Simple Programming Language using Lex and Yacc

Number of Students per Group

3-4 Students

Requirements

It is required to design and implement a programming language using the Lex and Yacc compiler generating package.

The Project comprises the following

1. Design a suitable programming language; you may use an existing one. The important constructs to be considered are:
   - Variables and Constants declaration.
   - Mathematical and logical expressions.
   - Assignment statement.
   - If-then-else statement, while loops, repeat-until loops, for loops, switch statement.
   - Block structure (nested scopes where variables may be declared at the beginning of blocks).
   - Enums
   - Functions
2. Design a suitable and extensible format for the symbol table.
3. Implement the lexical analyzer using Lex.
4. Design suitable action rules to produce the output quadruples and implement your parser using YACC.
5. Implement a proper syntax error handler.
6. Build a simple semantic analyzer to check for the following:
   - Variable declaration conflicts. i.e. multiple declarations of the same variable.
   - Improper usage of variables regarding their type.
   - Variables used before being initialized and unused variables.
   - The addition of type conversion quadruples to match operators' semantic requirements, i.e. converting integer to real, …
   - Warning in case of if statement whose condition is always false.

Bonus: Code Playground GUI

For some bonus points (limited to compensate loss of point within the project, not extendable to any other year work of the subject), you're encouraged to design and develop a GUI (Any Platform) :

- Level 1 (input/output fields)
  - Should have input and output fields with "Compile" button.
  - User inputs the code in the input field (better be a code editor field e.g. https://codemirror.net/try/ supports line numbering) and sees the output quadruples in the output field.
  - Should also print the error if any.
  - Should show the final symbol table.
- Level 2 (Highlight the line with an error):

  Should highlight the line with error (if any) in the "user input" field after compilation.

- Level 3 (Print the Symbol table step-by-step):

  - Should execute the input step-by-step.
  - Should have a button to step to the next line
  - Should show the symbol table as part of the UI
  - Should update the symbol table every step if changed

The design is left for your creativity, but should include the components mentioned for each level. Kindly note that to receive the bonus points for a level, it should be complete and fully functioning with all the mentioned components. Completing each level (1 up to 3 **strictly**) increases your bonus points.

Project Phases

Phase I: In this phase, it is required to do the following:

- Implement the lexical analyzer using Lex.
- Implement the parser using YACC.
- Deliver your Lex and YACC files.

Phase II: In this phase, it is required to modify your implementations to include the following:

- Design a suitable and extensible format for the symbol table.
- Design suitable action rules to produce the output quadruples.
- Implement a proper syntax error handler.
- Build a simple semantic analyzer.
- Bonus Code Playground GUI

Deliverables

1. Source code of your project.
2. A Document that contains the following:
   - Project Overview
   - Tools and Technologies used
   - A list of tokens and a description of each.
   - A list of the language production rules.
   - A list of the quadruples and a short description of each. e.g. :

Quadruple	Description
JMP L	Unconditional jump to Label L
NEG V1, V2	V2= -V1

   - [BONUS] Code Playground GUI

Program evaluation

1. CLI works as expected, i.e. The program is to be fed by a source code file containing your language and produce the corresponding quadruples and the symbol table log saved to files.
2. Display the syntax errors that exist in your program as a response in the terminal (command prompt).
3. Display the semantic errors that exist in your program as a response in the terminal (command prompt).
4. Display the symbol table (save it to a comma-separated file “symtbl.log” after each change with “====” separation in between each log and the following).
   Example

```
sym_key, sym_val, decleration_line
x, 1, 12
y, 2, 13
z, 3, 14
=====
sym_key, sym_val, decleration_line
x, 1, 12
y, 2, 13
z, 3, 14
```

Note: This example illustrates ONLY the file format, NOT the fields of the symbol table itself. You should design your own symbol table according to the design of your language.

Evaluation Criteria

1. The correctness of your quadruples.
2. The Syntax error handling.
3. The Semantic error handling.
4. Project understanding for the whole team.
5. The document

Notes

1. Anything listed as optional will be considered a bonus.
2. Everything else mentioned is mandatory.
3. Any other semantic checks than the ones mentioned above will be considered a bonus.

Due Dates:

Phase I delivery: week 12 – 1 May 2023
Phase II delivery: week 14 – 15 May 2023