Project 2 Summary Report

Jade Pearl

CMSC 430: Compiler Theory and Design

Professor Gregory Williams

Due Date: April 9th, 2024

Project Approach

I went through the course learning resources for the weeks 3 and 4 to further understand syntactical analysis. I downloaded all required materials for project 2 and prepped everything in VSCode like I did with the previous project, again copying all given test case files into the skeleton code folder to test as I progressed. The Project 2 Approach was a helpful outline but some parts of it were a little vague in what I needed to do, so those parts required a lot more critical thinking than some others. While going through the approach, I also used the grammar written in the project requirements as a guide and example to what I needed to change in my parser and what I needed to leave alone. The issues I ran into were purely logical and I was able to work through them for the most part. The hardest part of it was implementing the rest of the binary operators like the exponent, remainder, and unary minus operators. Implementing it with the order of precedence was difficult because I had to make more productions which I was not sure how to adjust the pre-existing productions in the parser to fit in the new ones. I got feedback from 2 classmates on what I needed to change about my approach and it helped me eventually get on the right track. After that obstacle, it was easier for me to understand and I was able to implement the relational operators | and ! in a similar way which took me a lot less time to solve. I then implemented the error handling by following the rest of the approach and things went smoothly from there.

Test Cases

For this project, I used all provided test cases while progressing through the assignment to make sure the compiler was working as intended. In addition, I created three more test cases which followed the criteria for the assignment: one test case with no errors, another with multiple errors on one line, and a third containing all lexemes from the flex file.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Results | Actual Results | Pass/Fail |
| example1.txt: tests all grammar productions including the added binary operators and relational operators | Compiles successfully | Compiles successfully | Pass |
| example2.txt | Returns that there are 3 syntax errors | Returns 3 syntax errors: one for an error in the function\_header, one for variable production, and one for the case production | Pass |
| example3.txt | Contains 6 random syntax errors | Contains 6 syntax errors | Pass |

Here are the screenshots for each additional test case I created:

A screen shot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A computer screen shot of a program

Description automatically generated

Lessons Learned and/or Possible Improvements

This project has improved my skills in syntactical analysis and implementation of grammar productions. Having predefined grammar productions and having examples of changes in the project requirements helped me reason through making the new grammar productions that needed to be added. This helped me improve on creating new grammar productions following a particular set of rules by which I want the compiler to abide. I had struggled implementing the operator productions because they had to be separated based on order of precedence and that was a new concept to get used to. I hope to bring the knowledge I have learned from this project with me in the future so that creating a language with a compiler will be an easier endeavor. I also hope to learn more about semantic analysis soon so that I can implement the last part of constructing this compiler.