

Institución: Universidad de San Carlos de Guatemala

Carrera: Ingeniería en ciencias y sistemas

Curso: Compiladores 1

Nombre: José Daniel Alvarado Fajardo

Carne: 201904061

Fecha: 07/03/2021

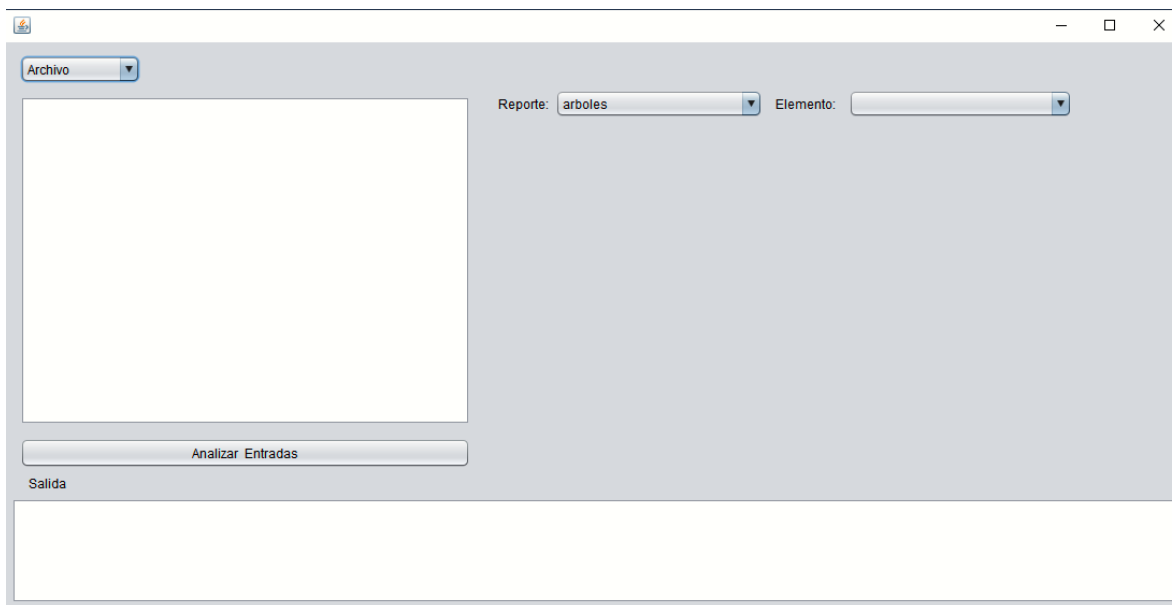


Manual de usuario de proyecto #1

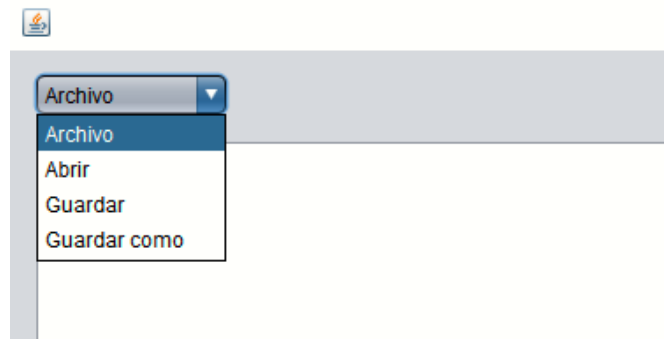
Descripción:

Este programa se realiza con la finalidad de que la facultad cuente con sistema que se capas de realizar el método del árbol y el método de Thomson, para que los estudiantes futuros puedan verificar las respuestas de sus tareas y exámenes del con curso.

Vista general del programa:



Dentro de la funcionalidades del programa se encuentran las siguientes:

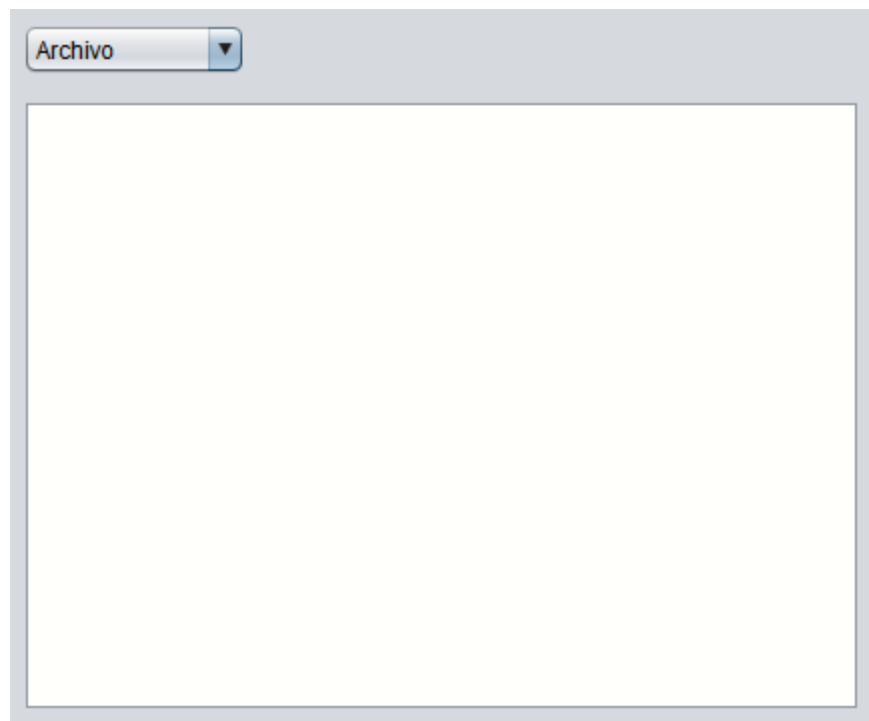


Nuevo Archivo: Por medio de esta opción se podrá crear un nuevo archivo con la extensión “.olc”.

Guardar: Permite guardar cambios realizados en el archivo actual.

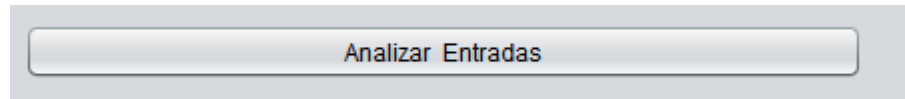
Guardar como: Nos permite guardar el archivo que se tiene abierto con otro nombre

Los archivos olc se podrán escribir dentro del siguiente panel:



Al presionar el botón de Analizar entradas, se procederá a generar árboles, tabla de siguientes, tabla de transiciones, autómatas finitos determinista, autómatas no

deterministas de Thomson y se validaran los lexemas de las disintitas expresiones regulares definidas con anterioridad.



En estos dos combos boxes se podrá elegir el tipo de reporte que se desea visualizar y la expresión correspondiente.

Reporte: Elemento:

Dentro de la consola se podrá visualizar cunado una cadena es valdia y cuando no lo es.

Salida

Ejemplo:

La imagen muestra una interfaz de usuario de un software de análisis léxico. En la parte superior izquierda hay un menú "Archivo". A la izquierda hay un editor de texto con el siguiente contenido:

```
{
<!
    Este es un comentario multilinea
    Si no da error, ya salio el proyecto
}>
CONJ: letra -> a-z; // declarando conjunto de letras desde a hasta z en r
CONJ: digito -> 0~3; // creamos el conjunto de digitos solo para 0, 1, 2 y

//agregamos Expresiones regulares
identificador -> . {letra} * [ "_ " ] {letra} {digito};
decimales -> . +{digito} . " " + {digito};
```

Debajo del editor hay un botón "Analizar Entradas". En la parte superior derecha hay dos menús desplegables: "Reporte:" con el valor "automatas" y "Elemento:" con el valor "decimales". En el centro hay un diagrama de estados de un autómata finito:

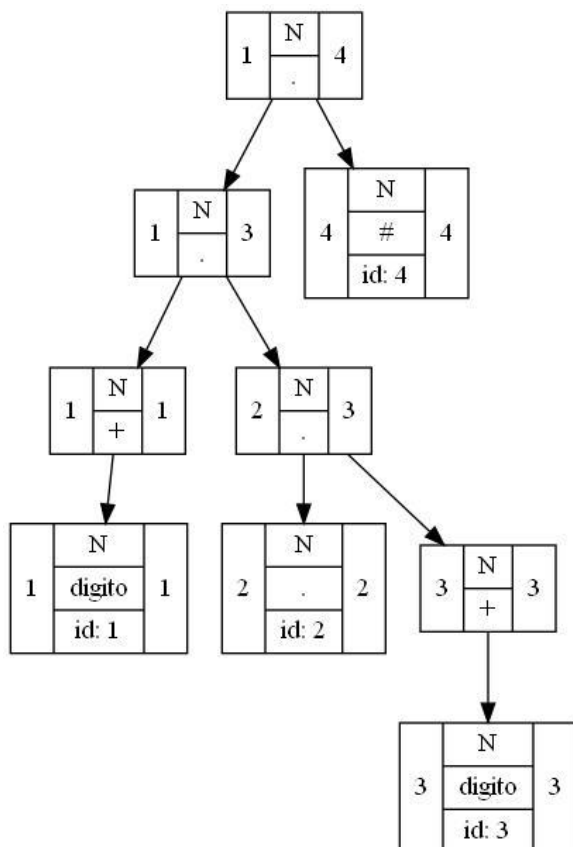
```
graph LR
    S0((S0)) -- digito --> S1((S1))
    S1 -- digito --> S1
    S1 -- ' ' --> S2((S2))
    S2 -- digito --> S3(((S3)))
    S3 -- digito --> S3
```

En la parte inferior hay una consola con el título "Salida" que muestra los siguientes mensajes:

```
La expresion: HoLA, es No valida con la expresion regular: identificador.
La expresion: 301.59, es No valida con la expresion regular: decimales.
La expresion: 1200.31, es valida con la expresion regular: decimales.
```

Ejemplo de diagrama de árbol:

Dentro del árbol se podrá observar del lado izquierdo los primeros de cada nodo y en el lado derecho los ultimos, su anulabilidad estará representada por una A en caso de que el nodo sea anulable y de no serlo tendrá una N, mientras que el valor del nodo se encuentra en medio y aquellos nodos que sean hojas contarán con id.



Ejemplo de diagrama de transiciones:

En esta tabla se muestra cada uno de los id de las hojas con su valor y valor de siguientes.

CARACTER	ID	SIGUIENTES
digito	1	1, 2
.	2	3
digito	3	3, 4
#	4	--

Ejemplo de tala de transacciones:

Aquí se puede observar los estados, los elementos que lo conforman y a que estado se puede mover con los elementos del alfabeto que correspondan.

ESTADO	digito	.
S0 = [1]	S1	---
S1 = [1, 2]	S1	S2
S2 = [3]	S3	---
S3 = [3, 4]	S3	---

Ejemplo diagrama de Thomson:

En este diagrama se mostrara un autómata finito no determinista el cual posee transiciones con épsilon.

