Programming Assignment 4
A Parser[1]

Assignment spec may change depending upon discussion in the class, this is a working doc

Release Time	Due Date
November 7, 2023	**November 21, 2023**

Objectives

To implement a parser for a self-designed Cminus programming language (a tiny simplified subset of the C language) using PLY (Python Lex-Yacc).

Problem Specification

Language Specification: The self-designed language is called **Cminus** and it has the following features:

- It is a case-sensitive language that uses ASCII characters.
- It supports only two data types: basic type int and a standard data type float.
- It supports arithmetic, logical, relational, and assignment operators.
- It supports if-else, while, and compound statements for control flow.
- It supports single-line and multi-line comments that start with /* and end with */
- It supports identifiers that start with a letter and can contain alphanumeric characters.
- It supports literals that are enclosed in single quotes for strings, and supports only decimal notation for floating point numbers.
- It supports keywords that are reserved for the language and cannot be used as identifiers. The keywords are: int, float, if, else, exit, while, read, write, and return.

Full specifications of Cminus were provided in the attachment as a separate document.

A lexical analyzer for Cminus was implemented in your HW3 using PLY - Ply is a Python library that provides a set of tools to write lexical and syntactic analyzers. You will now add the parsing code to generate parse tree (abstract syntax tree - AST) of the given Cminus code snippet.

You can use the following hints to write the parser rules:

- Use the p_<name> function syntax to define a rule. The <name> part should match the corresponding non-terminal in the grammar.

- Use the p[0], p[1], …, p[n] syntax to access the symbols on the right-hand side of the rule. The p[0] symbol is the result of the rule.

[1] Assignment idea courtesy Steve Carr @ WMU-CS

- Use the p.slice[n] syntax to access the token object of the n-th symbol. The token object has attributes such as type, value, lineno, and lexpos.
- Use the p_error function to handle syntax errors. The function takes a single argument which is the token object where the error occurred. You can print an error message and skip the rest of the input.
- Use the yacc function to create the parser object. You can pass the lexer object and the start symbol as arguments.

For more information and examples, you can refer to the PLY documentation or the PLY tutorial.

For your homework, you should write a report that explains your design choices, your implementation details, and your testing results. You should also include the source code of your parser and some sample inputs and outputs.

Example Input:

An example input source code snippet written in Cminus could look like this:

```
/* This is a multi-line

          comment */

int c;

if (c == d) { IF = a; }
```

Example LEX Output:

An example output list of tokens produced by the lexical analyzer could look like this:

```
[(COMMENT,' ….'), (KEYWORD, int), (IDENTIFIER, c), (SEPERATOR, ;),
(KEYWORD, if), (SEPARATOR, (), (IDENTIFIER, c), (LOGIC-OP, ==),
(IDENTIFIER, d), (SEPERATOR, )), (SEPERATOR, {), (IDENTIFIER, IF),
(ARITH-OP, =), (IDENTIFIER, a), (SEPERATOR, ;), (SEPERATOR, })]
```

Example YACC Output:

A level traversal of your parse tree, separate each level by two blank lines, and separate each node within a level of the parse tree by " : ".

As time permits – We will draw the AST or a traversal of the parse tree.

Congratulations! You're well on your way to becoming a compiler designer! 🔍

Notes

References

- ChatGPT-like tools
- GeeksforGeeks - https://www.geeksforgeeks.org/introduction-of-lexical-analysis/#
- Lex - https://en.wikipedia.org/wiki/Lex_(software)
- Wiki - https://en.wikipedia.org/wiki/Lexical_analysis
- YACC: https://en.wikipedia.org/wiki/Yacc
- Parsing Wiki - https://en.wikipedia.org/wiki/Parsing
- flex, bison, jflex, ply

If you have creative ideas for extensions or making it more interesting, run them by the course staff, and we'd be happy to give you guidance!

Design Requirements

Code Documentation

For this assignment, you must properly document your code and use good software development practices.

Github

Use github to store your repository. Use good revision-cotrol-system practices as you develop various pieces of the search engine.

Testing

Make sure you test your application with several different values capturing different cases, to make sure it works.

Assignment Submission

- Generate a .zip file that contains all your files, including:
  - Source code files
  - any input or output files
- Don't forget to follow the naming convention specified for submitting assignments
- You will also show execution of your application to grader / instructor. They may give you a test case or two on the spot.