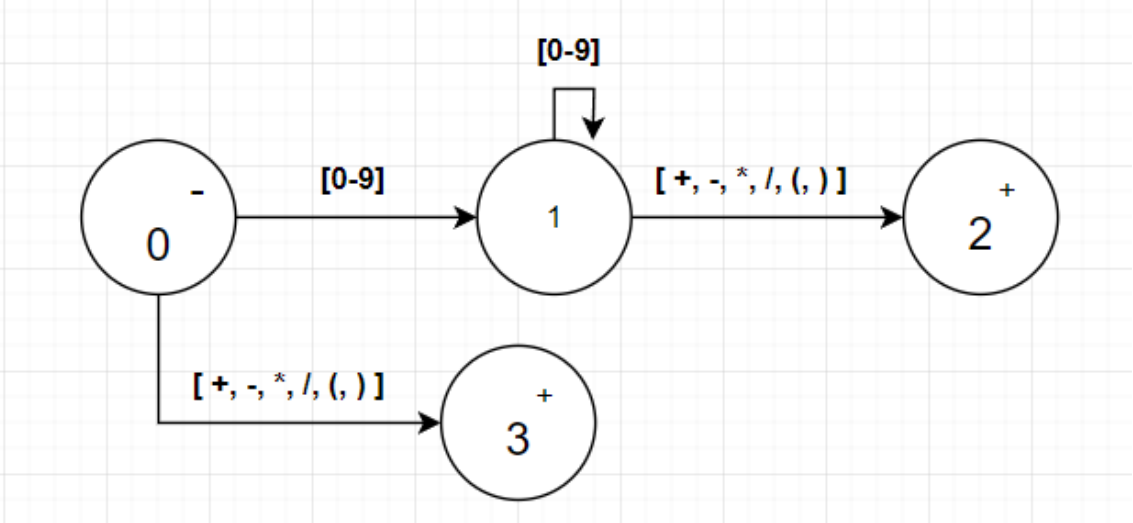
Trabajo Práctico N°2

Diagrama y Tabla de transición



Estados	[0-9]	[+, -, *, /, (,)]
0-	1	3
1	1	2
2+	-	-
3+	-	-

Estructura del TP

```
02-Lexer
 |_ docs
 |      |_ gramática descrita en formato bnf en archivo de texto
 |      \_ Informe y diagramas
 |_ Makefile
 |_ README.md
 |_ main.c
 |_ tests.c
 |_ reconocedor.h
 |_ reconocedor_automata.c
 \_ reconocedor_control.c
```

Objetivo

Este programa es un ejemplo de un autómata finito determinístico (AFD) que se utiliza para reconocer un lenguaje de expresiones regulares. El autómata analiza un archivo de entrada llamado "archivo.txt" y divide el texto en tokens.

main.c

int main(int argc, char argv): Abre un archivo llamado "archivo.txt" y comienza a procesar su contenido.

Para cada carácter leído del archivo, utiliza un autómata finito determinístico para determinar el estado del análisis y reconoce tokens basados en ese estado. Luego, imprime los tokens y las ubicaciones correspondientes.

void vaciarString(char str[]): se utiliza para borrar el contenido de una cadena str. Esto se usa para reutilizar la cadena cuando se analizan tokens.

void conseguirToken(char token[20], int estado_actual): determina el tipo de token basado en el estado actual del análisis y devuelve el token correspondiente.

void imprimirCaracterRechazado(int x, int fila, int columna): se utiliza cuando se encuentra un carácter que no se ajusta a ninguna regla del autómata. Imprime un mensaje de rechazo y la ubicación (fila y columna) del carácter.

void inicioTabla() y void finTabla(): Estas funciones se utilizan para imprimir encabezados y marcas visuales para formatear la salida del programa. inicioTabla() imprime la cabecera de una tabla que muestra los tokens, y finTabla() imprime una línea divisoria para separar las entradas en la tabla.

void imprimirToken(char string[LENGTH], char token[20]): se utiliza para imprimir los tokens y las cadenas correspondientes en una tabla. Formatea la salida de manera legible y utiliza los valores de string y token para llenar las columnas de la tabla.

reconocedor.h

- es un archivo Header que contiene las declaraciones de las funciones utilizadas en el programa principal para el reconocimiento de tokens.

reconocedor_automata.c

- Contiene funciones para el funcionamiento del programa que reconoce tokens en un archivo de entrada.

int conseguirIndice(int x): toma un entero x como argumento y devuelve un entero que representa un índice. La función se utiliza para determinar en qué categoría se encuentra un carácter dado.

En este caso, clasifica los caracteres en tres categorías: dígitos del 0 al 9, caracteres operadores [(,), +, *, -, /], y caracteres no válidos.

void conseguirToken(char token[20], int estado_actual): toma un array de caracteres token y un entero estado_actual como argumentos.

Su propósito es asignar un token basado en el estado actual del análisis léxico. Si el estado actual es 2, asigna el token "Constante numerica"; de lo contrario, asigna el token "Operador".

int T(int q, int x): toma el estado actual q y el carácter de entrada x. La función utiliza un arreglo bidimensional llamado tabla para determinar la transición a un nuevo estado según el estado actual y el carácter de entrada.

Si el carácter no es válido según la función conseguirIndice, devuelve -1 (indicando un carácter no válido). De lo contrario, devuelve el nuevo estado basado en la tabla.

tests

Verifica el funcionamiento de las funciones conseguirIndice y T definidas en el archivo "reconocedor_automata.c".

Estas pruebas están diseñadas para verificar si las funciones se comportan según lo esperado.

Makefile

```
all: tests reconocedor

reconocedor: reconocedor_automata.o
		gcc  main.c reconocedor_automata.o -o tablaToken

reconocedor_automata.o : reconocedor_automata.c
		gcc -c reconocedor_automata.c

ReconocedorTest: reconocedor_automata.o
	gcc tests.c reconocedor_automata.o -o test

tests: ReconocedorTest
	test

clean:
	del /q test.exe reconocedor_automata.o tablaToken.exe
```

Este Makefile se usa para compilar un programa llamado tablaToken y un programa de prueba llamado test.

reconocedor

- Define una regla para construir el programa tablaToken.
- Si reconocedor_automata.o no existe o ha cambiado desde la última compilación, se ejecuta el comando gcc para compilar main.c y reconocedor_automata.o en el archivo ejecutable tablaToken.

reconocedor_automata_o

- Se define una regla para compilar reconocedor_automata.c en reconocedor_automata.o.
- La opción -c de gcc indica que solo se debe compilar el código fuente en un archivo objeto sin intentar crear un ejecutable.

ReconocedorTest

- Compila el programa de prueba test.
- Dependiendo de reconocedor_automata.o, compila tests.c y reconocedor_automata.o en el ejecutable test.

tests

- Se asegurará de que "ReconocedorTest" esté actualizado y luego ejecutará el comando test.

clean

- Se utiliza para eliminar los archivos generados por las compilaciones anteriores.
- Utiliza el comando del para borrar los archivos ejecutables y el archivo objeto reconocedor_automata.o.