UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 1
ING MARIO BATISTA

MANUAL TECNICO

Proyecto 1

DAVID ROBERTO DIAZ PRADO

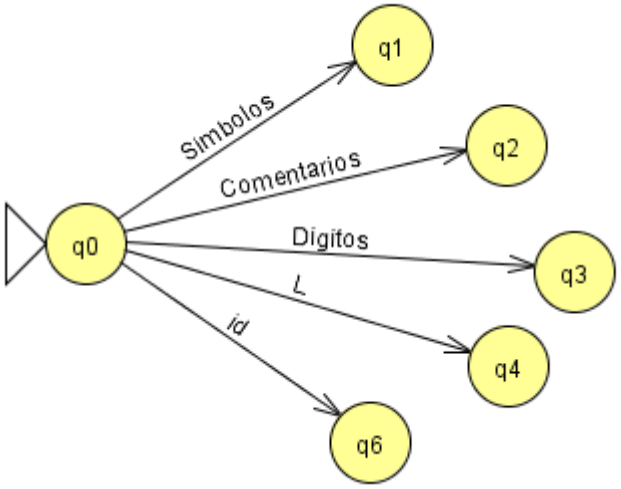
201807420

SECCION A

Aux: Luis Fernando Mazariegos

Para poder crear nuestra Solución debemos de empezar por saber que es un Analizador léxico, El analizador léxico es la primera parte de un compilador, este se encarga de verificar que todo lo que viene en la entrada sea del lenguaje y da una salida a componentes léxicos conocidos como tokens.

Para crear un Analizador léxico es importante hacer un Autómata finito Determinista, para el objetivo de la práctica se creó un mapa de estados como representación del AFD:



Este autómata se tomo como base para los diferentes tipos de lenguajes que se estarán evaluando en el proyecto, para las diferentes transiciones de este en un lenguaje especifico se deben de utilizar patrones.

En la parte de código de nuestro programa se realizó un analizador que lee lo que está dentro de un Text Area que en Tkinter suele ser solamente tx a la hora de ejecutar la función que da paso al análisis léxico este dependiendo de la extensión del archivo que ha sido seleccionado manda a un análisis especifico sobre es lenguaje de programación,

Pasando por esto el programa convierte todo el texto a minúsculas para que no haya confucion con las palabras reservadas y los identificadores

```
def EstadosIniciales(est,caracter,idError,fila,columna,consola):
    if caracter.isdigit():
        if existeT(listaTransicion,"D",est,2):
            NuevaT=Transiciones("D",est,2)
            listaTransicion.append(NuevaT)
        return 2
    elif caracter.isalnum():
        if existeT(listaTransicion,"L",est,1):
            NuevaT=Transiciones("L",est,1)
            listaTransicion.append(NuevaT)
        return 1
```

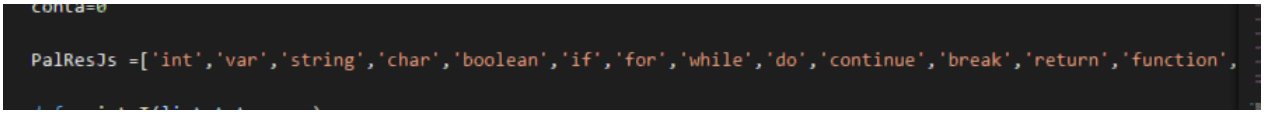
Este compara carácter por carácter y al igual que nuestro AFD va comparando estados, si este llega a su estado de aceptación se crea un lexema, esto se logró gracias al uso de recursividad.

El código maneja condiciones, si el carácter es una letra pasa a su estado correspondiente, y si es un numero pasa a su estado siguiente respectivo.

```
    columna=0
if estado==0:
    estado=EstadosIniciales(estado,caracter,idError,fila,columna, consola)
if estado==1:
    if caracter.isalpha() or caracter=='_':
        lexema=lexema+caracter
        if caracter=='_':
            if existeT(listaTransicion,'_',1,1):
                NuevaT=Transiciones("_",1,1)
                listaTransicion.append(NuevaT)
        else:
            if existeT(listaTransicion,'L',1,1):
                NuevaT=Transiciones("L",1,1)
                listaTransicion.append(NuevaT)
        estado=1
    else:
        if es_Reservada(lexema):
            NuevoToken=Token(conta,fila,columna,lexema)
            conta=conta+1
            lexema=""
            listaToken.append(NuevoToken)
        else: #aqui puedo poner para que solo sean las conjunciones como . + - = ! / *
            NuevoToken=Token(conta,fila,columna,lexema)
```

Al crear un lexema este puede ser un número, cadena de caracteres, símbolo o una palabra Reservada, las palabras reservadas se tienen ya definidas con un lexema especifico.

Para cada analizador léxico se toma un punto de partida en nuestro caso es el estado ‘0’ a partir de ahí el analizador busca coincidencia en los caracteres que vienen en la entrada y este a partir de patrones va cambiando de estado hasta llegar a un estado de aceptación, en dado caso que algún carácter no cumpla con lo anteriormente descrito este se ira guardando en una lista de errores



En un diccionario,tupla o lista se pueden guardar todas las palabras reservadas del lenguaje para que este los compare con los lexemas y al encontrar coincidencias