CS 315
Project 1

Assigned: Oct. 22, 2018
Due: Oct. 30, 2018, 23:59

Lexical Analyzer for a Robot Programming Language

This semester's projects are about the design of a new language for programming robots. Assume that you are working in the software department of a company that builds robots. The hardware department constructs robots that can turn, move, grab/release objects, send/receive data to/from the master. The robot has also some sensors to perceive the environment. The sensors return floating point values. The robot can turn 1 degree, to left or right, at a time, and move ahead in steps of 1 mm. Your team is asked to design a Programming Language to program the robots constructed by the hardware department. People who purchase these robots may not be computer engineers; so it is important that the programming language is readable, writable and reliable.

Part A - Language Design (30 points)

First, you will **give a name** to your language and design its syntax. Note that the best way to hand in your design is its grammar in BNF form, followed by a description of each of your language components. The following is a list of features that are required in your language:

- Primitive functions: move, turn, grab, release, read data from a sensor given the sensor ID, send and receive data from/to another robot or master.
- Variable names,
- Assignment operator,
- Arithmetic expressions,
- Conditional statements (e.g., if-then, if-then-else)
- Loops (e.g., for, while),
- Components for function definitions and function calls,
- Comments.

You are encouraged to use your imagination to extend the list given above.

Do not panic! You will have a chance to do minor revisions on your syntax design for Project 2 (later in the semester). Language designs are almost never exactly right in the first iteration. Just try your best to make it as readable/writable/reliable as you can and keep your eyes open for what does and what does not work :)

Part B - Lexical Analysis (40 points)

In the second part of this project, you will design and implement a lexical analyzer for your language, using the lex tool available on all Unix style systems. Your scanner should read its input stream and output a sequence of tokens corresponding to the lexemes you will have defined in your language. Since at this stage you will not be able to connect the output to a parser, your scanner will print the names of the tokens on the screen. For instance, if we were designing a C like syntax, for the input

```
if ( answer == 2 ) { ...
```

the lexical analyzer should produce the output

IF LP IDENTIFIER EQUAL NUMBER RP LBRACE ...

Part C - Example Programs (30 points)

Finally, you will prepare test programs of your choice that exercise all of the language constructs in your language, including the ones that you may have defined in addition to the required list given above. Be creative, have some fun. Make sure your lex implementation correctly identifies all the tokens. The TA will test your lexical analyzer with these example programs along with other programs written in your language.

Do not panic! You are not required to write an interpreter or compiler for this language. Just write a few programs in the language you designed and make sure that the lexical analyzer produces the right sequence of tokens.

Groups

The project will be implemented in groups of two or three students. The members of the groups will be the same for this and the next project, in which you will write a parser for your robot programming language. Note that all members of a group will get the same grade.

Submission

- There are several parts that you will hand in.
    1. A project report including the following components:
        - Name, ID and section for all of the project group members.
        - The name of your language.
        - The complete BNF description of your language.
        - A paragraph explanation for each language construct (i.e. nonterminals) detailing their intended usage and meaning, as well as **all of the associated conventions.**
        - Descriptions of how nontrivial tokens (comments, identifiers, literals, reserved words, etc) are defined in your language. For all of these, explain what your motivations and constraints were and how they relate to various language criteria such as readability, writability, reliability, etc.
    2. Your lex description file
    3. The example program described above, written in your language
- Make sure your lexical analyzer compiles and runs on dijkstra.cs.bilkent.edu.tr . The TA will test your project on the dijkstra machine, and any project that does not compile or run on this machine will get 0 on Part-B.
- If you do not remember your password for your dijkstra account, contact Ahmet Göktaş (< lastname> @ bilkent.edu.tr) at BCC.
- Please email **all of the above items** to Duygu Durmuş <name>.<astname>@bilkent.edu.tr) before the due date. PDF format is preferred for the project reports. Late submissions will be accepted, with 20 points (out of 100) deduction for each extra day..

Resources

- Running lex and yacc on Linux systems (accessible in Bilkent Campus)
- The Lex & Yacc Page (dinosaur.compilertools.net)

Good Luck! - Have fun.