Instituto Tecnológico de Costa Rica

Departamento de Computación

Sede Alajuela

Curso:

Compiladores e intérpretes

Profesor:

Aurelio Sanabria

Estudiantes:

Carlos Adán Arguello Calderón - 201173805

Jefri Cárdenas Villatoro -2013101392

Tipo de evaluación:

Laboratorio 2: Parser y AST

Introducción

Para este laboratorio se ha implementado el ‘parser’ (analizador sintáctico) y el ‘ast’ (árbol sintáctico abstracto), para comenzar con esta implementación se realizo anteriormente el laboratorio 1 que consistía en la realización del ‘scanner’. Es importante saber que un parser y un ast, un parser es un programa informático que analiza una serie de símbolos de acuerdo a las reglas de una gramática formal y un ast es una representación de un árbol de la estructura sintáctica simplificada del código fuente escrito en cierto lenguaje de programación.

Implementación del parser

Dado el siguiente código de implementación del parser mas la impresión del resultado. Se explicará paso por paso el código fuente:

# Clase parser

class parser():

#Estas son las variables que inicializan el parser, los tokens que vienen del scanner son enviados aca al parser

def \_\_init\_\_(self, TOKENS):

self.TOKENS = TOKENS #token

self.TOKEN = TOKENS[0][0] # cabecera de token

self.TOKENVALUE = TOKENS[0][1] # valor de token

self.CONTADOR = -1 # contador

self.EXIST = len(self.TOKENS)-1 #contador verifica si existe

def nextt(self): # función que aumenta en 1 para seguir recorriendo

self.CONTADOR += 1

self.TOKEN = self.TOKENS[self.CONTADOR][0]

def nexte(self): #verifica si hay existencia o sea exist debe ser mayor a contador

return self.CONTADOR < self.EXIST

def direccion(self):

pass

def direccionTica(self): # hace el recorrido al finalizar

while(self.nexte()):

self.nextt()

self.inicio()

def inicio(self): # comienza el recorrido de clase tipo token inicio

if(self.TOKEN == "Inicio"):

self.nextt()

self.lugar()

else:

self.error("Se esperaba recibir el Token 'Inicio',", self.TOKEN)

def lugar(self): # comienza el recorrido de clase tipo token lugar

if(self.TOKEN == "Lugar"):

self.nextt()

self.detalles()

else:

self.error("Se esperaba recibir el Token 'Lugar',", self.TOKEN)

def detalles(self): # comienza el recorrido de clase tipo token detalles

if(self.TOKEN == "Detalles"):

self.nextt()

self.distancia()

else:

self.error("Se esperaba recibir el Token 'Detalles',", self.TOKEN)

def distancia(self): # comienza el recorrido de clase tipo token distancia

if(self.TOKEN == "Distancia"):

self.nextt()

self.medida()

else:

self.error("Se esperaba recibir el Token 'Distancia',", self.TOKEN)

def medida(self): # comienza el recorrido de clase tipo token medida

if(self.TOKEN == "Medida"):

self.nextt()

self.conectores()

else:

self.error("Se esperaba recibir el Token 'Medida',", self.TOKEN)

def conectores(self): # comienza el recorrido de clase tipo token conectores

if(self.TOKEN == "Conectores"):

self.nextt()

self.cardinales()

else:

self.error("Se esperaba recibir el Token 'Conectores',", self.TOKEN)

def cardinales(self): # comienza el recorrido de clase tipo token cardinales

if(self.TOKEN == "Cardinales"):

self.nextt()

self.fin()

else:

self.error("Se esperaba recibir el Token 'Cardinales',", self.TOKEN)

def fin(self): # comienza el recorrido de clase tipo token fin

if(self.TOKEN == "Fin"):

pass

else:

self.error("Se esperaba recibir el Token 'Fin',", self.TOKEN)

def error(self, error, esperado): #en caso de que el recorrido no encuentre equivalencia de tipo dará error

print(error, "se recibió el token '"+esperado+"'")

# Función principal

if(\_\_name\_\_ == '\_\_main\_\_'):

scanner = scanner()

scanner.agregar()

scanner.generador\_tokens()

scanner.imprimir\_tokens()

parser = parser(scanner.TOKENS) # se envían los tokens

parser.direccionTica() # se hace recorrido y se va imprimiendo

Ejemplo de impresión:

Ingrese el archivo: C:\Users\Caili\Desktop\laboratorio1\Ejemplos\ejemplo1.txt

Error: El caracter < D > no es válido. Posición inicial: 0 - Posición final: 1

Error: El caracter < , > no es válido. Posición inicial: 55 - Posición final: 56

Error: El caracter < Ã¡ > no es válido. Posición inicial: 75 - Posición final: 77

Lista de TOKENS

<"Detalles", "ele directo como ">

<"Distancia", "6 ">

<"Medida", "cuadras ">

<"Conectores", "y ">

<"Detalles", "luego agarre a la ">

<"Cardinales", "derecha ">

<"Detalles", " ">

<"Conectores", "y ">

<"Detalles", "siga ">

<"Distancia", "3 ">

<"Medida", "cuadras ">

<"Detalles", "m ">

<"Detalles", "s ">

Se esperaba recibir el Token 'Inicio', se recibió el token 'Detalles'

Se esperaba recibir el Token 'Inicio', se recibió el token 'Distancia'

Se esperaba recibir el Token 'Inicio', se recibió el token 'Medida'

Se esperaba recibir el Token 'Inicio', se recibió el token 'Conectores'

Se esperaba recibir el Token 'Inicio', se recibió el token 'Detalles'

Se esperaba recibir el Token 'Inicio', se recibió el token 'Cardinales'

Se esperaba recibir el Token 'Inicio', se recibió el token 'Detalles'

Se esperaba recibir el Token 'Inicio', se recibió el token 'Conectores'

Se esperaba recibir el Token 'Inicio', se recibió el token 'Detalles'

Se esperaba recibir el Token 'Inicio', se recibió el token 'Distancia'

Se esperaba recibir el Token 'Inicio', se recibió el token 'Medida'

Se esperaba recibir el Token 'Inicio', se recibió el token 'Detalles'

Se esperaba recibir el Token 'Inicio', se recibió el token 'Detalles'

Implementación del AST