale Darstellung erfolgt in der Wirthsche`Erweiterte Backus-Naur-Form, kurz EBNF.

In Folge sind Regeln definiert, welche sich in zwei Symbolen ausdrücken, den Terminalsymbolen und den Nonterminalsymbolen.

z.B.:

```
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“ .
```

Somit wurde eine Grammatik dargestellt, die der Symbolik Syntax und Semantik verleiht, welche in Folge durch einen Scanner korrekt eingelesen, interpretiert und evaluiert werden kann. Das Scannermodul verwendet vielfältige Funktionen laut Dokumentation für diese Aufgabe.

Der Parser, besteht im wesentlichen daraus, die durch den Scanner bereitgestellten Funktionsumfang, für die gewünschte Darstellung der Grammatik. Die für dieses Beispiel gewählte Infix-Notation, wird im Parser gekapselt über rekursive Abstiege erzielt. Die dadurch erzielte Evaluierung wird weiters, falls korrekt, ausgegeben, andernfalls werden geeignete Ausnahmefehler dargestellt.

Der Funktionsumfang besteht aus einer Eingabeschnittstelle, der Symbolik angepassten Identifikationsevaluierung und der gewünschten Grammatik zugeordneten Bestandteilsevaluierung, realisiert über einen rekursiven Abstieg. Zur Testung wurde ein Potpourri von möglichen Testfällen gewählt.

Tests:

```
Test0: '11 / (-17.3 * 22 + 3)' ...
result: -0.029

Test1: '62 / (-17 * 2 + 3)' ...
result: -2.000

Test2: '((17 * -4) + 8) / -6' ...
result: 10.000

Test3: '1 + 2.0 + 3 * (-5 + -4)' ...
result: -24.000

Test4: '1 + 2' ...
result: 3.000

Test5: '1 - 2' ...
result: -1.000

Test6: '2*2' ...
result: 4.000

Test7: '2 / 2' ...
result: 1.000

Test8: '2 / 0' ...
Error: Error. Division by zero.

Test9: '( (' ...
Error: Error parsing 'Expression'.

Test10: ')' ...
Error: Error parsing 'Expression'.

Test11: '(2 + 3))' ...
result: 5.000

Test12: '((2 +3)' ...
Error: Expected 'right parenthesis' but have {{end of file,ts,7}}.

Test13: '(2+3)(2+3)' ...
result: 5.000

Test14: '{' ...
Error: Unknown character '{' encountered.

Test15: '' ...
Error: Error parsing 'Expression'.
```

```
Test13: '(2+3)(2+3)' ...
result: 5.000

Test14: '{' ...
Error: Unknown character '{' encountered.

Test15: '' ...
Error: Error parsing 'Expression'.

Test16: '4' ...
result: 4.000

Test17: '-3' ...
result: -3.000

Test18: '*' ...
Error: Error parsing 'Expression'.

C:\Users\wae\Documents\FHOOE\3.WS\SWE3\swe.ws.2022\SWE3
To automatically close the console when debugging stops
Press any key to close this window . . .
```

Beispiel 2: „Arithmetische Ausdrücke (präfix)“

Code

Das Programm befindet sich in der Solution SWE3_EDER_UE06 unter der Solution: „Arithmetische Ausdrücke (präfix)“.

Lösungsansatz:

Für die zweite Übungsaufgabe wurde eine andere Notation gewünscht, welche sich prinzipiell ähnlich aufbauen lässt, wie in Beispiel 1 ausgeführt, nur leicht abgeändert und simpler ist.

Grammatik:

Expression = Term|Digit

Term = Operation „*space*“ Term|Digit „*space*“ Term|Digit

Operator = „+“ | „-“ | „:“ | „*“

Digit = „0“ | „1“ | „2“ | „3“ | „4“ | „5“ | „6“ | „7“ | „8“ | „9“

Ein Testmodul wurde wiederrum gewählt und zeigt einen der Zeit geschuldeten…

Tests:

```
Test0: '*22' ...
Error: Error parsing 'AddOp'.

Test1: ':22' ...
Error: Error parsing 'Expression'.

Test2: ':20' ...
Error: Error parsing 'Expression'.

Test3: '( (' ...
Error: Error parsing 'Expression'.

Test4: ')' ...
Error: Error parsing 'Expression'.
```

Beispiel 3: „Rechnen mit Variablen“

Code

Das Programm befindet sich in der Solution SWE3_EDER_UE06 unter der Solution: „Rechnen mit Variablen“.

Lösungsansatz:

Zu den Ausführung des Beispieles 1 wurde hier wiederrum als Prämisse die Verwendung von Variablen gefordert, welche sich mittels additiver Implementierung realisieren lies.

Folglich wurde im Parsermodul ein zusätzlicher Container gewählt in Form einer Map, im Sinne einer Namensliste, welche es zulässt, bei den arithmetischen Ausdrücken auch Variablen zu verwenden.

Als Testmodul, wurde zusätzlich ein Variablenmodul hinzugefügt.

Tests:

```
----------------------------------------
test_variables

TEST00:
Input: x = 3
Expected output: 3
Actual output: 3.000

----------------------------------------

TEST01:
Input: y = 0
Expected output: 0
Actual output: 0.000

----------------------------------------

TEST02:
Input: x = 5
Expected output: 5
Actual output: 5.000

----------------------------------------

TEST03:
Input: z = 5
Expected output: Not found.
Actual output: No variable found.

----------------------------------------

TEST04:
Input: x+5 = x+5
Expected output: 8
Actual output: 8.000

----------------------------------------

TEST05:
Input: x-5 = x-5
Expected output: -3
Actual output: -3.000

----------------------------------------
```

```
TEST06:
Input: x-1 = x-1
Expected output: 5
Actual output: 5.000

----------------------------------------

TEST07:
Input: x*3 = x*3
Expected output: 6
Actual output: 6.000

----------------------------------------

TEST08:
Input: x/2 = x/2
Expected output: 3
Actual output: 3.000

----------------------------------------

TEST09:
Input: x/0 = x/0
Expected output: Error.
Actual output: Error. Division by zero.

----------------------------------------

TEST10:
Input: 10/x = 10/x
Expected output: 2
Actual output: 2.000

----------------------------------------

TEST11:
Input: 10/x = 10/x
Expected output: Error.
Actual output: Error. Division by zero.

----------------------------------------
1

C:\Users\wae\Documents\FHOOE\3.WS\SWE3\swe.ws.2022\SWE3.UE
Press any key to close this window . . .
```