

Linguaggi e Traduttori

Esercitazione n° 2

Esercizio 1:

Realizzare mediante Jflex e Cup uno scanner e un parser che riconoscano un linguaggio per la gestione di una biblioteca.

Il file di ingresso è suddiviso in due sezioni separate dal simbolo “%%” (Due simboli percentuale).

La prima sezione è composta da una lista non vuota di scrittori e dei relativi libri da loro scritti. Ogni elemento della lista ha i seguenti campi:

“<nome scrittore>” -> <lista di libri> ;

Dove <nome scrittore> è una stringa di lettere racchiusa tra i caratteri “ (Doppio apice).

<lista di libri> è una lista non vuota di libri scritti da un determinato scrittore e separati da una “,” (virgola). Ogni elemento è così composto:

<codice ISBN>: <titolo libro>: <numero di pagine>: <collocazione>

<codice ISBN> è formato da due caratteri numerici, seguiti da un trattino, seguito da due caratteri numerici, seguiti da un trattino, seguito da 5 caratteri esadecimali, seguiti da un trattino e seguito da una lettera o da un carattere numerico.

<collocazione> (è opzionale) ed è composta dalla parola LI o LS (letteratura straniera o italiana) seguita dal genere AV, BO o SO (Avventuroso, biografico o sociale), seguito da un numero intero ed eventualmente seguito da una lettera. Non esiste però il genere LI BO, lo si gestisca.

La seconda sezione è composta da una lista non vuota di utenti. Ogni elemento della lista è così definito:

“<nome utente>” : <lista di prestiti> ;

<lista di prestiti> è un insieme di prestiti fatti da un determinato utente della biblioteca, separati dal carattere “,” (virgola).

Per ogni prestito viene riportata una data e il codice ISBN del libro preso in prestito. La data ha il formato “GG/MM/AAAA”, dove GG è un numero tra 01 e 31, MM è un numero tra 01 e 12

Il programma dovrà riconoscere il linguaggio precedente descritto. Si allega un file di esempio:

“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;

Esercizio 2:

Scrivere un analizzatore lessicale mediante JFLEX in grado di riconoscere gli elementi principali di un documento HTML.

Un documento HTML è costituito da un testo ASCII annotato mediante opportune parole chiave.

Tutte le parole chiave sono racchiuse tra i simboli "<" e ">". All'interno di tali simboli possono essere presenti anche eventuali modificatori e parametri delle parole chiave. Per le parole chiave ed i parametri è indifferente l'uso di lettere maiuscole o minuscole. I due caratteri delimitatori con il loro contenuto costituiscono un **tag**. Le parole chiave sono costituite da caratteri alfanumerici, e possono iniziare con un carattere alfabetico. Inoltre, i tag di chiusura possono essere preceduti dal carattere "/". Un documento HTML può anche contenere dei commenti, che iniziano con la serie di caratteri "<!--" e terminano con i caratteri "-->". Tra le varie parole chiave che possono comparire, è richiesto di riconoscere esplicitamente le parole chiave "head", "body", "html", "title", "table", "h1", "h2", "h3", "h4".

L'analizzatore lessicale deve produrre in uscita il documento HTML in ingresso, depurato dei commenti. Inoltre, in coda deve stampare una serie di statistiche:

- il numero totale di tag che compaiono;
- il numero di tag table, h1, h2, h3 e h4 (e dei corrispondenti tag di chiusura).

Suggerimenti:

- Utilizzare la direttiva `%caseless` di `jflex` per generare uno scanner case insensitive
- Utilizzare gli stati per distinguere commenti e tag dal testo generico

Dato il seguente documento in ingresso:

```
<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"></a>
<h2>Titolo_2_2</H2>
<table border=0><tr><td>
<table border=0><tr><td> Tabella annidata livello 1</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>Tabella annidata livello 2</td></tr>
</table>
</td></tr>
</table>
</table>

<hr>
```

```
<!-- Fine dell'esempio -->
</body></HTML>
```

Il traduttore dovrebbe visualizzare il seguente output:

```
<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"></a>
<h2>Titolo_2_2</H2>
<table border=0><tr><td>
<table border=0><tr><td> Tabella annidata livello 1</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>Tabella annidata livello 2</td></tr>
</table>
</td></tr>
</table>
</table>

<hr>
</body></HTML>
```

Numero totale di tag: 67

Numero di tag table: 6

Numero di tag h1: 4

Numero di tag h2: 6

Numero di tag h3: 0

Numero di tag h4: 0

Esercizio 3:

Realizzare uno scanner per il linguaggio C.

Lo scanner dovrà:

- Eliminare i commenti tipici del linguaggio C (/* e */)
- Riconoscere le parole chiavi del C: signed, unsigned, int, long, short, float, double, char, long, short, extern, register, auto, static, void, if, else, while, switch, case, break, for, return, default
- Riconoscere i simboli: “{“, “}”, “(“, “)”, “[“, “]”, “+”, “-”, “*”, “/”, “%”, “=”, “;”, “:”, “.”, “,”

Gli interi, i floating point, gli identificatori, le stringhe

- Gli operatori relazionali e logici
- Eventuali #include, spazi, tabulazioni e newline dovranno essere scartati

Lo scanner dovrà fornire in output le unità riconosciute (i TOKEN) stampandoli separati da uno spazio. Per quanto riguarda gli interi, i floating point, gli identificatori e le stringhe, dovrà stampare oltre al TOKEN riconosciuto anche il valore (Es. INTEGER:30)

Si osservi l'esempio. Dato il seguente file di ingresso:

```
#include <stdio.h>
int prova(int i) {
```

```
float b;  
b = 3.433E-02;  
i = 3;  
if (i==1) return 1;  
/* questo Ã un commento */  
else return 0;  
}
```

Si dovrÃ ottenere il seguente output:

```
INT ID:prova TO INT ID:i TC GO FLOAT ID:b PV ID:b UGUALE  
NUM_FLOAT:3.433E-02 PV ID:i UGUALE NUM_INT:3 PV IF TO ID:i UGUALE  
UGUALE NUM_INT:1 TC RETURN NUM_INT:1 PV ELSE RETURN NUM_INT:0 PV  
GC
```