Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Lenguajes Formales y de programación
Primer Semestre 2024
Proyecto #1

Autómata Finito

Manual Técnico

Nombre: Jose Manuel Arana Velásquez

Carné: 201909158

Tabla de contenido

Tabla de contenido 2
Requerimiento mínimo del entorno de desarrollo..... 3
Paradigma: programación Objetos 4
Librerías..... 9

Descripción de la solución

La aplicación fue desarrollado en Python con la librería de Tkinter que nos permite crear unas interfaces esto unido a clases y funciones pudimos crear el proyecto para poder darle un uso sin necesidad de consola y poder generar todo con interfaces.

Requerimiento mínimo del entorno de desarrollo

Una distribución de Microsoft System

- Python: versión 10.10.2

-Visual Estudio Code: versión 1.87

- Procesador: Pentium II, compatible con PC a 266 MHz

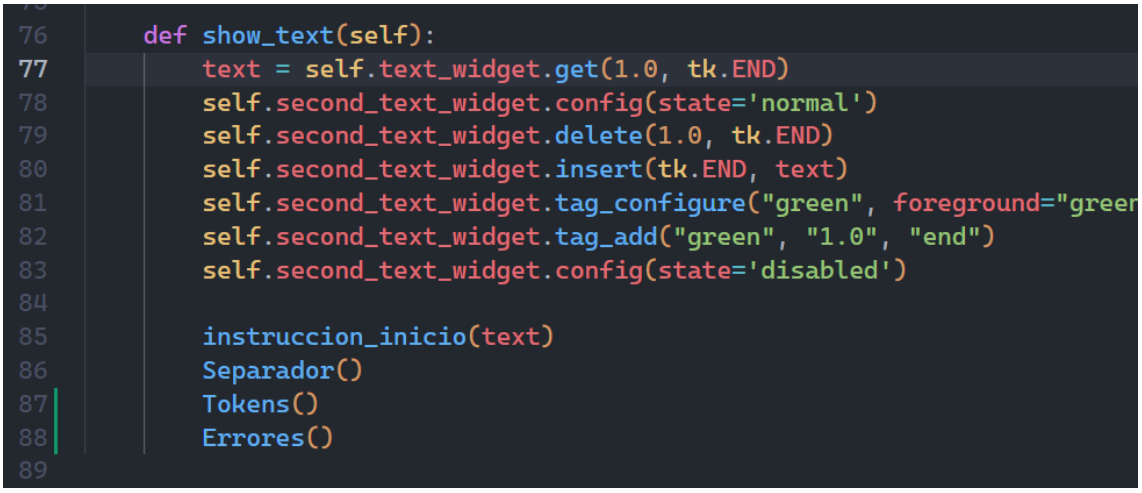
- Memoria RAM: 128 MB

- Espacio de disco duro disponible: 4 GB

Paradigma: programación Objetos

1. Main: Se Utilizo Tkinter para la generación de interfaces y poder darle unión con funciones después de diferentes acciones

```
class TextEditorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Editor de texto")

        self.editor_frame = tk.Frame(root)
        self.editor_frame.pack(fill=tk.BOTH, expand=True)

        self.line_number_bar = tk.Text(self.editor_frame, width=4, padx=4, takefocus=0, borde
        self.line_number_bar.pack(side=tk.LEFT, fill=tk.Y)

        self.text_widget = ScrolledText(self.editor_frame, wrap=tk.WORD)
        self.text_widget.pack(expand=True, fill='both')

        self.text_widget.bind('<Key>', self.update_line_numbers)
        self.text_widget.bind('<MouseWheel>', self.update_line_numbers)

        self.current_line = 1
```



```
76      def show_text(self):
77          text = self.text_widget.get(1.0, tk.END)
78          self.second_text_widget.config(state='normal')
79          self.second_text_widget.delete(1.0, tk.END)
80          self.second_text_widget.insert(tk.END, text)
81          self.second_text_widget.tag_configure("green", foreground="green
82          self.second_text_widget.tag_add("green", "1.0", "end")
83          self.second_text_widget.config(state='disabled')
84
85          instruccion_inicio(text)
86          Separador()
87          Tokens()
88          Errores()
89
```

2. Lexemas: Se creo Una instancia que obtenida la cadena y la mandaba a la clase Lexemas para poder Leerlas

```python
def instruccion_inicio(cadena):
    global n_linea
    global n_columna
    global lista_lexemas

    n_columna=0
    lexema = ''
    puntero = 0

    if not cadena:  # Verificar si la cadena está vacía
        return

    while puntero < len(cadena):  # Verificar si puntero está dentro del rango de la cadena
        char = cadena[puntero]
        puntero += 1
        if char == '\"':        #! leemos nuestra cadena y al encontrar un "" que habre empieza
            lexema, cadena = armar_lexema(cadena[puntero:])
            if lexema and cadena:
                n_columna += 1
                #Armar lexema como clase
                l = Lexema(lexema, n_linea, n_columna)
```

3. Creación Lexema: Se creo una función que creaba el lexema luego de las validaciones

```python
def armar_lexema(cadena):
    global n_linea
    global n_columna
    global lista_lexemas
    lexema = ''
    puntero = ''
    for char in cadena:
        puntero += char
        if char == ':' or char == '\"' or char == '' or char == '=':
            return lexema, cadena[len(puntero):]    #! si encuentra una  : o " termino de le
        else:
            lexema += char   #! creamos nuestros Token
    return None, None
```

4. abstracción: Se creo una Clase Abstracción que era la clase donde se alojaban las lecturas del lexema

```python
from abstraccion.abstraccion import Expression


class Lexema(Expression):
    def __init__(self, lexema, fila, columna):
        self.lexema = lexema
        super().__init__(fila, columna)

    def execute(self, environment):
        return self.lexema

    def getFila(self):
        return super().getFila()

    def getColumna(self):
        return super().getColumna()
```

5. abstracción: Se creo la clase abstracción

```python
from abc import ABC, abstractmethod

class Expression(ABC):

    def __init__(self, fila, columna):
        self.fila = fila
        self.columna = columna

    @abstractmethod
    def execute(self, environment):
        pass

    @abstractmethod
    def getFila(self):
        return self.fila

    @abstractmethod
    def getColumna(self):
        return self.columna
```

6. Separador: Se creo la clase separador que, separaba los lexemas de inicio y cuerpo

```python
def Separador():

    global Inicio
    global Cuerpo

    # Inicializa la variable de condición
    condicion = True

    for lexema in lista_lexemas:
        if lexema.lexema == 'Inicio':
            condicion = True
            continue  # Salta al siguiente ciclo sin ejecutar el resto del código en este cicl

        if lexema.lexema == 'Cuerpo':
            condicion = False
            continue  # Salta al siguiente ciclo sin ejecutar el resto del código en este cicl

        # Dependiendo de la condición, agrega el lexema a la lista correspondiente
        if condicion:
            Inicio.append(lexema.lexema)
```

7. Errores: Los errores se agregaron de los datos no agregado en Lexemas

```
else:
    lista_errores.append(char)
    cadena = cadena[1:]
    puntero = 0
    n_columna +=1
```

8. Creación De Html(Datos): Se utilizo una clase con funciones para unirlas y crear así el archivo HTML

```
def CreateHtml(cuerpo,inicio):
    # Inicializar la variable html_resultante fuera del bucle
    html_resultante = ""

    # Buscar el Head
    for elemento in inicio:
        if('Encabezado'):
            title= elemento['Encabezado']['TituloPagina']
            html_resultante+= generar_html_inicio(title)

    # Buscar el elemento Fondo primero
    for elemento in cuerpo:
        if 'Fondo' in elemento:
            color_fondo = elemento['Fondo']['color']
            html_resultante += generar_html_fondo(color_fondo)
            break  # Terminar la búsqueda una vez que se haya encontrado el fondo

    # Procesar los otros elementos
    for elemento in cuerpo:
        if 'Titulo' in elemento:
            html_resultante += generar_html_titulo(elemento['Titulo'])
        if 'Parrafo' in elemento:
```

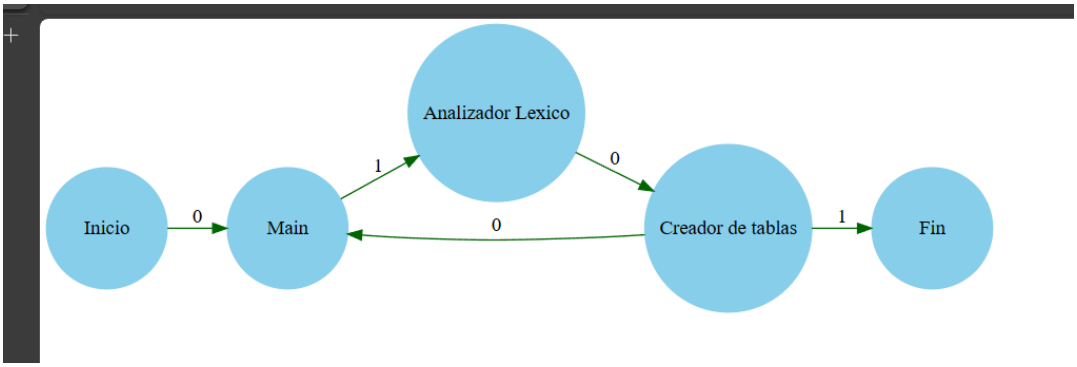
9. HTML(Errores): Se creo una tabla aparte para el manejo de errores

```
from analizador import lista_lexemas,lista_errores

def Tokens():

    llaveA = "}"
    llaveC = "{"


    with open('tabla.html', 'w', encoding='utf-8') as archivo:
        archivo.write("<table border='1'>\n")
        archivo.write("<tr>")
        archivo.write(f"<td><h1>Tokens</h1></td>")
        archivo.write(f"<td><h1>Fila</h1></td>")
        archivo.write(f"<td><h1>Columna</h1></td>")
        archivo.write("</tr>\n")



        for lista in lista_lexemas:
            archivo.write("<tr>")
            archivo.write(f"<td>{lista.lexema}</td>")
            archivo.write(f"<td>{lista.fila}</td>")
            archivo.write(f"<td>{lista.columna}</td>")
            archivo.write("</tr>\n")
```

10. HTML(Tokens): Se creo una tabla para leer los lexemas y ser puestos es una tabla


```
from analizador import lista_lexemas,lista_errores

def Tokens():

    llaveA = "}"
    llaveC = "{"


    with open('tabla.html', 'w', encoding='utf-8') as archivo:
        archivo.write("<table border='1'>\n")
        archivo.write("<tr>")
        archivo.write(f"<td><h1>Tokens</h1></td>")
        archivo.write(f"<td><h1>Fila</h1></td>")
        archivo.write(f"<td><h1>Columna</h1></td>")
        archivo.write("</tr>\n")



        for lista in lista_lexemas:
            archivo.write("<tr>")
            archivo.write(f"<td>{lista.lexema}</td>")
            archivo.write(f"<td>{lista.fila}</td>")
            archivo.write(f"<td>{lista.columna}</td>")
            archivo.write("</tr>\n")
```

11. HTML(Tokens): Se Utilizo GrapFiz para el uso de la grafica del automata



Librerías

- Tkinter: se utilizó para el manejo de Interfaces