CS 315
Project 1

Assigned: Oct. 15, 2021
Due: Oct. 22, 2021, 23:59

A Programming Language for Drones and its Lexical Analyzer

This semester's projects are about the design of a new language for drones. Assume that you are working in the software department of a company that builds drones. These drones are used for spraying pesticides or fertilizers over grain or vegetable fields or fruit plantations. The hardware department constructs drones that are equipped with

- a compass to detect the heading of the drone (an integer value between 0 and 359), the direction of the drone, measured as the angular distance relative to north),
- and a barometer for measuring its altitude,
- controls for the motors,
  - to turn the heading to left or right for 1 degree,
  - to move the drone so that it climbs up or drops down with a speed of 0.1 m/s.
  - to move the drone so that it moves in the heading direction forward or backward with a speed of 1 m/s.
- a spray nozzle that can be turned on and off for spraying the chemical in its tank,
- connection to the base computer through wi-fi. The base computer may be a desktop or a mobile device.

The programs written in your language will be executed on the base computer, which will send the appropriate commands to the drone over the wi-fi connection.

Drone for Agricultural Spraying.

Traversing the whole field.

Images are from www.uavfordrone.com/agriculture-spraying-drone-2

In this project, you will design a simple programming language and its parser (in project 2) only.

Part A - Language Design (40 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 required in your language:

- variable identifiers
- assignment operator
- precedence, associativity of the operators
- expressions (arithmetic, relational, boolean, their combination)
- loops
- conditional statements
- statements for input / output
- function definitions and function calls.
- comments
- primitive functions for
  - reading the heading
  - reading the altitude
  - reading the temperature
  - vertically, climb up, drop down, or stop
  - horizontally, move forward, backward, or stop
  - turn the heading to left or right
  - turning on or off spray nozzle
  - connecting to the base computer through wi-fi

All of these features must be built-in in your language. Do not assume importing from a library.

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

You will have a chance to do minor revisions on your syntax design for Project 2, to be assigned later. 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 (30 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 Unix style systems. Your scanner should read its input stream and output a sequence of tokens corresponding to the lexemes 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 an output, similar to the following:

```
IF LP IDENTIFIER EQUAL_OP 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. The example programs should be extensive and readable.

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 can be implemented in groups of two or three students. The members of the groups will be the same for both parts of the project, this and the next. The members of a group can be from different sections.

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.
     - One 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.
     - Evaluate your language in terms of
       a. Readability
       b. Writability
       c. Reliability
  2. The lex description file.
  3. Example programs, written in your language. These example programs should be meaningful.
- 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 and/or run on this machine will get 0 on Part-B.
- Please upload **all of the above items** to Moodle (CS 315 (All Sections) Programming Languages) 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)
- Discrete Mathematics - Sets

Good Luck! - Have fun.