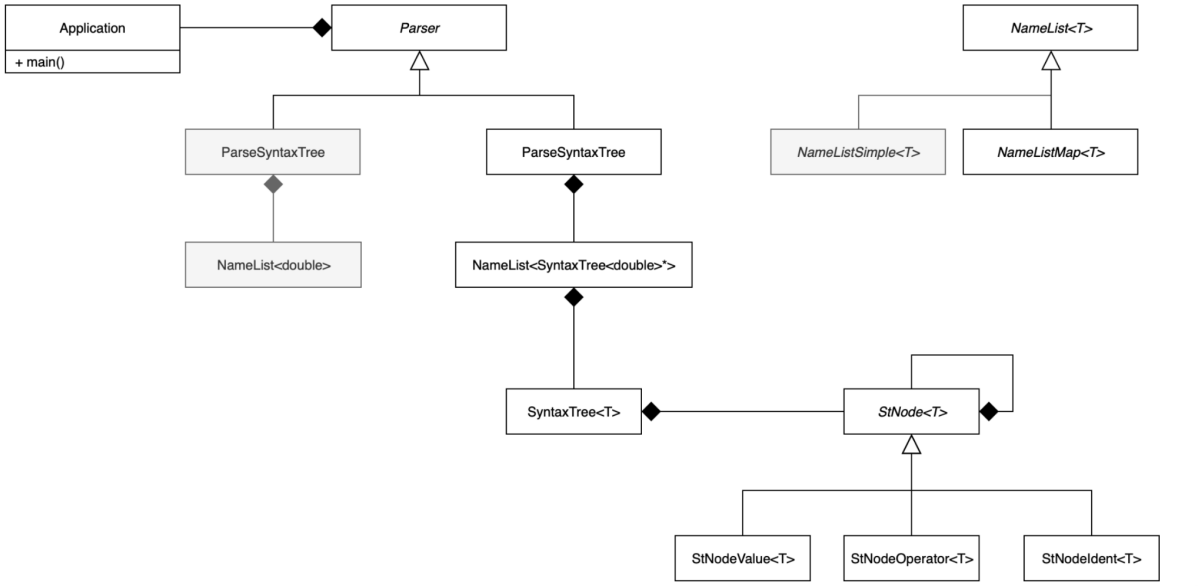
Protokoll Übung 7

benötigte Zeit: 10

Ausdrucksbaum

Lösungsidee

Die Idee ist, mittels des pfc:scanner einen Parser zu implementieren, der aus gültigen Sätzen (Grammatik in Angabe) einen Ausdrucksbaum erstellt. Dazu werden ein paar Klassen benötigt:



Parser ist abstrakt und bietet eine Signatur für die Methode parse an (pure virtual). Der Konstruktor wird in den geschützten Bereich verschoben, wodurch eine wiederverwendbare Base-Level-Implementierung für das Instanzieren aller Kind-Klassen gebildet wird. Zusätzlich werden in der Klasse Parser auch alle is_tb_ - Funktionen implementiert. Da diese immer gleich bleiben und auch in allen möglichen Spezialisierungen des Parsers verwendet werden. Die dazugehörigen parse_ - Funktionen, hängen aber sehr stark von der Implementierungsart ab, weshalb diese erst in der Kind-Klasse definiert werden.

ParseSyntaxTree ist eine Spezialisierung des Parsers. In den parse_ - Funktionen wird anhand der vordefinierten Grammatik der Scanner durchlaufen und dabei wird ein Syntax-Tree aufgebaut. Diese Implementierung verwendet die Klasse NameListMap<SyntaxTree<double>*>.

NameListMap<T> dient dazu, einen Identifier und seinen zugehörigen Syntax-Tree zu speichern und bei späterer Verwendung den Wert abfragen. Hierfür wird eine Map verwendet, als Key wird eine Zeichenkette verwendet, Value ist ein generischer Typ.

Syntax-Tree<T> enthält den den Wurzel-Knoten des Baumes welcher vom Typ StNode<T> ist. Zusätzlich kann ein Baum evaluiert werden, in dem er durchlaufen wird und Werte/Zwischenergebnisse durch die Operatoren verknüpft werden.

StNode<T> dient als polymorphe Basisklasse und speichert neben das linke und rechte Kind. Weiters bietet es die Signatur der Methode evaluate welche in den Kind-Klassen zu auswertung implementiert wird.
Für diese Klasse gibt es drei Spezialisierungen: St NodeValue, StNodeOperator und StNodeIdent.

- StNodeValue speichert einen Wert von Typ T welcher durch die evaluate-Methode retourniert wird.
- StNodeOperator enthält eine der fünf Operationen [+, -, *, /, ^] und verknüpft in der evaluate-Methode den linken und rechten Teilbaum damit.
- StNodeIdent speichert eine Zeichenkette, welche ein Variable/Identifier symbolisiert. In der evaluate-Methode wird in einer NameList der zum Identifier passende Syntax-Tree ausgewertet und das Ergebnis wird retourniert.

Implementierung

siehe: "/expression_trees"

Testfälle

Fall 1

```
Test: output of simple number
Testing: print(3);
expected: 3
actual: 3
Result: 1
```

Fall 2

```
Test: output of expression
Testing: print(12 * 3 + 20 / 5);
expected: 40
actual: 40
Result: 1
```

Fall 3

```
Test: output of nested expression
Testing: print(4 / (2 * (3 - 1)));
expected: 1
actual: 1
Result: 1
```

Fall 4

```
Test: assign a simple number to a variable and use it in output
Testing: set(x,3);
print(x/2);
expected: 1.5
actual: 1.5
Result: 1
```

Fall 5

```
Test: assign an expression to a variable and use it in output
Testing: set(x,12 * 3 + 20 / 5);
print(x/5);
expected: 8
actual: 8
Result: 1
```

Fall 6

```
Test: assign nested expression to var and use it multiple times in output
Testing: set(x,4 / (2 * (3 - 1)));
print((x * 5) / (x / 2));expected: 10
actual: 10
Result: 1
```

Fall 7

```
Test: implementation of exponent
Testing: set(x,4);
print(x^2);expected: 16
actual: 16
Result: 1
```

Fall 8

```
Test: implementation negative number
Testing: set(x,-4);
print(5 - x);expected: 9
actual: 9
Result: 1
```

Fall 9

```
Test: implementation of negative expression
Testing: set(x,12.5);
print(0 + -(x*3));expected: -37.5
actual: -37.5
Result: 1
```

Fall 10

```
Test: assign and reassign a value to var
Testing: set(x,12.5);
print(x);
set(x,5);
print(x);expected: 12.5, 5
actual: 12.5, 5
Result: 1
```

Fall 11

```
Test: assign an expression to a var that uses an unassigned var
    should work because the lookup for the variable happens when evaluating
Testing: set(x,y^2 - 4);
set(y,6);
print(x);expected: 32
actual: 32
Result: 1
```

Fall 12

```
Test: use unassigned var in output
Testing: set(x,y^2 - 4);
print(x);expected: Variable 'y' not found
Variable y was not found!
Result: 1
```

Fall 13

```
Test: use unrecognised keyword
Testing: zet(x,y^2 - 4);
print(x);expected: Error parsing 'program'
Error parsing 'program'
Result: 1
```

Fall 14

```
Test: division by zero
Testing: print(7 / 0);expected: Division by Zero
Division by Zero!
Result: 1
```

Fall 15

```
Test: division by zero through variable
Testing: set(x,0);
print(7 / x);expected: Division by Zero
Division by Zero!
Result: 1
```

Fall 16

```
Test: invalid format of print statement
Testing: print (7 / x, 1);expected: Expected ')' but have ...
Expected 'right parenthesis' but have {{comma,ts,4}}.
Result: 1
```

Fall 17

```
Test: invalid format of set statement
Testing: set(x);
expected: Expected ',' but have ...
Expected 'comma' but have {{right parenthesis,ts,14}}.
Result: 1
```

Fall 18

```
Test: invalid format of expression
Testing: set(x, / 4 5);
expected: Error parsing 'expression'
Error parsing 'expression'
Result: 1
```

Fall 19

```
Test: define multiple variables and print every (subtree) of each variable
Testing: set(x,y^2 - 4);
set(y,6+4);
print((x * 5) / (x / 2));
x:
     4
 -
          2
     ^
          y
```

```
y:
     4
 +
     6
```