情報科学演習 D
課題 1

基礎工学部情報科学科ソフトウェア科学コース
学籍番号: 09B20033

城間大幹

2022 年 10 月 13 日

1　仕様

本章では Lexer[*1]の仕様について説明する．Lexer への入力は pas ファイルから行い，出力は ts ファイルで行う．字句解析を行うメソッドである Lexer.run(String, String) の仕様は以下の通りである．

- 第一引数で指定された pas ファイルを読み込み，トークン列に分割する．
- トークン列は第二引数で指定された ts ファイルに書き出す．
- 正常に処理が終了した場合は標準出力 (System.out) に”OK”を出力する．
- 入力ファイルが見つからない場合は標準エラー (System.err) に”File not found”と出力する．

なお切り出すトークンの種類や出力方法は指導書の通りであるため, ここでは説明を割愛する．
次に指導書で詳細に言及されていなかった仕様について, 以下説明する．

- 例えば 1a といった数字と文字列を含むようなトークンについては, 識別子 (SIDENTIFIER) と判定する．
- 文字列を示す, ’ ’ 内については予約語や記号があったとしても’ ’ 内は一つの文字列 (SSTRING) と判定する．
- aaa[bb など識別子の途中に記号などを含む場合は, aaa, [, bb と分ける．

2　方針と設計

本章では Lexer の方針と設計について説明する．
pas ファイルから一文字ずつ読み出して単語と記号に分け, それぞれがトークン一覧に該当するかどうかを判定した．単語は, 記号や改行文字に当たるまでの文字列とした．

3　実装プログラム

本章では作成した二つのプログラム, Token.java と Lexer.java について説明する．
まず Token.java についてだが, Lexer.java で切り出した文字列 str がどのトークンに当たるかを判定する．str が ID42 番までのトークンを記載した二次元配列 reserved_word の中に該当すれば, その ID と字句解析器上でのトークン名を返す．str が数字であれば ID44 と SCONSTANT を返す．上記に該当しなければ ID43 と SIDENTIFIER を返す．
続いて Lexer.java について説明する．Lexer.java では Token.java に渡す文字列 str の作成を行なう．BufferedReader を用いて一文字ずつ c に読み出し, それをもとに文字列 str を作るが, いくつかのパターンがある．
コメント {.....} に対しては, c={ となった場合, c=} となるまで読み進め, str は作成しない．
シングルクォーテーション に対しては, 二つ目のシングルクォーテーションまでの文字列を str に格納し, その str と ID45 と SSTRING を返す．

[*1] 今回作成したプログラムの名前

上記二つに該当しない場合, トークン一覧にある記号や改行文字, 空白などが c に格納された際, まずここまで作成された str を Token.java に渡す. 続いて < は <,<>, <= のいずれかに, > は >,>= のいずれかに, : は : , :=のいずれかに, . は . , .. のいずれかに, 次の文字次第で当たる可能性がある. 故に c_next に次の文字を格納し, 状況に応じて str に c か c+c_next を格納し Token.java に渡す. その他の四則演算記号や括弧などは str にそのまま c を格納し, Token.java に渡す.

なお, 行番号については改行文字が c に格納される度に height でカウントした.

また outputResult は Token.java から返る ID, 字句解析器上でのトークン名と行数を扱うメソッドであり, outputResult を呼び出す度に戻り値を result に格納し続けた. 入力ファイルの読み込みが終了した後に, FileWriter で result の内容を ts ファイルに書き込んだ.

4 考察や工夫点

トークンの判定や字句の切り出し, ファイルへの出力などできる限り機能を分けて記述した. これによりテストケースを通した際のデバッグで, 原因の追求がしやすいことや, コードを改変した際にこれまで通っていたテストケースが通らなくなることが激減した. 全体を機能に分け, なるべくお互いが干渉しないように設計し, 部分的, 段階的に実現していくことの重要性を学んだ.

5 感想

一つ一つ単語を切り出して, トークン一覧に該当するかどうかを判定するという極めてシンプルな方針で作成したが, 条件分岐が複数に渡りプログラムを作成するのがとても大変だった. オートマンを意識して作成すればもう少し楽に作れたかもしれない.

個人的に良かった点は, コードを書く前に入力の pas ファイルを全て見て, 一つテストケースに依存しすぎないよう, 多くのテストケースで発生する問題から優先的に対処したことだ. テストケースを通すためのプログラムではなく字句解析を行うプログラムを作ることが求められており, テストケースを用いた体系的な理解が最も重要だと考えたからである. 課題 2 以降も同様にして取り組みたい.

6 謝辞

終始熱心なご指導を頂いた情報科学演習 D の松本 真佑，桝井 晃基担当教員, また TA の方々には心より感謝の意を表します.

参考文献

[1] 小山博史, 株式会社ガリレオ, コンパイラの入り口, 「字句解析」のための文字列操作, https://atmarkit.itmedia.co.jp/ait/articles/0705/15/news135.html, 2007 年 05 月 15 日

[2] quwahara, Qiita, 1 単純な字句解析を Java で実装する, https://qiita.com/quwahara/items/d7ea5d0e0dbc0409a01f, 2017 年 07 月 18 日