2 Analisi lessicale

Gli esercizi di questa sezione riguardano l'implementazione di un analizzatore lessicale per un semplice linguaggio di programmazione. Lo scopo di un analizzatore lessicale è di leggere un testo e di ottenere una corrispondente sequenza di token, dove un token corrisponde ad un'unità lessicale, come un numero, un identificatore, un operatore relazionale, una parola chiave, ecc. Nelle sezioni successive, l'analizzatore lessicale da implementare sarà poi utilizzato per fornire l'input a programmi di analisi sintattica e di traduzione.

I token del linguaggio sono descritti nel modo illustrato in Tabella 1. La prima colonna contiene le varie categorie di token, la seconda presenta descrizioni dei possibili lessemi dei token, mentre la terza colonna descrive i nomi dei token, espressi come costanti numeriche.

Token	Pattern	Nome
Numeri	Costante numerica	256
Identificatore	Lettera seguita da lettere e cifre	257
Relop	Operatore relazionale (<,>,<=,>=,==,<>)	258
Assegnamento	`assign`	259
To	`to`	260
If	`if`	261
Else	`else`	262
While	`while`	263
Begin	`begin`	264
End	`end`	265
Print	`print`	266
Read	`read`	267
Disgiunzione	`||`	268
Congiunzione	`&&`	269
Negazione	`!`	33
Parentesi tonda sinistra	`(`	40
Parentesi tonda destra	`)`	41
Parentesi graffa sinistra	{	123
Parentesi graffa destra	}	125
Somma	`+`	43
Sottrazione	`-`	45
Moltiplicazione	`*`	42
Divisione	`/`	47
Punto e virgola	`;`	59
Virgola	`,`	44
EOF	Fine dell'input	-1

Tabella 1: Descrizione dei token del linguaggio

Gli identificatori corrispondono all'espressione regolare:

$$(\texttt{a} + ... + \texttt{z} + \texttt{A} + ... + \texttt{Z})(\texttt{a} + ... + \texttt{z} + \texttt{A} + ... + \texttt{Z} + \texttt{0} + ... + \texttt{9})^*$$

e i numeri corrispondono all'espressione regolare $\texttt{0} + (\texttt{1} + ... + \texttt{9})(\texttt{0} + ... + \texttt{9})^*$.

L'analizzatore lessicale dovrà ignorare tutti i caratteri riconosciuti come "spazi" (incluse le tabulazioni e i ritorni a capo), ma dovrà segnalare la presenza di caratteri illeciti, quali ad esempio `#` o `@`.

L'output dell'analizzatore lessicale dovrà avere la forma $\langle \text{token}_0 \rangle \langle \text{token}_1 \rangle \cdots \langle \text{token}_n \rangle$. Ad esempio:

- per l'input `assign 300 to d;` l'output sarà $\langle 259, \text{assign} \rangle$ $\langle 256, 300 \rangle$ $\langle 260, \text{to} \rangle$ $\langle 257, \text{d} \rangle$ $\langle 59 \rangle$ $\langle -1 \rangle$;