Project 2

Introduction

This project completes the final step of the compiler front end. From `lexcial` to `syntax` and finally to `semantic`, we've done everything we need for the front end of the compiler, and the next step is to generate the intermediate code to compile.

Basic Function

1. 15 error types were identified
2. create symbol table for variable and function

Method

1. Create symbol table to look up symbols
2. handle the syntax sentences and give a type checking

Program Structure

1. main.c
    - - Program entry, read in the file, call BISON internal function for syntax analysis, and then do a semantic analyze.
2. tree.h with the tree.c
    - Defined node types of tree and define the function to insert node and the print syntax
3. lex.l
    - Conduct lexical analysis
4. syntax.y
    - Perform grammatical analysis
5. semantic.h with the semantic.c
    - The Symbol is generated and type checked

Lexical Analysis

Basically, the match is performed using the identifiers given by the documentation, and the unspecified identifiers are uniformly specified as a token called LEX_ERROR to avoid undesirable results from parsing.

In this project, I just use the code last time and without any changes;

Syntax Analysis

Syntax parsing basically refers to the expressions in the given document, adding error handling where possible to print syntax errors, and it should be noted that the relationship between nodes needs to be handled during the matching process.

In this project, I just use the code last time and without any changes;

Syntax Tree

The syntax tree is realized by a simple multi-tree, which implements the insertion method and prints the syntax tree through a simple preorder traversal.

In this project, I just use the code last time and without any changes.

Symbol Table

The Symbol determines if there is an error by traversing the Syntax Tree while performing a type check.

Test

I passed the 15 tests given and 10 self-written tests.

It is worth mentioning that in the given sample, tend to repeat an error for a mistake, for example, in the 7th sample, it gives the type 7 and type 5 two mistakes, however, the type 5 error is caused by the type 7, so the type 5 is not a mistake essentially, if we need to modify the code, we also just to repair the type 7 this mistake, so I only print type 7 mistake rather than print all.