Manual Técnico

En este manual se explicará de manera detallada los componentes y métodos utilizados para la realización del programa.

Para la realización del programa se utilizó el lenguaje de programación Python en el programa de pycharm. Inicialmente se pidió crear una interfaz grafica para la cual se utilizo la iberia de tkinter, con ella fue posible crear la interfaz que consta de una venta con un panel donde se colocarían el resto de los objetos como los son los botones o la caja de texto.

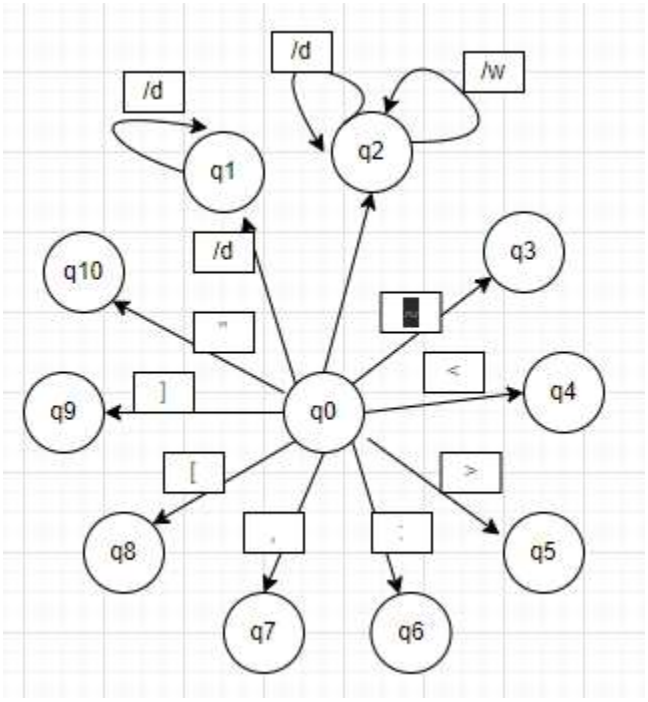
```
from tkinter import *
import tkinter as tk
```

```
ventana=tk.Tk()
ventana.title("Proyecto 1 LFP")

frame = Frame(ventana, width=600, height=450)
frame.pack()

menu = tk.Menu(ventana)
ventana.config(menu=menu)
opciones1 = tk.Menu(menu)
opciones1.add_command(label="Reperte de Tokens", command=tokenhtml)
opciones1.add_command(label="Reporte de Errores", command=erroreshtml)
opciones1.add_command(label="Manual de Usuario")
opciones1.add_command(label="Manual Tecnico")
menu.add_cascade(label="Reportes", menu=opciones1)

area = Text(frame, width=72, height=20)
area.place(x=10, y=50)

scroll = Scrollbar(frame, command=area.yview())
area.config(yscrollcommand=scroll.set)
```

Por otra parte, la librería también se utilizo para poder utilizar el navegador de archivos de la máquina que se usara.

```
def leer(self):
    self.archivo = filedialog.askopenfilename(initialdir="/",
                                              title="Select a File",
                                              filetypes=(("Text files", "*.form*"), ("all files", "*.*")))
```

para poder realizar un autómata capaz de reconocer los distintos tipos de texto fue necesario aplicar el método de grafica del árbol para poder llegar a una conclusión al momento de construir el autómata que es el que se muestra en el programa.



Con esto en mente se separaron los símbolos y los textos apartados que se tomarían como variables en el futuro del programa.

```
def q3(self, caracter: str):
    self.agregar_token(self.buffer, self.x, self.y, 'virgilla')
    self.estado = 0
    self.i -= 1

def q4(self, caracter: str):
    self.agregar_token(self.buffer, self.x, self.y, 'menor que')
    self.estado = 0
    self.i -= 1

def q5(self, caracter: str):
    self.agregar_token(self.buffer, self.x, self.y, 'mayor que')
    self.estado = 0
    self.i -= 1
```

Como se mira en la imagen estos fueron los métodos que se utilizaron para poder apartar los símbolos que reconocería el programa.

Como parte del proyecto también se pidió crear reportes de los errores y los tokens que se pudieran incluir en el archivo de entrada, para ello se crearon dos clases uno que almacenaría los tokens y otro los errores, haciendo uso del método constructor se ingresaron en una lista para poder separar y mostrar posteriormente en las páginas web.

```
class Error:
    def __init__(self, descripcion: str, x: int, y: int):
        self.descripcion = descripcion
        self.x = x
        self.y = y

    def imprimirData(self):
        print(self.descripcion, self.x, self.y)

#////////////////////////////////////////////////clase Token/////////
class Token:
    def __init__(self,lexema : str,x : int,y : int,tipo : str) -> None:
        self.lexema = lexema
        self.x = x
        self.y = y
        self.tipo = tipo

    def imprimirData(self):
        print(self.lexema, self.x, self.y, self.tipo)
```

Para realizar los html que mostrarían los reportes de los tokens y los errores se inicio por importar la librería webbrowser que serviría para trabajar con los archivos que se crearon, los cuales se escribirían en ellos el código para formar el html.

```
import webbrowser
```

Se inicio con el código básico de html para el titulo y cabeza de la página, para el cuerpo se creó una tabla con que muestra los datos que se mostrarían al usuario, tanto en el reporte de errores como en el de tokens, luego mediante unos métodos con los cuales se arreglaría el texto que iría en el cuerpo del archivo html se almacenaría ese texto en una variable y así poder agregarla al código sin ningún problema.

```
def erroreshtml():
    f = open('proyecto.html', 'w')
    t = lexico.mostrarerrores()
    mensaje = '<!DOCTYPE html>' \
              '<html>' \
              '<head>' \
              '<meta charset = "utf-8"> ' \
              '<title>Reporte de Errores</title>' \
              '</head>' \
              '<body> ' \
              '<h1>Reporte de Errores</h1> '\
              '<table class="default">' \
              '<tr>' \
              '<td>'+"Error"+'</td>' \
              '<td>'+"linea"+'</td>' \
              '<td>'+"columna"+'</td>' \
              '</tr>' \
              +t+\
              '</table>' \
              '</body> ' \
              '</html>'
```

```
def mostrartokens(self):
    d = ""
    for i in self.listaTokens:
        dato = '<tr><td>' + str(i.lexema) + '</td>' \
               '<td>' + str(i.x) + '</td>' \
               '<td>' + str(i.y) + '</td>' \
               '<td>' + str(i.tipo) + '</td>'
        d += dato
    return d
```