Esercitazione di Laboratorio di Linguaggi e Computabilità

LeX e Yacc per Java: JFlex e BYACC/J

Esercizio 1

Modificare la definizione della grammatica della calcolatrice in modo che accetti l'operazione modulo (%), il logaritmo in base 10 (log) e il logaritmo naturale (ln).

Svolgimento

- *Operazione modulo*

1. Per prima cosa è necessario identificare il token che rappresenta l'operazione modulo, cioè il simbolo '%';
2. Una volta identificato il simbolo da gestire:
   a. È necessario definire una produzione che permetta al parser di riconoscere l'operazione modulo (file `.y`); in particolare, bisogna modificare la produzione 'exp', aggiungendo una parte destra che gestisca l'operazione modulo:

```
exp:     NUM                { $$ = $1; }
       | exp '+' exp        { $$ = $1 + $3; }
       | exp '-' exp        { $$ = $1 - $3; }
       | exp '*' exp        { $$ = $1 * $3; }
       | exp '/' exp        { $$ = $1 / $3; }
       | '-' exp  %prec NEG { $$ = -$2; }
       | exp '^' exp        { $$ = Math.pow($1, $3); }
       | '(' exp ')'        { $$ = $2; }
       ;
```

->

```
exp:     NUM                { $$ = $1; }
       | exp '+' exp        { $$ = $1 + $3; }
       | exp '-' exp        { $$ = $1 - $3; }
       | exp '*' exp        { $$ = $1 * $3; }
       | exp '/' exp        { $$ = $1 / $3; }
       | exp '%' exp        { $$ = $1 % $3; }
       | '-' exp  %prec NEG { $$ = -$2; }
       | exp '^' exp        { $$ = Math.pow($1, $3); }
       | '(' exp ')'        { $$ = $2; }
       ;
```

Modificare poi la definizione di precedenza tra gli operatori "`%left '*' '/'` " in "`%left '*' '/' '%'`", nella parte delle definizioni del file `.y`.

   b. È necessario definire una regola lessicale che permetta al lexer di riconoscere il carattere '%' ed identificarlo come token (file`.flex`); in particolare, bisogna modificare la regola sotto il commento 'operators', aggiungendo un pattern che riconosca il carattere '%':

```
/* operators */
"+" |
"-" |
"*" |
"/" |
"^" |
"(" |
")"    { return (int) yycharat(0); }
```

->

```
/* operators */
"+" |
"-" |
"*" |
"/" |
"^" |
"%" |
"(" |
")"    { return (int) yycharat(0); }
```

- *Logaritmo in base 10 (log)*

3. Per prima cosa è necessario identificare il token che rappresenta l'operazione logaritmo in base 10, che chiameremo LOG10, e dichiararlo nella sezione delle dichiarazioni del file `.y` con il comando

"`%token LOG10`"; inoltre si può assegnare al token una associatività a sinistra con il comando "`%left LOG10`";

4. Una volta identificato il simbolo da gestire:
    a. È necessario definire una produzione che permetta al parser di riconoscere l'operazione logaritmo (file `.y`); in particolare, bisogna modificare la produzione 'exp', aggiungendo una parte destra che gestisca l'operazione modulo:

```
LOG10 exp            { $$ = Math.log10($2); }
```

    b. È necessario definire una regola lessicale che permetta al lexer di riconoscere i caratteri che identificarlo il token LOG10 (file .flex); in particolare bisogna:
        i. Nelle definizioni, aggiungere una macro che permetta di riconoscere la sequenza di caratteri "log10" oppure "LOG10":

```
LOG10 = ("log10"|"LOG10")
```

        ii. Aggiungere una regola lessicale che informi il parser che è stato riconosciuto il token LOG10:

```
{LOG10}    { return Parser.LOG10; }
```

- *Logaritmo naturale (ln)*

  Simile al caso sopra…

Esercizio 2: nelle produzioni di calc.y restituire (in `$$`) una stringa (sval, non dval) con l'espressione ben 'impaginata'

E stampare la stringa risultante quando viene dato NL. Esempio: con spazi regolari 1+2 /3 → 1 + 2 / 3
N.B. il risultato dell'esressione non viene più calcolato