**README ESERCITAZIONE 3**

**LINK:** https://gitlab.com/g4660/compilatori-a.a.-2021\_22/hand-coded-parser-es3\_hcp/panichella\_es3\_hcp.git

L’esercitazione 5 è stata sviluppata seguendo con il “Project SDK” e il “Project Language Level” seguente:

Graphical user interface, application

Description automatically generated

Nell’esercitazione 3 si è costruito un parser a discesa ricorsivo tramite delle funzioni che vanno a simulare lo sviluppo di ogni non terminale.

La grammatica in esame è la seguente, anche se è stata aggiornata per far si che rispettasse delle regole adatte al parsing top down.

(  
 **N** = {S, Program, Stmt, Expr},  
 **T** = {EOF. ';', IF, THEN, ELSE, END, ID, ASSIGN, WHILE, LOOP, RELOP, NUMBER},    
 **S**  
 **P** = {

          S -> Program  EOF

          Program -> Program ; Stmt  
                           |  Stmt  
           Stmt -> IF Expr THEN Stmt ELSE Stmt END IF  
                     | ID ASSIGN Expr

                     | WHILE Epr LOOP Stmt END LOOP

          Expr ->  Expr  RELOP Expr

          Expr ->   ID  
                     |  NUMBER  
       }  
)

Trasformata in

S -- > Program EOF

Program 🡪 Stmt Program1

Program1 🡪; Stmt Program1 | epsilon

Stmt 🡪 IF Expr THEN Stmt ELSE Stmt END IF | ID ASSIGN Expr | WHILE Expr LOOP Stmt END LOOP

Expr 🡪 ID Expr1 | NUMBER Expr1

Expr1 🡪 Relop Expr Expr1 | epsilon

RELOP 🡪 OP\_MIN | OP\_MIN\_EQ | OP\_MIN\_MAX | OP\_MAX | OP\_MAX\_EQ

All’interno di questa esercitazione, 2 sono stati i file principalmente sviluppati:

* Token (che gestisce gli oggetti token creati dal lexer);
* Lexer;
* Tester.

Il Tester è il programma principale che effettua la verifica se un determinato testo appartiene o meno al linguaggio in esame. Attravero la funzione nextToken() del lexer, va ad aggiungere ad un array di token, il token che il lexer gli restituisce e una volta che ha finito di analizzare il file, effettua il controllo che effettivamente, quella successione di token mandatagli dal lexer appartenga alla grammatica.

Il loken EOF è aggiunto alla fine dell’array direttamente dal Tester, in modo tale da evitare di fare un ulteriore ciclo nel lexer.

All’interno del lexer quello che si è fatto, è stato quello di andare ad analizzare, carattere per carattere il file e nel momento in cui viene trovata una certa corrispondenza, viene creato un nuovo token e restituito al RecDecParser.

Vari sono stati i test che sono stati svolti per verificare il funzionamento e per ogni test creato, è presente sia la versione giusta che la versione contenente l’errore.

TESTS:

* testFile8\_errato: all’interno dell’if, dopo dell’else è ammesso un altro stmt e non un confronto;
* testFile9\_errato: l’ultimo statement non deve chiudersi con il punto e virgola;
* testFile10\_errato: dopo di uno stmt non ci può essere ID e poi Stmt di nuovo;
* testFile11\_errato: dopo del THEN nell’if deve per forza essere un stmt, non può andare in vuoto;
* testFile12\_errato: un if deve avere l’else;
* testFile13\_errato: dopo del LOOP (nel while) deve esserci uno Stmt;
* testFile14\_giusto: test per vedere se un input generico senza spazi non dà errori lessicali;