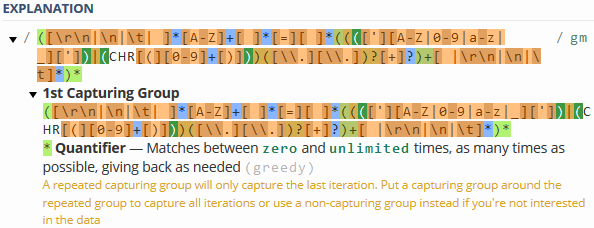
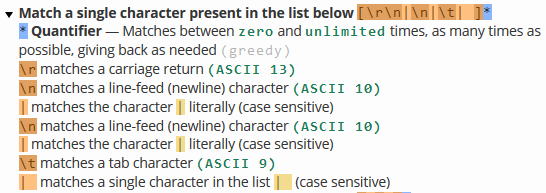
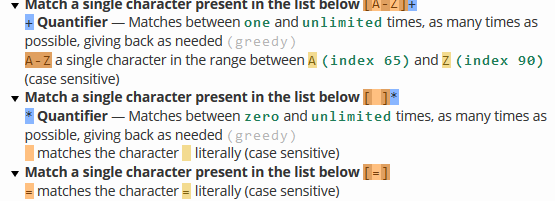
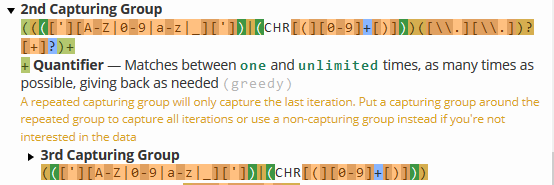
**Expresion Regular**

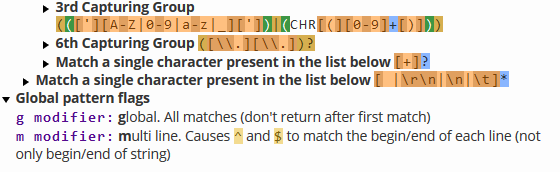
**([\r\n|\n|\t| ]\*[A-Z]+[ ]\*[=][ ]\*(((['][A-Z|0-9|a-z|\_]['])|(CHR[(][0-9]+[)]))([\\.][\\.])?[+]?)+[ |\r\n|\n|\t]\*)\***





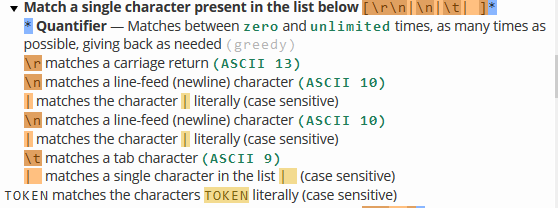


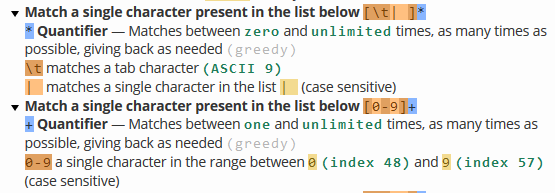


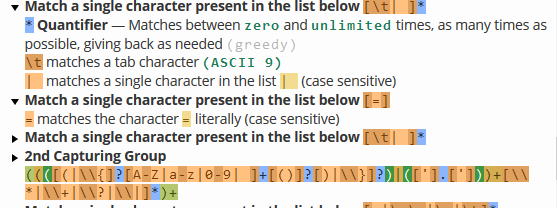


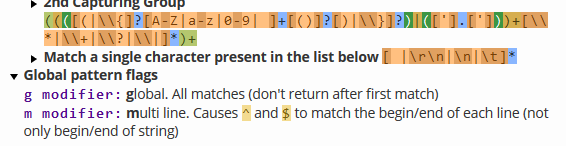
**([\r\n|\n|\t| ]\*TOKEN[\t| ]\*[0-9]+[\t| ]\*[=][\t| ]\*((([(|\\{]?[A-Z|a-z|0-9| ]+[()]?[)|\\}]?)|(['].[']))+[\\\*|\\+|\\?|\\|]\*)+[ |\r\n|\n|\t]\*)\***



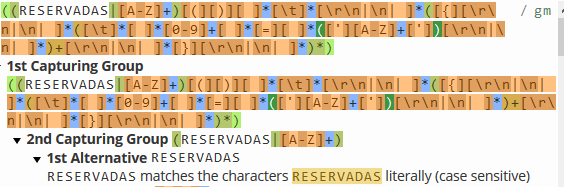


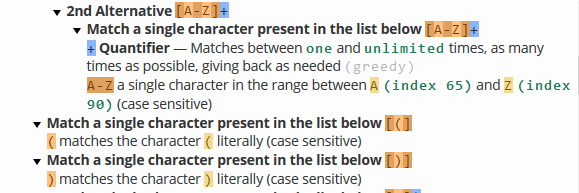


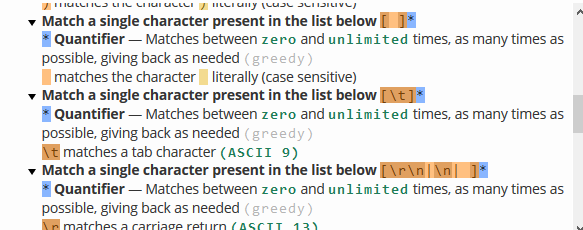


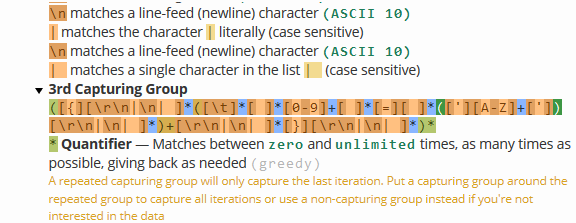


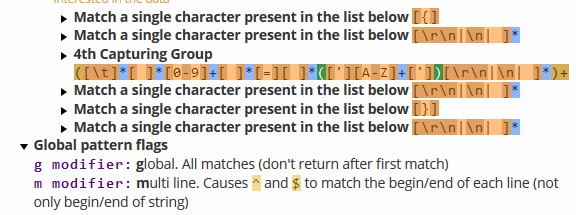
**((RESERVADAS|[A-Z]+)[(][)][ ]\*[\t]\*[\r\n|\n| ]\*([{][\r\n|\n| ]\*([\t]\*[ ]\*[0-9]+[ ]\*[=][ ]\*(['][A-Z]+['])[\r\n|\n| ]\*)+[\r\n|\n| ]\*[}][\r\n|\n| ]\*)\*)**











**Fase Numero 2:**

Los cambios realizados al árbol de expresión son acordes a los cambios que el Ingeniero hizo a la lógica así como se cambio como es usa la concatenación (.) para poder ajustarse a la expresión regular presente en los archivos de prueba.

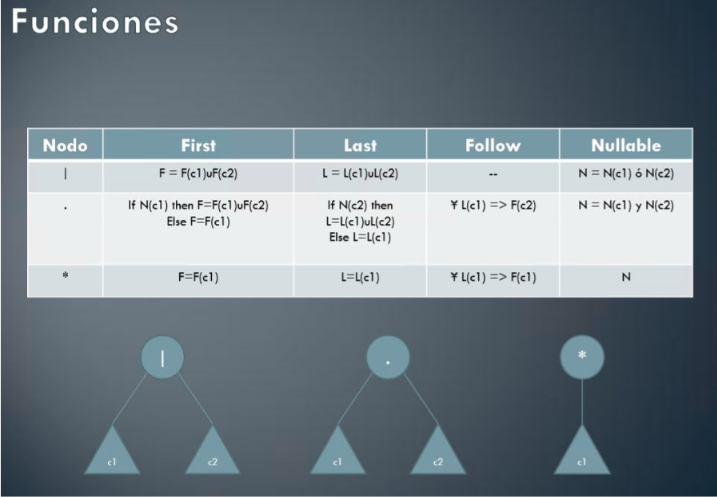
Para la lógica de como hacer los first y last lo único que se hizo fue:

1. Recorrer el árbol encontrando hojas y enumerándolas para el uso de los nodos intermedios.
2. Recorrer el árbol utilizando los nodos intermedios y haciendo los first y las acorde a la siguiente figura:

Imagen que contiene texto

Descripción generada automáticamente

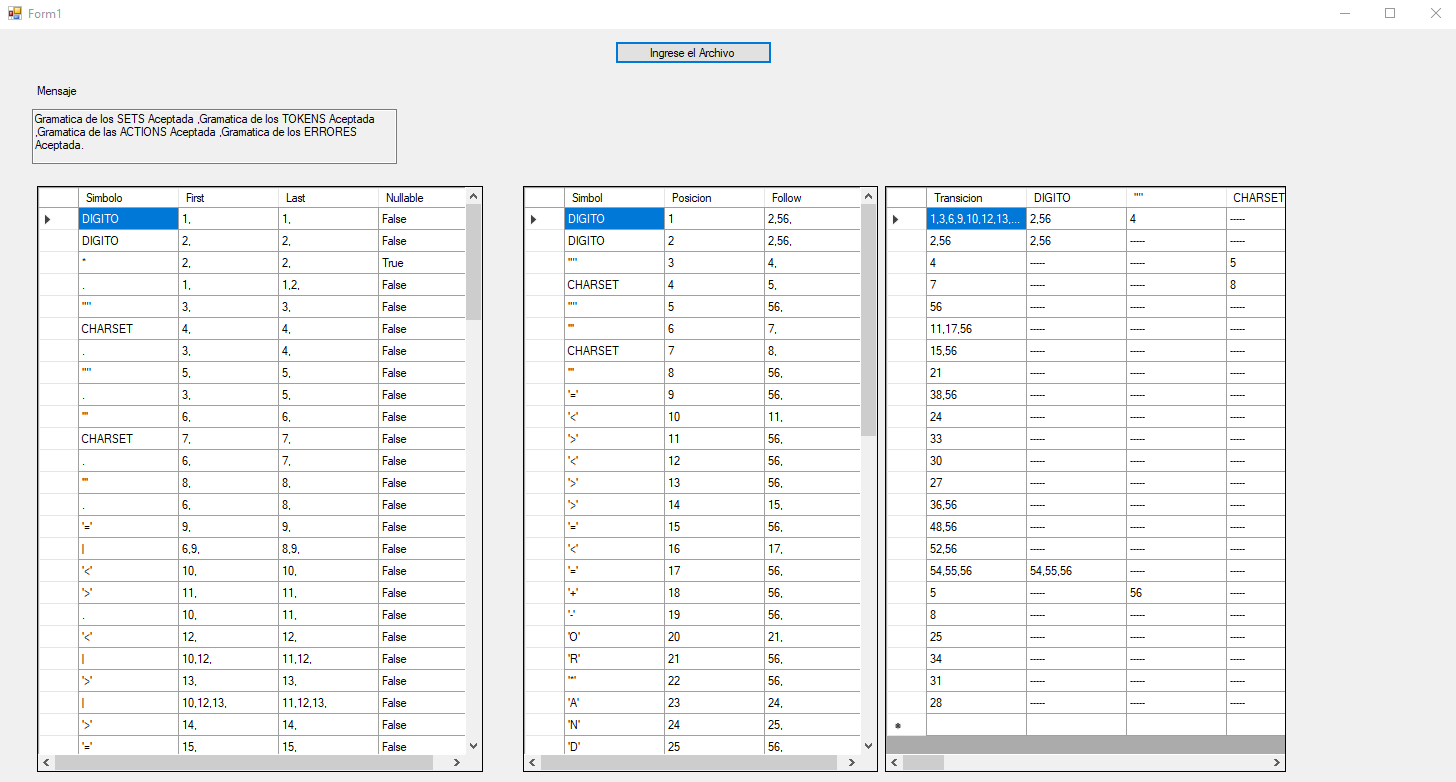
Para hacer la tabla de follows se recorrio de nuevo el árbol pero esta vez solo se tomaron en cuenta los nodos intermedios que contenían operadores exceptuando el “|” que no tiene follow, para el “?” y el “+” se utilizo la lógica que tiene “\*” en cuenta de la siguiente figura:



Y finalmente para la tabla de transiciones (debo de añadir que fue lo que mas me rompió la cabeza) lo que se hizo fue:

1. Llenar una lista con todos los símbolos no repetidos que estuvieran presentes en el árbol.
2. Después tomar todos los first del nodo raíz y encontrar su nodo en el árbol en base al numero de posición y finalmente se comparaban con la lista de símbolos para ver con cual era igual y al símbolo actual en el loop y si lo era se toma su follow de la tabla de follow.
3. Primero se hace con los first de la raíz para llenar una Queue con todas las transiciones encontradas y despues trabajar en un loop donde se iba vaciando la queue o agregando si se encontraba una transición nueva.

**Vista General de el Programa con el archivo de prueba de Gramática.txt**

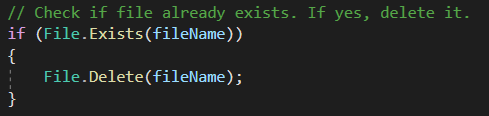


Para la tercera fase del proyecto se tiene que crear un ejecutable a través de código generado por el programa en si después de crear las tablas de los first, last, follow y la tabla de transiciones.



Como se puede ver se tomo la tabla de transiciones, las variables obtenidas de los sets, las palabras reservada y los Tokens.

A partir de la tabla de transiciones se crea un autómata este creado a partir de un ciclo y el uso de un caso (switch) donde se navega a través del autómata para reconocer una cadena si es valida en el contexto de nuestros tokens y sets. Así como se toma de los follow el símbolo terminal el cual nos ayuda a determinar cual de los estados dentro de la tabla de transiciones son estados terminales y así poder validar si la cadena finaliza en un estado válido en la gramática.



Se crea un nuevo archivo con extensión .cs y se escribe en el.



Ejemplo del código generado por el programa que después se hace un ejecutable

Finalmente después de escribir todos los casos de transición así como los estados de aceptación estos se ejecutan directamente dentro del programa usando [VBCodeProvider](https://docs.microsoft.com/en-us/dotnet/api/microsoft.visualbasic.vbcodeprovider?view=dotnet-plat-ext-3.1).