# MANUAL TÉCNICO

## PROYECTO 1, LFP

Universidad de San Carlos de Guatemala Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Laboratorio de Lenguajes Formales de Programación
Sección A-
Aux. Mario Josué Solis Solórzano
Lourdes Patricia Reyes Castillo
201902259

# REQUERIMIENTOS

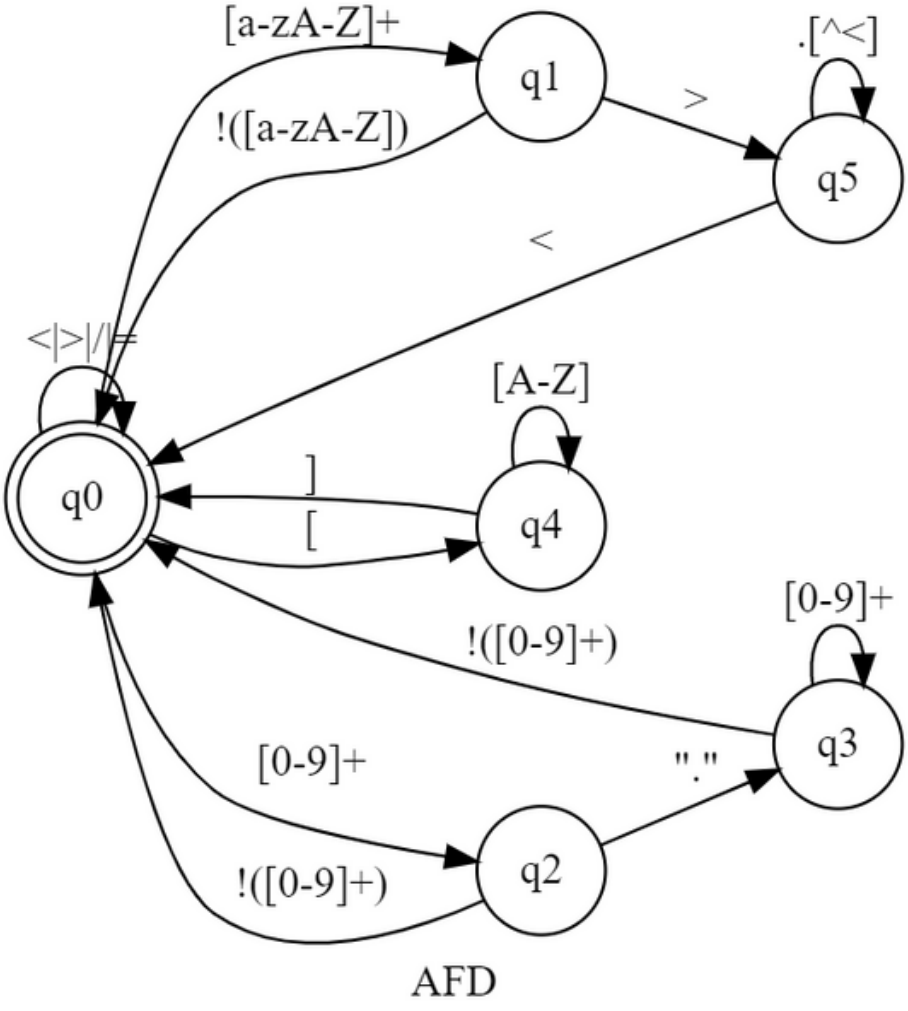
| | |
|---|---|
| CPU: | Intel de 64 bits, Recomendado: Intel Core i3 o superior |
| RAM: | 4GB, Recomendado: 8 GB o más |
| Espacio disponible en disco: | 8GB |
| Sistema Operativo: | Windows 10 |
| Resolución de Pantalla: | Básico: pantalla de 1280 x 800 |
| Programas Adicionales | Python 3.10.5 - Python 3.10.6 - Python 3.10.4 |

**Objetivos:**

- Otorgar un respaldo sobre cómo se estructura el programa.
- Resolver las operaciones básicas establecidas, leyéndolas desde un archivo de lenguaje de etiquetas, mediante el lenguaje de programación Python.

# ANÁLISIS LÉXICO

El analizador presentado se ha basado en el siguiente Autómata Finito Determinista



# I. CLASES

**Clase Token:** guarda todos los patrones válidos en el lenguaje.

```python
class token():
    def __init__(self, lexema, tipo, linea, columna):
        self.lexema = lexema
        self.tipo = tipo
        self.linea = linea
        self.columna = columna

    def mostrartoken(self):
        print("°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°")
        print("Lexema: ", self.lexema)
        print("Tipo: ", self.tipo)
        print("Linea: ", self.linea)
        print("Columna: ", self.columna)
```

**Clase Error:** guarda todos los errores encontrados en el lenguaje.

```python
class error():
    def __init__(self, caracter, descripcion, tipo, linea, columna):
        self.caracter = caracter
        self.descripcion = descripcion
        self.tipo = tipo
        self.linea = linea
        self.columna = columna

    def mostrarerror(self):
        print("°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°")
        print("Caracter: ", self.caracter)
        print("Descripción: ", self.descripcion)
        print("Tipo: ", self.tipo)
        print("Linea: ", self.linea)
        print("Columna: ", self.columna)
```

# 2. FUNCIONES

**Analizar:** esta función llama al analizador léxico y realiza las operaciones

```
def analizar():
    global scanner
    global operaciones
    if text_area.get(1.0, END)!="":
        scanner.analizar(text_area.get(1.0, END))
        if (len(scanner.listaerrores)==0):
            generarreportetokens(scanner.listatokens)
            obteneroperaciones(scanner.listatokens)
            generarreporteoperaciones(scanner.titulo, scanner.texto, operaciones)
```

**Obtener operaciones:** esta función realiza las operaciones.

```
def obteneroperaciones(listatokens):
    global operaciones
    for i in range(len(listatokens)):
        if listatokens[i].tipo=="OPERACION" and listatokens[i+1].lexema=="=":
            print(listatokens[i+2].lexema)
            op = operacion(listatokens[i+2].lexema)
            contador = i+5
            t = listatokens[contador]
            while t.tipo=="Numero":
                if t.tipo == "Numero" and listatokens[contador+2].tipo=="NUMERO":
                    op.numeros.append(listatokens[contador+2].lexema)
                    print(listatokens[contador+2].lexema)
                    contador+=8
                else:
                    break
            if op.tipo=="SUMA":
                for n in op.numeros:
                    op.total+=float(n)
                print("Total: "+ str(op.total))
```