Örebro University

Compilers and Interpreters

Assignment 1

Author:

Linus Baumgärtner
lbaumgaertner@culba.de

September 23, 2022

Part A

1. Where was the problem?
   - in line 247 in *abscissa.cpp*
2. Why did the program crash, and how did you find that reason?
   - The program crashes because of an access to a wrong address, the function *anslutningspropp* is called with **terminalhantering = 0* which is a faulty address for the int pointer *terminalhantering*, found by debugging with gdb
3. What did you do to fix the problem?
   - comment out the line 247 in *abscissa.cpp* that didn't work

Part B

4. Which of the phases and other parts of a compiler are present in the 2.9 program?
   - The program contains the phases lexical analyzer (scanner) and syntax analyzer (parser). Also there is a semantic action contained that prints out the numbers and the operators in postfix notation.
5. How are they implemented?
   - There is one function *lexan* for the lexical analyzer that scans the next char, saves the token value and then returns the token type. The parser is implemented as a predictive recursive-descent parser. There is one function for each non-terminal and a match function to ensure the right syntax. As a semantic action after each number or operator the token value is outputted.
6. Which are missing?
   - Semantic Analizer, Intermediate Code Generator, Code Optimizer, Code Generator, Machine Dependant Code Optimizer
7. If you were to modify the 2.9 program so it actually calculates the values of the expressions, and not just prints out postfix code, how would you do that?
   - To calculate the values you can use a stack machine on the generated postfix output:
     - push numbers in top of stack
     - if operator: take two top numbers and push the result of the operation

Part C

8. Does the macro work? (Addendum: Also show the macro!)

   Listing 1: Macro for factorial

   ```
   1 #define FACTORIAL(n) (n==0 ? 1 : n*FACTORIAL(n-1))
   ```

   - No, it does not work.
9. Why, or why not?
   - The self reference is not considered a macro call to prevent an infinitely large expansion. So the preprocessing result of e.g. "FACTORIAL(3)" is just "(3==0 ? 1 : 3*FACTORIAL(3 -1))".
10. If it doesn't work: Explain how the C preprocessor would have to be modified for the macro to work!
    - The preprocessor would have to expand the self reference and evaluate the calculation of the values to reach the stopping criterium and prevent a infinitely large expansion.