PLDI Group 10, Assignment 2

Rithik Kumar S, Devashish Vij

September 2023

The Lexer

- **TOKEN:**

  {KEYWORD} | {IDENTIFIER} | {CONSTANT} | {STRING_LITERAL} | {PUNCTUATOR}

  This matches any of the five types of tokens: keyword, identifier, constant, string literal, or punctuator.

- **KEYWORD:**

  (char | else | for | if | int | return | void)

  Matches any of the specific keywords, such as char, else, for, etc.

- **IDENTIFIER:**

  {IDENTIFIER_NONDIGIT}+({IDENTIFIER_NONDIGIT} | {DIGIT})*

  Matches an identifier, which starts with one or more non-digit characters (like letters or an underscore), followed by zero or more characters that can either be non-digit or digit.

- **IDENTIFIER-NONDIGIT:**

  [a–zA–Z_]

  Matches any single uppercase or lowercase letter, or an underscore.

- **DIGIT:**

  [0 – 9]

  Matches any single numeric digit.

- **CONSTANT:**

  {INTEGER_CONSTANT} | {CHARACTER_CONSTANT}

  Matches either an integer constant or a character constant.

- **INTEGER-CONSTANT:**

  ({SIGN}?{NONZERO–DIGIT}{DIGIT}*)|((0)*{DIGIT}*)

  Matches optionally a sign followed by a nonzero digit and then any number of digits, or any number of zeroes followed by any number of digits.

- **NONZERO-DIGIT:**

  [1–9]

  Matches any single digit that's not zero.

- **SIGN:**

  [+\–]

  Matches a plus or minus sign.

- **CHARACTER-CONSTANT:**

  \'{C–CHAR–SEQUENCE}\'

  Matches a character sequence enclosed in single quotes.

- **C-CHAR-SEQUENCE:**

  {C–CHAR}+

  Matches one or more characters.

- **C-CHAR:**

  [^\'\\\n\r]|({ESCAPE–SEQUENCE})

  Matches any character except a single quote, backslash, newline, or carriage return, or matches any of the defined escape sequences.

- **ESCAPE-SEQUENCE:**

  (\\\')|(\\\")|(\\\?)|(\\\\)|(\\a)|(\\b)|(\\f)|(\\n)|(\\r)|(\\t)|(\\v)

  Matches specific escape sequences like ', ", \, *etc.*

- **STRING-LITERAL:**

  \"{S–CHAR–SEQUENCE}\"

  Matches a set of 2 double quotes, either enclosing a character sequence or not.

- **S-CHAR-SEQUENCE:**

  {S–CHAR}*

  Matches zero or more characters.

- **S-CHAR:**

  [^\"\\\n\r]|({ESCAPE_SEQUENCE})

  Matches any character except a double quote, backslash, or newline, carriage return or matches any of the defined escape sequences.

- **PUNCTUATOR:**

  (\[|\]|\(|\)|\{|\}|(->)|&|*|\+|-|\/|%|!|\?|<|>|
  (<=)|(>=)|(==)|(!=)|(&&)|(\|\|)|=|:|;|,)

  Matches any of the specified punctuators like brackets, parentheses, relational operators, etc.

- **COMMENT:**

  {MULTI_LINE_COMMENT}|{SINGLE_LINE_COMMENT}

  Matches either a multi-line comment or a single-line comment.

- **MULTI-LINE-COMMENT:**

  (\/*([^*]|*[^/])**\/)

  Matches a sequence starting with /* and ending with */, and doesn't match nested comments. This works because the part of the RegEx within the nested parentheses wouldn't match if there's a nested */, which would make the part of the RegEx after that match the end of the comment, forcing this expression to match the smallest possible comment match.

- **SINGLE-LINE-COMMENT:**

  (\/\/[^\n\r]*)

  Matches a sequence starting with // and matches any number of characters that are not newlines or carriage returns. When it reaches a newline or a carriage return, (i.e., end of the line), the match terminates there.

- **WS:**

  [\n\t\r]

  Matches whitespace characters: space, newline, tab, or carriage return.

- **INVALID-TOKEN:**

  .+

  Matches all characters, with one or more occurrences. When this is matched at the end of the rules section, it matches any token that doesn't match the rest of the defined rules, thus matching any invalid characters.

The Makefile

The Makefile has these targets:

- `all`: This target depends on the `clean`, `lexer`, and `test-all` targets. When you run `make` or `make all`, it will execute these three targets in the specified order.
- `lexer`: This target is responsible for compiling the lexer. It depends on `lex.yy.c` and `lexer_generator.c`:

  ```
  lexer: lex.yy.c lexer_generator.c
  gcc lex.yy.c lexer_generator.c -o lexer -ll
  ```

- `lex.yy.c`: This target generates the `lex.yy.c` file using `flex` and depends on the file `10_A2.l`:

  ```
  lex.yy.c: 10_A2.l
  flex 10_A2.l
  ```

- `clean`: This target is used to clean up the generated files and compiled lexer:

  ```
  clean:
  rm -rf lex.yy.c lexer
  ```

- `test-all`: This target runs the lexer on multiple test files. After each test, an echo statement indicates the beginning of the next test:

  ```
  test-all:
  ./lexer < test.nc
  echo "\n\n\nRunning next test"
  ./lexer < test1.nc
  echo "\n\n\nRunning next test"
  ./lexer < test2.nc
  echo "\n\n\nRunning next test"
  ./lexer < test3.nc

  ...

  echo "\n\n\nRunning next test"
  ./lexer < test10.nc
  ```

The Test Files

There are 11 test files, with the first one (`test.nc`) having various kinds of different tests for the lexer, including an invalid token, and the rest (`test1.nc`, `test2.nc`, ... `test10.nc`) taken from the question's PDF.