Μεταγλωττιστές

Άσκηση Μέρος 1

Λεκτική Ανάλυση

Στο μέρος αυτο υλοποιούμε ένα λεκτικό αναλυτή, ο οποίος υποστηρίζει την χρήση βασικών εκφράσεων που μοιάζουν με την C γλώσσα. Χρησιμοποιώντας το εργαλείο JFlex κάνουμε λεκτική ανάλυση με τις εξής αρχές:

• Τις λέξεις κλειδιά
float, int, char, while, if, else, print, void, return, break, continue, new, delete
• Τα αναγνωριστικά που αποτελούνται από ένα γράμμα του λατινικού αλφαβήτου (πεζό ή κεφαλαίο) και πιθανώς ακολουθείται από μια σειρά γραμμάτων του λατινικού αλφαβήτου ή δεκαδικών ψηφίων. Τα πεζά γράμματα θεωρούνται διαφορετικά από τα αντίστοιχα κεφαλαία. Οι λέξεις κλειδιά δεν μπορούν να είναι αναγνωριστικά.
• Ακέραιες σταθερές χωρίς πρόσημο αποτελούμενες από δεκαδικά ψηφία.
• Σταθερές κινητής υποδιαστολής.
• Σταθερές χαρακτήρα που είναι ένας χαρακτήρας μέσα σε μονά εισαγωγικά. Οι ακολουθίες διαφυγής που υποστηρίζονται είναι οι \n, \t και \0.
• Σταθερές συμβολοσειρές, που είναι ακολουθίες κοινών χαρακτήρων ή ακολουθιών διαφυγής μέσα σε διπλά εισαγωγικά. Οι ακολουθίες διαφυγής που υποστηρίζονται είναι οι \n και \t.
• Τελεστές: = > < ! = <= >= + - * / % == && || ! .
• Διαχωριστές: { } [ ] ; ( )

Για να το πετύχουμε αυτό δημιουργήσαμε ένα project βασισμένο στη δομή του εργαστηρίου. Το αρχείο lexer.flex έχει τον κώδικα για το εργαλείο jflex. Ο κώδικας αυτός μεταφράζεται σε κώδικα java αυτόματα από το εργαλείο.

Έτσι, ορίσαμε τα παρακάτω βασικά patterns για την αναγνώριση των tokens :

```
LineTerminator      = \r|\n|\r\n
WhiteSpace          = {LineTerminator} | [ \t\f]
Comment             = "/*" [^*] ~"*/" | "/*" "*"+ "/"
Exponent            = [eE][\+\-]?[0-9]+
Float1              = [0-9]+ \. [0-9]+ {Exponent}?
Float2              = \.[0-9]+  {Exponent}?
Float3              = [0-9]+ \. {Exponent}?
Float4              = [0-9]+ {Exponent}
FloatLiteral                = {Float1} | {Float2} | {Float3} | {Float4}
Identifier          = [:jletter:] [:jletterdigit:]*
IntegerLiteral      = 0 | [1-9][0-9]*
```

Χρησιμοποιούμε regular expressions για την περιγραφή των λεξημάτων που πρέπει να αναγνωρίζονται. Για γράμματα και αριθμούς το jflex παρέχει έτοιμα character classes, που χρησιμοποιούμε στην αναγνώριση των identifiers. Επιπλέον, ορίζουμε και 2 ακόμα αρχικές

καταστάσεις (state) για το scan, με στόχο να ερμηνεύσουμε τις σειρές χαρακτήρων και τους μεμονωμένους χαρακτήρες, μία για την περίπτωση που αναγνωρίσεται string, με “ (Double Quotes) και μία για char που αναγνωρίζεται με ‘ (Single Quote).

```
%state STRING
%state CHAR
```

Στη συνέχεια στην φάση των rules και actions, ορίζουμε για κάθε λέξημα, ένα προκαθορισμένο action, δηλαδή κώδικα που εκτελείται στην προκειμένη περίπτωση. Στην συγκεκριμένη φάση μονάχα εκτυπώνουμε το κάθε εύρημα σε μια γραμμή, αλλά στη συνέχεια θα πρέπει να αλλάξει κάθε τέτοια εντολή σε input προς τον συντακτικό αναλυτή. Στις περιπτώσεις string και char, το action είναι μόνο η αρχικοποίηση ενός StringBuffer στον οποίο κρατάμε την τιμή, και αρχικοποίηση μίας ξεχωριστής διαδικασίας scan. Σε whitespace και σχόλια, το action είναι κενό.

```
//Main initial state
<YYINITIAL> {
        /* reserved keywords */
        "print"                 { out.println("PRINT"); }
        "float"                 { out.println("FLOAT"); }
        "int"                   { out.println("INT"); }
        "char"                  { out.println("CHAR"); }
        "while"                 { out.println("WHILE"); }
        "if"                    { out.println("IF"); }
        "else"                  { out.println("ELSE"); }
        "void"                  { out.println("VOID"); }
        "return"                { out.println("RETURN"); }
        "break"                         { out.println("BREAK"); }
        "continue"              { out.println("CONTINUE"); }
        "new"                   { out.println("NEW"); }
        "delete"                { out.println("DELETE"); }

        /* identifiers */
        {Identifier}            { out.println("id:" + yytext()); }

        /* literals */
        {IntegerLiteral}                { out.println("integer:" + yytext()); }
        {FloatLiteral}          { out.println("float:" + yytext()); }

        /* String or Char (New Initial State) */
        \"                      { sb.setLength(0); yybegin(STRING); }
        \'                      { sb.setLength(0); yybegin(CHAR); }

        /* operators */
        "="                     { out.println("ASSIGN"); }
        ">"                     { out.println("BIGGER THAN"); }
        "<"                     { out.println("SMALLER THAN"); }
```

```
        "!="                    { out.println("DIFFERENT THAN"); }
        "<="                    { out.println("SMALLER OR EQUAL THAN"); }
        ">="                    { out.println("BIGGER OR EQUAL THAN"); }
        "+"                     { out.println("PLUS"); }
        "-"                     { out.println("MINUS"); }
        "*"                     { out.println("MULTI"); }
        "/"                     { out.println("DIV"); }
        "%"                     { out.println("MOD"); }
        "=="                    { out.println("EQUAL"); }
        "&&"                    { out.println("AND"); }
        "||"                    { out.println("OR"); }
        "!"                     { out.println("NOT"); }
        "."                     { out.println("DOT"); }

        /* Separators */
        "{"                     { out.println("LEFT CURLY BRACKET"); }
         "}"                    { out.println("RIGHT CURLY BRACKET"); }
        "["                     { out.println("LEFT BRACKET"); }
        "]"                     { out.println("RIGHT BRACKET"); }
        ";"                     { out.println("SEMICOLON"); }
        "("                     { out.println("LEFT PARENTHESIS"); }
        ")"                     { out.println("RIGHT PARENTHESIS"); }
        ","                     { out.println("COMMA"); }

        /* comments */
        {Comment}               { /* ignore */ }

        /* whitespace */
        {WhiteSpace}            { /* ignore */ }
}
```

Στην περίπτωση του string:

```
/*Initial state for String.*/
<STRING> {
        \"                      { yybegin(YYINITIAL); out.println("string:" + sb.toString()); }

        [^\n\r\"\\]+            { sb.append(yytext()); }
        \\t                     { sb.append('\t'); }
        \\n                     { sb.append('\n'); }
        \\r                     { sb.append('\r'); }
        \\\"                    { sb.append('\"'); }
         \\                     { sb.append('\\'); }
}
```

Στην γλώσσα που υποστηρίζεται, δεν επιτρέπονται strings με single quotes, και μόνο ο τύπος char υποστηρίζεται στην περίπτωση που αναγνωρίζεται single quote. Γι'αυτό και πριν τον ορισμό των patterns προσθέσαμε στο custom code block και μία μέθοδο που ελέγχει τον buffer να μήν έχει μέγεθος μεγαλύτερο του 1. Αν βρεθεί με μεγαλύτερο του ένα έχουμε exception.

```
%{
   // user custom code
   StringBuffer sb = new StringBuffer();//Hold a String.

   public void checkCharError(StringBuffer sb){//ONLY 1 CHARACTER WITHIN '', and no
more!
       if( sb.length() > 1 ) throw new RuntimeException("Error at line: "+(yyline+1) + "
,column:" + (yycolumn+1) + " : illegal character <"+ yytext()+">");
   }

%}
```

Έτσι στην περίπτωση του char:

```
/*Initial state for character. This language doesn't support strings like 'Hi!'. ' only for
character.*/
<CHAR> {
   \'                      { yybegin(YYINITIAL);
                             out.println("char: " + sb.toString());
                           }

   [^\n\r'\\]+               { sb.append(yytext()); checkCharError(sb); }
   \\t                     { sb.append('\t'); checkCharError(sb); }
   \\n                      { sb.append('\n'); checkCharError(sb); }
   \\r                     {/* ignore sb.append('\r'); */} //Carriage return not supported (?)
   \\0                      { sb.append('\0'); checkCharError(sb); }
   \\\'                    { sb.append('\''); }
   \\                      { sb.append('\\'); checkCharError(sb); }
}
```

Και τέλος για κάθε περίπτωση που βρεθεί μη υποστηριζόμενος χαρακτήρας, έχουμε exception.

```
/* error fallback */
[^]                         { throw new RuntimeException((yyline+1) + ":" + (yycolumn+1) + ":
illegal character <"+ yytext()+">"); }
```

Για έλεγχο του project, στο αρχικό directory του project:

```
mvn package
java -jar target/compiler-0.0.1.jar erg1.txt
java -jar target/compiler-0.0.1.jar erg2.txt
java -jar target/compiler-0.0.1.jar erg3.txt
```

Τα αρχεία txt στο αρχικό directory έχουν sample code.