10/11/2008

Formal Languages and Compilers – JFlex & Cup

Using the JFLEX lexer generator and the CUP parser generator, realize a Java program capable of recognizing a simple language that allows formulating and computing mathematical expressions.

Input Language.

The input file present an header section followed by a set of mathematical expressions applied to variables.

```
File_Name: example.txt ;
Date: 10/11/2008 ;
Time: 09:00 am;
E-mail: FCL@polito.it ;
```

Figure 1 - Header Section Example

The header section contains a file name, a date, a time, and one or more e-mail addresses (up to a maximum of 3). There is one field for line and each field is prefixed by a label followed by a colon as shown in the figure and terminated by a semicolon symbol.
The order of the fields is the same shown in the example with the exception of Date and Time fields that can be inverted (FileName, Date, Time and E-mail(s) OR FileName, Time, Date and E-mail(s)).

Fields details:

- **File Name:** prefixed by "File_Name :" label, the field consists of a string of numbers and letters where the starting character is a letter and with a mandatory file extension of 3 charcters separated by a dot '.'
- **Date:** prefixed by "Date :" the field is expressed with the format gg/mm/aaaa.
- **Time:** prefixed by the label "Time :" the field is expressed with the format hh:mm followed by one of the 2 keywords "am" or "pm"
- **E-mail:** prefixed by the label "E-mail", the field consists of a string of letter, numbers and the symbols '_' and '.' Followed by the symbol "@" and then two strings of letters (ONLY letters) separated by means of a '.'

The mathematical operations section consists in a list (that could be empty) of mathematical operations and their assignments to variables. The allowed operators are just '+' and '-' allowing for sum and subtraction and operates either on previously defined variables or on numbers. The numbers are floating point numbers that could also be written omitting the decimal part (i.e. 23.4, 45, .56, 4.34); the scientific notation is not supported.
The result of such operations can be assigned either to a single variable or to a list of variables.
Variables are identified by a string containing letter, number and the symbol '_' where the starting character MUST be a letter or an '_'. Variables are declared using the keyword "VAR".

```
2.0: VAR A_0, VAR a_1;
2 : VAR B;
A_0 * B + 3 : VAR _res1 , VAR _res2 , VAR _res3;
```

Figure 2 - Mathematical Operations Section Example

The first operation assigns 2.0 to the variables A_0 and a_1, the second assigns 2 to variable B and the last one assigns 7.0 to variables _res1, _res2, _res3. The syntax of each line of the section is then a mathematical operation on numbers and existing variables followed by a ":" symbol followed by a list of variables (could not be empty!) and terminated by a semicolon.
The name and the value of the variables MUST be saved into a symbol table in order to be reused in the consecutive expressions. The **ONLY** global variable that can be used in the presented solution is the symbol table, therefore the values to be assigned to the variables should be accessed by means of inherited attributes.
Precedence and Associativity of the algebraic operations should be managed properly.

The proposed solution should print an error message if the input file is not well formed and alternatively provide in output (printing to screen) the list of the different assignments; for the previous example the output should be:

```
A_0: 2.0
A_1: 2.0
B: 2.0
_res1: 7.0
_res2: 7.0
_res3: 7.0
```

Figure 3 - Output Example