Linguaggi e Traduttori

Esercitazione di laboratorio n° 4

1 Esercizio (mini C - Gestione errori sintattici)

Partendo dallo scanner e dal parser per il riconoscimento del linguaggio **mini C** realizzato nell'esercitazione precedente, usare il simbolo predefinito **error** per gestire gli errori sintattici di tale linguaggio.

Ad esempio il parser potrà segnalare i seguenti errori sintattici, indicando in caso di file di input errato la riga e la colonna dove si è verificato l'errore:

- *Error in declaration*: errore in una dichiarazione di variabile
- *Missing ; before* }: mancato inserimento del simbolo ';' dopo uno statement
- *Error in expression*: errore in un'espressione matematica, booleana o di confronto
- *Error in assignment*: errore in un assegnamento
- *Error in 'print' instruction*: errore in un'istruzione di tipo `print`
- *Error 'else' expected in 'if' instruction*: se manca la keyword `else` in un costrutto di tipo `if`
- *Error in 'if' condition*: un errore nella condizione del costrutto `if`
- *Error '(' expected in 'if' instruction* o *Error ')' expected in 'if' instruction*: se rispettivamente non è stato messo il simbolo '(' o ')' in un'istruzione di tipo `if`
- *Error in 'while' condition*: un errore nella condizione del costrutto `while`
- *Error '(' expected in 'while' instruction* o *Error ')' expected in 'while' instruction*: se rispettivamente non è stato messo il simbolo '(' o ')' in un'istruzione di tipo `while`
- *Error in vector*: errore nell'utilizzo di un vettore (manca il simbolo '[' o tra le due parentesi quadre vi è un simbolo o una sequenza di simboli non corretti)
- *Error in statement*: un errore generico all'interno di uno statement

2 Esercizio

Scrivere, utilizzando i programmi JFLEX e CUP, un parser che riconosca il linguaggio di seguito descritto e che sia in grado di indicare le strutture (regole, fatti o interrogazioni) errate, attraverso l'uso del simbolo predefinito **error**.

2.1 Linguaggio di ingresso

Un programma logico è costituito da un insieme non vuoto di **fatti**, da un insieme, eventualmente vuoto, di **regole**, da una e una sola **interrogazione** e da un numero qualsiasi di **commenti**, tra loro inframezzati in qualsiasi ordine.

Un fatto è costituito da un **predicato** seguito dal carattere '.'.

Una regola è costituita da un predicato seguito dal simbolo ':-' seguito da una lista non vuota di predicati separati dal carattere ',' e terminati dal carattere '.'.

Un'interrogazione è costituita dal simbolo '?-' seguito da una lista non vuota di predicati separati dal carattere ',' e terminata dal carattere '.'.

Un commento è una stringa di caratteri racchiusa tra i simboli '/*' e '*/'.

Un predicato è costituito da un **funtore** seguito da una lista non vuota di argomenti separati dal carattere ',' terminata dal carattere ')'; alternativamente un predicato è un semplice **atomo**.

Un funtore è un atomo immediatamente seguito dal carattere '('.

Un argomento è un predicato oppure una **variabile**.

Un atomo è una stringa di lettere, numeri e '_' il cui primo carattere sia una lettera minuscola, oppure un numero, intero o reale, con o senza esponente, con o senza segno.

Una variabile è una stringa di lettere, numeri e '_' il cui primo carattere sia una lettera maiuscola o il carattere '_'.

Il programma dovrà indicare riga e colonna in cui si è verificato un'errore.

2.2 Esempio file di input

```
/* Esempio di programma logico */
/* appartenenza ad una lista */
member(X,cons(X,_)).
member(X,cons(_,Y)):-
member(X,Y).
/* lista di partenza */
start_list(cons(a,cons(b,cons(c,nil)))).
/* interrogazione */
?- start_list(L), member(X,L), goal(X).
/* goal */
goal(c).
```