Project 4 Summary Report

Jade Pearl

CMSC 430: Compiler Theory and Design

Professor Gregory Williams

Due Date: May 7th, 2024

Project Approach

I went through the course learning resources for the weeks 7 and 8 to learn about semantic analysis which is what the Project 4 is all about. I downloaded all required materials for project 4 and set everything up in VSCode, again copying all given test case files into the skeleton code folder as I did previously. The project 4 approach document helped a lot more this time than in previous projects. This project was easier than project 3 by far but there were still parts that were unnecessarily hard because I don’t feel like I learned enough about this from the course material. Everything from the IF statement checking and onward was very difficult and I could not figure them out on my own without classmate feedback. Even then, I still got errors that no one else had gotten and no way to figure out what was wrong with my code, so it took a lot of time to work through with the insufficient amount of information I got from the course materials.

Test Cases

I used all provided test cases for this class but could not make any more since I could not think of different test cases that would test the same exact thing without being repetitive. The provided test cases seemed to cover all bases of the assignment.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Results | Actual Results | Pass/Fail |
| Semantic1.txt | Variable init type mismatch | Variable init type mismatch | Pass |
| Semantic2.txt | When type mismatch | When type mismatch | Pass |
| Semantic3.txt | Switch Expression not integer | Switch Expression not integer | Pass |
| Semantic4.txt | Case type mismatch | Case type mismatch | Pass |
| Semantic5.txt | Integer type required | Integer type required | Pass |
| Semantic6.txt | Undeclared scalar | Undeclared scalar | Pass |
| Semantic7.txt | Undeclared list | Undeclared list | Pass |
| Semantic8.txt | List element types do not match | List element types do not match | Pass |
| Semantic9.txt | List type does not match element types | List type does not match element types | Pass |
| Semantic10.txt | List subscript must be integer | List subscript must be integer | Pass |
| Semantic11.txt | Char literal comparison error | Char literal comparison error | Pass |
| Semantic12.txt | 2 arithmetic operators requires numeric types errors | 2 arithmetic operators requires numeric types errors | Pass |
| Semantic13.txt | Remainder operator requires integer | Remainder operator requires integer | Pass |
| Semantic14.txt | If-Elseif-Else type mismatch | If-elseif-else type mismatch | Pass |
| Semantic15.txt | Fold requires numeric list | Fold requires numeric list | Pass |
| Semantic16.txt | Illegal narrowing variable init | Illegal narrowing variable init | Pass |
| Semantic17.txt | Illegal narrowing function return | If-Elseif-Else type mismatch | Fail |
| Semantic18.txt | Duplicate scalar and duplicate list | Compiled successfully | Fail |
| Semantic19.txt | 7 semantic errors of various types | 8 semantic errors | Partial pass |
| Valid1.txt | Success | Success | Pass |
| Valid2.txt | Success | Success | Pass |
| Valid3.txt | Success | Success | Pass |

Here are the screenshots for each test case:A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated  
A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated  
A screenshot of a computer program

Description automatically generated

Lessons Learned

This project deepened my understanding of how semantic analysis interacts with error handling while creating a compiler. This required very deep thought and programming knowledge on the C++ side of things. It was helpful that part of the project was implemented in the skeleton code before starting, that way we could see an example of how to approach writing the other semantic statements. I hope now that I have finished this series of projects, I could better learn some day how to develop my own compiler for things I might implement such as my own game engine.