# Formal Languages and Compilers

Laboratory n° 1

## 1 Exercise

Given the following regular definitions related to a filename description with an optional path (in DOS format) write, using the JFLEX program, a scanner which recognizes a correct pathname.

### 1.1 Regular Definitions

```
PathFileName → ( Drive : )? (\)? (PathName \ )* FileName (. FileType)?
Drive        → letter
PathName     → id
FileName     → id
FileType     → id
Id           → (letter | digit )( letter | digit )*
```

**letter:** all the alphabet letters and the special characters except: slash, backslash, colon, star, question mark, double quotes, minus, greater, OR symbol.
**digit:** digits from 0 to 9

### 1.2 Examples

```
A:\DIR1\DIR2\FILE1.EXE
E:\FILE1.BAT
DIR1\DIR2\FILE2
\UNIX\FILE.C
A:README.TXT
PIPPO
```

## 2 Exercise

Write the regular definitions and the correspondent JFLEX program for recognizing a subset of the URL defined by HTTP. This subset must include:

- The recognition of more than one scheme (like http:, ftp:, gopher: and similar ones).
- The domains expressed as names (e.g. www.polito.it) or as IP addresses (e.g. 130.192.4.4)
- The use of ports different from the default one for the current scheme (domain followed by ":" and by the number of the alternative port).
- The use of anchors in the HTML file (file name followed by a "#" and by the anchor name).
- Sequences of escape (i.e. the character "%" followed by a pair of exadecimal digits) in any position.

### 2.1 Examples of valid URL

Strings of characters recognized by the lexical analyzer as FileName:

```
http://www.mysite.it/file.html
ftp://10.9.9.71/prova.zip
http://another.site.com/%7Euser/index.html#rif33
nntp://news.site.ch:8181/data/
```

## 3 Exercise

Write a lexical analyzer by means of a JFLEX program which, given one or more C sources as input, recognizes all the language elements (numerical constants, strings, preprocessor directives, keywords and comments) and produces as output an HTML file where each element results highlighted with a different color. All the input files must appear in a unique HTML file preceded by the name of the original source file. Note that in HTML colors are expressed with a "#" followed by three pairs of exadecimal digits, each one represented by a fundamental component of the RGB tern (e.g. "#FF0000" is a bright red, "#0000FF" is blue and so on).

### 3.1 Example

Given the following input file (named `main.c`)

```
#include <stdio.h>
int prova(int i) {
  if (i==1) return 1;
  /* questo è un commento */
  else return 0;
}
```

the HTML output will be:

```
<HTML>
<BODY bgcolor="#FFFFFF">
<H2>main.c</H2>
<CODE>
<FONT COLOR="#00FF00">#include &lt;stdio.h&gt;</FONT>
<BR>
<FONT COLOR="#0000FF">int</FONT> prova(
<FONT COLOR="#0000FF">int</FONT> i)<BR>
{<BR>
<FONT COLOR="#0000FF">if</FONT>
(i==<FONT COLOR="#FF0000">1</FONT>)
<FONT COLOR="#0000FF">return</FONT>
<FONT COLOR="#FF0000">1</FONT>;<BR>
<FONT COLOR="#C0C0C0">/* questo è un commento */
</FONT><BR>
<FONT COLOR="#0000FF">else</FONT>
<FONT COLOR="#0000FF">return</FONT>
<FONT COLOR="#FF0000">0</FONT>;<BR>
}<BR>
</CODE></BODY></HTML>
```