# Implementación de un parser

# Reconoce expresiones mediante la gramática:

# EXP -> EXP op EXP | EXP -> (EXP) | cte

# la cual fué modificada para eliminar ambigüedad a:

# EXP -> cte EXP1 | (EXP) EXP1

# EXP1 -> op EXP EXP1 | vacío

# los elementos léxicos (delimitadores, constantes y operadores)

# son reconocidos por el scanner

#

# Autor: Dr. Santiago Conant, Agosto 2014 (modificado Agosto 2015)

# A01064725

# A00816446

import sys

import obten\_token as scanner

# Empata y obtiene el siguiente token

def match(tokenEsperado):

global token

if token == tokenEsperado:

token = scanner.obten\_token()

else:

error("token equivocado")

# Función principal: implementa el análisis sintáctico

def parser():

global token

token = scanner.obten\_token() # inicializa con el primer token

init()

if token == scanner.END:

print("Expresion bien construida!!")

else:

error("expresion mal terminada")

# Módulo que reconoce expresiones

def exp():

if token == scanner.INT or token == scanner.FLT:

match(token) # reconoce Constantes

exp1()

elif token == scanner.LRP:

match(token) # reconoce Delimitador (

exp()

match(scanner.RRP)

exp1()

elif token == scanner.IDT:

match(token) # reconoce identificadores

param()

exp1()

else:

error("expresion mal iniciada")

# Módulo auxiliar para reconocimiento de expresiones

def exp1():

if token == scanner.OPB:

match(token) # reconoce operador

exp()

exp1()

# Módulo auxiliar para diferenciación entre asignaciones y expresiones

def init():

if token == scanner.IDT:

match(token) # reconoce un identificador

if token == scanner.ASG:

match(token) # reconoce el simbolo "=" determinando que es una asignación

exp()

elif token == scanner.LRP:

param() # encontramos un "(" por lo que es una función

elif token == scanner.OPB:

exp1() # encontramos un operador binario haciendo al IDT una variable parte de una expresión

else:

exp()

# Módulo auxiliar para convertir una variable en una función y su lista de argumentos

def param():

if token == scanner.LRP:

match(token) # reconoce Delimitador (

if token == scanner.RRP:

match(token) # reconoce Delimitador )

else:

exp()

params()

# Módulo auxiliar para agregar más argumentos a una función

def params():

if token == scanner.RRP:

match(token) # reconoce Delimitador )

elif token == scanner.CMA:

match(token) # reconoce Separador ,

exp()

params()

else:

error("función mal formada")

# Termina con un mensaje de error

def error(mensaje):

print("ERROR:", mensaje)

sys.exit(1)