Assignment 3 Documentation

Mach Bui
Rian Luzio
Jason Liu

1. Problem Statement

Utilizing the lexical analyzer from Assignment 1, the team was tasked with constructing a basic syntactical analyzer, the next and second step in the compilation process. We were allowed to use any top-down parser which included a RDP, predictive recursive descent parser, or a table driven predictive parser. Whichever approach was chosen had to remove left recursion and, if necessary, use left factorization. At the end, the analyzer should print a file of tokens, lexemes, the production rules used, and the assembly code instructions. To get more points, the team had to implement Scope Checking, Type Checking, Testing, or Error Handling.

2. How to Use Our Program

Our program's development took place in Visual Studio 2019

To run the program, place Lexer.cpp, Syn.cpp, Header.h, and the desired input file together in the same folder location. Compile and run the program in Visual Studio. When prompted, input the name of the input file to be used and tested. The program will take this file and process it lexically and syntactically analyze it.

Or double click Synt_Analysis_and_Lexer_Proj3.exe

3. Design of Our Program

R1: E -> E + T
R2: E -> E - T
R3: E -> T
R4: T -> T * F
R5: T -> T / F
R6: T -> F
R7: T -> (E)
R8: F -> i

The program consisting of 3 files Lexer.cpp, Syn.cpp, and Header.h utilizes a lexical analyzer to help in the tokenizing process. While processing the tokens, the lexical analyzer simultaneously syntactically analyzes the code to ensure correct formatting. After every token finishes being syntactically analyzed, assembly code listing will be printed.

Syn.cpp

This program file contains multiple functions. The id_search function searches the table for an id to read that a variable has been called and is present. The id_index function, that searches for the index, gets the memory location of the declared variable. In addition, the convert_char function reads in the character gained from the text file and returns a corresponding index. Function F is involved when the variable called is deemed an identifier, having different cases for integers, booleans, and floats. Function Tprime is involved when the multiplication and the division operators are called. Function T calls function F and function Tprime. Function Eprime is involved when the addition and subtraction operators are called. Function E call function T and function Eprime. Function A is the recursive descent parser that will go through the file to make sure it's valid syntax; if not, it'll throw an error. The main function builds the starting stack and calls the previous functions to undergo its process.

Lexer.cpp

This program file contains four functions that will determine the keyword of the inputted text file. These functions will see if the variable is a keyword, a number, an operator, or a separator. This program file also contains the update_col function that will update the column, and the open_file function to select which text file to scan, requiring a user to input the name of the desired text file. Finally, the last function is the lexer function, which reads the text file and creates a table that organizes the code into keywords, displaying the key word and the variable in question itself.

Header.h

This program file contains the state table, the arithmetic table, and the character table. The state table is a record of values that decides what the current character should be sorted to, as a keyword. The arithmetic table's function is to implement the starting stack. The character table is a record of specific words and characters that the lexer will use to scan if the current variable is a keyword, separator, or operator, and assign them as such.

4. Limitations

1)The program is unable to interpret characters that do not exist in the state table columns.
2)It only works in the specified confines of the assignment's outline
3)If an invalid character is imputed into the program, a crash may occur and the program would need to be restarted.

5. Shortcomings

1)Running the program in linux produces some bugs so the best environment is in VS2019.