**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

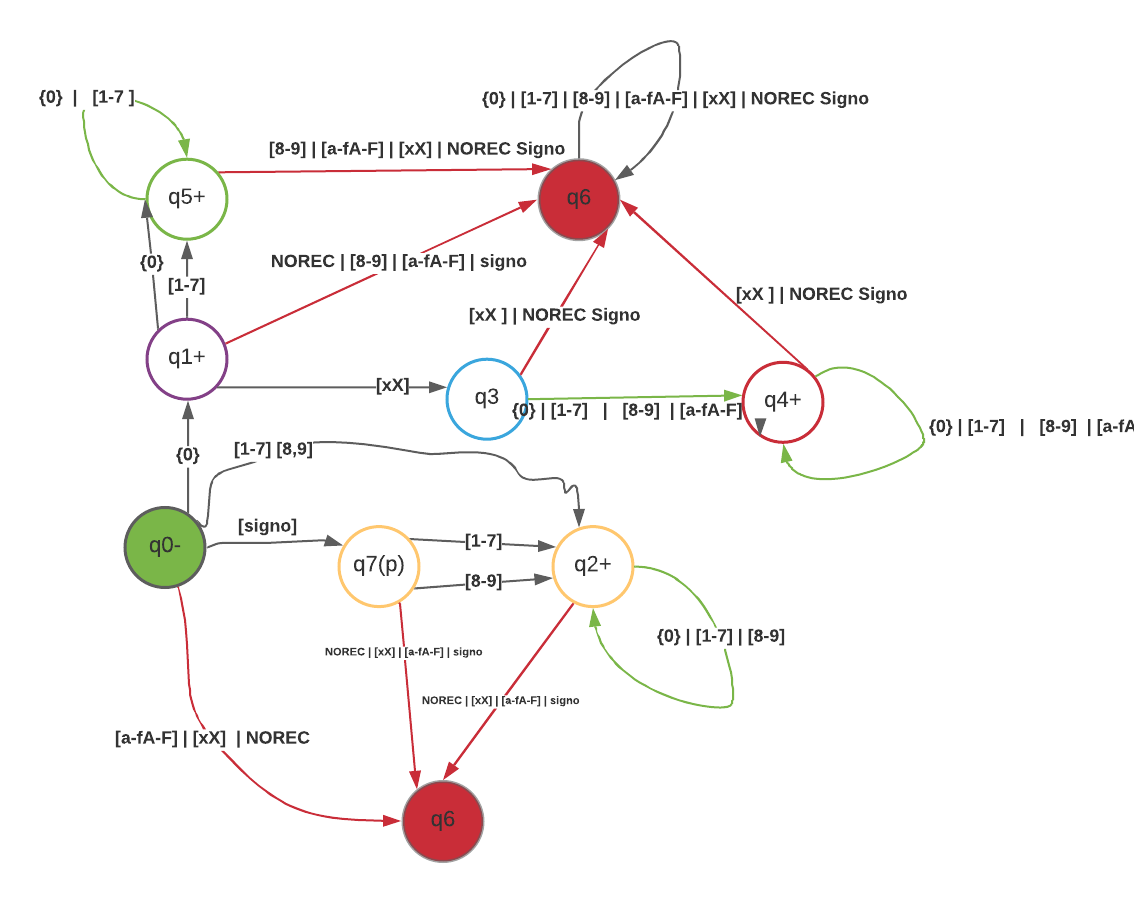
Descripción generada automáticamente**

**Explicación Del ejercicio 1:**

Para empezar el trabajo lo que hicimos fue dividirlo en dos partes, la primera consiste en resolver el autómata y la segunda en realizar el código. Para la resolución del autómata tuvimos en cuenta el ejemplo otorgado en trabajo práctico y junto con los conocimientos de resolución de autómatas vistos en el primer cuatrimestre lo logramos diagramar.

Para la segunda parte tuvimos como base 2 funciones antes vistas en clase las cuales son “EsPlabra2” y “Columna”, ellas funcionan en conjunto para determinar el estado del autómata al cual pertenece el carácter a analizar y así retornarlo (es aquí en donde se utiliza la matriz que es la tabla de transiciones).  Decidimos incluir otra función llamada “mensajeTipoDePalabra” la cual recibe el estado del autómata y con la ayuda de un switch muestra en pantalla a qué tipo de número se está haciendo referencia. A su vez decidimos usar los contadores en esta función.

En cuanto al proceso en sí, la idea principal que tuvimos fue guardar la cadena en un vector y con la utilización de un do while, ir iterando para guardar carácter por carácter, así cuando terminase la palabra poder analizarla y dar una respuesta .Esta iteración culminaría con la lectura de la última posición del vector, mientras se esté iterando pensamos en un if y else para separar las cadenas en el momento que se lea el “ & “, es decir que siempre que el dígito que provenga sea distinto del “&” el mismo iba ser guardado en una vector auxiliar, una vez leído el “&” se entra al else, en donde terminada la lectura de la palabra se iba analizar el estado que devolvía la misma para poder luego mostrar en pantalla el tipo de número que pertenece. Algo muy importante fue vaciar el vector antes que terminase el else para así poder guardar sin inconvenientes la nueva palabra. Una vez terminada la iteración mostramos en pantalla los contadores de los tipos de números junto a los errores léxicos que pudiera presentar la cadena.

AUTÓMATA UTILIZADO PARA EL EJERCICIO 1:Ded

**Definición formal del autómata:**

A= ({q0, q1,q2,q3,q4,q5,q6,q7} , {0,1-7, 8-9, a-f A-F, signo, xX, no rec} , q0 , {q1,q2,q4,q5})

**NOTA**: Para una mejor visualización se duplicó el estado de rechazo (q6) con el fin de que no se crucen las flechas.

**Explicación Del ejercicio 2:**

Al tener que utilizar el ejercicio 1, lo que hicimos fue lo siguiente:

Reutilizamos la tabla de transición del autómata, las funciones: esPalabra2, columna y mensajeTipoPalabra. Para resolver el problema primero debemos ver si es una cadena válida (decimales, +, -, \*). Para esto se reemplaza cada “+”, “-” y “\*” por un “&” en un vector auxiliar.

Luego se hace el chequeo para ver si la operación es válida. (Se reutiliza código ejercicio 1).

La operación será válida siempre que no se hayan detectado hexadecimales, octales y errores léxicos.

En caso de no ser válida, se imprime por pantalla: “CADENA INVALIDA, INGRESE UNA CADENA VÁLIDA”

Si la cadena es válida, procedemos a realizar la precedencia de la multiplicación, donde se lee posición por posición hasta encontrar un “ \* ”.

En caso de encontrarlo se leen los caracteres **anteriores** a ese \* y los **siguientes**

Un ejemplo: 3+**4**\***7**+3.

Se leerán los caracteres anteriores hasta llegar a un “+” o “-” , estos se irán  guardando en un vector para después convertirlos a un entero y con la ayuda de un algoritmo se los invertirá (ya que cuando leemos de atrás para adelante se nos invierte el orden). También recalcar que si se lee un “-” este signo se guardará en una variable signo con la ayuda de un “if”, para así poder identificar al número que sea negativo. Lo mismo se hará para los números siguientes salteando el paso del uso de algoritmo para invertir.

Obtenidos los números anteriores y siguientes se multiplicarán y se guardarán en una variable para después sumarla en otra variable que servirá para almacenar el resultado de las multiplicaciones (resultado acumulado).

**Nota**: Contaremos con una cadena idéntica a la cadena original a la cual cada vez que se lea un “ \* “, los elementos usados en esa multiplicación  se reemplazan por una “a”. Luego este nuevo vector se filtrará para conseguir el resto de la cadena de la operación y así poder operarlo de manera más sencilla.

Una vez resuelta la multiplicación de la operación original, solo queda resolver las sumas y restas. Se va leyendo cada posición del vector que las contiene hasta llegar al fin de cadena.

Dentro del segundo while (que se encarga de leer los caracteres que conforman los números), se va guardando en un vector auxiliar los caracteres para la operación. El fin de este while es leer un signo o el fin de cadena.

Cuando se lee un signo, el valor del vector auxiliar se convierte en un entero y se suma este al resultado acumulado.

Se reinicia el vector auxiliar, se guarda en la posición 0 del vector auxiliar el SIGNO que hizo salir del while inicialmente, acto seguido se setea moverposicionporsigno  en 1 , para que cuando se vuelva a entrar al while, no borre el signo guardado.

Este proceso se repite hasta llegar al fin de cadena.

Terminada la resolución de la operación se mostrará en pantalla el resultado final de la operación.