|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Parser LL | | Parser top-down per riconoscimento di costrutti “if-then-else” |  Introduzione L’obbiettivo del parser di cui segue è quello di riconoscere la corretta scrittura e nidificazione di costrutti condizionali “if-then-else”, mostrando a fine elaborazione quali siano i punti di inizio di ciascun costrutto e, nel caso, quali termini non siano validi e riconosciuti dalla grammatica adottata. Grammatica  1. < All > ::= < Text > $ 2. < Text > ::= < IfteConstruct > < Text > | notIfte < Text > | ε 3. < IfteConstruct > ::= if notIfte then < Block > < OptElse > endif 4. < Block > ::= < IfteConstruct > | notIfte 5. < OptElse > ::= else < Block > | ε   I simboli terminali sono: if, then, else, endif, notIfte.  First Sets:   * All : { notIfte, if, ε } * Text : { notIfte, if, ε } * Ifte : { if } * Block : { notIfte, if } * OptElse : { else, ε }   Follow Sets:   * All : { $ } * Text : { $ } * Ifte : { notIfte, if, else, endif } * Block : { else, endif } * OptElse : { endif }   La grammatica è LL(1), poiché non sono presenti ambiguità nelle produzioni della grammatica adottata. Parsing Table  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | if | then | else | notIfte | endif | $ | | All | R0 | Error | Error | R0 | Error | R7 | | Text | R1 | Error | Error | R2 | Error | R7 | | IfteConstruct | R3 | Error | Error | Error | Error | Error | | Block | R4 | Error | Error | R5 | Error | Error | | OptElse | Error | Error | R6 | Error | R7 | Error | |
| Implementazione Il parser LL è stato implementato in Python 2.7 e utilizza un sistema di indicizzazione per ciascuno dei vari token, regole e routines semantiche per districarsi nell’analisi: a ciascuna di queste viene data una tag per la tipologia (Terminale, Regola, Routine) e un corrispondente indice univoco all’interno della classe previamente specificata per distinguere ogni passaggio.  Per esempio: ***(Rule, N\_IFTE)*** è una coppia (int, int) che sta a indicare che stiamo trattando una regola e precisamente quella relativa al costrutto “if-then-else”.  Sulla base di queste indicazioni il metodo che si occupa dell’analisi sintattica può gestire lo stack, che, in base al contenuto della parsing table sopra citata e delle regole (comprensive di routine semantiche dove necessario) sostituirà ogni regola con le derivazioni adeguate e prosegurie nell’analisi. Output Di seguito gli output relativi ai corrispondenti input inseriti nel programma. Per ogni file di testo viene specificato se vengono trovati costrutti “if-then-else” o meno e, nel caso, si indica la posizione della parola con cui inizia il costrutto. Nel caso di errore viene notificata la posizione dell’elemento che non rispetta la grammatica somministrata.   * “abcdef”, solo testo:   Number of 'if' constructs recognised: 0.  No if constructs found.   * “if ciao then come else stai endif”, if-then-else semplice:   Number of 'if' constructs recognised: 1.  If construct number 1 found at 1.   * “if proviamo then un else così endif testo in mezzo if perchè then si endif”, if separati da testo:   Number of 'if' constructs recognised: 2.  If construct number 1 found at 1.  If construct number 2 found at 11.   * “testo all'inizio if prova then if questo then è nidificato endif else continua endif”, testo iniziale e if nidificati:   Number of 'if' constructs recognised: 1.  If construct number 1 found at 3.   * “stavolta sbaglio then apposta endif”, if mancante all’inizio:   Error: unexpected value (token index:3) at word n° 3.   * “if ultima then if prova then vedere else cosa endif else funziona”, endif mancante al termine del costrutto:   Error: unexpected value (token index:5) at word n° 12. |