Formal Languages and Compilers

Laboratory n° 2

1 Exercise

Write a scanner for the C programming language. The scanner must:

- Recognize the C language comments (/* and */)
- Recognize the symbols: “{“, “}”, “(“, “)”, “[“, “]”, “+”, “-”, “*”, “/”, “=”, “;”, “.”, “,”, “<”, “>”, “&”, “|”, “!”.
- Recognize some C keywords: `int`, `double`, `if`, `else`, `while`, `print`
- Recognize integer and double numbers
- Any `#include`, space, tab and newline must be discarded

The scanner must provide as output the recognized units (the TOKENS) printing them separated by a space. Regarding the integers, the double numbers and the identifiers, it must also print, in addition to the recognized token, the token value (e.g. `INT:30`)

The recognized C language, as shown in the example, is a simplified version where functions do not exist and the variables can be declared only at the beginning of the file.

NB: The scanner written for this laboratory will be used in the next laboratories for building a complete compiler. This compiler will be able to compile programs similar to the one shown in the following example.

1.1 Input file example

Observe the example. Given the following input file:

```
/* Example of Bubble sort sorting algorithm*/

double x[5];
int i, j;
double swap;
int pos;

/* Vector initialization */
x[0] = -2.0;
x[1] = -3.0;
x[2] = 3.0;
x[3] = 5.0;
x[4] = 2.5;

/* Bubble sort */
pos = 5;
while(pos > 0){
  i = 0;
  while (i < pos - 1){
    j = i + 1;
    if (x[i] > x[j]){
      swap = x[j];
      x[j] = x[i];
      x[i] = swap;
    }
    i = i + 1;
  }
  pos = pos-1;

}

/* Print results */
i = 0;
while(i<5){
  print (x[i]);
  i = i + 1;
}
```

1.2 Output example

the output must be the following:

```
DOUBLE_TYPE ID:x SO INT:5 SC S INT_TYPE ID:i C ID:j
S DOUBLE_TYPE ID:swap S INT_TYPE ID:pos S ID:x SO
INT:0 SC EQ MIN DOUBLE:2.0 S ID:x SO INT:1 SC ...
...
ID:pos EQ INT:5 S WHILE RO ID:pos MAJ INT:0 RC BO ID:i
EQ INT:0 S WHILE RO ID:i MIN ID:pos MIN ID:1 RC BO ID:j
...
```

2 Exercise (Grammar derived from an exam)

Using Jflex and Cup, write a scanner and a parser which recognize the language for the management of a library.

The input file is subdivided into two sections separated by the symbol “%%” (two percent symbols).

The first section is composed by a non-empty list of writers and the books written by them. Each list element has the following fields:

<writer name> > <books list>;

Where <writer name> is a string of letters enclosed by the characters " (Double quote). <books list> is a non-empty list of books written by a writer and separated by a "," (comma).

Each list element is composed as follow:

<ISBN code>:<book title>:<number of pages>:<collocation>

<ISBN code> consists of two numeric characters, followed by a dash, followed by two numeric characters, followed by a dash, followed by 5 hexadecimal characters, followed by a dash and followed by a letter or a numeric character. <collocation> (is optional) and is composed by the word LI or LS (*letteratura italiana* or *letteratura straniera*) followed by the genre AV, BO o SO (*Avventuroso*, *biografico* or *sociale*), followed by an integer number and eventually followed by a letter. The genre LI BO does not exist: handle this case.

The second section is composed of a non-empty list of users. Each list element is defined as follows:

<user name>:<loans list>;

<loans list> is a set of loans associated to a library user, separated by the character "," (comma).

For each loan the loan date and the book ISBN code are reported. The date is in the format "DD/MM/YYYY", where DD is a number between 01 and 31, MM is a number between 01 and 12.

The program must recognize the previously described language and write if it is grammatically correct.

2.1 Input file example

```
"Hesse Herman" -> 88-17-83457-X:"Narciso e Boccadoro":200:LS SO 127 A,
                  88-14-24B43-2:"Siddhartha" : 236 : LS SO 127 B,
                  88-12-34AA3-B:"Lupo della steppa, Il":262:LS SO 127 C;
"Baricco Alessandro"-> 88-17-10625-9:"Seta":100:LI AV 1,
                       88-17-86563-X:"City":319:LI AV 2 A;
"F. Christiane"-> 88-17-11520-7:"Noi, i ragazzi dello zoo di Berlino":346:LS BO 1;
%%
"Giovanni": 02/10/2006 88-17-11520-7;
"Stefano" : 12/04/2007 88-17-83457-X,
            20/09/2007 88-14-24B43-2,
            29/09/2007 88-17-11520-7;
"Giovanni": 02/10/2007 88-17-10625-9,
            02/10/2007 88-17-86563-X;
```

3 Exercise

Write a lexical analyzer using JFLEX which is able to recognize the principal elements of an HTML document. An HTML document consists of an ASCII text annotated by appropriate keywords.

All the keyword are enclosed between the symbols "<" and ">". Within these symbols there could also be some modifiers and keyword parameters. Keywords and parameters are case insensitive. The two delimiter characters and their content define a tag. The keywords consist of alphanumeric characters and begin with an alphabetic character. Furthermore, the closing tags can be preceded by the character "/". An HTML document can contain some comments. Comments start with characters "<!−−" and end with characters "−− >". Among the keywords that could appear, the lexical analyzer must explicitly recognize the keywords "head", "body", "html", "title", "table", "h1", "h2", "h3", "h4".

The lexical analyzer must produce as output the HTML input document with all comments removed. Furthermore, at the end of the file it must print the following statistics:

- the total number of tags;
- the number of table, h1, h2, h3 and h4 tags (and the corresponding closing tags).

Some hints

- Use the %caseless jflex directive to generate a case insensitive scanner
- Use the states to distinguish comments and tags from generic text

3.1 Input file example

```
<HTML><HEAD><TITLE>Prova</TITLE></HEAD>
<BODY>
<!-- .... <table>Finta tabella (da non contare)</table> -->
<H1>Titolo_1</h1>
<h2>Titolo_1_1</H2>
Testo <b>vario</b>
<H1>Titolo_2</h1>
<h2>Titolo_2_1</H2>
<table border=2><tr><td>Idem</td></tr></table>
<a href="top.html"><img src="/img/top.gif" alt="top"></a>
<h2>Titolo_2_2</H2>
<table border=0><tr><td>
<table border=0><tr><td> Tabella annidata livello1</td></tr>
</table>
</table>
<!-- I tag che seguono indicano una lista non numerata -->
<ul>
<li><a href="pippo.htm">pippo</a>
<li><a href="pluto.htm">pluto<i>pi&ugrave;</i>pippo</a>
</ul>
<table border=0><tr><td>
<table border=0><tr><td>
<table border=0><tr><td>Tabel la annidata livello 2</td></tr>
</table>
</td></tr>
</table>
</table>
<img src="/img/bottom.gif " height="10" width="10" alt="bottom">
<hr>
```

3.2 Output example

```
<HTML><HEAD><TITLE>Prova</TITLE></HEAD>
<BODY>
<H1>Titolo_1</h1>
<h2>Titolo_1_1</H2>
Testo <b>vario</b>
<H1>Titolo_2</h1>
<h2>Titolo_2_1</H2>
<table border=2><tr><td>Idem</td></tr></table>
<a href="top.html"><img src="/img/top.gif " alt="top"></a>
<h2>Titolo_2_2</H2>
<table border=0><tr><td>
<table border=0><tr><td>Tabella annidata livello1</td></tr>
</table>
</table>
<ul>
<li><a href="pippo.htm">pippo</a>
<li><a href="pluto.htm">pluto<i>pi&ugrave;</i>pippo</a>
</ul>
<table border=0><tr><td>
<table border=0><tr><td>
<table border=0><tr><td>Tabel la annidata livello2</td></tr>
</table>
</td></tr>
</table>
</table>
<img src="/img/bottom.gif " height="10" width="10" alt="bottom">
<hr>
</body></HTML>

Total number of tags: 67
Total number of table tags: 12
Total number of h1 tags: 4
Total number of h2 tags: 6
Total number of h3 tags: 0
Total number of h4 tags: 0
```