CS75 Project 1: Lexical Analyzer for C- -

Spring 2007, Meeden

Parts 1-3: Due by midnight Monday, February 5

Part 4: Due by midnight Monday, February 12

For this project I would like each student to work individually. However, feel free to help each other and to discuss how to go about solving aspects of the project.

Project parts

Implementing the actual code is straight forward once you have developed a correct DFA that accepts all the tokens in the language. Therefore, you will complete parts 1-3 below and turn them in before beginning on the implementation in part 4.

1. List the set of tokens to be returned by your lexical analyzer.
2. Define regular expressions for this set of tokens.
3. Derive a single DFA from your regular expressions.
4. Implement the DFA in a Python program similar in structure to the scanner example provided for the infix to postfix translator we discussed in class. Specifically, each state of your DFA should be implemented as a method in your `LexicalAnalyzer` class.

Specifications

- The lexical analyzer should provide a method called `getToken` that returns the next token (and any associated value) read from a file.
- Keyword tokens should be distinct from identifier tokens, and different identifier tokens should be distinct from one another. These distinctions are typically made with the help of a symbol table.
- The lexical analyzer should convert integer tokens, including characters, into numeric values. For example, `'a'` should be read as an integer token with value `97`, the ASCII code of the character. Use `ord` in Python to determine the ASCII value of a character.
- The lexical analyzer should ignore comments which can be of the form:

```
/*
  a comment
  that can go for several lines
*/
// a single line comment, where everything to the right is ignored
```

- The lexical analyzer should report any error it finds by line number and continue trying to scan the file.
- At the end of the file, the lexical analyzer should return a special `'done'` token.

Project deliverables

Do an `update75` to populate your `cs75/projects/1a` and `cs75/projects/1b` directories with starting point files. When you run `handin75` to turn in your code, the script will look here for your solutions.

- The file `tokens` should contain the list of tokens you designated in part 1.
- The file `regular-expressions` should contain the list of regular expressions you created to describe the tokens in part 2.

- Turn in a hand-written drawing of your DFA from part 3, or provide a computer-generated drawing in this directory.
- The file `scanner.py` should contain the implementation of your lexical analyzer for part 4.