2. Analisi lessicale

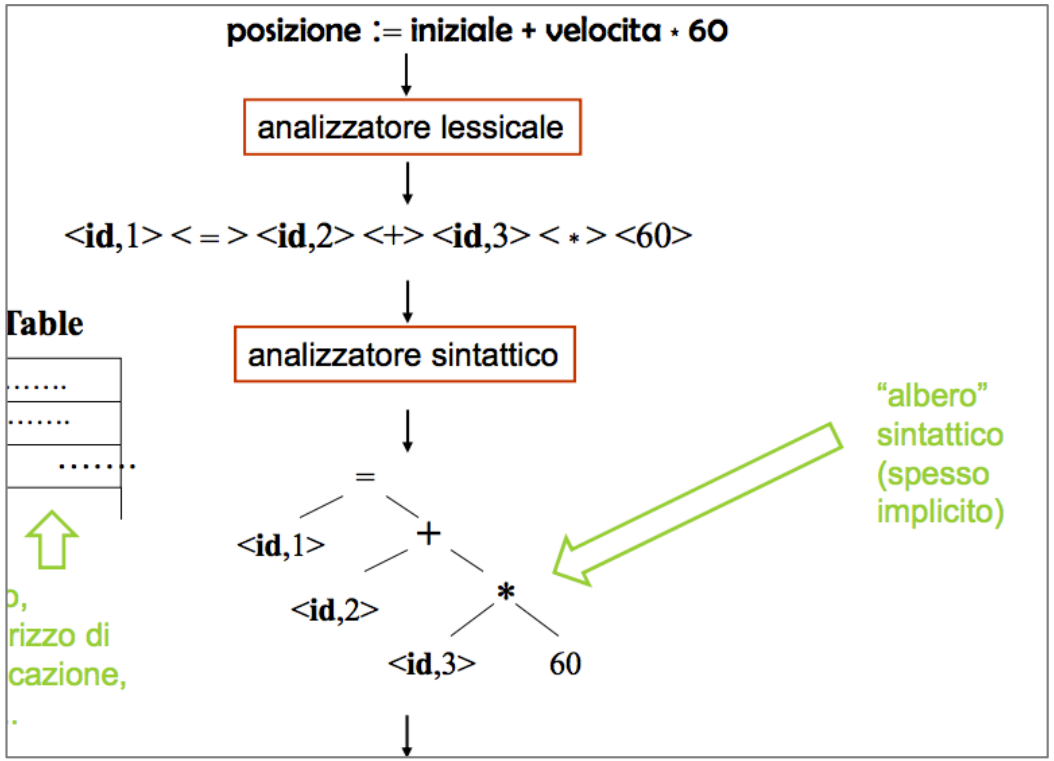
Implementazione in Java di un lexer per un semplice linguaggio di programmazione

Analizzatore lessicale

- Cosa fa:
  - **Input:**
    - legge un testo scritto in un certo linguaggio, nel nostro caso una **stringa di caratteri di un programma sorgente** scritto in un qualche linguaggio di programmazione dato e raggruppa i caratteri in sequenze chiamate **lessemi**
    - **Lessemi. Intuizione:** sequenze complete e dotate di senso per il linguaggio in oggetto; in altri termini: sequenze di caratteri ("parole") del programma che rispettano un pattern di un token per quel linguaggio di programmazione.
  - **Output:** restituisce una corrispondente **sequenza di token**, dove un token corrisponde ad un'unità lessicale per quel linguaggio, nel caso di un linguaggio di programmazione ad esempio:
    - un numero
    - un identificatore
    - un operatore relazionale
    - una parola chiave, (if, else, print)
    - …

Analizzatore lessicale

- Dove si posiziona in una *pipeline di traduzione*?
- Nelle sezioni successive, l'analizzatore lessicale implementato fornirà l'input a programmi di analisi sintattica e di traduzione (step successivi all'analisi sintattica).



(...continua...)

Analisi lessicale (teoria)

- *Token*: coppia <nome token, valore attributo>
  nome token: simbolo astratto che rappresenta un'unita` lessicale (una parola chiave, un identicatore, ecc.)
- *Pattern*: descrizione della forma che i lessemi di un'unita` lessicale possono avere.

  Esempio: se il token e` una parola chiave, il pattern e` la sequenza di caratteri che formano la parola chiave. Per gli identicatori, il pattern descrive stringhe di caratteri, che sono tutte identificatori.
- *Lessema*: sequenza di caratteri del programma sorgente che rispetta il pattern di un token

  Esempi:

  nome token **if**, pattern: caratteri i seguito da f, lessema: **if**
  nome token **id**, pattern: una lettera seguita da lettere e cifre
  esempi di lessemi: **posizione, iniziale**
  nome token **num**, pattern: un numero
  esempi di lessemi: **3.14159, 0, 6.2**

Il nostro linguaggio: i token

- I token del linguaggio sono descritti nel modo illustrato in **Tabella 1**.

Categorie di token

Possibili pattern dei token

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
Case	`case`	259
When	`when`	260
Then	`then`	261
Else	`else`	262
While	`while`	263
Do	`do`	264
Assegnamento	`:=`	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
EOF	Fine dell'input	-1

Tabella 1: Descrizione dei token del linguaggio

Esempio di programma sorgente

d :=300;
print(d*t)

- L'analizzatore lessicale deve ignorare tutti i caratteri riconosciuti come "spazi" (incluse le tabulazioni e i ritorni a capo), ma deve 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 `d:=300;` l'output sarà $\langle 257, \text{d} \rangle \langle 265, := \rangle \langle 256, 300 \rangle \langle 59 \rangle \langle -1 \rangle$;
- per l'input `print(d*t)` l'output sarà $\langle 266, \text{print} \rangle \langle 40 \rangle \langle 257, \text{d} \rangle \langle 42 \rangle \langle 257, \text{t} \rangle \langle 41 \rangle \langle -1 \rangle$;
- per l'input `case when x>y then x:=0 else y:=0` l'output sarà $\langle 259, \text{case} \rangle \langle 260, \text{when} \rangle \langle 257, \text{x} \rangle \langle 258, > \rangle \langle 257, \text{y} \rangle \langle 261, \text{then} \rangle \langle 257, \text{x} \rangle \langle 265, := \rangle \langle 256, 0 \rangle \langle 262, \text{else} \rangle \langle 257, \text{y} \rangle \langle 265, := \rangle \langle 256, 0 \rangle \langle -1 \rangle$;
- per l'input `while (cases<=printread) do cases:=cases+1` l'output sarà $\langle 263, \text{while} \rangle \langle 40 \rangle \langle 257, \text{cases} \rangle \langle 258, <= \rangle \langle 257, \text{printread} \rangle \langle 41 \rangle \langle 264, \text{do} \rangle \langle 257, \text{cases} \rangle \langle 265, := \rangle \langle 257, \text{cases} \rangle \langle 43 \rangle \langle 256, 1 \rangle \langle -1 \rangle$.

Token con e senza attributi

- In generale, i token della Tabella 1 hanno un attributo: ad esempio, l'attributo del token <256, 300 > è il numero 300, mentre l'attributo del token <259, case> è la stringa "case ".
- Alcuni token della Tabella 1 sono senza attributo: ad esempio, il segno di moltiplicazione * è rappresentato dal token <41> , e la parentesi tonda destra) è rappresentata dal token <125> .

Pattern di identificatori e numeri

- Gli identificatori corrispondono all'espressione regolare

$$[a - zA - Z][a - zA - Z0 - 9]*$$

- i numeri corrispondono all'espressione regolare

$$0 \mid [1 - 9][0 - 9]*$$

Analisi lessicali e errori

- Attenzione: l'analizzatore lessicale non è preposto al riconoscimento della struttura sintattica dei comandi del linguaggio. Pertanto, restituirà senza batter ciglio sequenze di token anche per frammenti di programma che vi suonano "errati", ad esempio:

```
5+)
else 5 == print < case
```

- Altri errori invece, come simboli non previsti o sequenze che non corrispondono a nessun pattern per nessun token previsto devono essere rilevate.

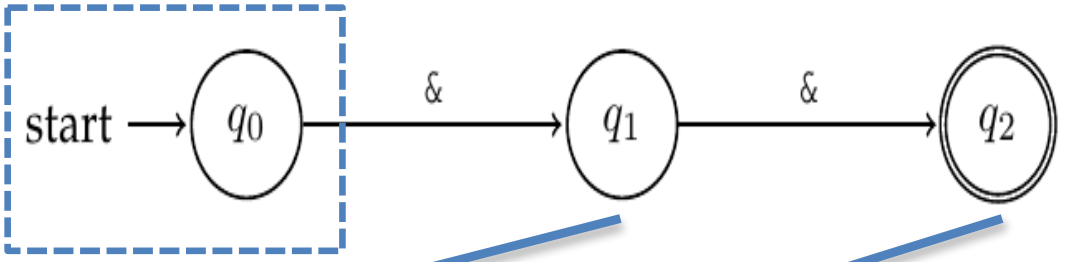
  - nel caso dell'input `17&5` ?
  - oppure dell'input `|||` ?

Esercizio 2.1

- Si scriva in Java un analizzatore lessicale che
  - legga da file un input e
  - stampi la sequenza di token corrispondente.
- Per questo esercizio, si possono utilizzare senza modifica le classi Tag, Token e Word (download da Moodle).
- Invece devono essere completate
  - la classe NumberTok
  - la classe Lexer
  - A partire dalle implementazioni di partenza scaricabili su Moodle

NumberTok

- Idea: ispirandosi alla classe Word estendere Listing 5 per definire una classe NumberTok e rappresentare i token che corrispondono ai numeri
  - Caso simile a word (token con attributo) ma …
    - Cos'e' il lessema nel caso dei numeri?
    - Ha senso trattarlo come una stringa di caratteri?




```
case '&':
    readch(br);
    if (peek == '&') {
        peek = ' ';
        return Word.and;
    } else {
        System.err.println("Erroneous character"
                + " after & : "  + peek );
        return null;
    }
```

Più complicati casi come > o < dove per capire cosa tokenizzare occorre andare a guardare il simbolo successivo

Input retraction:
Libro di testo 3.4

Identificatori e parole chiave

```
// ... gestire il caso degli identificatori e delle parole chiave //

        } else if (Character.isDigit(peek)) {
```

Restituire uno degli oggetti specificati nella classe Word

```
public class Word extends Token {
    public String lexeme = "";
    public Word(int tag, String s) { super(tag); lexeme=s; }
    public String toString() { return "<" + tag + ", " + lexeme + ">"; }
    public static final Word
    casetok = new Word(Tag.CASE, "case"),
    when = new Word(Tag.WHEN, "when"),
    then = new Word(Tag.THEN, "then"),
    elsetok = new Word(Tag.ELSE, "else"),
    whiletok = new Word(Tag.WHILE, "while"),
    dotok = new Word(Tag.DO, "do"),
    assign = new Word(Tag.ASSIGN, ":="),
    print = new Word(Tag.PRINT, "print"),
    read = new Word(Tag.READ, "read"),
```

Identificatori e parole chiave

```
// ... gestire il caso degli identificatori e delle parole chiave //

        } else if (Character.isDigit(peek)) {
```

Occorre istanziare e restituire
un nuovo oggetto Word che ha
ID come tag e la stringa che corrisponde
al lessema come secondo parametro