TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE NUEVO LEÓN

Unidad 2
Lenguajes y Autómatas

ALUMNOS- Christiam Iván Rodríguez Moreno
NO.CONTROL: 13480619
CARRERA: Ingeniería en Sistemas Comunicacionales
ASESOR INTERNO: Juan Pablo Rosas Baldazo

Cd. Guadalupe, N.L. 03 junio, 2018

Índice

Introducción

En este reporte se implementaron 4 partes

*la primera parte que el archivo tenia que leer un archivo .txt en java

*Una vez que el archivo .txt se pudiera compilar y se pudiera leer la información se continuaba,con el punto numero 2 era separar carácter por carácter esto incluye; (símbolos,signos de puntuación, palabras entre otros tipos de caracteres)

* separado el archivo de texto se continuaba con la siguiente fase que era crear una pequeña tabla que comparara los caracteres y asi se pudieran referenciar el archivo

*Ya por ultimo una vez que se obtuvo la información de los caracteres y se podían referenciar ,entonces ya seguía el ordenar el cuádruplas los caracteres asignados y por medio de una cadena acomodar los caracteres en dicha cuádrupla asignada

Descripcion

Primero que nada una de las complicaciones fue el como exportar un archivo de texto externo al código de java aunque este punto no fue tan complicado ya que el archivo lo acabamos en 4 días sin embargo el código debe estar fuera de una clase y meterlo como aplicación y tener un orden de las clases para poder hacer esto recurrimos a diferentes paginas web para poder crear una clase fuera de las clases

Una vez exportado el código y compilado fuera de las clases lo siguiente en la lista era separa la información y separarla en dichos caracteres lo cual con lleva un esfuerzo ya que muchas de las cosas que estábamos viendo eran complicadas porque en el semestre pasado tuvimos problemas con autómatas ya que la profesora no iba,después de eso lo siguiente era comparar los caracteres una de las funciones que se investigo fue el llamado archivo tokeneizer el cual nunca había utilizado y para ello creo que fue una de las partes mas difíciles y nos llevo mas tiempo de la investigación ya que despues de investigar que era decia que este metodo lo que hacia era comparar los caracteres despues de esto lo ultimo era crear un arreglo en donde se asignaban las tablas.

Pseudocodigo

```
Clase tokeneizer( dato):
 nodo left= null
 nodoright=null
 dato= String
clase  arbol():
//declaracion de variables
self:int
self.root
insert(self, a, dato):
si a == Null:
si a = node
si es falso:
d = a.dato
si dato < d:
si a.left = self.insert
entonces (a.left,dato)
si es falso:
a.right = self.insert(a.right, dato)
 retorna a:
def inorder(self, a):// set metodo in orden
si a == Null:
retorna a null
si es falso:
self.inorder(a.left) \\se defina el get del metodo in orden
se imprime(a.dato)
self.inorder(a.right)
```

```
def preorder(self, a):se define el get  del metodo pre orden
si a == Null:
return Null
Si es falso:
se imprime(a.dato)
self.preorder(a.left)
self.preorder(a.right)
def postorder(self, a):
si a == Null:
retorna a Null
si es falso:
self.postorder(a.left)
self.postorder(a.right)
se imprime(a.dato)
def buscar(self, dato, a):
si a == Null:
retorna Null
si es falso :
si dato == a.dato:
retorna a.dato
si es falso:
si  dato < a.dato:
retorna self.buscar(dato, a.left)
si es falso:
retorna self.buscar(dato, a.right)
SI tree = arbol()
Donde Verdadero es :
os.system("cls")
```

```
se imprime("Arbol ABB")

opc = input("\n1.-Insertar nodo \n2.-Inorden \n3.-Preorden \n4.-Postorden \n5.-
Buscar \n6.-Salir

\n\nElige una opcion -> ")

si opc == '1':

si nodo = input("\nIngresa el nodo -> ")

Si nodo.isdigit():

nodo = int(nodo)

si tree.root = tree.insert(tree.root, nodo)

si es falso:

se imprime("\nIngresa solo digitos…")

Si opc == '2':

Si tree.root == None:

se imprime("Vacio")

si es falso:

tree.inorder(tree.root)

si opc == '3':

si tree.root == None:

se imprime("Vacio")

si es falso entonces:

tree.preorder(tree.root)

si opc == '4':

if tree.root == Null:

print("Vacio")

else:

tree.postorder(tree.root)

si opc == '5':

nodo = input("\nIngresa el nodo a buscar -> ")
```

si nodo.es un digito():

nodo = int(nodo)

si tree.buscar(nodo, tree.root) == None:

se imprime("\n Nodo no encontrado...")

si es falso :

se imprime("\nNodo encontrado -> ",tree.buscar(nodo, tree.root), " si existe...")

si es falso:

se imprime ("\n Ingresa solo digitos...")

si opc == '6':

se imprime ("\n Elegiste salir...\n")

imprime ("pause")

break

si es falso :

se imprime("\nElige una opcion correcta...")

imprime(null)

se imprime("pause")

Conclusión

Si bien este proyecto fue un poco mas complicado por el motivo que no conociamos muchas funciones o como separar un documento de texto por caracteres aprendimos a investigar y crear tablas con arreglos y aprendimos a utlizar la clase tokeneizer y a hacer un codigo un poco mas complejo tan bien el como comparar diferentes tablas que era un poco mas complicado porque nunca lo habiamos echo implementamos algunas funciones del codigo pasado para que pudiera funcionar este codigo al final igual que el codigo pasado solo es cuestion de aprender el como utilizar la logica solamente salen una comparacion de tokens el cual te dice que es cada funcion por decir si hay 5 signos de puntuacion o si hay 3 signos matematicos y 3 palabras reservadas

Referencias

Creación de ficheros

programacion.jias.es › Ficheros

Clase tokeneizer

www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/colecciones/stringtokenizer.html

https://es.wikibooks.org/wiki/Programación_en_Java/La_clase_StringTokenizer

felinfo.blogspot.com/2010/01/uso-de-stringtokenizer-con-varios.html

Creación de las cuádruplas

arantxa.ii.uam.es/~alfonsec/docs/compila5.htm