言語処理系論 レポート課題1

- 所属: 東京大学教養学部学際科学科総合情報学4年
- 学生証番号: 08-192017
- 氏名: 江口 大志

1-1[必須] GT,GEQ,ELSE,TIMESのトークンを追加する

加えた変更

`lexer-by-hand.ml`に変更を加える。

tokenにGEQ,GT,TIMES,ELSEを追加。

```
@@ -3,7 +3,7 @@
-type token = INVALID | EQ | LEQ | LT | PLUS | IF | THEN | INT | ID
+type token = INVALID | EQ | LEQ | LT | GEQ | GT | PLUS | TIMES | IF |
THEN | ELSE | INT | ID
```

出力関数に追加。

```
@@ -34,9 +34,13 @@ let output_token token =
    EQ -> print_string "EQ "
  | LEQ -> print_string "LEQ "
  | LT -> print_string "LT "
+ | GEQ -> print_string "GEQ "
+ | GT -> print_string "GT "
  | PLUS -> print_string "PLUS "
+ | TIMES -> print_string "TIMES "
  | IF -> print_string "IF "
  | THEN -> print_string "THEN "
+ | ELSE -> print_string "ELSE "
  | INT -> (print_string ("INT("^(get_lexeme())^") "))
  | ID -> print_string ("ID("^(get_lexeme())^") ")
  | INVALID -> assert false
```

GT,GEQはLTと同様に, TIMESはPLUSと同様に, ELSEはTHENと同様に最初の文字の入力を受け取る。

```
@@ -59,9 +63,12 @@ and q0 () =
            q0())
   | '=' -> (save EQ; next())
   | '<' -> (save LT; q_lt())
+  | '>' -> (save GT; q_gt())
   | '+' -> (save PLUS; next())
+  | '*' -> (save TIMES; next())
```

```
    | 'i' -> (save ID; q_i())
    | 't' -> (save ID; q_t())
+   | 'e' -> (save ID; q_e())
    | '0' -> (save INT; next())
    | c -> if '1'<=c && c<='9' then (save INT; q_num())
           else if 'a'<= c && c<='z' then (save ID; q_sym())
```

q_lt()と同様にGT,GEQの判定ができる関数q_gt()を定義。

```
@@ -73,6 +80,11 @@ and q_lt() =
     '=' -> (save LEQ; next())
   | _ -> next()

+and q_gt() =
+  match readc() with
+    '=' -> (save GEQ; next())
+  | _ -> next()
+
 and q_num() =
   let c= readc() in
     if '0'<=c && c<='9' then (save INT; q_num()) else next()
```

THENの判定と同様、elseの文字を一文字ずつ受け取り、それ以外が来たときにIDとして識別する状態関数を定義。

```
@@ -101,6 +113,24 @@ and q_the() =
    | c -> if ('a'<= c && c<='z')||('0'<=c && c<='9') then (save ID; q_s
ym())
           else next()

+and q_e() =
+  match readc() with
+    'l' -> (save ID; q_el())
+  | c -> if ('a'<= c && c<='z')||('0'<=c && c<='9') then (save ID; q_s
ym())
+         else next()
+
+and q_el() =
+  match readc() with
+    's' -> (save ID; q_els())
+  | c -> if ('a'<= c && c<='z')||('0'<=c && c<='9') then (save ID; q_s
ym())
+         else next()
+
+and q_els() =
+  match readc() with
+    'e' -> (save ELSE; q_sym())
+  | c -> if ('a'<= c && c<='z')||('0'<=c && c<='9') then (save ID; q_s
ym())
+         else next()
```

```
+
 and q_sym() =
   let c = readc() in
       if ('a'<= c && c<='z')||('0'<=c && c<='9') then (save ID; q_sym(
))
(END)
```

テストケース

#use 'lexer-1-1.ml'とした後に、

```
# main "if x >= 0 then x = 0 else x = x * x";;
IF ID(x) GEQ INT(0) THEN ID(x) EQ INT(0) ELSE ID(x) EQ ID(x) TIMES ID(x) -
: unit = ()

# main "if if0 > 1 then then1 = 2 else else2 = 3";;
IF ID(if0) GT INT(1) THEN ID(then1) EQ INT(2) ELSE ID(else2) EQ INT(3) - :
unit = ()
```

となり、これは意図した動作になっている。

1-2[余力があれば]コメントをスペースと同一視して扱えるようにする

加えた変更

/でコメントを検知するための状態関数に移動する。

```
@@ -63,6 +63,8 @@ and q0 =
   | 'i' -> (save ID; q_i())
   | 't' -> (save ID; q_t())
   | '0' -> (save INT; next())
+  | '/' -> (pos_start := !pos_start+1;
+            q_lsl())
   | c -> if '1'<=c && c<='9' then (save INT; q_num())
          else if 'a'<= c && c<='z' then (save ID; q_sym())
```

*で次の状態に移りコメントを検知できるようにする。

```
@@ -106,6 +108,21 @@ and q_sym() =
       if ('a'<= c && c<='z')||('0'<=c && c<='9') then (save ID; q_sym())
       else next()

+and q_lsl() =
+  match readc() with
+    '*' -> (pos_start := !pos_start+1;q_com())
+  | _ -> report_error(!pos_current)
+
```

```
+and q_com() =
+  match readc() with
+    '*' -> (pos_start := !pos_start+1;q_ast())
+  | _ -> (pos_start := !pos_start+1;q_com())
+
+and q_ast() =
+  match readc() with
+    '/' -> (pos_start := !pos_start+1;q0())
+  | _ -> report_error(!pos_current)
+
 and next() =
      if !last_token=INVALID then
```

テストケース

`#use 'lexer-1-2.ml'`とした後に、以下を実行すると、意図した挙動になっていることが確認できる。

```
# main "if /* this is a comment */ x<y";;
IF ID(x) LT ID(y) - : unit = ()
```

ただし現在の実装だとnested commentに対応できない。nested commentに対応するには再起関数などを使って実装する。

```
# main "if /* /* this is a comment */ */ x<y";;
IF invalid token at position 9
- : unit = ()
```

1-3[必須]lexer.mll を拡張し、GT,GEQ,ELSEのトークンを追加する

加えた変更

必要な型を定義する。

```
@@ -10,8 +10,8 @@
 let line_no = ref 1  (* the current line number, used for error reporting
*)
 let end_of_previousline = ref 0
 (* data type declaration for tokens *)
-type token = EQ | LEQ | LT | PLUS | MINUS | TIMES | LPAREN | RPAREN
-           | IF | THEN | INT of int | ID of string | EOF
+type token = EQ | LEQ | LT | GEQ | GT | PLUS | MINUS | TIMES | LPAREN |
RPAREN
+           | IF | THEN | ELSE | INT of int | ID of string | EOF
 }
```

各文字列に対応する型を指定する。（この時GTをGEQより先に指定すると、>=にGTが反応してしまうので注意）

```
@@ -36,9 +36,12 @@ rule token = parse
  | "=" {EQ}
  | "<=" {LEQ}
  | "<" {LT}
+| ">=" {GEQ}
+| ">" {GT}
  | "0" {INT(0)}
  | "if" {IF}
  | "then" {THEN}
+| "else" {ELSE}
  | digitnz digit*
     {let s = Lexing.lexeme lexbuf in INT(int_of_string s)}
  | lower (digit|lower)*
```

テストケース

1-1と同様に、以下のテストケースをファイルとして用意した。

```
// test1.txt
if x >= 0 then x = 0 else x = x * x
```

```
// test2.txt
if if0 > 1 then then1 = 2 else else2 = 3
```

CLIでlexerを生成。

```
$ ocamllex lexer-1-3.mll
```

`#use 'lexer-1-3.ml'`とした後に、

```
# main "test1.txt";;
- : token list =
[IF; ID "x"; GEQ; INT 0; THEN; ID "x"; EQ; INT 0; ELSE; ID "x"; EQ; ID
"x"; TIMES; ID "x"]

# main "test2.txt";;
- : token list =
[IF; ID "if0"; GT; INT 1; THEN; ID "then1"; EQ; INT 2; ELSE; ID "else2";
EQ; INT 3]
```