令和 5 (2023) 年 10 月

言語処理系 (字句解析課題問題)

以下の演習を行い、kibaco へ提出せよ。

注意事項

- 提出するファイル形式は PDF のみ
- ソースファイルの変更部分および実行結果を単一の PDF にまとめる
- 実行結果として実行画面のスクリーンショットを含める

(1) `go run main.go` によりプログラムを実行し `3.14` と入力した際の実行結果を示せ。また、そのような実行結果となる理由を説明せよ。

(2) `repl/repl.go` 11 行目の `const PROMPT = ">> "` の `">> "` を `"Input> "` に変えるとプログラムを実行した際に、どの部分が変化するかを説明せよ。

(3) `lexer/lexer_test.go` の 35 行目の {`token.LET, "let"`}, を {`token.LET, "LET"`}, に変更した後に `go test ./lexer` を実行せよ。変更前と変更後で結果がどの様に変化するかを示し、なぜ変化が起きたかを説明せよ。

(4) Monkey 言語では、`@` はトークンとして認識されず、`ILLEGAL` として処理される。`@` を新しく 1 文字トークンとして追加し、`lexer` で処理される様にするため、以下の手順でプログラムを変更せよ。

  - `token/token.go` の定数の定義に `atmark = "@"` を追加する。
  - `lexer/lexer.go` の `switch` 文に `case '@':` を追加し、トークンへの変換を行う。

(5) Monkey 言語での識別子名には、アルファベットのみ使用可能である（例えば `a100` は識別子 `a` と整数 `100` の 2 つのトークンとして処理される）。以下の手順でプログラムを変更し、2 文字以降には数字文字も使用可能な様にせよ。

  - `lexer/lexer.go` に新しい関数 `isLetterNum()` を追加する。`isLetter()` を参考に、数字文字に対しても `true` を返却する。

- readIdentifier() で、2 文字目以降の判定に isLetterNum() を利用する様に変更する。(readIdentifier() での 1 文字目の判定にも isLetterNum() を利用することも出来る。理由を説明せよ。)

(6) 2 文字トークン "++" を追加して lexer が認識するようにプログラムを変更せよ。トークンタイプは++ とする。(token/token.go に定数を追加する際には、PLUSPLUS = "++" などとする。)

(7) 関数を記述する際のキーワードを "fn" に加え、 "function" でも可能となるようにしたい。キーワードに "function" を追加し、出力するトークンは "fn" と同じとなるようにプログラムを変更せよ。

(8) 整数リテラルとして、16 進数 (0x35AC など) が利用可能となるようにプログラムを変更せよ。以下の手順を参考にせよ。

- 新しいトークンタイプ HEX を追加する。リテラルは、入力された 16 進数から 0x を除いた部分とする。(例 0xab59 の場合、ab59 とする。)
- メソッド isDigit を参考にして新しいメソッド isHex を追加する。数字文字に加え a-f および A-F ならば true を返却する。
- メソッド readNumber を参考にして新しいメソッド readHex を追加する。isHex を呼び出して利用する。

(9*) Monkey 言語の識別子名を定義を変更し、一文字目はかならず $ から (例えば、$var などのように) 始まることが必要となるようにプログラムを変更せよ。

(10*) Monkey 言語の識別子名に '-' (ハイフン) を利用できるように変更せよ。この場合、どのような不具合が生じる可能性があるか説明せよ。

(11*) アルファベットの大文字・小文字の区別をしない様に変更せよ。例えば、let、Let、LET などを同じトークンに変換する様にプログラムを変更せよ。