Project 1: Scanner and Parser

COP4020: Programming Languages

Trent Wells

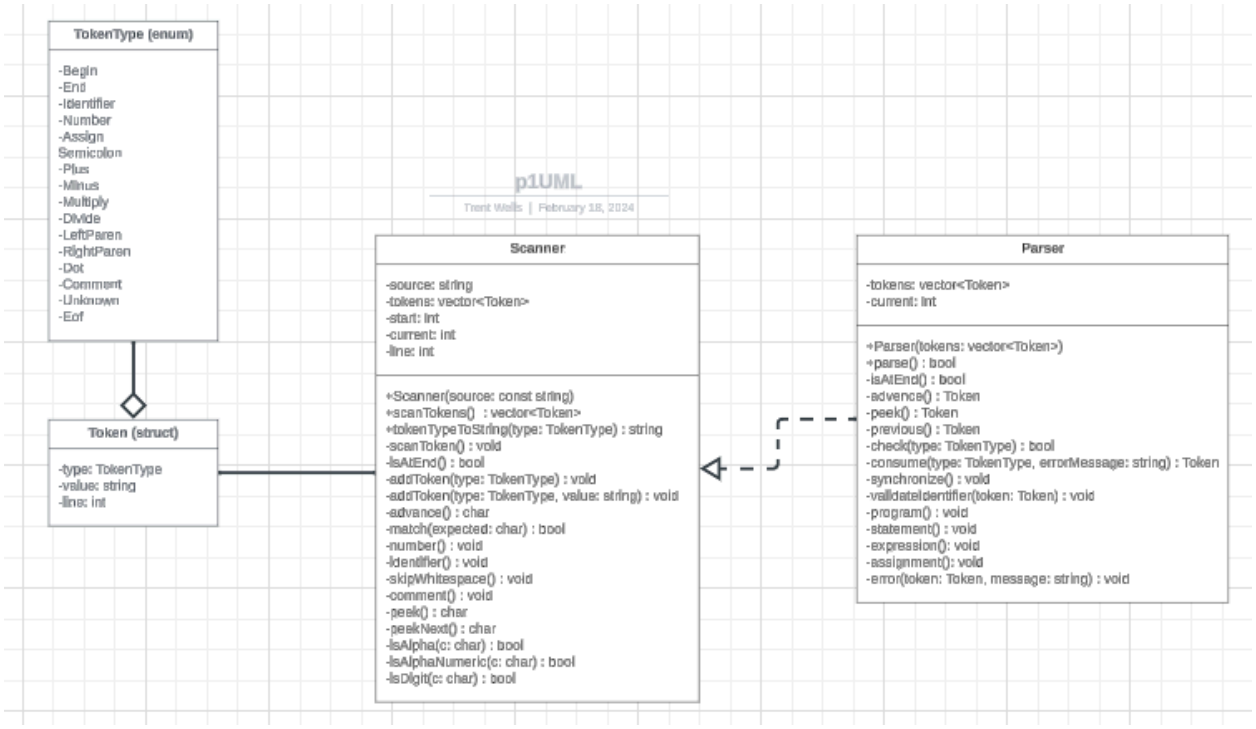
Lexeme Structure (found in p1Lexemes.xlsx):

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Lexeme description (RE or verbal)** | **Examples** | **Meaning** | **Token** | **Attribute 1** |
| 2 | Begin | begin | beginning of code | Begin | N/A |
| 3 | End | end | end of code | End | N/A |
| 4 | [a-zA-Z](?:_?(?![0-9])[a-zA-Z0-9])*[a-zA-Z0-9] | a, b_xy | identifier | Identifier | name |
| 5 | \d+ | 99, 25 | numeric literal | Number | numeric value |
| 6 | = | = | assignment operator | Assign | N/A |
| 7 | ; | ; | semicolon | Semicolon | N/A |
| 8 | + | + | addition operator | Plus | N/A |
| 9 | - | - | subtraction operator | Minus | N/A |
| 10 | * | * | multiple operator | Multiply | N/A |
| 11 | / | / | divide operator | Divide | N/A |
| 12 | ( | ( | left parenthesis | LeftParen | N/A |
| 13 | ) | ) | right parenthesis | RightParen | N/A |
| 14 | . | . | defines end of code along with “end” | Dot | N/A |
| 15 | ~ | ~ | comment denotation | Comment | N/A |
| 16 | N/A | N/A | unknown token | Unknown | N/A |
| 17 | eof | eof | end of file | Eof | N/A |

EBNF (found in p1Lexemes.xlsx):

| | A | B |
|---|---|---|
| 1 | **Nonterminal** | **RHS** |
| 2 | program | “begin” {assignment} “end” “.” |
| 3 | assignment | Identifier “=” expression “;” |
| 4 | expression | Term {(“+” \| “-”) term} |
| 5 | term | Factor {(“*” \| “\”) factor} |
| 6 | factor | Identifier \| number \| “(“ expression “)” |
| 7 | identifier | letter {letter \| digit} [letter \| digit] |
| 8 | letter | “a” \| “b” \| … \| “z” \| “A” \| “B” \| … \| “Z” |
| 9 | digit | “0” \| “1” \| .. \|”9” |
| 10 | number | digit {digit} |
| 11 | comment | “~” {all_characters} newline |
| 12 | newline | newline char |
| 13 | all_characters | any character |

UML (found in p1UML.png):




Running:

To run this program in Linux follow these steps-

1. Open a terminal and navigate to the directory holding the project
2. Run “make” in the terminal
3. Run “./main” in the terminal-
    a. this will run files a1-a8 through the scanner and parser and output whether the provided file contains valid language.
    b. Lines 38-42 of main.cpp contain commented out debugging code that outputs every token along with its line number.
    c. The main loads the input files in the scanner by providing their directory “/inputFiles/ax”