02332 Compiler Construction Mandatory Assignment: A Simple Interpreter

8/10 2020

Freja Nørgaard Jensen s195473
Oliver Comprés Poulsen s185112

WEEK 3

Task 1: **Subtraktion og division**

Vi har først defineret de forskellige expressions i g4 filen. Dernæst har vi brugt java filen til at skabe noget kode, som sørger for at programmet kan læse og udføre de forskellige kommandoer. Vores kode fra de to filer ses nedenfor. Plus og minus er defineret i samme variabel, hvilket gange og divider også er. Dette gør, at man nemmere kan angive den rigtige forrang.

impl.g4

```
expr :  e1=expr s=GangeDividerOperator e2=expr # GangeDividerOperator
    | e1=expr s=PlusMinusOperator e2=expr # PlusMinusOperator

PlusMinusOperator: ('+'|'-');
GangeDividerOperator: ('*'|'/');
```

main.java

```
    public Double visitPlusMinusOperator(implParser.PlusMinusOperatorContext ctx){
        if (ctx.s.getText().equals("+"))
        return visit(ctx.e1) + visit(ctx.e2);
    else
        return visit(ctx.e1) - visit(ctx.e2);
    };

    public Double visitGangeDividerOperator(implParser.GangeDividerOperatorContext ctx){
        if (ctx.s.getText().equals("*"))
        return visit(ctx.e1) * visit(ctx.e2);
    else
        return visit(ctx.e1) / visit(ctx.e2);
    };
```

Task 2: Conditional branching, For-loops og Arrays på papir

Nedenstående tekst er en udvidelse til g4 filen. De skal angives som kommandoer, der gør det muligt at gøre brug af hhv. while loops, if statements, for loop og arrays.

```
'while' '('c=condition')' p=program                          # WhileLoop
'if' '('c=condition')' p=program                          # IfStatement
'for' '('x=ID '=' e=expr  '..' e2=expr ')' p=program    # ForLoop
x=ID '[' e=expr ']' '=' e1=expr ';'                       # Array
```

Nedenunder ses et eksempel på to betingelser, som også skal implementeres i g4 filen. På samme måde laves resten af betingelserne.

```
e1=expr '>=' e2=expr                                      # SmallerEqualThan

e1=condition '&&' e2=condition                                # And
```

WEEK 4

Task 3: Implementation af af task 2 i g4 filen. Checke at de kan accepteres af parseren. Check rangfølgen.

Vi har implementeret kommandoerne fra task 2 som en del af vores grammar, og har tjekket at alle vores implementationer fungerer som de skal. I impl_input.txt har vi skrevet de forskellige testcases og hvad de hver især skal printe i konsollen for at bekræfte at de virker. Som eksempel ses nedenunder vores test case 2 og den tilhørende terminal besked.

impl_input.txt

```
// Testcase 2:
// Tensten fungerer, hvis konsolen printer 1
n = 4;
m = 6;

if(n<=6){
  output 1;
}
```

Terminal

```
frejajensen@LAPTOP-JQ9IQF72:~/compiler-project$ make test
java main impl_input.txt
1.0
```

I et udtryk evalueres operatorerne efter forrang. Det ses også i vores test cases, at det har lykkedes os at få implementeret dette. Det er rækkefølgen af de forskellige udtryk i grammar filen, som afgør hvilken rækkefølge operatorerne bliver læst i.

WEEK 5

Task 4: Implementation af visit metoder til de forskellige kommandoer fra task 2 i java filen
Visit metoderne implementeret i *main.java* filen, og her ses et eksempel på hvordan henholdsvis for loopet og arrayet er implementeret.

```
public Double visitForLoop(implParser.ForLoopContext ctx){

    double v = visit(ctx.e);
    env.setVariable(ctx.x.getText(),v);

    for (double i = v; i <= visit(ctx.e2); i++) {
        env.setVariable(ctx.x.getText(),i);
        visit(ctx.p);
    }
    return null;
}

public Double visitArray(implParser.ArrayContext ctx){
    double s = visit(ctx.e);
    double v = visit(ctx.e1);
    env.setVariable("#" + ctx.x.getText()+s,v);

    return null;
}
```