

## Lab 8 DOCUMENTATION

[https://github.com/lola23/LFTC/tree/master/Lab\\_08/src](https://github.com/lola23/LFTC/tree/master/Lab_08/src)

# LL 1

Fields:

grammar: Grammar

FIRST: map of non terminal with the non terminals that it has as first

FOLLOW: map of every non terminal and next symbol generated after it

numberedProductions: map of all productions with an index

alpha: stack for terminals

beta: stack for nonterminals

pi: stack of indexes of productions

parseTable()

PRE: -

POST: constructs the parsing table

parse():

In: sequence to be checked

Out: true or false if sequence is accepted or not

numberProductions():

pre: -

post: numberedProductions is populated with every production and its

LL_1		
grammar		Grammar
FIRST		Map<String, List<String>>
FOLLOW		Map<String, Set<String>>
parseTable		Map<Pair<String, String>, Pair<List<String>, Integer>>
numberedProduction:		Map<Pair<String, List<String>>, Integer>
alpha		Stack<String>
beta		Stack<String>
pi		Stack<String>
first(String)		List<String>
FIRST()		void
FOLLOW()		void
copy(Map<String, Set<String>>)		HashMap<String, Set<String>>
numberProductions()		void
generateParseTable()		void
parse(List<String>)		boolean
pushAll(List<String>, Stack<String>)		void
containsKey(Pair<String, String>)		boolean
getParseTable()		String
getProductionsNumbered()		Map<Pair<String, List<String>>, Integer>
getPi()		Stack<String>

Grammar		
N		List<String>
E		List<String>
S		String
P		Map<String, List<List<String>>>
readFromFile()		void
getN()		List<String>
getE()		List<String>
getS()		String
getP()		Map<String, List<List<String>>>
getRhsProductions(String)		Map<String, List<List<String>>>
getProductionsInWhichNTIsOnTheRight(String)		Map<String, List<List<String>>>

Main		
grammar		Grammar
ll1		LL_1
main(String[])		void
parseSequence(List<String>)		void
displayProductions(Stack<String>)		String
printMenu()		void
printNonTerminals()		void
printTerminals()		void
printProductions()		void
printProductionsForANonTerminal(String)		void
buildString(String)		String

P DS SL D BS AD TY IDL AD AA T AAL S SI ST AS E EX EX2 OP IS ST CS IFS EIFS ES C TRY EXT R LS  
TREX

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36

P

P -> 2 DS SL 3

DS -> D DS | ~

D -> BS | AD

BS -> TY 0 IDL 23

TY -> 5 | 6

IDL -> 24 0 IDL | ~

AD -> 7 19 TY 20 0 AA AAL 23

AA -> 19 T 20 | ~

T -> 0 AA | 1

AAL -> 24 0 AA AAL | ~

SL -> S SL | ~

S -> SI | ST

SI -> AS | IS

AS -> 0 13 E 23

E -> 16 EX | EX

EX -> T EX2

EX2 -> OP T | ~

OP -> 31 | 32 | 33 | 34 | 35 | 14 | 15

IS -> 36 17 0 18 23 | 4 17 0 18 23

ST -> CS | IFS | LS

CS -> 21 SL 22

IFS -> 8 C CS EIFS ES

EIFS -> 9 C CS | ~

ES -> 10 CS | ~

C -> 17 TREX 18

TREX -> TRY EXT

EXT -> R TREX | OP TREX | ~

TRY -> 0 | 1

R -> 25 | 26 | 27 | 28 | 29 | 30 | 14 | 15

LS -> 11 17 5 0 23 E 12 E 23 1 18 21 S 22

Example:

*pl.in*

begin\_appy

inty a, b, c, max;

a is 10;

b is 20;

c is 12;

ify ( a >= b and a >= c ) {

max is a;

} elseify ( b >= a and b >= c ) {

max is b;

} elsy {

max is c;

}

end\_appy

*plPIF.out*

2 5 0 24 0 24 0 24 0 23 0 13 1 23 0 13 1 23 0 13 1 23 8 17 0 25 0 14 0 25 0 18 21 0 13 0 23  
22 9 17 0 25 0 14 0 25 0 18 21 0 13 0 23 22 10 21 0 13 0 23 22 3

=> Sequence is accepted

Code review:

<https://github.com/eduardCeausoglu/FLCDLabs/commit/65dae928271b006df17092093080a5336c5ceb05>