Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Lenguajes Formales de Programación
Primer Semestre de 2023

Sección	Catedrático	Tutor Académico
A+	Ing. Otto Amilcar Rodríguez Acosta	P.E.M. Carlos Esteban Godínez Delgado
A-	Ing. Vivian Damaris Campos Gonzales	Mario Josué Solís Solórsano
B+	Ing. David Estuardo Morales Ajcot	Diego Andrés Obín Rosales
B-	Inga. Zulma Karina Aguirre Ordoñez	Pablo Daniel Rivas Marroquin

PROYECTO NO. 1

1. DESCRIPCIÓN GENERAL

1.1 OBJETIVO GENERAL

Que el estudiante cree una herramienta la cual sea capaz de reconocer un lenguaje, dado por medio de un analizador léxico el cual cumple con las reglas establecidas, manejando la lectura y escritura de archivos para el manejo de la información. A través de un entorno gráfico.

1.2 OBJETIVOS ESPECÍFICOS

- Implementar por medio de estados un analizador léxico.
- Utilizar funciones de manejo de cadenas de caracteres en lenguaje Python.
- Programar un Scanner para el análisis léxico.
- Construir un scanner basándose en un autómata finito determinístico.
- Crear una herramienta para interactuar de forma visual con el usuario con Tkinter

1.3 DESCRIPCIÓN

Se solicita la lectura de código fuente, el cual tendrá un formato JSON, creando un programa el cual sea capaz de identificar un lenguaje dado, identificando los errores léxicos y ejecutando las instrucciones correspondientes.

Se listarán una serie de instrucciones las cuales deben de ser ejecutadas, cumpliendo con el formato asignado, generándolo un resultado y graficarlos en un archivo según la jerarquía operacional de cada instrucción. **Colocando el resultado en cada nodo que aplique.**

Los errores deben ser generados en un archivo JSON.

2. CARACTERÍSTICAS DE LA SOLUCIÓN

Para responder a las necesidades que se le plantean, se ha pensado en el desarrollo de una aplicación en lenguaje Python que permita reconocer las distintas instrucciones, y la ejecución de las mismas. Con el objetivo que se implemente el análisis léxico correspondiente.

Se deben de mostrar de manera funcional y agradable al usuario resumen de errores detectados, así como el resultado de las operaciones realizadas en cada una de las funciones que se describen más adelante, así como su respectivo archivo en forma gráfica (árbol de operaciones, incluyendo el resultado en cada nodo que aplique).

2.1 DISEÑO DE LA INTERFAZ

En la aplicación se deberán demostrar como mínimo los siguientes menús, tomando en cuenta que deberá de ser totalmente gráfica.

Archivo	**Ayuda**
Abrir	Manual de Usuario
Guardar	Manual Técnico
Guardar Como	Temas de Ayuda
Analizar	
Errores	
Salir	

- Menú
    - **Abrir**: Permite abrir un archivo para poder seguir editándolo en la aplicación
    - **Guardar**: Permite guardar el archivo que está siendo editado.
    - **Analizar**: Analiza el texto y mostrará los elementos reconocidos.
    - **Errores**: Muestra los errores del último archivo compilado.
    - **Salir**: Con esta opción se cerrará la aplicación.

- Ayuda
    - **Manual de Usuario**: Se deberá mostrar el manual de usuario realizado por el estudiante para guiar a la persona en la utilización de la aplicación. Este debe de estar en formato PDF.
    - **Manual Técnico**: Se deberá mostrar el manual técnico realizado por el estudiante para entender como se ha realizado la aplicación. En donde se evidencie el diseño y análisis previo para darle solución al proyecto. Este debe estar en formato PDF. Debe incluir el autómata utilizado para el analizador léxico, AFD encontrado por medio del método del árbol; esto debe coincidir con el código fuente de su programa.
    - **Temas de Ayuda**: Permite mostrar información del estudiante que ha creado la aplicación.

2.1.2 DEFINICIÓN DE OPERACIONES VÁLIDAS

Función	Operación
SUMA	Suma de 2 o más números u operaciones anidadas.
RESTA	Resta de 2 o más números u operaciones anidadas.
MULTIPLICACION	Multiplicación de 2 o más números u operaciones anidadas.
DIVISION	División entre números u operaciones anidadas.

POTENCIA	Potencia N de un número u operación anidadas.
RAIZ	Raíz N de un número u operación anidadas.
INVERSO	Inverso de un número u operación anidadas.
SENO	Función trigonométrica seno de un número u operación anidadas.
COSENO	Función trigonométrica coseno de un número u operación anidadas.
TANGENTE	Función trigonométrica tangente de un número u operación anidadas.
MOD	Residuo entre números u operaciones anidadas.

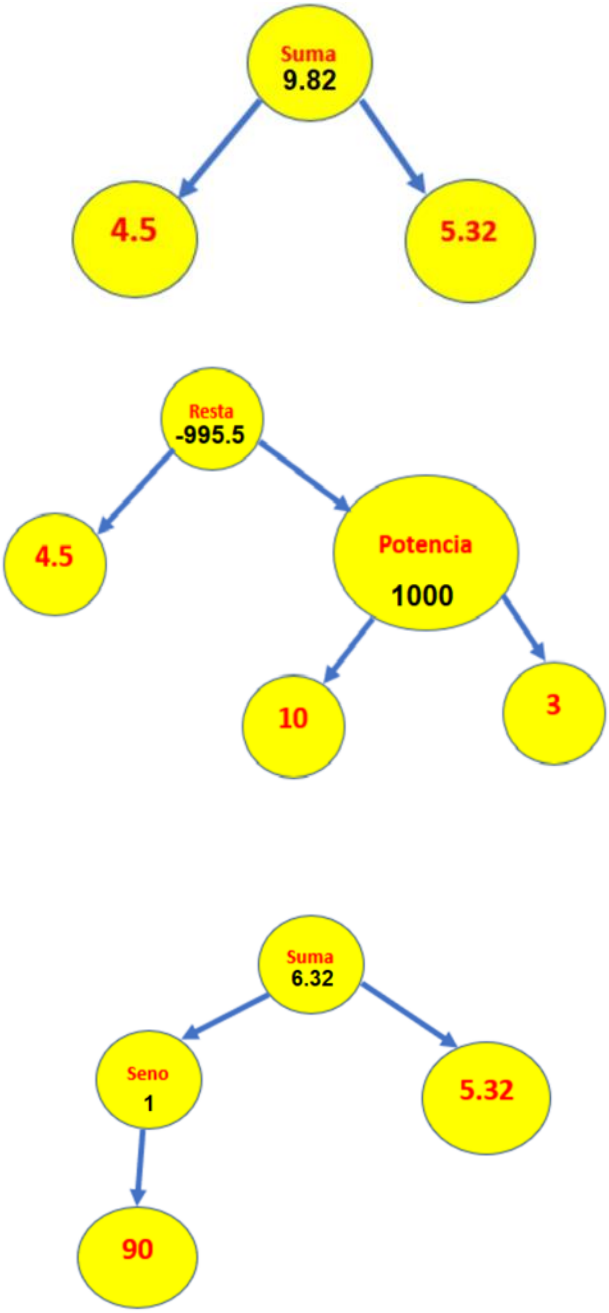
2.2 ESTRUCTURA DEL ARCHIVO DE ENTRADA

```
{
  {
        "Operacion":"Suma"
        "Valor1":4.5
        "Valor2":5.32
  },
  {
        "Operacion":"Resta"
        "Valor1":4.5
        "Valor2": [
              "Operacion":"Potencia"
              "Valor1":10
              "Valor2":3
  ]},
  {
        "Operacion":"Suma"
        "Valor1":[
              "Operacion":"Seno"
              "Valor1":90
  ]
        "Valor2":5.32
  }

  "Texto":"Realizacion de Operaciones"
  "Color-Fondo-Nodo":"Amarillo"
  "Color-Fuente-Nodo":"Rojo"
  "Forma-Nodo":"Circulo"

}
```

2.3 ESTRUCTURA DEL ARCHIVO DE SALIDA



*Ejemplo del diagrama, puede ser creativo.

2.4 ESTRUCTURA DEL ARCHIVO DE ERRORES

Si existieran errores léxicos, debe de generar una tabla de errores en formato **JSON**, con nombre **ERRORES_<<no_carne>>**, cumpliendo con la siguiente información:

```
{
    {
        "No.":1
        "Descripcion-Token":{
            "Lexema": ?
            "Tipo": Error
            "Columna": 5
            "Fila": 6
        }
    },
    {
        "No.":2
        "Descripcion-Token":{
            "Lexema": !
            "Tipo": Error
            "Columna": 8
            "Fila": 17
        }
    }
}
```

2.5 FUNCIONALIDAD DE LA APLICACIÓN

El código será cargado en un área de texto, se podrá modificar, guardar el archivo con el mismo nombre, o bien guardar el archivo con diferente nombre, y al momento de analizar se deberán mostrar los respectivos diagramas del árbol creado de acuerdo a la operación, siempre considerando la jerarquía operacional, además deberá mostrar los errores encontrados en un archivo JSON, indicando que provocó el error. **RESULTADOS_<<no_carne>>**

2.6 DOCUMENTACIÓN

- Manual técnico
- Manual de usuario
- Archivos de prueba

3 NOTAS IMPORTANTES

- El proyecto se deberá realizar en forma individual.
- El proyecto se implementará en lenguaje Python, en caso contrario no será calificado.
- Se valorará la calidad de la información proporcionada por la aplicación cuando se produzcan errores, así como la presentación de la interfaz gráfica y amigabilidad de la aplicación.
- Copias, totales o parciales de proyectos tendrán una nota de 0 puntos y las sanciones por parte de la Escuela de Sistemas.
- Sistema Operativo Libre
- Utilizar para el entorno gráfico únicamente la librería TKINTER

3.1 ENTREGA

- Ejecutable
- Código Fuente
- La entrega se realizará en la plataforma UEDI. Todos los archivos solicitados deberán ser entregados en repositorio de GitHub identificado de la siguiente forma: **[LFP]Proyecto2_carnet,** el estudiante es responsable de verificar que dentro del repositorio se encuentren todos los archivos necesarios para su calificación.
- **NO ESTA PERMITIDO EL USO DE LIBRERÍAS EXTERNAS QUE FACILITEN LA GENERACIÓN DEL ANALIZADOR LÉXICO.**
- No será permitida la modificación de archivos de entrada durante la calificación.
- La calificación deberá ser en línea y se estará grabando la reunión para tener un respaldo de la forma en que se procedió.
- La calificación de la práctica será personal y durará como máximo 30 minutos, en un horario que posteriormente será establecido.
- **Copia parcial o total del proyecto tendrá una nota de 0 puntos, y se notificará a la Escuela de Sistemas para que se apliquen las sanciones correspondientes.**
- **En el caso de no cumplir con alguna de las indicaciones antes mencionadas, NO se calificará la practica; por lo cual, se tendrá una nota de cero puntos.**

FECHA DE ENTREGA: 24 DE MARZO DE 2023 A LAS 23:59.