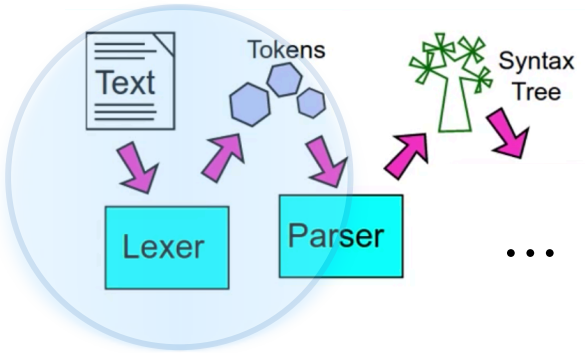

2. Analisi lessicale (parte II - es. 2.2. e 2.3)

Implementazione in Java di un lexer per un semplice linguaggio di programmazione

Esercizio 2.2

- **Estensione della definizione degli identificatori**
- Consideriamo la seguente nuova definizione di identificatori): un identificatore è composto da una sequenza non vuota di lettere, numeri, ed il simbolo di 'underscore' _ che:
  - non comincia con un numero e che
  - non può essere composto solo dal simbolo _. Più precisamente, gli identificatori corrispondono all'espressione regolare:

$$\left([\mathtt{a}-\mathtt{zA}-\mathtt{Z}] \mid (\_(\_)^*[\mathtt{a}-\mathtt{zA}-\mathtt{Z0}-\mathtt{9}])\right)\left([\mathtt{a}-\mathtt{zA}-\mathtt{Z0}-\mathtt{9}] \mid \_\right)^*$$

- Estendere il metodo lexical_scan per gestire identificatori che corrispondono alla nuova definizione.

Esercizio 2.3

- **Riconoscimento di commenti nel file sorgente**
- Estendere il metodo lexical_scan in modo tale che possa trattare la presenza di commenti nel file di input. I commenti possono essere scritti in due modi:
  - commenti delimitati con /* e */ ;
  - commenti che iniziano con // e che terminano con un a capo oppure con EOF.
- I commenti devono essere ignorati dal programma per l'analisi lessicale;
- In altre parole: per le parti dell'input che contengono commenti, non deve essere generato nessun token.

Esercizio 2.3

- Ad esempio, consideriamo il seguente input, l'output del programma per l'analisi lessicale sarà:

⟨259, assign ⟩
⟨256, 300 ⟩
⟨260, to ⟩
⟨257, d ⟩
⟨59 ⟩
⟨259, assign ⟩
⟨256, 10 ⟩
⟨260, to ⟩
⟨257, t ⟩
⟨59 ⟩
⟨268, print ⟩
⟨40 ⟩
⟨42 ⟩
⟨257, d ⟩
⟨257, t ⟩
⟨41 ⟩
⟨−1 ⟩

```
/* calcolare la velocita` */
assign 300 to d; // distanza
assign 10 to t; // tempo
print(* d t)
```

Esercizio 2.3

- Oltre alle coppie di simboli /*, */ e //, un commento può contenere simboli che non fanno parte del pattern di nessun token, ad esempio:
  - `/*@#?*/` o
  - `/* calcolare la velocita' */` (apostrofo!)
- Se un commento di forma `/* ... */` è aperto ma non chiuso prima della fine del file deve essere segnalato un errore
  - ad esempio il caso di input

```
assign 300 to d; /* distanza
```

Esercizio 2.3

- Si noti che ci possono essere due commenti di seguito non separati da nessun token, ad esempio:

```
assign 300 to d; /*distanza*//*da Torino a Lione*/
```

Esercizio 2.3

- La coppia di simboli */ se scritta al di fuori di un commento, deve essere trattata dal lexer come il segno di moltiplicazione seguito dal segno di divisione
  - ad esempio, per l'input

    `x*/y`

  - l'output sarà

    ⟨257, x⟩

    ⟨42⟩

    ⟨47⟩

    ⟨257,y⟩

    ⟨−1⟩

- In altre parole: l'idea è che in questo caso la sequenza di simboli */ non verrà interpretata come un commento da saltare ma come una sequenza dei due token menzionati (moltiplicazione e divisione).

FAQ - lezione precedente

Salve,

Sono una studentessa del suo laboratorio di Linguaggi Formali e Traduttori Corso A.

Vorrei chiederle delucidazioni riguardo all'esercizio 2.1/2.2 del Lexer.

Mi chiedevo se, nella definizione di identificatore, sono incluse anche le parole che contengono numeri al suo interno.

Per esempio, "abcd1234bs" viene riconosciuta come "acbd1234bs" o come "abcd" "1234" "bs"?

E invece, "1234abcd" come viene riconosciuta?

E invece "0245"?