Assignment 3 Project documentation

Group: Estrella Mejia, Khoi Nguyen, Marcos Sanchez-Cruz, Steven Chiang, Thomas Tran

Problem Statement

The goal of Assignment 3 is to generate intermediate code for the simplified version of our grammar from our previous assignment. The grammar we're using is the same used in class, with the exception that it is a more simplified version and it has no <Function Definitions>. On top of that, no "real" type is allowed. In the program, a symbol table will hold all of the identifiers declared in the program and these identifiers will be accessed by the symbol table handling procedures. A symbol table is a database for symbols that occur during the compilation process. It also provides sets of procedures including lookup(), insert(), remove(), and list() among others.

How to use your program

Instructions on executing the Syntax Analyzer program on a Windows Operating System:

- Download and Install Python from www.python.org/downloads
- Run the Python installer and follow the onscreen instructions.

Downloading and extracting the files:

- Create a folder with a name of your choosing.
- We will call it Intercode. This is where we will be extracting the zip.
- Download and unzip the Assignment3 zip to the created folder.
- Files in the folder are main.py, SyntaxAnalyzer.py, LexicalAnalyzer.py, intercode.py, sample2.txt

Running the code:

Option 1:Using Command Prompt

- Run Windows command prompt and navigate to the directory where the files were extracted.
- If in the c:\ directory run the following command:
  - cd \ cd Intercode
- Once in theIntercode Folder use the following command to run the program: py main.py

Option 2: Using Visual Studio Code

- If you have not already done so, Install VS code: Install the Python Extension for Vs Code from the Visual Studio MarketPlace
  - https://marketplace.visualstudio.com/items?itemName=ms-python.python
- Open Visual Studio Code and click on open Folder and navigate to the folder containing the program files.

- Once the files show up in the Explorer pane on the left side, click on it to open and click on the green run button on the top right corner in the file tab section.

After running program

- Once the code is running the program will ask for a file to analyze.
    - Enter sample2 which is included in zip file
    - Or enter the name of the file you would like to have analyzed
- Then you will be asked which analyzer you would like to use
    - For assignment 3 enter 3 for intercode analyzer

Design of your program

- To make the program more organized and easier to read, the whole program is split into 4 different files.
    - The lexical (LexicalAnalyzer.py) and syntax analyzer (SyntaxAnalyzer.py), the intercode (intercode.py), and the main file (main.py).
    - The main file will use all 3 classes to produce the expected output.
- The input that the program asks for needs to be a text file, as the program will view lines with the "!" mark at the beginning and end of the line as comments.
    - The input text file will be read by main.py and separate each lexem into a list which is pushed into the lexical analyzer.
- A specific state is assigned to each lexeme read by the lexical analyzer, the assigned state determined by the lexeme that sets them to a token.
- The syntax analyzer separates each token and lexeme line by line into different dictionaries and lists.
    - The analyze function accepts these dictionaries and lists, adding rules to each lexeme using the "set_up" functions.
    - This function will continue to loop through the entire token list and the rules appended to each token will be based on the current token
- The main file proceeds to print out each of the rules line by line
    - Each line prints out the lexeme, token, and rules set by the class in SyntaxAnalyzer.py
- intercode.py will take in the list of lexemes and solve basic arithmetic operations.
    - This is done by looking through the list of lexemes and checking if they are the same as certain operators such as ";" or "+"
    - An instruction from a list of instructions will be then appended depending on the operator found, making note of the address as well.
- The construct table function from intercode.py takes in the token list and stores important values that will be used to print the table.
    - The table will contain the lexeme, its location in memory, and its token type.
- After constructing the symbol table, it is then printed by main.py

Any Limitation

- Boolean types are not accounted for in our lexical analyzer
    - Therefore we don’t account for it in our intercode.

Any shortcomings

- There is no error handling implemented into the program.
    - No error message is generated if there are any syntax errors.
- Other than that, we've been able to meet the rest of the requirements of the assignment and kept our shortcomings to a minimum.