

言語処理系（字句解析課題問題）

- (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

```
>>go run main.go
```

```
Hello sentooooooooon! This is the Monkey programming language!
```

```
Feel free to type in commands
```

```
>> 3.14
```

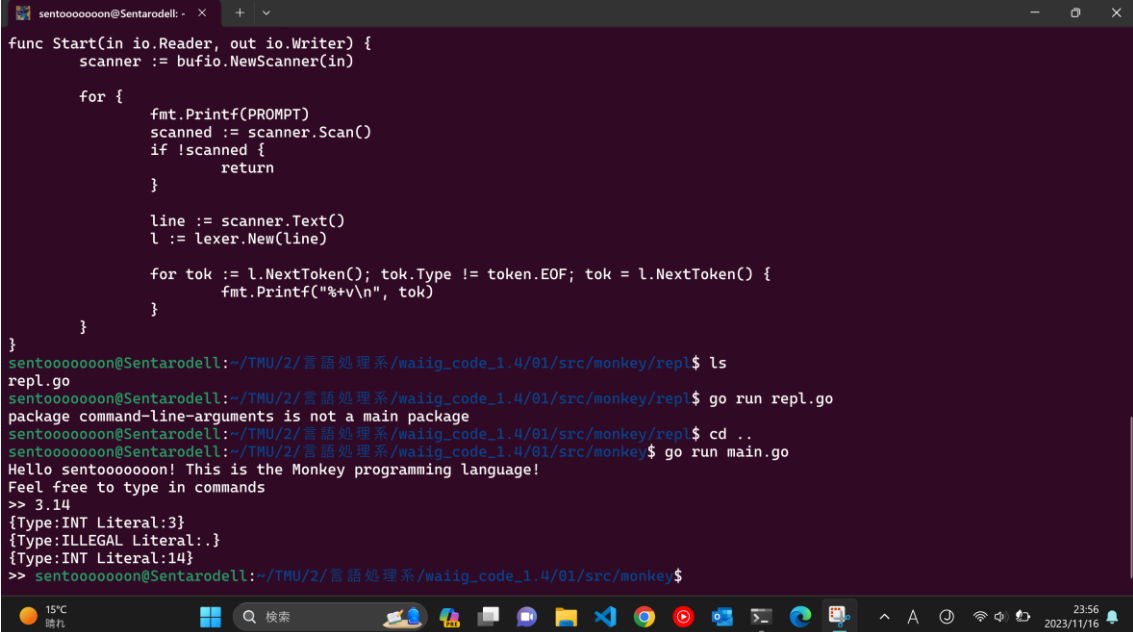
```
{Type:INT Literal:3}
```

```
{Type:ILLEGAL Literal:..}
```

```
{Type:INT Literal:14}
```

まず最初のメッセージは、fmt.Printf()でメッセージを表示。user, err := user.Current()でユーザー名（sentooooooooon）を表示させている。

3.14 と入力すると、トークンが読み込まれ、それぞれの型が変換される。



```
func Start(in io.Reader, out io.Writer) {
    scanner := bufio.NewScanner(in)

    for {
        fmt.Printf(PROMPT)
        scanned := scanner.Scan()
        if !scanned {
            return
        }

        line := scanner.Text()
        l := lexer.New(line)

        for tok := l.NextToken(); tok.Type != token.EOF; tok = l.NextToken() {
            fmt.Printf("%v\n", tok)
        }
    }
}

sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4/01/src/monkey/repl$ ls
repl.go
sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4/01/src/monkey/repl$ go run repl.go
package command-line-arguments is not a main package
sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4/01/src/monkey/repl$ cd ..
sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4/01/src/monkey$ go run main.go
Hello sentooooooooon! This is the Monkey programming language!
Feel free to type in commands
>> 3.14
{Type:INT Literal:3}
{Type:ILLEGAL Literal:..}
{Type:INT Literal:14}
>> sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4/01/src/monkey$
```

- (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。

main.go を実行したときに表示される、入力を求める文字列が変化する。

- (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わするかを示し、なぜ変化が起きたかを説明せよ。

```
sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4/01/src/monkey$ go test ./lexer/
--- FAIL: TestNextToken (0.00s)
    lexer_test.go:122: tests[0] - literal wrong. expected="LET", got="let"
FAIL
FAIL    monkey/lexer    0.003s
FAIL
sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4/01/src/monkey$ go test ./lexer/
ok      monkey/lexer    0.002s
```

変更する前はエラーが出ないが変更するとエラーが出る。

- (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。

```
sentooooooooon@Sentarodell: ~
+
sentooooooooon@Sentarodell:~$ cd TMU/2/言語処理系/
sentooooooooon@Sentarodell:~/TMU/2/言語処理系$ cd waiig_code_1.4/0
-bash: cd: waiig_code_1.4/0: No such file or directory
sentooooooooon@Sentarodell:~/TMU/2/言語処理系$ cd waiig_code_1.4/
sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4$ cd 0
-bash: cd: 0: No such file or directory
sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4$ cd 01
direnv: loading ~/TMU/2/言語処理系/waiig_code_1.4/01/.envrc
direnv: export +GOPATH
sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4/01$ cd src/monkey/
sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4/01/src/monkey$ ls
lexer  main.go  repl  token
sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4/01/src/monkey$ go run main.go
Hello sentooooooooon! This is the Monkey programming language!
Feel free to type in commands
Input> @
{Type:@ Literal:0}
Input>
```

- (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。

```
Input> sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4/01/src/monkey$ go run main.go
Hello sentooooooooon! This is the Monkey programming language!
Feel free to type in commands
Input> 1f2
{Type:INT Literal:1}
{Type:IDENT Literal:f2}
Input> 22f3
{Type:INT Literal:22}
{Type:IDENT Literal:f3}
Input> a100
{Type:IDENT Literal:a100}
Input>
```

```
func (l *Lexer) readIdentifier() string {
    position := l.position
    for isLetterNum(l.ch) {
        l.readChar()
    }
    return l.input[position:l.position]
}
```

```
func isLetterNum(ch byte) bool {
    return isLetter(ch) || isDigit(ch)
}
```

- (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。)

```
sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4/01/src/monkey$ go run main.go
Hello sentooooooooon! This is the Monkey programming language!
Feel free to type in commands
Input> 2
{Type:INT Literal:2}
Input> ++
{Type:++ Literal:++}
Input> +
{Type:+ Literal:+}
Input> +++
{Type:++ Literal:++}
{Type:+ Literal:+}
Input>
```

```
case '+':
    if l.peekChar() == '+' {
        ch := l.ch
        l.readChar()
        literal := string(ch) + string(l.ch)
        tok = token.Token{Type: token.PLUSPLUS, Literal: literal}
    } else {
        tok = newToken(token.PLUS, l.ch)
    }
}
```

- (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。

```

sentooooooooon@Sentarodell:~/TMU/2/言語処理系/waiig_code_1.4/01/src/monkey$ go run main.go
Hello sentooooooooon! This is the Monkey programming language!
Feel free to type in commands
Input> fn
{Type:FUNCTION Literal:fn}
Input> FN
{Type:IDENT Literal:FN}
Input> function
{Type:IDENT Literal:function}
Input> function
{Type:IDENT Literal:function}
Input> fn
{Type:FUNCTION Literal:fn}
Input>

```

```

switch l.readIdentifier() {
    case "fn", "function":
        return token.Token{Type: token.FUNCTION, Literal: "fn"}
}

```

- (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。

```

(9) func (l *Lexer) readNumber() string {
(10)     position := l.position
(11)     isHex := false
(12)
(13)     for {
(14)         if isHex && !isHexDigit(l.ch) {
(15)             break
(16)         }
(17)         if !isHex && !isDigit(l.ch) {
(18)             break
(19)         }
(20)
(21)         l.readChar()
(22)         if l.ch == 'x' {
(23)             isHex = true
(24)         }
(25)     }
(26)
(27)     return l.input[position:l.position]
(28) }

```

```

func isHexDigit(ch byte) bool {

```

```
    return '0' <= ch && ch <= '9' || 'a' <= ch && ch <= 'f' || 'A' <= ch  
&& ch <= 'F'  
}
```

```
{token.INT, "0x35AC"},
```