GIT LINK: https://github.com/denisababeii/Parser

Class Grammar

read(String file) -> read a grammar from a given file

printNonTerminals() -> print the set of nonterminals

printTerminals() -> print the set of terminals

printProductions() -> print the set of productions

printProductionsForNonTerminal() -> productions for a given nonterminal

checkCFG() -> checks whether the grammar is CFG or not

Class Production

Used to represent a production which includes a List of string “symbols” and a list of list of strings that are the “rules”.

The program reads a grammar from a given file and through a menu, offers the user the possibility to see the nonTerminals, terminals, productions, productions for a certain nonterminal and whether the grammar is CFG(Context Free Grammar) or not.

Two example files are given: OurGrammar.txt and SimpleGrammar.txt

Syntax used:

- program::=function | function program

- function::= "func" "\s" identifier "\s" "{" inputList "}" "=" ">" "{" outputList "}" ":" "\n" "\t stmtlist

- inputList ::= identifier | "," inputList

- outputList ::= identifier | "," outputList

- declarations ::= identifier "=" operator

- stmtlist ::= stmt | stmt "\n" "\t" stmtlist

- stmt ::= instmt | iostmt | ifstmt | forstmt | whilestmt | operationstmt | declarations

- instmt ::= "read" "(" identifier ")"

- iostmt ::= "write" "(" operator ")"

- ifstmt ::= "if" "\s" condition ":" "\n" "\t" stmt [orifstmt] [orstmt]

- orifstmt ::= "\n" orif "\s" condition : stmt

- orstmt ::= "\n" or "\s" : stmt

- forstmt ::= "for" "\s" identifier "\s" "from" "\s" operator "\s" "to" "\s" operator ":"

- whilestmt ::= "while" "\s" condition ":" "\n" "\t" stmt

- condition ::= identifier relation operator

- relation ::= "<" | "<" "=" | "<" "=" ">" | "<" ">" | ">" "=" | ">"

- operationstmt ::= identifier "=" operator operation operator

- operator ::= identifier | constant

- operation ::= "+" | "-" | "/" | "*" | "%"

- identifier::= letter | letter recursive

- recursive::= letter | digit | "_" | letter recursive | digit recursive | "_" recursive

- letter ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"

- digit ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

- constant ::= integer | real | character | string | boolean

- integer ::= "-" number | number | "0"

- number ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" | number digit

- real ::= "-" integerPart "." fractionalPart | integerPart "." fractionalPart

- integerPart ::= digit | number

- fractionalPart ::= digit | fractionalPart digit

- character ::= "'" letter "'" | "'" digit "'" | "'" "_" "'"

- string ::= "'" recursive "'"

- boolean ::= "True" | "False"