

Parser Document

Trevor Mee

03/16/2025

Project 2

COP4020 - Programming Languages

Table of Contents

Table of Contents.....	2
1 File Structure.....	3
2 Usage.....	3
2.1 Error Message Explanation.....	6

1 File Structure

The file structure for project 2 is as follows:

Root Directory (contains all files)

- scanner.hpp (scanner logic header file)
- scanner.cpp (scanner logic source file)
- parser.hpp (parser logic header file)
- parser.cpp (parser logic source file)
- main.cpp (contains main entry point for running program)
- a1.in, a2.in, a3.in, a4.in, a5.in, a6.in, a7.in, a8.in (input test files)
- Makefile
- run_script.sh (script to parse through a1.in-a8.in input files)
- Proj2 Parser Docuemntation.pdf (this file)

2 Usage

Please use g++ 11.4.0 for running this project. There are a couple of options when it comes to compiling and running this project.

Option 1: Manual Compilation

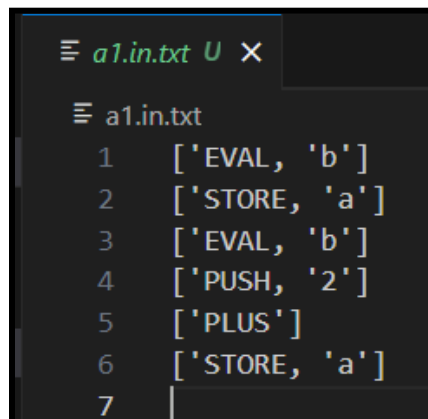
- 1) Navigate into the projects root directory
- 2) Type `make`. This will generate an executable named `proj2`
- 3) Then type `./proj2 <source_file>` to scan and parse a source file. For example, type `./proj2 a1.in` to scan and parse input file 'a1.in'.

If an input file is successfully parsed, you will see an output message to the console and an output file will be generated with the name '<source_file>.txt'.

The below image shows what output should be similar to on a successfully parsed input file to the console:

```
trevormee@trevor:/mnt/c/Users/trevo/source/repos/cop4020/p2$ ./proj2 a1.in
Compiling a1.in...
Success! The program is legal!
Generated RPN code written to a1.in.txt
```

The below image shows what the output file ('a1.in.txt' in this example) will look like when an input file is successfully parsed:



```
a1.in.txt
1  ['EVAL', 'b']
2  ['STORE', 'a']
3  ['EVAL', 'b']
4  ['PUSH', '2']
5  ['PLUS']
6  ['STORE', 'a']
7
```

If an input file is NOT successfully parsed (meaning that an error occurred), no output file will be generated. Instead, you will only see an output message from the console indicating that an error occurred. For example, the input file 'a4.in' contains an illegal redefinition of the variable b. The following image shows what output should be similar to on an unsuccessful attempt at parsing an input file:

```
trevormee@trevor:/mnt/c/Users/trevo/source/repos/cop4020/p2$ ./proj2 a4.in
Compiling a4.in...
>>> Error line 3: Illegal redefinition of variable b
trevormee@trevor:/mnt/c/Users/trevo/source/repos/cop4020/p2$
```

Option 2: Using run_script.sh

- 1) Navigate into the projects root directory
- 2) Type `make`. This will generate an executable named `proj1`
- 3) Type `chmod +x run_script.sh` to ensure that the script is executable
- 4) Type `./run_script.sh` to see the program automatically scan and parse input files `a1.in`, `a2.in`, `a3.in`, ..., `a8.in`

Output should be similar to:

```

trevormee@trevor:/mnt/c/Users/trevo/source/repos/cop4020/p2$ ./run_script.sh
Parsing file: a1.in
Compiling a1.in...
Success! The program is legal!
Generated RPN code written to a1.in.txt
-----
Parsing file: a2.in
Compiling a2.in...
>>> Error at line 2: Identifier cannot end with an underscore.
>>> Error line 2: Identifier expected
-----
Parsing file: a3.in
Compiling a3.in...
Success! The program is legal!
Generated RPN code written to a3.in.txt
-----
Parsing file: a4.in
Compiling a4.in...
>>> Error line 3: Illegal redefinition of variable b
-----
Parsing file: a5.in
Compiling a5.in...
Success! The program is legal!
Generated RPN code written to a5.in.txt
-----
Parsing file: a6.in
Compiling a6.in...
>>> Error line 5: id, if or while expected
-----
Parsing file: a7.in
Compiling a7.in...
Success! The program is legal!
Generated RPN code written to a7.in.txt
-----
Parsing file: a8.in
Compiling a8.in...
>>> Error line 4: Expected 'beginSym' but found 'identifier' with lexeme 'began'

```

2.1 Error Message Explanation

If an input source file does not contain any syntax errors, you will see the phrase “Success! The program is legal!” followed by another statement of the following form:

“Generated RPN code written to <source_file>.txt”. If an input source file does contain syntactic or semantic errors, an error message will be output to the console. Error messages for this project are of the following forms:

- “Error line “ <line_number> “: Expected” <expected_token> “ but found
““ <current_token> ““ with lexeme `” <current_token_type>
- “Error line “ <line_number> “: Illegal redefinition of variable “
<variable_name>
- “Error line “ <line_number> “: undefined variable <variable_name>